Social Interactions under Uncertainty in Multi Agent Systems

Noam Hazon
Computer Science Department

Ph.D. Thesis

Submitted to the Senate of Bar-Ilan University
Ramat Gan, Israel
October 2010

This work was carried out under the supervision of Prof. Yonatan Aumann and Prof. Sarit Kraus, Computer Science Department, Bar-Ilan University.

Acknowledgments

This research could not have been initiated nor completed without the immense support of my advisors, Yonatan Aumann and Sarit Kraus. Their advice and productive discussions greatly contributed to my understanding of the various problems. I want to thank Yonatan for teaching me how to prove results in an elegant way. He helped me realize that even if you know something is true, you must show it to others in the clearest way possible. I want to thank Sarit for teaching me what is to be dedicated to research and quality, for her hard work and helpful comments when reviewing my papers, for introducing me to various lines of research and to various researchers, and for her professional support and encouragement. To make a long story short, I would like to thank her for her invaluable guidance.

I would also like to thank Gal Kaminka, my M.Sc. supervisor, who introduced me to the world of research, and taught me basic principles and methods.

I would like to thank my lab members from the MAS groups for creating a friendly and productive environment. Also to my excellent coauthors: Paul E. Dunne, Edith Elkind, David Sarne, and Michael Wooldridge I convey my sincere appreciation.

I wish to express my gratitude to my parents, Shmuel and Tzipi, for their support and encouragement of my academic education. They have taught me important lessons that are not covered in any classroom.

Most of all, I would like to thank my wife Shira for the years of love and unlimited support, and for being there for me for every occasion. Also to my sons, Matan and Harel, who have helped in their special ways in making my life much richer throughout my graduate studies, thank you.

Finally, my thank you to G-d.

Abstract

Multi-agent Systems (MAS) deal with environments in which there are several agents that may interact. The field of multi-agent systems began its rapid advancement with the development of distributed, interconnected computer systems, such as the Internet and multi-robot teams. Such interconnected settings, where one agent interacts with another, involve studying interactions such as coordination, cooperation and collective decision making. Many researches in the field of multi-agent systems have investigated these social interactions, but with the assumption of complete information. However, agents must also be able to make good decisions in situations that involve a substantial degree of uncertainty. In our work, we provide the foundation for building such agents. Specifically, we have investigated the computational aspects of two common social interactions, collective decision making by voting and collaborative search. However, we use probabilistic models that shed a new light on these known settings.

The first part of our research investigates computational aspects of voting procedures, with the presence of uncertainty. We begin by considering the winner determination problem, which is termed "evaluation" in the probabilistic knowledge setting. In the evaluation problem a probabilistic model of voter preferences and a particular voting rule are given and the probability of a particular candidate winning needs to be computed. We provide a polynomial algorithm to solve this evaluation problem when the number of candidates is a constant, and we present experimental results illustrating the algorithm's performance in practice. However, when the number of candidates is not bounded, we prove that the problem becomes hard for many prominent voting rules. We further show that even evaluating whether a candidate has *any* chance of winning is hard in many cases, and we proffer an approximation algorithm for both problems.

We then consider another probabilistic model, where only the probability that a candidate will be preferred over another is known. This information is useful in voting trees, which are widely used in sports tournaments. In these settings the problem is not only to calculate the probability of a candidate to be chosen (i.e. the evaluation problem), but also the possibility of malicious manipulation by the election organizers (i.e. the control problem). In the setting of voting trees, the election officers may control the election by rigging the ballot agenda, i.e. the voting order. We show that the evaluation problem can be solved efficiently, while the control problem of agenda rigging is provably hard in some settings. We thus present heuristics for agenda rigging. We investigate the performance of these heuristics for both randomly generated data sets and real-world data sets from tennis and basketball competitions.

Finally, we consider computational aspects of manipulation. In this case, we do not assume to have probabilistic knowledge. Rather, we assume that we have imperfect information concerning which voters will join the coalition of manipulators. This new model of strategic voting, which is called *safe manipulation*, was recently put forward by Slinko and White [101]. We study the complexity of finding a safe manipulative vote for a number of common voting rules, while providing algorithms and hardness results for both weighted and unweighted voters. We also propose two ways to extend the notion of safe manipulation and study the computational properties of the resulting extensions.

The second part of our research investigates collaborative physical search problems with uncertain knowledge. In these settings, an agent or a team of agents (e.g., robots) seeks a given item, potentially available at different locations in a physical environment. We assume that the cost of acquiring the resource or item at a given source is uncertain (a-priori), and the agents can observe its true value only after physically arriving at the source. We first introduce and analyze the problem with a single agent, either providing a polynomial solution to the problem or proving its hardness. We also introduce a fully polynomial time approximation scheme algorithm for a specific variant of our problem. We then generalize our results to the multi-agent settings, where we analyze two models for handling resources, shared and private budget models. We present polynomial algorithms that work for any fixed number of agents, both for shared and private budget models. For non-communicating agents in the private budget model, we propose a polynomial algorithm that is suitable for any number of agents. Finally, we define our problem in an environment with self-interested agents. We show how to find a Nash Equilibrium in polynomial time, and prove that the bound on the performance of our algorithms, with respect to social welfare, is tight.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multi-agent Systems (MAS) deal with environments in which there are several agents that may interact. The field of multi-agent systems began its rapid development with the advancement of distributed, interconnected computer systems, such as the Internet and Multi-robot teams. Such interconnected settings, where one agent interacts with another, involve studying interactions such as coordination, cooperation and collective decision making [114]. Many works in the field of multi-agent systems have investigated these social interactions, but with the assumption of complete information. However, agents must also be able to make good decisions in situations that involve a substantial degree of uncertainty. In our work, we provide the foundation for building such agents. Specifically, we investigate the computational aspects of two common social interactions: collective decision making by voting and collaborative search. Nonetheless, we use probabilistic models that shed new light on these known settings.

Reaching a collective decision is a very common social interaction among people. In many multi-agent environments it is also desirable to have a mechanism which enables the agents in a system to make a collective decision on a given issue. The means by which such a collective decision is made is typically a *voting procedure* [28]. A classic, much studied issue in the political science literature is the design of voting procedures that, given the preferences of voters within a system, will result in an outcome that will be acceptable to most of the voters, i.e., that will as closely as possible reflect the preferences of voters [5, 6]. The first part of our research investigates computational aspects of voting procedures, with the presence of uncertainty.

When considering voting procedures from a computational perspective, many interesting questions arise [41]. Perhaps the most natural question from a computer science perspective is the *winner determination* problem: given the preferences of all the agents, is it possible to efficiently compute the winning outcome according to a particular voting rule? Fortunately, in this sense it seems that relatively few voting rules are hard to compute [16]. Perhaps more intriguing are questions related to the complexity of *manipulating* a voting procedure. It can be computationally infeasible for an agent to compute a beneficial manipulation [15], implying that while manipulation is possible in theory, it is infeasible in practice. Most work on the manipulation of voting procedures has considered the manipulation of elections by *voters*; specifically, strategic misrepresentation of preferences in order to bring about a more favored outcome. However, manipulation is also possible by election officers – those responsible for organizing an election, which is sometimes called "control" [17]. While the complexity of manipulation and control has been extensively studied in previous work (see, e.g., [15,34,37,38,89]), a common underlying assumption is *perfect information* about voter preferences: when computing the outcome, we have complete and correct knowledge of the preferences of all voters. However, there are many settings whereby this is not a realistic assumption, as we will discuss below.

We begin by considering the winner determination problem, which is termed "evaluation" in the probabilistic knowledge setting. We assume that for each voter, we have a probability distribution over a set of preference orderings. Thus, for each voter, we have a number of possible preference orderings – we do not know which of these orderings actually represents the preferences of the voter, but for each ordering, we know the probability that it does. The evaluation problem is therefore to compute the probability that a particular candidate will win, given the probabilistic knowledge about the preferences of the electorate and a specific voting rule. We proffer a polynomial algorithm to solve the evaluation problem when the number of candidates is a constant. We present experimental results obtained from implementation of the algorithm, illustrating that the algorithm's performance in practice is better than its predicted theoretical bound. However, when the number of candidates is not bounded, we prove that the problem becomes hard for many prominent voting rules. We further show that even evaluating whether a candidate has *any* chance of winning is hard in many cases, and we provide an approximation algorithm for both problems.

We then consider another probabilistic model, where only the probability that a candidate will be preferred over another is known. This information is useful in voting trees, which are widely used in sports tournaments. In these settings the problem is not only to calculate the probability of a candidate to be chosen (i.e. the evaluation problem), but also the possibility of malicious manipulation by the election organizers (i.e. the control problem). We investigate two common voting tree rules: the balanced voting tree, where every candidate has to participate in the same number of matches in order to win, and the linear order, which is a completely unbalanced tree. In these settings, the election officers may control the election by rigging the ballot agenda, i.e. the voting order. We show that the evaluation problem can be solved efficiently both for the balanced and unbalanced tree, while the control problem of agenda rigging is provably hard for the balanced tree. As a result we present heuristics for agenda rigging. We investigate the performance of these heuristics for both randomly generated data sets and real-world data sets from tennis and basketball competitions.

Finally, we consider computational aspects of manipulation. In this case, we do not assume to have probabilistic knowledge. Rather, we assume that we have imperfect information regarding which voters will join the coalition of manipulators. This new model of strategic voting, which we call *safe manipulation*, was recently introduced by Slinko and White [101]. In this model, a potential manipulator v announces how he intends to vote, and some of the other voters whose preferences coincide with those of v may follow suit. Depending on the number of followers, the outcome could be better or worse for v than the outcome of truthful voting. A manipulative vote is called *safe* if for some number of followers it improves the outcome from v's perspective, and can never lead to a worse outcome. We study the complexity of finding a safe manipulative vote for a number of common voting rules, providing algorithms and hardness results for both weighted and unweighted voters. We also propose two ways of extending the notion of safe manipulation to heterogeneous group of manipulators, and initiate the study of computational complexity of related questions. Our first extension of Slinko and White's model [101] is very simple and natural, and seems to behave similarly to the original model from the algorithmic perspective. However, arguably, it does not capture some of the scenarios that may occur in practice. Our second model is considerably richer, but many of the associated computational problems become intractable.

The second part of our research investigates collaborative physical search problems with uncertain knowledge. In these settings, an agent or a team of agents (e.g., robots) seeks a given item, potentially available at different locations in a physical environment. Traveling between locations, as well as acquiring the item at any given location consumes resources available to the agents. The cost of acquiring the resource or item at a given source is uncertain (a-priori), and the agents can observe its true value only when after physically arriving at the source. Sample applications involving this type of search include agents on exploration and patrol missions (e.g., an agent seeking the best location to deploy sensing equipment along its path).

Given such settings, we analyze three variants of the problem, differing in their objective: minimizing the total expected cost, maximizing the success probability given an initial budget, and minimizing the budget necessary to obtain a given success probability. Although this model captures many real world scenarios, to date it has not been investigated by other researchers. We first introduce and analyze the problem with a single agent, and either provide a polynomial solution to the problem or prove its hardness. We also introduce a fully polynomial time approximation scheme algorithm for the minimum budget variant. We then generalize our results to the multi-agent settings, where we analyze two models for handling resources, shared and private budget models. We present polynomial algorithms that work for any fixed number of agents, both for shared and private budget models. For non-communicating agents in the private budget model, we present a polynomial algorithm that is suitable for any number of agents. We also analyze the difference between homogeneous and heterogeneous agents, both with respect to their allotted resources and with respect to their capabilities. Finally, we define our variants in an environment with self-interested agents. We show how to find a Nash Equilibrium in polynomial time, and prove that the bound on the performance of our algorithms, with respect to social welfare, is tight.

Below we discuss the contribution of this dissertation in more detail. In Section 1.1 we describe our contribution to the problem of evaluation of election outcomes under uncertainty. In Section 1.2 we introduce our work on evaluation and control problems with voting trees and in Section 1.3 we present our work on the complexity of safe strategic voting. In Section 1.4 we summarize our contribution to the analysis of physical search problems with uncertain knowledge.

1.1 Evaluation of Election Outcomes under Uncertainty

Determining the winner of an election in an efficient way is the most natural computational problem of voting theory. With perfect information this problem is usually easy. However, there are many settings where this assumption is not realistic. Hence, in this part of our work we investigate the evaluation of voting rules, which is the probabilistic variant of the winner determination problem. We assume that what is known about an electorate is the following. For each voter, we have *a probability distribution over a set of preference orderings.* The idea is that although we do not know a voter's preference ordering exactly, we know that it is one of a set of possible orderings (typically a subset of the overall set of possible preference orders), and we have a probability distribution over these. This information may, for example, be obtained from historical voting data, or by sampling. In this setting, the following fundamental question arises: given such a probabilistic model of voter preferences and a particular voting rule, how hard is it to compute the probability that a particular candidate will win? We refer to this as the EVALUATION problem, and to the best of our knowledge, this question has not been addressed in the existing literature[1].

The motivation for investigating this question is not merely theoretical interest (which is, of course, in itself legitimate). In many situations, it might be beneficial to try to foresee the probability of a candidate being chosen using only partial knowledge about the other agents' preferences, which is modeled by a probability distribution as we have described. One area is the avoidance of strategic voting by a coalition of manipulators. Suppose that agent A wants to vote for its most preferred candidate. Another manipulator agent, B, could try to convince A that his preferred candidate does not have any chance of winning so he should directly vote for his second preferred candidate; otherwise this candidate will also lose to A's least preferred candidate. Due to lack of exact knowledge of how the other agents will vote, A may be convinced by B. Alternatively, A can estimate the other agent's probabilities to vote for the candidates, by asking people who know these agents, or by using the history of their former votes on the same issue. The ability to calculate the probability of a candidate winning should then assist

[1]The exception to this is the work of [35] and we discuss their work in relationship to ours.

A in deciding whether B has a valid point.

This ability to calculate the probability of a candidate winning might also be useful in other domains. For example, (and somewhat more speculatively), consider large multi-agent environments, in which there is a need to keep communication to a minimum. The voting process inevitably requires communication between the election officer and the voters in order to elicit their preferences. However, one way to reduce the communication load is to calculate the probabilities on the agents' preferences from their voting history and then calculate the probability of each candidate to win: the winner is then the candidate with the highest probability of winning. In this manner, we simulate a voting process by choosing the successful candidate without the need of communication at all. (This method might be extended to a more sophisticated protocol which uses limited communication by asking only a subset of the voters about their current preferences, though we do not investigate this possibility in this work.)

We analyze the ability to calculate the probability of a candidate winning with a variety of different settings. We first formally define the above mentioned "evaluation" question in Definition 4.1. We then give a polynomial algorithm to answer the evaluation problem when the number of candidates is a constant. While a result in [35] establishes that Evaluation is NP-hard for several key voting rules, even under quite stringent assumptions about probability distributions, we show that this result holds only for weighted voting rules with weights that are not bounded by $Poly(n)$. We then experimentally evaluate our algorithm, showing that the actual running time and space are smaller than the asymptotic bound. Therefore, we also test how many voters the polynomial time algorithm can handle for a given set of candidates. The results demonstrate that even with 6 or 7 candidates, the algorithm can handle more than 100 voters, which suggests that it may be used in many real-world voting scenarios. If the number of candidates is not bounded, the evaluation problem becomes much harder: we show that even for the well-known Plurality, k-approval Borda, Copeland, and Bucklin voting rules the problem is #P-hard. We then analyze a simpler question, known as the problem (Definition 4.2). This question simply asks whether a candidate has *any* chance of being the winner, i.e., whether the probability that the candidate will be a winner is greater than 0. Surprisingly, this problem is shown to be NP-complete (in the strong sense) even for the Plurality voting rule, when voters do not have equal weights. We give a polynomial time algorithm in cases where all voters have equal weights, for Plurality, and show that the problem is hard for many other voting rules (including k-approval, Borda, Copeland and Bucklin). This is done by establishing the connection to a related problem, the possible-winner problem [75]. Finally, we present a Monte Carlo algorithm that is able to approximately answer even the EVALUATION problem where the number of candidates is a parameter, with an error as small as desired.

Table 1.1 summarizes our key results. For comparison, we also include the results attained by Conitzer and Sandholm [35].

No. of Candidates	Weights	CHANCE-EVALUATION	EVALUATION
constant	equal	$P_{(p,k,b,c,bu,m,s,...)}$	$P_{(p,k,b,c,bu,m,s,...)}$
	bounded by $Poly(n)$	$P_{(p,k,b,c,bu,m,s,...)}$	$P_{(p,k,b,c,bu,m,s,...)}$
	otherwise	NP-hard$_{(b,c,m,s)}$ [35]	NP-hard$_{(b,c,m,s)}$ [35]
parameter	equal	$P_{(p)}$, NP-complete$_{(k,b,c,bu,m)}$	#P-hard$_{(p,k,b,c,bu)}$
	bounded by $Poly(n)$	NP-complete$_{(p,k,b,c,bu,m)}$	#P-hard$_{(p,k,b,c,bu)}$
	otherwise	NP-complete$_{(p,k,b,c,bu,m)}$	#P-hard$_{(p,k,b,c,bu)}$
approximation	any	$P_{(p,k,b,c,bu,m,s...)}$	$P_{(p,k,b,c,bu,m,s...)}$

Table 1.1: Summary of key results. The abbreviations appearing in parentheses near a complexity class indicate the voting rules for which the results have been proved. Key: p=plurality, k=k-approval, b=borda, c=copeland, bu=bucklin, m=maximin, s=stv, ...=many more voting rules.

1.2 How to Rig Elections and Competitions

Voting procedures may seem as efficient and elegant solutions for reaching a collective decision among agents. However, when the multi-agent system consists of a large number of agents, we might have hundreds of elections occurring every minute. In such settings, it is very hard to supervise elections to ensure that they are fair. Consequently, there is an increasing chance that the election officers, those responsible for organizing an election, will attempt to tilt the election results in their favor. This is, of course, a negative phenomenon, sometimes termed *control*. Bartholdi et al. [17] were the first to investigate the computational complexity of finding a successful control by changing the set of voters or candidates. In our work we investigate another type of election control – rigging the ballot agenda in order to favor a particular candidate. It is well-known that some sequential pairwise majority elections may be rigged in this way (e.g., [28, p.177] and [94]). In such an election, the candidates are voted pairwise, and the winner remains to challenge the next candidates while the loser is eliminated. The order of the pairwise elections is usually done according to a binary tree. Initially, the candidates are placed at the leaf nodes of the binary tree. Candidates at sibling nodes compete against each other in a pairwise match, and the winner of the match moves up the tree. The candidate who reaches the root node is the winner of the tournament. The chairman's role is to fix the initial ordering of the candidates (the voting agenda). If the election officer knows the preferences of the electorate – or more specifically, who would win in every possible ballot – then he may be able to fix the election agenda to the benefit of a favored candidate [79].

However, the assumption that the chairman knows exactly how a voter would vote in any given ballot is very strong, and ultimately unrealistic. It ignores the possibility of strategic voting, for one thing, but more generally, the preferences of voters will *not* be public – the chairman will have at best only *partial* knowledge about them. In light of this, the present work considers the extent to which it is possible to rig an election agenda (and, more generally, running orders for competitions) in the manner described above in the presence of *uncertain* information. We assume that an election officer knows the *probability* that a given candidate will beat another in a pairwise ballot. This probability may be obtained from opinion polls, in the case of governmental elections or similar; or it may be from form tables, in the case of sporting competitions.

In our work we focus mainly on the two most common ways to organize a set of pairwise elections. One obvious way is to use a balanced binary tree. With this tree structure, every candidate has to participate in the same number of matches in order to win. Due to its fairness, this voting tree is widely used in many social settings, as well as in sporting competitions (e.g. the soccer world cup). We also investigate a rather unfair voting tree, where elections are ordered according to a linear order. In such an election, the first two candidates in the ordering will be in a simple majority ballot against each other, with the winner then going on to face a ballot against the third candidate, and so on, until the winner of the final ballot is the overall winner. This tree structure has been used in boxing competitions, and to elect the city to host the Olympic Games. In the voting literature, Koutsoupias et al. [78] suggested to use this structure for voting in multi-criteria elections and Xia et al. [117] analyzed the assumption which makes it applicable.

Our problem may at first seem narrow – a very restricted class of voting rules, and a very specific design objective. But this seemingly simple question has turned out to be surprisingly subtle and some of the answers are counter-intuitive. To begin with, note that the number of possible agendas grows extremely quickly with the number of candidates, i.e., $O(\frac{m!}{2^{m-1}})$ when the possible tree structure is limited to be of balanced tree only. Thus, even for a small number of candidates it can be hard to answer the agenda rigging question. For $m = 2; 4; 8; 16; 32$, the numbers of possible, non-duplicate agendas are $1; 3; 315; 638 \times 10^6; 122 \times 10^{24}$, respectively. We also note that the two classes of voting trees that we analyze, linear order and balanced tree, are widely used in the field of social choice and sporting competitions. They also play a key role in other social and commercial settings, ranging from employment interview processes to patent races and rent-seeking contests (see [82, 91, 103] for details and further discussion).

Consequently we need to investigate the complexity of finding an agenda for unbalanced and balanced voting trees. First we formally define the underlying assumptions and problems. Before analyzing the problem of rigging an agenda, we must check whether evaluating an agenda, i.e. computing the winning probabilities of the candidates, can be done efficiently. Hence we present a polynomial time algorithm for evaluating an agenda with any voting tree, and show an optimized version of this algorithm for balanced voting trees. We then show that rigging an agenda for balanced voting trees is provably hard (the complete proof is due to [104, 105]). In the linear order case, we first show how to improve the general agenda evaluation algorithm for linear orders, and prove the unfairness of the linear order rule; a candidate can only benefit by going late in a voting order. Thus, the election officer can try to increase a candidate's chance of winning by placing it last in the voting order. We then show that a relaxed version of the original rigging agenda problem is hard to solve. However, it is also natural to ask whether there is any agenda which would make a specific candidate the winner with a non-zero probability. In the linear order case, we show that this problem can be solved in polynomial time. Our hardness results may lead us to conjuncture that a designer cannot benefit from having the probabilistic information, since it is hard to rig an election agenda even with this input. However, in practical terms, the worst-case analysis is not enough. We thus present heuristics for agenda rigging. We investigate the performance of these heuristics for both randomly generated data sets and real-world data sets from tennis and basketball competitions. Our heuristics achieved over 96% of the optimal solution on average for the randomly generated and the basketball data set, and performed reasonably well for the tennis data set. Finally, it is important to clarify our motivation for this work. We are, of course, *not* advocating election manipulation, or trying to develop techniques to make it easier! If we can identify cases where election manipulation is easy in practice (even if it is hard in theory), then we can use this information to design elections so as to avoid the possibility of manipulation.

1.3 Complexity of Safe Strategic Voting

Computational aspects of voting manipulation is an active topic of current research [41]. While the complexity of the manipulation problem for a single voter is quite well understood, more recently researchers have begun looking on coalitional manipulation, i.e., manipulation by multiple, possibly weighted voters. In this setting, the standard formulation taken by all recent works is as follows: we are given a set of votes that have been cast, and a set of manipulators. We are asked whether the manipulators can cast their vote in a way that makes a specific candidate win the election [35]. In this model, the manipulators want to get a particular candidate elected, and select their votes based on the non-manipulators preferences that are publicly known. Unlike the sincere voters, the manipulators are not endowed with preferences, i.e., ordering of candidates. This model is somewhat unsatisfactory for two reasons. First, it departs from the standard model of manipulation considered by Gibbard [57] and Satterthwaite [96], in which the manipulator, too, has a preference over the candidates, and a manipulation is deemed successful if it leads to an election outcome that the manipulator prefers to the outcome of truthful voting. Second, it is asymmetric in its treatment of sincere voters and manipulators, and thus does not explain how the manipulating coalition forms. Therefore, it is desirable to have a plausible model of the coalition formation process that would enable us to develop a better understanding of coalitional manipulation. In such a model the manipulators would start out by having the same type of preferences as sincere voters, and then some agents—those who are not satisfied with the current outcome and are willing to submit an insincere ballot—would get together and decide to coordinate their efforts.

However, it is quite difficult to formalize this intuition so as to obtain a realistic model of how the manipulating coalition forms. In particular, it is not clear how the voters who are interested in manipulation should identify each other, and then reach an agreement about which candidate to promote. Indeed, the latter decision seems to call for a voting procedure, and therefore is in itself vulnerable to strategic behavior. Further, even assuming that suitable coalition formation and decision-making procedures exist, their practical implementation may be hindered by the absence of reliable two-way communication among the manipulators.

In a recent paper [101], Slinko and White put forward a model that provides a partial answer to these questions. They consider a setting where a single voter v announces his manipulative vote L (the truthful preferences of all agents are, as usual, common knowledge) to his set of associates F, i.e., the voters whose true preferences coincide with those of v. As a result, some of the voters in F switch to voting L, while others (as well as all voters not in F) vote truthfully. This can happen if, e.g., v's instructions are broadcast via an unreliable channel, i.e., some of the voters in F simply do not receive the announcement, or if some voters in F consider it unethical to vote untruthfully. Such a situation is not unusual in politics, where a public figure may announce her decision to vote in a particular manner, and may be followed by a subset of like-minded people. That is, in this model, the manipulating coalition always consists of voters with identical preferences (and thus the problem of which candidate to promote is trivially resolved), and, moreover, the manipulators always vote in the same way. Further, it relies on minimal communication, i.e., a single broadcast message. However, due to lack of two-way communication, v does not know how many voters will support him in his decision to vote L. Thus, he faces a dilemma: it might be the case that if x voters from F follow him, then the outcome improves, while if some $y \neq x$ voters from F switch to voting L, the outcome becomes even less desirable to v than the current alternative (we provide an example in Section 6.1). If v is conservatively-minded, in such situations he would choose not to manipulate at all. In other words, he would view L as a successful manipulation only if (1) there exists a subset $U \subseteq F$ such that if the voters in U switch to voting L, the outcome improves; (2) for any $W \subseteq F$, if the voters in W switch to voting L the outcome does not get worse. Slinko and White [101] call any manipulation that satisfies (1) and (2) *safe*. The main result of [101] is a generalization of the Gibbard–Satterthwaite theorem [57, 96] to safe manipulation: the authors prove that any onto, non-dictatorial voting rule with at least 3 alternatives is safely manipulable, i.e., there exists a profile in which at least one voter has a safe manipulation. However, Slinko and White do not explore the computational complexity of the related problems.

In our work we focus on algorithmic complexity of safe manipulation, as defined in [101]. We first formalize the relevant computational questions and discuss some basic relationships between them. We then study the complexity of these questions for several classic voting rules, such as Plurality, Veto, k-approval, Bucklin, and Borda, for both weighted and unweighted voters. For instance, we show that finding a safe manipulation is easy for k-approval and for Bucklin, even if the voters are weighted. In contrast, for Borda, finding a safe manipulation—or even checking that a given vote is safe—turns out to be hard for weighted voters even if the number of candidates is bounded by a small constant.

We then explore whether it is possible to extend the model of safe manipulation to settings where the manipulator may be joined by voters whose preferences differ from his own. Indeed, in real life a voter may follow advice to vote in a certain way if it comes from a person whose preferences are similar (rather than identical) to hers, or simply because she thinks that voting in this manner can be beneficial to her. For instance, in politics, a popular personality may influence many different voters at once by announcing his decision to vote in a particular manner. We propose two ways of formalizing this idea, which differ in their approach to defining the set of a voter's potential followers, and provide some results on the complexity of safe manipulation in these models.

In our first extension, a manipulator v may be followed by all voters who rank the same candidates above the current winner as v does. That is, in this model a voter u may follow v if any change of outcome that is beneficial to v is also beneficial to u. We show that some of the positive algorithmic results for the standard model also hold in this more general setting. In our second model, a voter u may follow a manipulator v that proposes to vote L, if, roughly, there are circumstances when voting L is beneficial to u. This model tends to be computationally more challenging: we show that finding a safe strategic vote in this setting is hard even for very simple voting rules.

1.4 Physical Search Problems with Uncertain Knowledge

Frequently, in order to successfully complete its task, an agent may need to *explore* (i.e., search) its environment and choose among different available options. For example, an agent seeking to purchase a product over the internet needs to query several electronic merchants in order to learn their posted prices; a robot searching for a resource or a tangible item needs to travel to possible locations where the resource is available and learn the configuration in which it is available as well as the difficulty of obtaining it there. In these environments, the benefit associated with an opportunity is revealed only upon observing it. The only knowledge available to the agent prior to observing the opportunity is the probability associated with each possible benefit value of each prospect.

While the exploration in virtual environments can sometimes be considered costless, in physical environments traveling and observing typically also entails a cost. Furthermore, traveling to a new location may increase or decrease the distance to other locations, so the cost associated with exploring other unexplored locations changes. For example, consider a Rover robot with the goal of mining a certain mineral. Potential mining locations may be identified based on a satellite image, each associated with some uncertainty regarding the difficulty of mining there. In order to assess the amount of battery power required for mining at a specific location, the robot needs to physically visit there. The robot's battery is thus used not only for mining the mineral but also for traveling from one potential location to another. Consequently, an agent's strategy in an environment associated with search costs should maximize the *overall* benefit resulting from the search process, defined as the value of the option eventually used minus the costs accumulated along the process, rather than merely finding the best valued option.

In physical environments, it is common to use a team of agents rather than a single agent. Extending the single agent solution to multi-agent strategy may require subdividing the search space among the different agents. However, if agents have means of communication, then they may not wish to become too distant, as they can call upon each other for assistance. For example, even if a Rover does not have sufficient battery power for mining at a given location, it may be useful for it to travel to the site in order to determine the exact mining cost, and call for other robots that do have the necessary battery power. In this case, the scheduling of the robots' travel times is key, and must be carefully planned. If the agents are not fully cooperative, a selfish behavior should also be handled. Each one of the agents will try to minimize its traveling costs while still achieving the group's goal.

Finally, agents may be of different types, or with different amounts of resources. For example, Rover robots may enter the mission with different initial battery charges. They may also differ in their capabilities, like a team of rovers where some were specifically designed for mining missions, and thus require less battery power for the same mining task.

This part of our work aims at taking the first steps in understanding the characteristics of such physical environment settings, both for single and multi-agent cases, and developing efficient exploration strategies for the like. To the best of our knowledge, this is the first work to do so. Our main focus is on environments where the opportunities are aligned along a path, as in the case of perimeter patrol. We note that many single and multi-agent coverage algorithms convert their complex environment into a simple long path [51, 70, 102]. Furthermore, we show that the problem in more general metric spaces is NP-complete, even for tree graphs. For exposition purposes we use in the remaining of this work the classical procurement application where the goal of the search is purchasing a product and the value of each observed opportunity represents a price. Of course, this is only one example of the general setting of exploration in a physical environment, and the discussion and results of this work are relevant to any such setting, provided that exploration and fulfilling the task use the same type of resource.

We consider three variants of the problem, differing in their objective. The first (*Min-Expected-Cost*) is the problem of an agent that aims to minimize the expected total cost of completing its task. The second (*Max-Probability*) considers an agent that is given an initial budget for the task (which it cannot exceed) and needs to act in a way that maximizes the probability that it will complete its task (e.g., reach at least one opportunity with a budget large enough to successfully buy the product). In the last variant (*Min-Budget*) the agent is requested to guarantee a pre-defined probability of completing the task, and needs to minimize the overall budget that will be required to achieve the said success probability. We also consider the multi-agent extensions of these variants. While the first variant fits mostly product procurement applications, the two latter variants fit well into applications of robots engaged in remote exploration, operating with a limited amount of battery power (i.e., a budget).

Summary of Results. We first consider the single agent case. We prove that in general metric spaces all three problem variants are NP-hard. Thus, as mentioned, we focus on the path setting. For this case we provide a polynomial algorithm for the *Min-Expected-Cost* problem. We show the other two problems (*Min-Budget* and *Max-Probability*) to be NP-complete even for the path setting. Thus, we consider further restrictions and also provide an approximation scheme. We show that both problems are polynomial if the number of possible prices is constant. Even with this restriction, we show that these problems are NP-complete on a tree graph. For the *Min-Budget* problem, we provide an FPTAS (fully-polynomial-time-approximation-scheme), such that for any $\epsilon > 0$, the *Min-Budget* problem can be approximated with a $(1 + \epsilon)$ factor in $O(n\epsilon^{-6})$ time, where n is the size of the input.

For the multi-agent case, we first analyze a shared budget model, where all the resources and costs are shared among all the agents. We show that if the number of agents is fixed, then all of the single-agent algorithms extend to k-agents, with the time bounds growing exponentially in k. Therefore the computation of the agents' strategies can be performed whenever the number of agents is relatively moderate, a scenario characterizing most physical environments where several agents cooperate in exploration and search. If the number of agents is part of the input then the multi-agent version of *Min-Budget* and *Max-Probability* are NP-complete even on the path and even with a single price.

We then investigate a model of private budgets, where each agent has its own initial budget. We again assume that the number of possible prices is bounded. In this case, we separately consider the setting where agents can communicate and the setting where they cannot. For non-communicating agents we show a polynomial algorithm for the *Max-Probability* problem that is suitable for any number of agents. For the *Min-Budget* problem with non-communicating agents, we present a polynomial algorithm for the case in which all agents must be allotted identical resources, but show that the problem is NP-hard for the general case (unless the number of agents is fixed). Next we consider agents that can communicate, and can call upon each other for assistance. As noted above, in this case the scheduling of the different agents' moves must also be carefully planned. We present polynomial algorithms for both the *Max-Probability* problem and the *Min-Budget* problem that work for any constant number of agents (but become non-polynomial when the number of agents is not constant).

We then move to the self-interested agents setting, where the agents seek an item but each agent tries to minimize the use of its own private budget for traveling. We define two games, a sequential game, *Min-Expected-Cost-Game*, and a simultaneous game, *Min-Budget-Game*. We show that when the number of possible prices is bounded and there are a fixed number of agents, the strategy that maximizes the social welfare can be found in polynomial time. We also show that in the *Min-Budget-Game* this strategy is a Nash Equilibrium, but this is not always the case in the *Min-Expected-Cost-Game*. We therefore present a polynomial algorithm that guarantees a strategy which is a Nash Equilibrium will be found. Furthermore, we show an upper bound on the algorithm's performance, and prove that it is tight.

Finally, we extend our results to the case of heterogenous agents with different capabilities, and discuss the assumptions that we made throughout our work. Tables 1.2 presents a summary of the results. Empty entries represent open problems.

1.5 Thesis Overview

This dissertation comprises 9 chapters and 3 appendixes, organized into two main parts (see Figure 1.1). This chapter constitutes the introduction of the thesis. The next chapter surveys the related work. Chapters $3 - -6$ constitute Part 1 of the dissertation, which deal with the computational aspects of elections with uncertainty. Chapters $7 - -8$ constitute Part 2 of the dissertation, which consider physical search problems with uncertain knowledge. In Chapter 10 we provide our conclusions and discuss future work. The appendixes consist of the proofs we have omitted from the main text for ease of reading.

		Min-Expected-Cost	*Max-Probability*	*Min-Budget*
General metric spaces		NP-hard	NP-complete	NP-complete
Trees			NP-complete	NP-complete
Path	Single price	n/a	$O(m)$	$O(m)$
	d prices	$O(d^2m^2)$	$O(2^d\frac{e\cdot m}{d})^{2d})$	$O(2^d\frac{e\cdot m}{d})^{2d})$
	General case	$O(d^2m^2)$	NP-complete	NP-complete
	$(1+\epsilon)$ approximation	n/a		$O(n\epsilon^{-6})$

(a) Single agent.

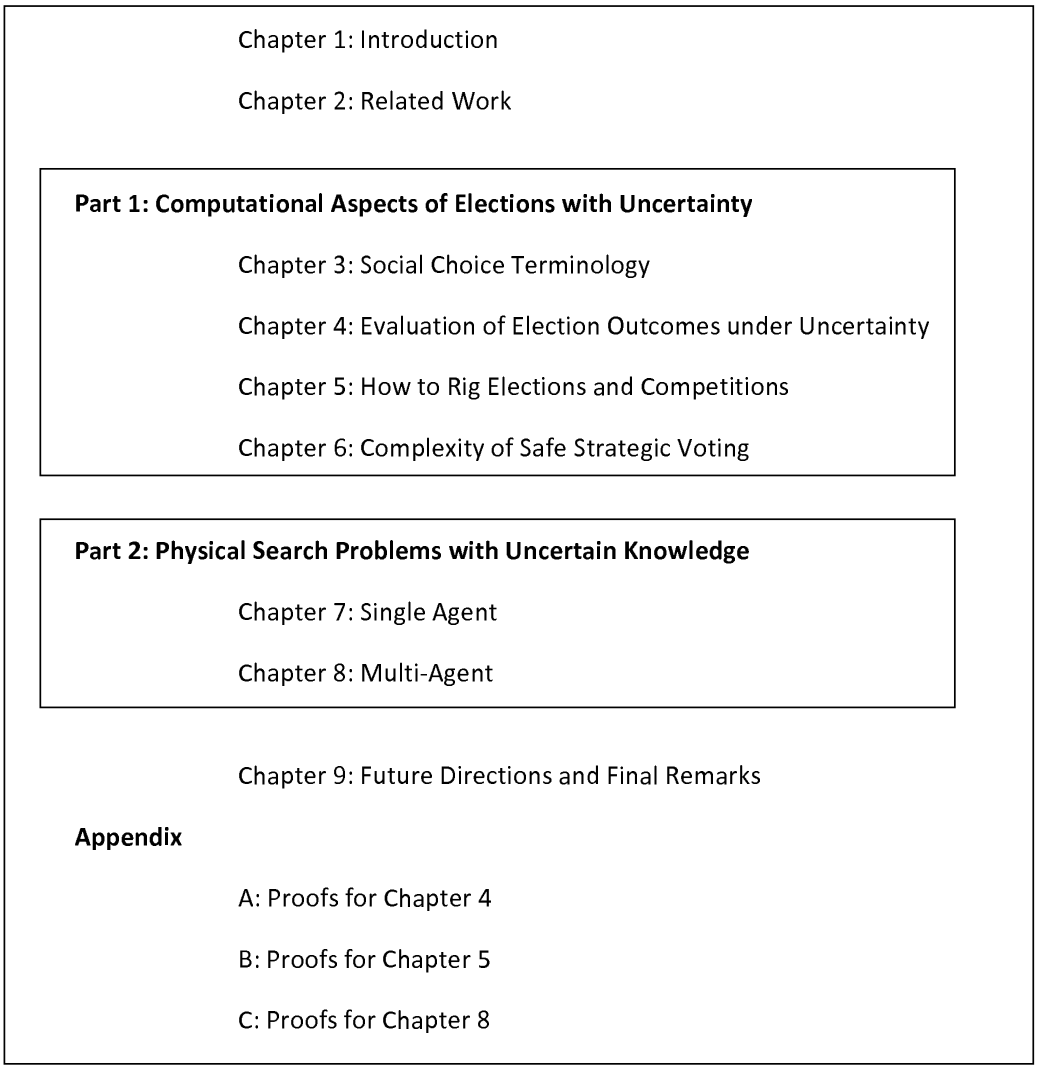
	Min-Expected-Cost	*Max-Probability*	*Min-Budget*
k agents	$O(2^d2^k\frac{m}{k})^{2k})$	$O(m2^{kd}\frac{em}{2kd})^{2kd})$	$O(m2^{kd}\frac{em}{2kd})^{2kd})$
General case		NP-complete	NP-complete
$(1+k\epsilon)$ approximation	n/a		$O(n\epsilon^{-6k})$

(b) Multi-agent, shared budget, on the path.

		Max-Probability	*Min-Budget*$^{\text{identical}}$	*Min-Budget*$^{\text{distinct}}$
No-communication	fixed k	$O(m^3k^2)$	$O(m^3k^2\log n)$	NP-complete
	otherwise	$O(m^3k^2)$	$O(m^3k^2\log n)$	NP-complete
With-communication	fixed k	$f(m2^{k\bar{d}}(\frac{em}{2k\bar{d}})^{2k\bar{d}}, k, d, k) \in P$		
	otherwise			

(c) Multi-agent, private budget, on the path.

Table 1.2: Summary of results for physical search problems with uncertain knowledge: n is the input size, m - the number of points (store locations), d - the number of different possible prices, $\bar{d} = d + 1$, k - the number of agents, n/a - the problem was not defined in that case or there is no need for a solution, f - the polynomial function defined in Lemma 8.15.



Chapter 1: Introduction

Chapter 2: Related Work

Part 1: Computational Aspects of Elections with Uncertainty

Chapter 3: Social Choice Terminology

Chapter 4: Evaluation of Election Outcomes under Uncertainty

Chapter 5: How to Rig Elections and Competitions

Chapter 6: Complexity of Safe Strategic Voting

Part 2: Physical Search Problems with Uncertain Knowledge

Chapter 7: Single Agent

Chapter 8: Multi-Agent

Chapter 9: Future Directions and Final Remarks

Appendix

A: Proofs for Chapter 4

B: Proofs for Chapter 5

C: Proofs for Chapter 8

Figure 1.1: Thesis Structure.

1.6 Publications

Results that appear in this dissertation have been published in the proceedings of the following refereed conferences and workshops:

- Noam Hazon and Edith Elkind. Complexity of Safe Strategic Voting. The 3rd International Symposium on Algorithmic Game Theory (SAGT-10), 2010. Short version presented in The Third International Workshop on Computational Social Choice (COMSOC-10), 2010. Early version presented in The First Workshop on Cooperative Games in Multiagent Systems (CoopMAS-2010), 2010. [67–69].

- Noam Hazon, Yonatan Aumann and Sarit Kraus. Collaborative Multi Agent Physical Search with Probabilistic Knowledge. The Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09), 2009. [61]

- Noam Hazon, Paul E. Dunne, Sarit Kraus and Michael Wooldridge. How to Rig Elections and Competitions. The 2nd International Workshop on Computational Social Choice (COMSOC-08), 2008. Early version presented in The 9th Bar-Ilan Symposium on the Foundations of Artificial Intelligence (BISFAI-2007), 2007. [65,66]

- Yonatan Aumann, Noam Hazon, Sarit Kraus and David Sarne. Physical Search Problems Applying Economic Search Models. The Twenty-Third Conference on Artificial Intelligence (AAAI-08), 2008. [7]

- Noam Hazon. Social Interaction under Uncertainty in Multi Agent Systems. The Thirteenth Annual AAAI/SIGART Doctoral Consortium (In association with AAAI-08), 2008. [60]

- Noam Hazon, Yonatan Aumann, Sarit Kraus and Michael Wooldridge. Evaluation of Election Outcomes under Uncertainty. The Seventh International Conference on Autonomous Agents and Multiagent System (AAMAS-08), 2008. Also presented in DIMACS Workshop on the Boundary between Economic Theory and Computer Science, 2007. [62,63]

In addition, results that appear in this dissertation have been accepted (with revisions) to the following refereed journals:

- Thuc Vu, Noam Hazon, Alon Altman, Yoav Shoham, Sarit Kraus and Michael Wooldridge. On the Complexity of Schedule Control Problems for Knock-out Tournaments. Under submission to the Journal of Artificial Intelligence Research (JAIR), 2010. [105]
- Noam Hazon, Yonatan Aumann, Sarit Kraus and Michael Wooldridge. On the Evaluation of Election Outcomes under Uncertainty. Under submission to Artificial Intelligence Journal, 2010. [64]

Chapter 2

Related Work

2.1 Computational Aspects of Elections with Uncertainty

This part of our work lies in the general domain of *computational social choice theory*, a research area at the intersection of artificial intelligence (AI), theoretical computer science, and social choice theory, that has attracted much interest recently [41]. In general, social choice theory is concerned with the design and analysis of methods for collective decision making. In multi-agent settings we often have self-interested agents with different preferences and capabilities, which need to reach a decision. Consequently, researchers in AI have become increasingly interested in social choice theory, especially concerning its computational aspects. One of the main topics with which computational social choice deals, is *computational voting theory* (for a brief survey of other topics, see [31]). In our work we analyze computational aspects of three major problems within computational voting theory: winner determination, control and manipulation, with the assumption of uncertain knowledge. In Chapter 4 we deal with winner determination under uncertainty. In Chapter 5 we investigate the control problem of rigging an election agenda, and in Chapter 6 we introduce our work on the complexity of safe manipulation. Next we present each problem separately with a review of the related work that has been done and we show the uniqueness of our settings.

2.1.1 Winner Determination

When considering voting procedures from a computational standpoint, the most natural question, from the computer science perspective, is whether it is possible to efficiently compute the winning outcome according to a particular voting rule given the preferences of all the agents. Bartholdi et al. [16] were the first to show that, surprisingly, there is a reasonable rule where determining the winner of the election is a hard computational problem. Specifically, evaluating the prospective winner according to Dodgson's rule is NP-hard. Later, Hemaspaandra et al. [72] proved that it is complete for the complexity class Θ_2^p. The work of Rothe et al. [92] subsequently showed that winner determination under Young's voting rule [120] is also complete for Θ_2^p. Computing the Banks winner has also been shown to be NP-complete [74, 113]. Despite these notable exceptions, most common voting rules are easy to evaluate. However, this holds only when perfect information about voter preferences is assumed, which is often a problematic assumption in real world settings. Our work, as described in Chapter 4, investigates voting rules under an uncertain information model.

The limiting assumption of perfect information has also motivated other researchers to seek a different, more realistic model. Konczak and Lang [75] and later Pini et al. [87], investigated the case whereby for each voter we have a correct but incomplete model of their preference relation. For this incomplete information setting, the researchers considered questions such as whether there was some completion of the incompletely known preferences for the candidates that would make a desired candidate a winner. This question, known as the possible-winner problem, is indeed important since it is also strongly connected to preference elicitation [36] and manipulation [75]. This incomplete information model was further investigated under sequential majority voting [79,88], under almost all scoring rules [19,20], and under other common voting rules [115]. The complexity of the possible-winner problem has also been studied for bounded parameters such as the number of candidates, the number of voters, and the total number of undetermined candidate pairs [21]. However, in contrast to our model, the incomplete information model cannot utilize any prior knowledge of the voters' preferences. It is "pessimistic" in nature, as it assumes that the missing data is completely unknown, and thus it ignores any probabilistic estimation of the voters preferences that could be learned from their voting history. Still, if very little is known on the voters' preferences, this model is more appropriate than ours. For example, if we are certain about the ranking of 2 candidates, but believe that all ordering of intermediate candidates are equally likely. Moreover, one of our problems (Chance-Evaluation) ignores the exact values of the probabilities, so it is very close to the settings of the possible-winner problem. We will show the connection between these problems in Section 4.3.2, where we present many hardness results, which we attained, as corollaries of [115](extended version). In a recent paper [10], Bachrach et al. studied the computational complexity of the counting version of the possible-winner problem. They proved #P-hardness results for Plurality and Veto, and provide a randomized approximation algorithm for all voting rules that are polynomial-time computable. Their algorithm may be used to generate the probabilistic input for our problem, where there is a lack of such knowledge on the voters' preferences.

Previous work that is most closely related to ours was conducted by Conitzer and Sandholm [35], who used a probability distribution over the votes as we do. Their results were derived with a restricted model of probability distributions. Their key result shows that if a manipulation for some voting rule is hard when complete information is provided, then it will be hard to even evaluate a candidate's winning probability with this protocol when there is uncertainty about the votes. However, as we will show in Section 4.2, this result holds only for weights that are not bounded by $Poly(n)$, where n is the number of voters. They further analyzed the un-weighted voters case, but only with a probability distribution that allows for perfect correlations among the voters (which actually simulates weights to the voters). This is also the case presented by Walsh [106], who proved some results regarding the connection between incomplete preference settings (i.e. the possible-winner problem) and the settings with a probability distribution over the votes, but with perfectly correlated votes. Probabilities over voters' preferences were also used by Hazon et al. [66] and Vu et al. [104]. However, in both of their works it was assumed that the only known information about an electorate is the probability that any given candidate will beat another. This information was then used to investigate the extent to which it is possible to rig the agenda of an election or competition so as to favor a particular candidate. We investigate this model in Chapter 5.

2.1.2 Control

Most work on the manipulation of voting procedures has considered the manipulation of elections by *voters*, which we will discuss in the next subsection. However, manipulation is also possible by election officers – those responsible for organizing an election, which in this context is sometimes called "control". In their paper on election control Bartholdi et al. [17] introduced the problem of election organizers trying to influence the outcome of the elections by changing the set of voters or candidates. For example, the election officers may try to remove some strong candidates so their preferred candidate will have a higher probability of winning. The authors showed that different voting rules differ significantly in terms of their resistance to control. The work of Hemaspaandra et al. [73] tried to find a voting rule which would be fully resistant to control. The authors showed that an artificially built rule is resistant to twenty different types of control. As for natural voting rules, Faliszewski and colleagues [45, 46] showed that some common voting rules are very close to the ideal of total resistance to control.

In our work we analyze a different type of control, which is unique to voting tree rules. In these rules the election organizers are responsible for defining the *order* of competition between the candidates. As a result they may rig it in order to favor a particular candidate. It is well-known that these voting trees may be rigged in this manner – see, e.g., [28, p.177] and [94]. Specifically, in the linear order case, if the election officer knows the preferences of the electorate – or more specifically, who will win in every possible ballot – then he can compute in polynomial time how to fix the election agenda to the benefit of a favored candidate [79]. However, in our work we analyze the rigging agenda problem under the assumption of uncertain information, which is a more realistic assumption

A closely related stream of work is the problem of optimal seeding for tournaments. This problem considers how to determine an agenda for a voting tree that will result in an "interesting" sporting competition. Schwenk [97] for example, assumed an imperfect information ballot matrix as we do (in Section 5.1), and used it to produce an interesting competition agenda that still satisfies some fairness criterions. Groh et al. [59] investigated a 4-player scenario with an auction-like analysis; instead of knowing the probability of winning, each player exerts some effort according to its private valuation. Earlier work by Searls [98] analyzed voting trees for 8 players, among 3 other tournament types. He used an imperfect information matrix to investigate the effect of the initial agenda on the probability that the best player will win the game. Note that none of these papers analyzed the complexity of finding the optimal agenda for a specific candidate, as in our work. We also focus on asymptotic complexity results, so that our results are not limited to very small numbers of candidates, m, such as $m = 4$ or $m = 8$.

A different model of partial information was studied by Lang et al. in [79], where they assumed a correct but incomplete model of preference relations for each voter. With this incomplete information setting, they considered questions such as whether there was some completion of the incompletely known preferences and some voting tree for the candidates that would make a desired candidate a winner. Roughly, our aim in this work is to study election control in much the same way as Lang et al. but with the probabilistic model described above, instead of the incomplete profile model.

2.1.3 Manipulation

Much of the interest in computational social choice stems from the possibility that computational complexity may provide a “solution” to some impossibility results in voting theory [28]. Specifically, a very well-known result in voting theory is the result of Gibbard [57] and Satterthwaite [96], 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 Bartholdi et al. [15] 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 Second-order Copeland was computationally hard to manipulate. This work subsequently led to many other researchers studying circumstances under which voting protocols are computationally easy or hard to manipulate, as we shall show shortly.

For a single voter, the complexity of the manipulation problem is quite well understood. Specifically, this problem is known to be efficiently solvable for most common voting rules with the notable exception of Second-order Copeland and Single Transferable Vote (STV) [14, 15].

The more recent work has focused, for the most part, on coalitional manipulation, i.e., manipulation by multiple, possibly weighted voters. In contrast to the single-voter case, coalitional manipulation tends to be hard. Indeed, it has been shown to be NP-hard for weighted voters even when the number of candidates is bounded by a small constant [35, 37]. Some of these results were later generalized by E. Hemaspaandra and L.A. Hemaspaandra [71], who characterized the scoring functions in which manipulation is NP-hard. Elkind and Lipmaa, in their work [39], discussed cryptographic techniques to make coalitional manipulation hard, and in another paper [38] they showed general approaches to designing hard-to-manipulate voting procedures, based on the idea of combining protocols.

For unweighted voters, nailing the complexity of coalitional manipulation proved to be more challenging. However, Faliszewski and colleagues [47] have established that this problem is hard for most variants of Copeland, and Zuckerman et al. [121] have shown that it is easy for Veto and Plurality with Runoff. Furthermore, in a recent paper Zuckerman and colleagues [118] make substantial progress in this direction, showing, for example, that unweighted coalitional manipulation is hard for Maximin and Ranked Pairs, but easy for Bucklin.

All of these papers (as well as the classic work of Barholdi et al. [15]) assume that the set of manipulators is given exogenously, and the manipulators are not endowed with preferences over the entire set of candidates; rather, they simply would like a particular candidate to get elected, and they select their votes based on the non-manipulators' preferences that are publicly known. That is, this model abstracts away the question of how the manipulating coalition forms. However, to develop a better understanding of coalitional manipulation, it would be desirable to have a plausible model of the coalition formation process. In such a model the manipulators would begin by having the same type of preferences as sincere voters, and then some agents— those who are not satisfied with the current outcome and are willing to submit an insincere ballot —would get together and decide to coordinate their efforts. Slinko and White [101] provide such a model, but they does not explore its complexity properties. They also assume that all manipulators have identical preferences. In our work, we investigate the algorithmic complexity of their safe manipulation model, and propose two ways of extending this notion of manipulation to heterogeneous group of manipulators.

2.2 Physical Search Problems with Uncertain Knowledge

Models of a single agent search process with prior probabilistic knowledge have attracted the attention of many researchers in various areas, mainly in economics and operational research, prompting several reviews over the years [81, 84]. These search models have developed to a point where their total contribution is referred to as *search theory*. Probably the most famous problem within this field is the "secretary problem", which has a remarkably long list of articles that have been dedicated to its variations (see [48] for an extensive bibliography). Nevertheless, these economic-based search models, as well as their extensions over the years into multi-agent environments [33, 95], assume that the cost associated with observing a given opportunity is stationary (i.e., does not change along the search process). While this permissive assumption facilitates the analysis of search models, it is frequently impractical in the physical world. Therefore, in our work, we assume that the cost associated with observing a given opportunity may change along the search process. The use of changing search costs suggests an optimal search strategy structure different from the one used in traditional economic search models; other than merely deciding when to terminate its search, the agent also needs to integrate exploration sequence considerations into its decision making process.

Search with changing search costs has been previously considered in the computer science domain in the contexts of Prize-Collecting Traveling Salesman problems (PC-TSP) [11] and the Graph Searching Problem (GSP) [77]. In PC-TSP we are given a graph with non-negative "prize" values associated with each node, and a salesman needs to pick a subset of the nodes to visit in order to minimize the total distance traveled while maximizing the total prize collected. All the variants of PC-TSP are NP-hard, as they are generalizations of the famous Traveling Salesman Problem (TSP). One variant of PC-TSP is the k-TSP, where every node has a prize of one and the goal is to minimize the total distance traveled, while visiting at least k nodes. This variant is similar to our *Min-Budget* problem, where we try to minimize the budget necessary to obtain at least a given success probability. There are several constant-factor approximations known for the k-TSP [4, 9, 27, 55, 56]. Another variant of PC-TSP is the Orienteering problem, where the goal is to maximize the total prize collected, while keeping the distance traveled below a certain threshold. This variant is similar to our *Max-Probability* problem, where we try to maximize the success probability while keeping the total traveling cost (plus the final purchase cost) below the initial budget. For points in the plane, Arkin et al. [2] gave a constant-factor approximation. For points in arbitrary metric spaces, Blum et al. [26] gave the first approximation algorithm, which was improved by Bansal et al. [13], and later by Chekuri et al. [30]. Nevertheless, the family of PC-TSP differs from our investigated model in in reference to two main aspects. First, the model of the PC-TSP does not contain probabilities, only costs on the edges and prizes on the nodes and thus constraints are additive. Second, in the PC-TSP there is a single prize at each node and whenever the salesman visits that node he collects the prize. In our setting, there may be several probabilities to "collect" at each node, and the actual amount collected depends on the remaining budget when reaching the node. There may be cases where visiting a node does not increase the success probability at all, even though there is some success probability at the node (for instance if the agent does not have enough budget when it reaches the node).

In the GSP case, the agent seeks a single item that resides at some node of a fixed graph, and a distribution is defined over all probabilities of finding the item at each of the graph's nodes. The goal is to minimize the expected cost, as in our *Min-Expected-Cost* problem. The GSP was shown to be strictly related to a classic well-studied problem, the minimum latency problem (MLP) [93], also called the traveling repairman problem [1], the school-bus driver problem [111], and the delivery man problem [49, 85]. In this problem an agent is supposed to visit the nodes of a graph in a way that minimizes the sum of the latencies to the nodes, where the latency of a node is its distance along the agent's tour. The minimum latency problem was shown to be NP-complete even when the metric space is induced by a tree [100], but can be solved in linear time when the underlying graph is a path [1, 53]. In the operations research community, there are several exact exponential time algorithms for the MLP, e.g. [22,49,83,99,119]. Researchers have also evaluated various heuristic approaches [108, 110]. In the computer science community, there is a large branch of research dealing with approximate solutions to the MLP. For general metrics, Blum et.al. [25] gave the first constant factor approximation. This was improved by Goemans and Kleinberg [58], and later by Chaudhuri et al. [29]. Koutsoupias et al. [77] provided a constant factor approximation for the unweighted case (i.e. for a shortest path metric on an unweighted graph), and Arora and Karakostas [3] gave a quasi-polynomial $O(n^{O(logn)})$ time approximation scheme for weighted trees and points in $\mathbb{R}^d$. The MLP was also generalized to multi-agent settings (with k repairmen) by Fakcharoenphol et al. [42, 43].

More important, Koutsoupias et al. [77] and later Ausiello et al. [8] showed how to extend results obtained for the MLP to the GSP. For example, in some cases, approximation developed for the MLP can be applied to the GSP. It is not clear, however, if results obtained for the GSP can be extended to work in our setting since in the GSP the success factor is binary: upon arriving at a node either the item is there or not. Extensions of the GSP to scenarios where the item is mobile are of the same character [52, 76]. In our work, however, similar to other works in economic search theory, we assume to have a probability distribution in each node.

Our work thus tries to bridge the gap between classical economic search theory (which is mainly suitable for virtual or non-dimensional worlds) and the changing search cost constraint imposed by operating in physical multi-agent environments.

Part I

Computational Aspects of Elections with Uncertainty

Chapter 3

Social Choice Terminology

An *election* is given by a set of *candidates* (also referred to as *alternatives*) $C = \{c_1, \dots, c_m\}$ and a set of *voters* $V = \{1, \dots, n\}$. Each voter i is represented by his *preference order* R_i, which is a total order over C; we will sometimes refer to total orders over C as *votes*. The vector $\mathcal{R} = (R_1, \dots, R_n)$ is called a *preference profile*. We say that two voters i and j are of the same *type* if $R_i = R_j$; we write $V_i = \{j \mid R_j = R_i\}$.

A *voting rule* $\mathcal{F}$ is a mapping from the set of all preference profiles to the set of candidates; if $\mathcal{F}(\mathcal{R}) = c$, we say that c *wins* under $\mathcal{F}$ in $\mathcal{R}$. A voting rule is said to be *anonymous* if $\mathcal{F}(\mathcal{R}) = \mathcal{F}(\mathcal{R}')$, where $\mathcal{R}'$ is a preference profile obtained by permuting the entries of $\mathcal{R}$. In this work we consider anonymous voting rules only. In addition, we restrict ourselves to voting rules that are polynomial-time computable.

During the election, each voter i submits a preference order L_i; the outcome of the election is then given by $\mathcal{F}(L_1, \dots, L_n)$. We say that a voter i is *truthful* if $L_i = R_i$. For any $U \subseteq V$ and a vote L, we use $\mathcal{R}_{-U}(L)$ to denote the profile obtained from $\mathcal{R}$ by replacing R_i with L for all $i \in U$.

Voting rules We will now define the main voting rules considered in this work. Our first family of voting rules consists of rules that assign scores to all candidates; the winner is then selected among the candidates with the highest score using a *tie-breaking rule*, i.e., a mapping $T : 2^C \to C$ that satisfies $T(S) \in S$. We consider two tie-breaking rules; *random*, where the winner is randomly selected among all the tied candidates, or, alternatively, *lexicographic*, where given a set of tied candidates the winner is the candidate which is maximal with respect to a fixed ordering $\succ$. (Our results can be easily extended to hold for other tie-breaking rules.)

Given a vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ with $\alpha_1 \geq \cdots \geq \alpha_m$, the *score* $s_\alpha(c)$ of a candidate $c \in C$ under a *positional scoring rule* F_α is given by $\sum_{i \in V} \alpha_{j(i,c)}$, where $j(i,c)$ is the position in which voter i ranks candidate c. Many classic voting rules can be represented using this framework. Indeed, *Plurality* is the scoring rule with $\alpha = (1, 0, \ldots, 0)$, *Veto* (also known as *Antiplurality*) is the scoring rule with $\alpha = (1, \ldots, 1, 0)$, *Borda* is the scoring rule with $\alpha = (m-1, m-2, \ldots, 1, 0)$, and k*-approval* is the scoring rule with α given by $\alpha_1 = \cdots = \alpha_k = 1$, $\alpha_{k+1} = \cdots = \alpha_m = 0$; we will sometimes refer to $(m-k)$-approval as k*-veto*.

Bucklin rule can be viewed as an adaptive version of k-approval. We say that k, $1 \leq k \leq m$, is the *Bucklin winning round* if for any $j < k$ no candidate is ranked in top j positions by at least $\lceil n/2 \rceil$ voters, and there exists some candidate that is ranked in top k positions by at least $\lceil n/2 \rceil$ voters. We say that the candidate c's *score in round* j is his j-approval score, and his *Bucklin score* $s_B(c)$ is his k-approval score, where k is the Bucklin winning round. The *Bucklin winner* is the candidate with the highest Bucklin score. Observe that the Bucklin score of the Bucklin winner is at least $\lceil n/2 \rceil$.

The *Copeland* rule is defined based on the notion of *pairwise elections*. We say that a candidate $c \in C$ beats another candidate $c' \in C$ in a pairwise election if the majority of voters rank c above c'. We will also refer to pairwise elections as *ballots*. The *Copeland score* $s_C(c)$ of a candidate c is given by the number of pairwise elections that c wins minus the number of pairwise elections that c loses.

We also consider another family of voting rules, known as *voting trees*, which are based on sequential pairwise elections along a binary tree. With these rules we often summarize voter preferences in a *majority graph*, $G \subseteq C \times C$, where $(c, c') \in G$ means that c would beat c' in a pairwise election. The majority graph is asymmetric and irreflexive, and since we assume that a tie-breaking rule is used, G is also complete. Thus, G is a *tournament* on C [80]. A voting tree on C is defined by:

- An m leaf binary tree, T, having a distinguished root $r(T)$.
- An *agenda* α, which is a one-to-one mapping between the leaf nodes of T and the candidates from C.

Given $\langle T, \alpha, G \rangle$, the *labeling* of the tree with respect to α and G is a function

$\ell : V \to C$, defined in the following recursive way:

$$\ell(v) = \begin{cases} \alpha(v) & \text{if } v \text{ is a leaf} \\ \text{w} & \begin{array}{l}\text{if } v_l \text{ and } v_r \text{ are the children of } v, \\ \text{and } w \text{ is the winner between } \ell(v_l) \text{ and } \ell(v_r) \text{ accoring to } G\end{array} \end{cases}$$

The winner of a voting tree rule is the candidate labeled at the root of the tree, i.e. $\ell(r(T))$. In our work we will mainly consider two common binary tree structures: the caterpillar structure, i.e. a completely unbalanced tree, and the balanced tree, which we term *linear order* and *fair tree order*, respectively.

Weighted voters Our model can be extended to the situation where not all voters are equally important by assigning an integer *weight* w_i to each voter i. To compute the winner of a profile $(R_1, \ldots, R_n)$ under a voting rule $\mathcal{F}$ given voters' weights $\mathbf{w} = (w_1, \ldots, w_n)$, we apply $\mathcal{F}$ on a modified profile which contains w_i copies of R_i for each $i = 1, \ldots, n$. When all the weights are equal, we say that the voters are *unweighted.* For each $U \subseteq V$, let $|U|$ be the number of voters in U and let $w(U)$ be the total weight of the voters in U. As an input we usually get a *voting domain*, i.e., a tuple $S = \langle C, V, \mathbf{w}, \mathcal{R} \rangle$, together with a specific voting rule. In the case of imperfect information about voter preferences, the voting domain will only contain the tuple $S' = \langle C, V, \mathbf{w} \rangle$.

Chapter 4

Evaluation of Election Outcomes under Uncertainty

In this chapter we analyze the winner determination problem, with the presence of uncertainty. We first formally define our probabilistic knowledge model, and our two main problems, Evaluation (Definition 4.1) and Chance-Evaluation (Definition 4.2). In Section 4.2, we first give a polynomial algorithm to answer the evaluation problem if the number of candidates is a constant. While a result in [35] establishes that Evaluation is NP-hard for several key voting rules, even under quite stringent assumptions about probability distributions, we show that this result holds only for weighted voting rules with weights that are not bounded by $Poly(n)$. We then evaluate the algorithm in practice, showing that the actual running time and space are smaller than the asymptomatic bound. Therefore, we also test how many voters the polynomial-time algorithm can handle for a given set of candidates. The results indicate that even with 6 or 7 candidates, the algorithm can handle more than 100 voters, , which suggests that it may be used in many real-world voting scenarios. If the number of candidates is not bounded, the evaluation problem becomes much harder we show in Section 4.3. Namely, even for the well-known Plurality, k-approval Borda, Copeland, and Bucklin voting rules the problem is #P-hard. We then analyze a simpler question, the Chance-Evaluation problem. Surprisingly, this problem is shown to be NP-complete (in the strong sense) even for the Plurality voting rule, when voters do not have equal weights. We give a polynomial time algorithm when all voters have equal weights, for Plurality, and show that the Chance-Evaluation problem is hard for many other voting rules (including k-approval, Borda, Copeland and Bucklin). This is done by establishing the connection to a related problem, the possible-winner problem [75]. Finally, we present a Monte Carlo algorithm that is able to approximately answer even the EVALUATION problem where the number of candidates is a parameter, with an error as small as desired.

4.1 Model and Problem Definitions

In many settings voter $i \in V$ will not usually know the preferences of the other individual voters – but he may know the *probability* that a voter will vote for a specific candidate, or the probability that he will prefer one candidate over another. If all probabilities are 0 or 1 then the scenario is one of *perfect information*, otherwise it is one of *imperfect information.* To model imperfect information, we assume that we have for each voter at most l possible preference orders, which are permutations over the available alternatives. Each such order is associated with a non-zero probability that this voter will choose to vote for it, and the sum of probabilities of the given preference orders is one; all the other possible preference orders, which are not explicitly given, are assumed to have a probability of zero. Consider the following illustrative example. Suppose we have 4 candidates, c_1, c_2, c_3 and c_4, and 3 voters, V_1, V_2 and V_3. The voters' preferences are summarized in Table 4.1 with a probability associated with each preference order. In this example $n = l = 3$ and $m = 4$.

V_1	V_2	V_3
$\frac{1}{3}$ (c_1, c_2, c_3, c_4)	$\frac{3}{4}$ (c_4, c_2, c_1, c_3)	$\frac{9}{10}$ (c_4, c_2, c_3, c_1)
$\frac{1}{2}$ (c_2, c_1, c_3, c_4)	$\frac{1}{4}$ (c_2, c_1, c_3, c_4)	$\frac{1}{10}$ (c_3, c_1, c_4, c_2)
$\frac{1}{6}$ (c_3, c_1, c_2, c_4)		

Table 4.1: An example of our imperfect information model of voter preferences.

We consider the case where voters' choices are independent. If we collect from each voter just one preference order (from the ones that are associated with him) we get one possible preference profile that we call a *voting scenario*, from which the winner can be calculated using one of the voting rules listed above (Plurality, Borda, ...). The probability of any given voting scenario occurring is simply the multiplication of the probabilities of its preference orders from the different voters. In these settings, we assume that the voters are *truthful*, i.e. for each voter i, $L_i = R_i$, thus one of the voting scenario must occur. We are now ready to define our main problems.

Definition 4.1 (EVALUATION). *Given a voting domain, an imperfect information model of voters' preferences, as described above, a specific candidate c^*, and a voting rule $\mathcal{F}$, what is the probability that c^* will be chosen using $\mathcal{F}$?*

The answer to this question is the sum of probabilities of all the voting scenarios where c^* wins using $\mathcal{F}$. For example, consider the imperfect information shown in Table 4.1. Assume that random tie-breaking rule is used. The winning probabilities for each candidate under the Plurality, Borda and Copeland voting rules are summarized in Table 4.2. Note that c_4 has the highest probability of winning under Plurality and Copeland, but c_2 has the highest probability of winning under Borda.

	Plurality	Borda	Copeland
c_1	0.036	0.058	0.052
c_2	0.178	**0.7**	0.256
c_3	0.053	0.017	0.017
c_4	**0.733**	0.225	**0.675**

Table 4.2: Winning probabilities for each candidate, rounded to 3 decimal places. Bold font represents the highest probability in each voting rule.

Note that the complexity of this problem is a function of the number of voters (n), the number of candidates (m), and the number of possible non-zero probability preference orders for each voter (l). We also define a related decision problem, which asks for a weaker question.

Definition 4.2 (CHANCE-EVALUATION). *Given a voting domain, an imperfect information model of voters' preferences, as described above, a specific candidate, c^*, and a voting rule $\mathcal{F}$, is the probability that c^* will be chosen using $\mathcal{F}$ greater than zero?*

Of course, an answer to the EVALUATION problem immediately yields an answer to the CHANCE-EVALUATION problem, so if the former problem is easy, then so is the latter. However, it could in principle be the case that EVALUATION is hard while CHANCE-EVALUATION is easy, which suggests that it is worth studying CHANCE-EVALUATION as a problem in its own right.

Note that the CHANCE-EVALUATION problem also seems to be a very natural one. In many cases there will be some candidates that do not have any chance of winning, and a voter might reasonably contemplate which candidates have no chance of winning when deciding how to vote.

In the following sections, we analyze the complexity of the EVALUATION and CHANCE-EVALUATION problems in two main different scenarios: when the number of candidates is bounded by a constant; and when it is not bounded.

4.2 Constant Number of Candidates

In many real-world scenarios, the number of alternatives is small and can be bounded by a constant. For example, if a group of agents want to decide on a full hour to meet in a given day, the number of alternatives is always 24. In this section we give a polynomial algorithm for the EVALUATION problem under the assumption of a constant number of alternatives[1]. Obviously, this algorithm also answers the CHANCE-EVALUATION problem in polynomial time. We then present some experimental results obtained with this algorithm, evaluating its performance in practice.

4.2.1 The Algorithm

The key to the efficiency of our algorithm is the distinction between a voting scenario and a *voting result*. Intuitively, in a voting scenario we know for each voter which preference order he votes for. But to identify a winning candidate, we do not care actually exactly which voter votes for which candidate; we can aggregate the possible voting scenarios into a compact intermediate form, which is what we call a voting result. For example, suppose we use the Plurality rule. With Plurality, a voting result may be a vector which stores the total number of votes for each candidate. Now suppose that there are three voters and two candidates, c_1 and c_2, and all the voters do not have a probability of 1 to vote for one of the candidates. Thus, there are three

[1]We thank Efrat Manisterski for her contribution in developing this algorithm.

voting scenarios with the same voting result of two votes for c_1 and one vote for c_2. A little more formally:

Definition 4.3. *Given a voting rule, a* voting result *is a succinct way to represent one or more voting scenarios over i voters, $0 \leq i \leq m$, such that:*

1. *For $i = m$, the winner can be determined from the voting result over the m voters in polynomial time.*
2. *A voting result over $i + 1$ voters can be generated from combining the voting result for i voters and one additional preference order, in polynomial time.*

After we present the algorithm, we describe different ways to represent voting results for many common voting rules. Let us first describe the algorithm where all the voters' weights are equal. A formal proof of correctness can be found in the Appendix. We use a dynamic programming approach to enumerate the possible voting results from the preferences of n voters and calculate their probability. This is done by using possible voting results from the preferences of $n - 1$ voters and their probabilities, which is in turn done by using the voting results from the preferences of $n - 2$ voters, and so on. Our algorithm builds a Table where the rows are possible voting results and the columns represent the voters. We denote by $T[\vec{i}, j]$ the cell in the table at the row which represents the voting result vector $\vec{i}$, and at column j. In any stage, the algorithm only requires storing 2 columns in memory.

Algorithm 1 VotingResult(table T, preference orders for each voter)

```
1: Init $T[.,.] \leftarrow 0$,   $T[\vec{0}, 0] \leftarrow 1$.
2: for $i \leftarrow 0$ to $n - 1$ do
3:     for all cells in column $i$ do
4:         $\vec{r} \leftarrow$ the voting results of the cell's row
5:         for $j \leftarrow 1$ to $l$ do
6:             $\vec{cur} \leftarrow$ preference order $j$ of voter $i + 1$
7:             $\vec{next} \leftarrow$ the voting result from adding $\vec{cur}$ to $\vec{r}$
8:             $T[\vec{next}, i + 1] \leftarrow T[\vec{next}, i + 1] +$ ( probability of $\vec{cur} \times T[\vec{r}, i]$)
```

When the algorithm terminates, each cell in the last column contains the probability of that cell's row voting result occurring. We can identify the winner for each voting result according to the specific voting rule. So, we can answer the EVALUATION problem from Definition 4.1 by simply summing for c^* the probabilities of the voting results where it wins. Consider the following small example. Suppose we use the plurality voting rule with 3 candidates, c_1, c_2 and c_3 and 2 voters, V_1 and V_2. The voters' preferences are summarized in table 4.3a. Table 4.3b shows the table, T, that is built by the algorithm for this data. Every row represents a voting result which is a vector such that index i counts the number of votes for candidate c_i. The last column shows the probabilities for every possible voting result with voters V_1 and V_2. Thus, the probability that c_1 is the winner, assuming a random tie-breaking method is used, is $\frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} \cdot (\frac{1}{3} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{3}{4}) + \frac{1}{2} \cdot (\frac{1}{6} \cdot \frac{1}{4})$. Note that in this example, $(2, 0, 0)$ is not a possible voting result.

V_1	V_2
$\frac{1}{2}\ c_1$	$\frac{1}{4}\ c_1$
$\frac{1}{3}\ c_2$	$\frac{3}{4}\ c_2$
$\frac{1}{6}\ c_3$	

(a) A set of voters' preferences.

Voting result (c_1, c_2, c_3)	0	1	2
$0, 0, 0$	1	0	0
$1, 0, 0$	0	$\frac{1}{2}$	0
$0, 1, 0$	0	$\frac{1}{3}$	0
$0, 0, 1$	0	$\frac{1}{6}$	0
$2, 0, 0$	0	0	$\frac{1}{2} \cdot \frac{1}{4}$
$1, 1, 0$	0	0	$\frac{1}{3} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{3}{4}$
$1, 0, 1$	0	0	$\frac{1}{6} \cdot \frac{1}{4}$
$0, 2, 0$	0	0	$\frac{1}{3} \cdot \frac{3}{4}$
$0, 1, 1$	0	0	$\frac{1}{6} \cdot \frac{3}{4}$

(b) The corresponding table T, that is built by the algorithm.

Table 4.3: An example of how algorithm 1 builds a table from a given set of preferences.

The time complexity of the algorithm is $O(n\times$ number of rows of $T \times l)$, and the space complexity is $O($ number of rows of $T)$. The specific voting rule determines how to express the possible voting results which in turn determines the number of rows. Clearly, we seek a representation which is as compact as possible to reduce time and space complexity. This issue, of how to summarize votes in a compact way, was also studied by by Chevealeyre et al. [32] and Xia and Conitzer [116], and we reuse some of their techniques. It also seems that their results, showing that their upper bound is tight, can be imported to our settings to show that our ways for the representation of voting results are optimal. Now, for many voting rules one of the following methods can be used to express the possible voting results:

1. A vector of $[0, n]^m$ such that index i represents the number of voters who voted for candidate i.
2. A vector of $[0, n]^{m(m-1)/2}$ which represents the number of voters who preferred the first candidate in each possible pair of candidates.
3. A vector of $[0, n]^{m^2}$ which represents the number of voters who ranked each candidate in each position (taken from [116]).
4. A vector of $[0, n]^{m \cdot 2^{m-1}}$. For each candidate i, let z_{-i} be a possible subset of candidates not containing i. The voting result vector represents the number of voters who preferred each candidate i over any candidate in z_{-i}, for each possible z_{-i} (taken from [32]).
5. A vector of $[0, n]^{m!}$ which represents the number of voters who voted for each possible preference order permutation.

We now show which method to use for some voting rules.

- *Plurality.* The first method can be used so the number of rows is $O((n + 1)^m)$ and the time complexity is $O((n + 1)^m \cdot l)$. However, the actual number of voting results will never be $(n + 1)^m$, since the total number of points given by **all** the candidates is n. Instead, the actual number of voting results with n voters is exactly the number of options to split the integer number n to exactly m non-negative integers, such that their sum is equal to n. Two sums which differ in the order of their summands are considered to be different compositions. This is called a *weak composition of* n *with exactly* m *parts*; we denote this value by $WC(n, m)$, $WC(n, m) = \binom{n+m-1}{m-1} = \frac{(n+m-1)!}{n!(m-1)!}$. Accordingly the running time complexity is $O(l \sum_{i=0}^{n} WC(i, m))$ and the space required is $O(WC(n - 1, m) + WC(n, m))$.
- *k-approval.* The first method can be used, with a running time complexity of $O(l \sum_{i=0}^{n} WC(i \cdot k, m))$ and a space complexity of $O(WC((n - 1) \cdot k, m) + WC(n \cdot k, m))$.

- *Borda.* We can use a modified version of the first method – a vector of $[0,(m-1)n]^m$, which represents the total score for each candidate. Thus, the time and space complexity are $O(l\sum_{i=0}^{n} WC(i\cdot\frac{m(m-1)}{2},m))$ and $O(WC((n-1)\cdot\frac{m(m-1)}{2},m)+WC(n\cdot\frac{m(m-1)}{2},m))$, respectively.

- *Bucklin.* The third method can be used so the time and space complexity are $O(l\sum_{i=0}^{n} WC(i\cdot m,m))$ and $O(WC((n-1)\cdot m,m)+WC(n\cdot m,m))$, respectively.

- *Copeland.* The second method can be used, so the number of rows is $O((n+1)^{m(m-1)/2})$ and the time complexity is $O((n+1)^{m(m-1)/2}l)$. This method can be used for any other Condorcet-consistent rule, i.e., Maximin, Ranked pairs, Voting trees, etc. (For an extensive discussion on voting rules, we refer the reader to [6]).

- *STV* (see definition in [6]). The fourth method can be used, so the number of rows is $O((n+1)^{m\cdot 2^{m-1}})$ and the time complexity is $O((n+1)^{m\cdot 2^{m-1}}l)$. If $m\leq 4$, the last method shall be used to calculate the number of scores for each candidate from the preference orders.

When we move to the weighted voters case, Connitzer and Sandholm [35] expressed the Evaluation problem as the following decision problem: given a number r, $0\leq r\leq 1$, is the probability of c^* winning greater than r? They showed that this problem is NP-hard for Borda, Copeland, Minimax and STV, even with extremely restricted probability distributions. We show that their results hold only for weights that are not bounded by *Poly*(n).

Claim 4.4. *For a constant number of candidates, the* Evaluation *problem is in P even for weighted voters, when the weights are in* $O(Poly(n))$

Proof. Our dynamic programming approach (algorithm 1) can be easily extended to work with weighted voters. Actually, the only thing that has to be changed is the range of possible voting results which determines the number of rows in the table. The number of rows will now become at most $O(Poly(n)^m)$, $O(Poly(n)^{m(m-1)/2})$, $O(Poly(n)^{m^2})$, $O(Poly(n)^{m\cdot 2^{m-1}})$ or $O(Poly(n)^{m!})$, depending on the specific voting rule (as described earlier). In all cases it is still in P. □

This result may be understood with reference to the proofs of Conitzer and Sandholm [35], who used a reduction from the Partition problem.

Partition is known to have a pseudo-polynomial time dynamic programming solution [54].

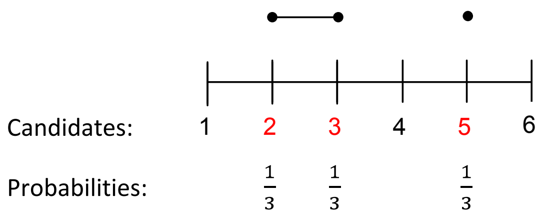
4.2.2 Experiments

In the preceding section, we gave analytical results for the case where the number of candidates is a constant. We showed that the complexity of our algorithm grows exponentially with the number of candidates. As with many other problems that have worst-case exponential running time, it is in interesting to ask whether we do indeed see worst case performance in practice. Our hypothesis was that in our problem, the actual number of voting results that the algorithm stores is much smaller than the asymptotic bound in most cases (and hence the required memory and time are smaller too). In particular, we investigated the effect of the probabilistic structure of the imperfect information on the number of stored voting results. Additionally, we tested the effect of the parameter l on the actual number of voting results. In this section we present experimental results obtained with an implementation of the algorithm for the Plurality rule with unweighted voters, which validate our hypothesis. Accordingly, we also found it interesting to check how many voters the polynomial-time algorithm can reasonably handle in practice, for a given number of candidates.

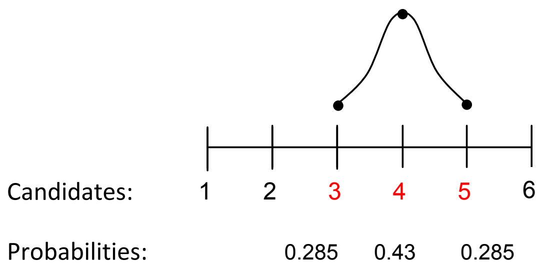
Our implementation (written in C++) ran on a 64-bit Linux PC, with 8 GB of RAM. The large amount of main memory was needed for the algorithm to store the table of the voting results. This table was implemented using Judy array [18], a complex but very fast associative array data structure for storing and looking up values. We chose to use this data structure since it typically requires much less memory than a conventional hash table. We measured the algorithm's performance by counting the total number of cells that were produced during run-time, to avoid the effect of the computer's hardware on the results (in contrast to time, which depends on the actual testing hardware). In most cases we ran 15 iterations and took the average; in the extreme cases, where the running time was too long, we took the average of only 5 iterations.

As an input, the algorithm takes an imperfect information matrix. Unlike in other experimental work in social choice which generates random *preferences* or random voters' *weights* (see [107] for example), we need to randomly generate *probabilities* over possible preferences. As noted before, in this work we assume that some knowledge on the preferences can be derived, and only

l preference orders have a non-zero probability for each voter. Therefore, the impartial culture assumption [24], which is a model of an electorate in which all preference orders are equally likely, can not be used. Alternatively, we considered two methods for selecting l candidates for each voter (Plurality needs the top choice candidate only) and generating the probabilities, using uniform and normal distributions. In the first method, l candidates were randomly chosen for each voter and the probability that she will vote for each one of them was set to $1/l$. The second method defined an arbitrary fixed order over the candidates. It then randomly chose one candidate to be the mean of the normal distribution, for each voter. The other $l-1$ candidates were chosen by their proximity to the mean candidate. The probability that each voter will vote for each one of the candidates was set according to the normal distribution, with the selected mean and a variance of 1. Figure 4.1 demonstrates the difference between these 2 methods.



(a) uniform distribution.



(b) normal distribution.

Figure 4.1: An example of how to generate random probabilities where $m = 6$, $l = 3$.

In the first set of experiments, we tested the effect of the random methods that we used to generate the voters preferences. In these experiments we fixed l to be 3, and we evaluated the effect of the two methods on the running time (in terms of the number of generated cells) for 5 and 6 candidates and $20-100$ voters. The results are shown in Figure 4.2.

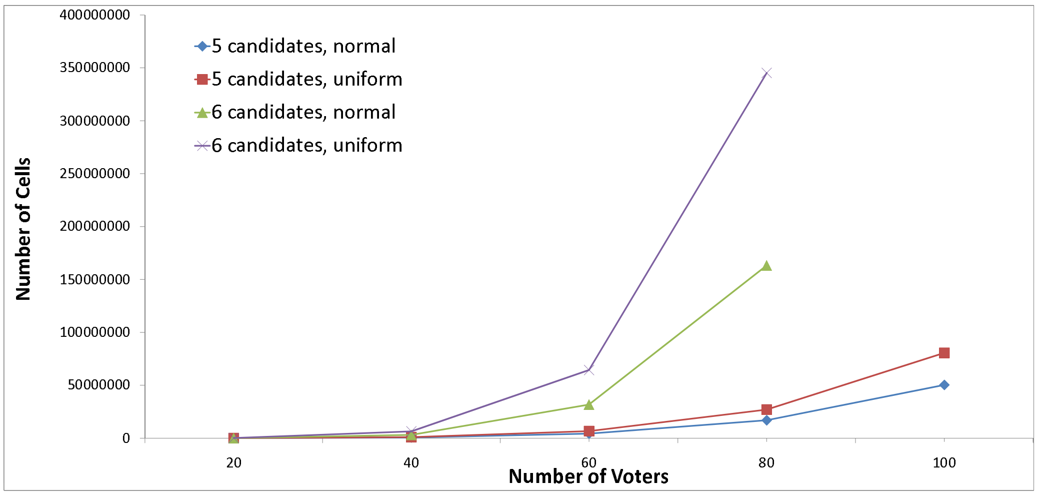


Figure 4.2: Results of first set of experiments.

Clearly, using the uniform distribution to generate the preferences results in more options to split the total number of votes among the candidates. Thus, increasing the number of voting results yields a higher running time. The second method simulates a more realistic scenario, the "single-peaked preference" principle [23]. In this case, there is some predetermined linear ordering of the candidate set. Every voter has some special place he likes best along that line, and his dislike for a candidate grows larger as the candidate goes further away from that spot. Similarly, in our case every voter has some special place that we believe has the highest probability to be selected, and the probability that the voter will vote for a candidate decreases as the candidate goes further away from that spot. In this case the votes are less scattered among the candidates, and thus the number of voting results is lower, yielding a lower running time.

We also used these settings to demonstrate how the ratio between the actual number of voting results to the theoretical number behaves. Since we used Plurality, the theoretical bound was computed using the WC function (as described above). The results are summarized in Figure 4.3. As there are more voting scenarios which lead to the same voting result, the gap between the theoretical bound to the actual number of voting results increases. Thus, this ratio is lower when there are more candidates or when a normal distribution rather than a uniform distribution is used to generate the imperfect information. On the other hand, the number of voters does not have a significant effect on this ratio.

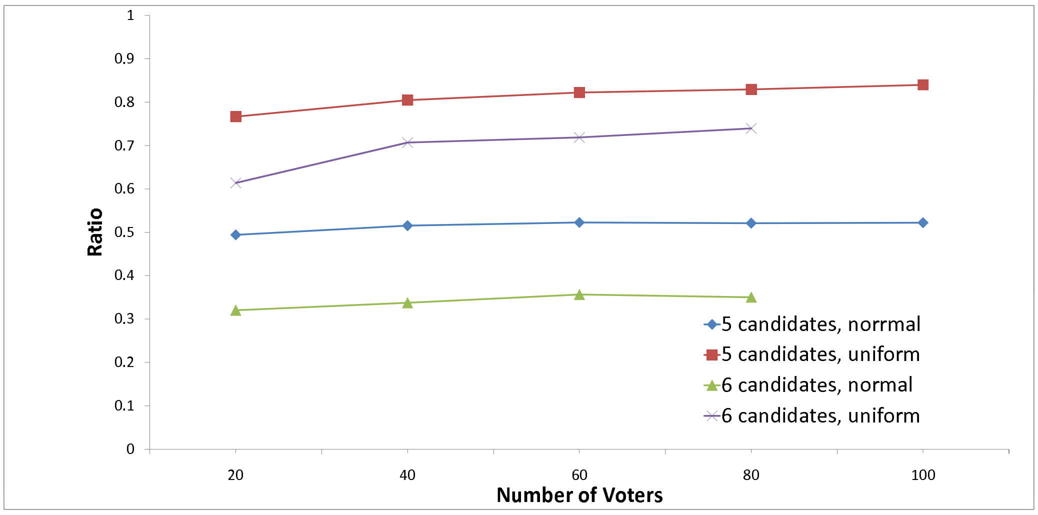


Figure 4.3: Ratio of actual number to theoretical number of voting results with increasing numbers of voters.

In the second set of experiments, we investigated the effect of l on the actual number of voting results. Although the algorithm's running time is (asymptomatically) linear in l, it was interesting to check if l has the same effect on the actual number of voting results. In these experiments, we fixed the number of candidates to 5, and used the normal distribution to generate preferences. We measured the ratio between the actual number of voting results to the theoretical bound (computed using the WC function) for 50 and 100 voters and l between $2-5$. The results are shown in Figure 4.4. Fortunately, as l increases the ratio of the actual number to the theoretical number of voting results increases in the same manner. As in the previous experiment, this ratio is not affected by the number of voters. Note that since $m = 5$, if $l = 5$ too, every possible voting result may happen, thus the ratio is 1.

The consequence of what we have shown so far is that there is a gap between the theoretically predicted running time and the actual one. Therefore, in the last set of experiments we tested how many voters the polynomial-time algorithm can handle in practice, for a given number of candidates. We set

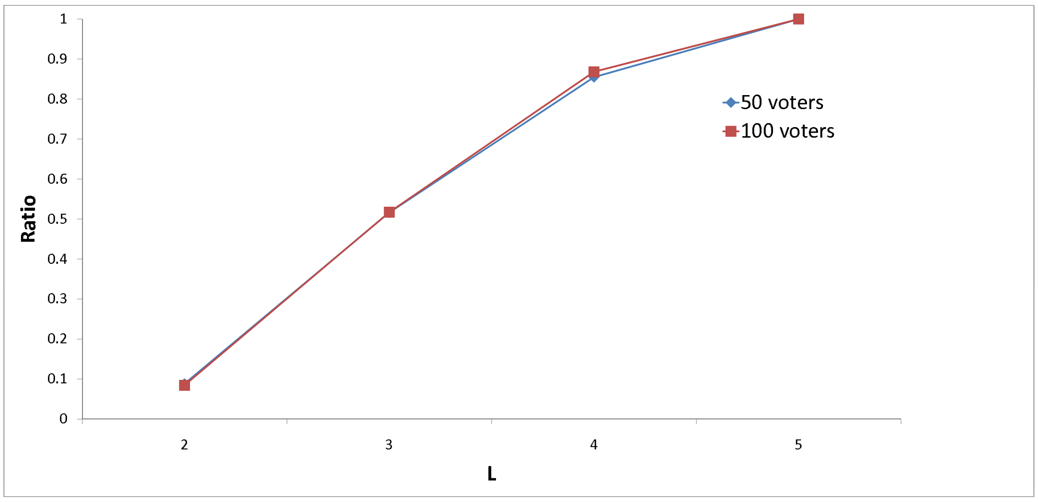


Figure 4.4: Ratio of actual number to theoretical number of voting results when increasing the number of non-zero probability preference orders for each voter (l).

l at its minimum value, 2, and we used the normal distribution to generate the preferences. We then tested for $4-7$ candidates how many voters the algorithm can handle. Clearly, we would have attained better results if we had allowed the algorithm to use a hard disk as a virtual memory. However, the I/O overheads would result in much higher running time, and we wanted to test our algorithm with reasonable limits. Therefore, the algorithm used only main memory, and the "extreme" results that are shown in Table 4.4 were achieved just before the algorithm ran out of space. The complete picture is shown in Figure 4.5. Note that the y-axis is shown on a logarithmic scale.

# of candidates	# of voters	Theoretical # of voting results	Actual # of voting results	Ratio	Time (sec)	Total # of cells
4	1100	223,045,351	47,331,609.2	0.212	111183.4	13,122,678,458.0
5	400	1,093,567,501	93,506,124.2	0.086	338200.4	7,640,484,607.0
6	140	498,187,404	18,146,578.2	0.036	4756	442,092,341.2
7	100	1,705,904,746	22,381,578.8	0.013	3792	346,504,543.2

Table 4.4: Extreme results. Fractions are rounded to 3 decimal places.

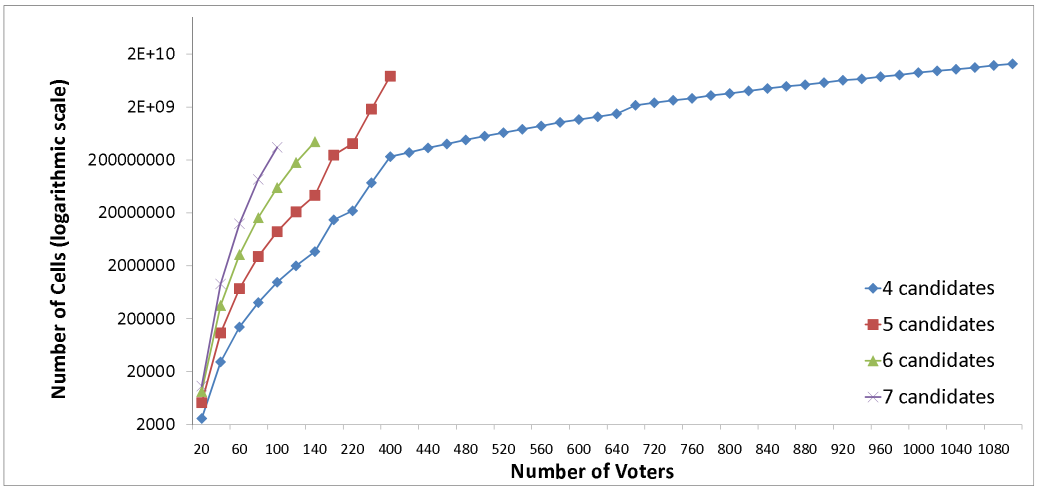


Figure 4.5: Results of the last set of experiments.

The results show that the actual running time (in terms of generated cells) heavily depends on the number of candidates, as expected. It is also apparent that the algorithm can handle a practical number of voters, even with 6 or 7 candidates. For example, the Israeli parliament (the Knesset) has 120 voters, and the United States Senate has 100 voters. Table 4.4 shows again the difference between the theoretical upper bound on the number of voting results (computed using WC function), and the actual number.

4.3 The Number of Candidates as a Parameter

If we cannot bound the number of candidates, then EVALUATION becomes much harder. In this section, we show that EVALUATION for k-approval, Borda, , Copeland, Bucklin and even for Plurality is #P-hard in this case. We also analyze the seemingly weaker question, the CHANCE-EVALUATION problem. Surprisingly, we show that even this problem is hard when voters do not all have equal weights under Plurality. We show that with equal weights, CHANCE-EVALUATION is still hard under k-approval, Approval, Range and Cumulative. However, we give a polynomial algorithm for the case where all voters have equal weights with Plurality.

4.3.1 The Evaluation Problem

Sometimes, the number of candidates cannot be assumed to be a constant, but is necessarily a parameter of the problem. For example, if a group of agents wants to choose one of them as a leader, $m = n$ and thus it is not a constant. There are some special cases where the number of voters is a constant and so a naive algorithm, which simply evaluates all possible options and runs in time polynomial of $O(m^n)$ will suffice. In most cases this is probably not going to happen. Unfortunately, as we will see, if both the number of voters, n, and the number of candidates, m, are given as parameters, the problem is #P-hard even for the Plurality, Borda, Range, Approval, Cumulative and Copeland voting rules.

All our #P-hard reductions will be from a well known #P-complete problem, PERMANENT, which is to calculate the permanent of a 01-matrix, (or, equivalently, to count the number of perfect matchings for a bipartite graph).

Definition 4.5. *Denote by S_n the set of all permutations of the numbers $1, 2, \ldots, n$. The* permanent *of an n-by-n matrix $A = (a_{i,j})$ is defined as*

$$perm(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

For a bipartite graph $G = (X \cup Y, E)$ such that $\forall (x, y) \in E, x \in X$ and $y \in Y$, and $|X| = |Y| = r$, a perfect matching *is a set of edges such that no two edges share a common vertex and every vertex is incident to exactly one edge. The permanent of G's adjacency matrix in fact counts the number of perfect matchings for G.*

We are now ready to show the proof for the Plurality voting rule.

Theorem 4.6. *If n and m are not constant, the* EVALUATION *problem is #P-hard for the Plurality voting rule.*

Proof. Given a bipartite graph $G = (X \cup Y, E)$, with $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_r\}$, for which we wish to count the number of perfect matchings, we construct an instance of the EVALUATION problem such that the probability of the chosen candidate to win is a function of (and only of) the number of perfect matchings in G. The voters are all the vertices of X plus two additional voters x_0 and $\hat{w}$, all with equal weights. The candidates are all the vertices of Y plus two additional candidates y_0 and $\hat{a}$. We first consider the case where the tie-breaking rule is lexicographic, and $\hat{a}$ has the top priority. For every $x \in X$, if $(x, y) \in E$, set the probability that voter x votes for candidate y to be $\frac{1}{r}$. With the remaining probability $(1 - \frac{\deg(x)}{r}$, where $\deg(x)$ is the degree of x) voter x votes for y_0. Finally, $\hat{w}$ votes for candidate $\hat{a}$ with probability 1, and x_0 votes for candidate y_0 with probability 1.

Consider a particular set of votes cast by the voters. Voters x_0 and $\hat{w}$ have no choice, so consider the choices made by voters in X. Each such set of choices naturally corresponds to a collection of r edges , M, between X and Y:

$$M = \{(x, y) \in X \times Y : x \text{ voted for } y\}$$

(note that if x voted for y_0 then this pair is not included in M). We show that $\hat{a}$ wins the election iff M is a perfect matching.

Suppose that M is a perfect matching, then all candidates in Y get exactly one vote (from the voters in X) as do $\hat{a}$ and y_0 (from $\hat{w}$ and x_0, respectively). Thus, all candidates obtain the same score, and $\hat{a}$ wins by the tie-breaking rule. Conversely, suppose that M is not a perfect matching. Then, either there is a candidate $y \in Y$ that gets more than one vote, or else there is a voter $x \in X$ that voted for y_0 (in addition to the vote y_0 surely received from x_0). In either case, there is a candidate that got more than one vote, while $\hat{a}$ received only one vote (from $\hat{w}$). Hence, $\hat{a}$ does not win the election.

The probability that the voters of X elect any specific perfect matching is r^{-r}. Thus

$$\Pr[\hat{a} \text{ wins the election}] = r^{-r} \cdot \mathrm{PM}(G)$$

where $\mathrm{PM}(G)$ denotes the number of perfect matchings in G. Hence, the answer to the Evaluation problem also gives us one for the number of perfect matchings.

The proof for random tie-breaking is essentially identical, only that in the case of an exact matching $\hat{a}$ does not necessarily win, but only wins with probability $\frac{1}{r+2}$. Hence, in this case $\Pr[\hat{a} \text{ wins the election}] = \frac{r^{-r}}{r+2} \cdot \mathrm{PM}(G)$. The rest of the proof remains the same. □

As for k-approval, we only need to slightly modify the reduction used in the proof for Plurality. Recall that in k-approval only the k first candidates get scores, so we don't care what is their order, and the order of the other $m - k$ candidates.

Theorem 4.7. *If n and m are not constant, the* Evaluation *problem is #P-hard for the k-approval voting rule, for every fixed k.*

Proof. Given a bipartite graph $G = (X \cup Y, E)$, with $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_r\}$, for which we wish to count the number of perfect matchings, we construct almost the same instance of the Evaluation problem as in the proof of Theorem 4.6. The set of voters is the same, but we add a set of dummy candidates $D = \{d^i_{y_j}\} \cup \{d^i_{\hat{a}}\}$, where $1 \leq i \leq k-1, 0 \leq j \leq r$. For every $x \in X$, if $(x, y) \in E$, set the probability that voter x gives one point to candidates $y, d^1_y, \ldots, d^{k-1}_y$ to be $\frac{1}{r}$. With the remaining probability ($1 - \frac{\deg(x)}{r}$, where $\deg(x)$ is the degree of x) voter x gives one point to $y_0, d^1_{y_0}, \ldots, d^{k-1}_{y_0}$. Finally, $\hat{w}$ gives one point to candidates $\hat{a}, d^1_{\hat{a}}, \ldots, d^{k-1}_{\hat{a}}$ with probability 1, and x_0 gives one point to candidates $y_0, d^1_{y_0}, \ldots, d^{k-1}_{y_0}$ with probability 1.

Now, each candidate $d^i_{y_j}$ and $d^i_{\hat{a}}$ gets the same number of points as y_j and $\hat{a}$, respectively. Therefore, the rest of the proof is essentially identical to that for the Plurality rule. □

We now turn to the Borda and Copeland protocols. We start with a simple lemma, the proof of which is trivial.

Lemma 4.8. *Let V be a set of voters, each with an individual preference order over a set of candidates. Suppose that all orders are different, and that for each preference order of any voter v, there exists another voter v' with the exact opposite preference order. Then:*

- *In the Borda protocol all candidates get the exact same score (which is also the average score).*
- *In the Copeland protocol, all pairwise contests are tied, for a total score of 0 for all candidates.*

Theorem 4.9. *If n and m are not constant, the* Evaluation *problem is #P-hard for the Borda voting rule.*

Proof. Let $G = (X \cup Y, E)$ be a bipartite graph, with $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_r\}$, for which we wish to count the number of perfect matchings. We construct an instance of the Evaluation problem as follows. There are $2(r+1)$ voters composed of two subsets: X^+ and W, with $r+1$ voters in each. The set X^+ consists of the set X plus one additional voter x_0. The set W consists of $r+1$ voters $w_0, \ldots, w_r$. All voters have equal weights.

There are $r+2$ candidates: $C = \{c_0, \ldots, c_r\}$ and one "special" candidate $\hat{a}$. We build the EVALUATION instance in such a way that every perfect matching in G corresponds to a voting scenario in which for every voter $x_i \in X^+$, there is a voter $w_j \in W$ with the exact reverse preference order. In this case, by Lemma 4.8 all candidates have the same score, and $\hat{a}$ wins by lexicographic tie-breaking rule. Furthermore, the EVALUATION instance is constructed so that $\hat{a}$ only wins in voting scenarios that correspond to perfect matchings in G. The details follow.

For ease of notation we denote $i \oplus j = (i+j) mod(r+1)$. Define the following set of orderings over the candidate set. For each $i = 0, \ldots, r$ let $s_i = (c_i, c_{i\oplus 1}, \ldots, c_{i\oplus r}, \hat{a})$, and denote by $(s_i)^R$ the reverse order to s_i. For each $(x_j, y_i) \in E$ (an edge in G), there is a probability of $1/r$ that voter x_j vote for order s_i. With the remaining probability $(1 - \frac{\deg(x_j)}{r})$ voter x_j votes for order s_0. Voter x_0 votes for s_0 with probability 1. For voters in W, voter w_j votes for order $(s_j)^R$ with probability 1. Note that, in particular, $\hat{a}$ is last in all votes of X^+ and first in all votes of W. See Figure 4.6 for an example of how to build an instance from a given bipartite graph where $r = 3$.

Consider a set of orders chosen by the voters. Only the voters of X have any choice, so consider their votes. Each such set of choices naturally corresponds to a collection of r edges, M, between X and Y:

$$M = \{(x_i, y_j) \in X \times Y : x_i \text{ voted } s_j\}$$

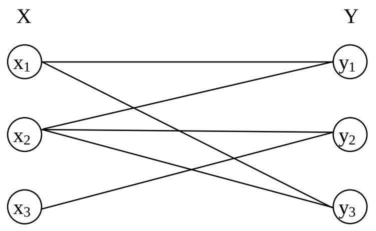
We show that for lexicographic tie-breaking, $\hat{a}$ wins the election iff M is a perfect matching in G.

Suppose that M is a perfect matching in G. Then, each s_i gets exactly one vote from the voters in X^+. However, each $(s_i)^R$ also receives exactly one vote, from the voters of W. Hence, for each preference order that received a vote, the exact opposite order was also voted for. In this case, by Lemma 4.8, $\hat{a}$ wins by lexicographic tie-breaking rule.

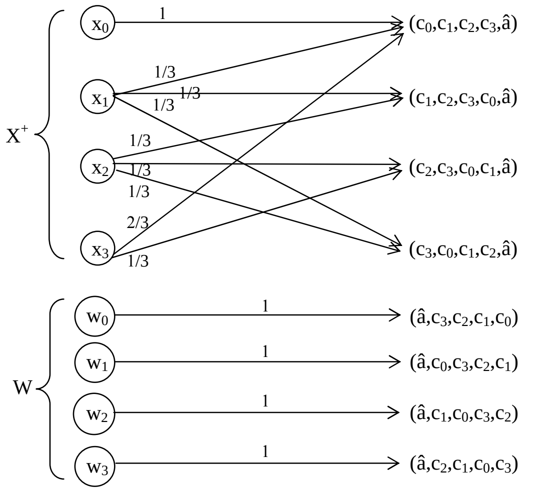
Conversely, suppose that M is not a perfect matching. Denote by α the average total score of the candidates, $\alpha = (r+1)^2$. Since α is an *average*, it is independent of the actual choices made by the voters. Consider M. Since M is not a perfect matching, there exists at least one order s_i that does note receive any vote from X^+. W.l.o.g. assume that this is s_r. Note that in all orders s_i with $i \neq r$ candidate c_r appears after candidate c_{r-1}. Hence, the total score that c_{r-1} gets from voters of X^+ must be higher than the total score they give c_r. The voters of W, on the other hand, in total give all candidates of C the exact same score (since the construction of the s_i's is symmetric). Hence, c_{r-1} gets a higher total score than c_r, and, in particular, it is not the case that all candidates get an identical total score. Thus, there must be a candidate c_{i_0} that gets a total score β strictly greater than the average α. On the other hand, the score of $\hat{a}$ is always the same (being always last in votes of X^+ and first in votes of W). Hence, its score is always identical to the one it gets in a perfect matching, namely α. Hence, $\hat{a}$ does not win the elections.

The probability that the voters of X elect any specific perfect matching is r^{-r}. Thus, $\Pr[\hat{a}$ wins the election$] = r^{-r} \cdot \mathrm{PM}(G)$. Hence, the answer to the Evaluation problem also gives us one for the number of perfect matchings.

The proof for random tie-breaking (instead of lexicographic) is essentially identical, as in the proof for Plurality. □



(a) Bipartite graph example, $r = 3$.



(b) The corresponding instance for the Evaluation algorithm.

Figure 4.6: Reduction from Permanent to Evaluation problem used in the proof of Theorems 4.9 and 4.10.

Theorem 4.10. *If n and m are not constant, the* Evaluation *problem is #P-hard for the Copeland voting rule.*

Proof. The proof is very similar to that of the Borda protocol, and uses the exact same construction. Following that proof, we show that also for the

Copeland protocol, $\hat{a}$ can win iff M (as defined in the Borda proof) is a perfect matching. Indeed, if M is a perfect matching, then as shown above, for each vote for a given preference order there is a vote for the exact reverse order. Thus, the conditions of Lemma 4.8 hold, and all candidates get an identical 0 score. Hence, $\hat{a}$ can win (either by lexicographic or by random choice, depending on the protocol).

Conversely, suppose that M is not a perfect matching. Then, there exists at least one order s_i that is not voted for by any voter of X^+. W.l.o.g. assume that this is s_r. In all orders s_i with $i \neq r$ candidate c_{r-1} appears before candidate c_r. In all orders $(s_i)^R$ with $i \neq (r-1)$ candidate c_{r-1} appears immediately after c_r, and in $(s_{r-1})^R$ it appears before candidate c_r. Hence, for any other candidate c_j, if c_r wins the pairwise contest with c_j, so does c_{r-1}. In addition, c_{r-1} beats c_r. Hence, in total, c_{r-1} must win strictly more pairwise contests than c_r. Hence, it cannot be the case that all candidates score exactly 0. Thus, since the average total score is necessarily 0, there must be at least one candidate that scores more than 0. On the other hand, $\hat{a}$ ties all pairwise contests (it is first in all votes by W and last in all those by X^+), for a total of 0. Thus, $\hat{a}$ cannot win the elections. The rest of the proof is identical to that for the Borda rule. $\square$

As for Bucklin, we use a slightly different construction. This proof does not assume the use of any specific tie-breaking rule.

Theorem 4.11. *If n and m are not constant, the* Evaluation *problem is #P-hard for the Bucklin voting rule.*

Proof. Let $G = (X \cup Y, E)$ be a bipartite graph, with $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_r\}$, for which we wish to count the number of perfect matchings. We construct an instance of the Evaluation problem as follows. There are $2(r+1)$ voters, thus the Bucklin score of the Bucklin winner will be at least $r+2$. The voters are composed of two subsets: X^+ and W, with $r+1$ voters in each. The set X^+ consists of the set X plus one additional voter x_0. The set W consists of $r+1$ voters $w_0, \ldots, w_r$. All voters have equal weights. There are $r+1$ regular candidates: $C = \{c_0, \ldots, c_r\}$ and one "special" candidate $\hat{a}$. Additionally, there are $(r+1)^2$ dummy candidates: $D = \{d^i_{y_j}\} \cup \{d^j_{\hat{a}}\}$, where $1 \leq i \leq r$, $0 \leq j \leq r$.

We build the Evaluation instance in such a way that every perfect matching in G corresponds to a voting scenario in which the Bucklin wining round is $r+2$ and candidate $\hat{a}$ wins. Furthermore, the Evaluation instance is constructed so that in other voting scenarios the Bucklin winning round is strictly less than $r+2$ and one of the candidates form C wins. The details follow.

For ease of notation we denote $i \oplus j = (i+j) mod (r+1)$. In our construction, the Bucklin wining round is always less than or equal to $r+2$, thus we show only the first $r+2$ candidates in each preference order (other candidates may be placed arbitrarily). For each $i = 0, \dots, r$ let $s_i = (c_i, d^1_{y_i}, \dots, d^r_{y_i}, \hat{a})$, and $t_i = (\hat{a}, c_i, c_{i\oplus 1}, \dots, c_{i \oplus r}, d^i_{\hat{a}})$. For each $(x_j, y_i) \in E$ (an edge in G), there is a probability of $1/r$ that voter x_j vote for order s_i. With the remaining probability $(1 - \frac{\deg(x_j)}{r})$ voter x_j votes for order s_0. Voter x_0 votes for s_0 with probability 1. For voters in W, voter w_j votes for order t_j with probability 1. Note that any order s_i gives one point to candidate $c \in C$ in the first round, and every order t_i gives one point to $c \in C$ on each round j, $2 \le j \le 4$.

Consider a set of orders chosen by the voters. Only the voters of X have any choice, so consider their votes. Each such set of choices naturally corresponds to a collection of r edges, M, between X and Y:

$$M = \{(x_i, y_j) \in X \times Y : x_i \text{ voted } s_j\}$$

We show that $\hat{a}$ wins the election iff M is a perfect matching in G.

Suppose that M is a perfect matching in G. Then, each s_i gets exactly one vote from the voters in X^+. For every j, $1 \le j \le r+1$, the score of every dummy candidate $d \in D$ in round j is less than or equals 1. The score of every candidate $c \in C$ in round j is j, and the score of $\hat{a}$ in round j is $r+1$. Since no candidate has more than $r+2$ points, every j, $1 \le j \le r+1$, is not the Bucklin winning round. On the other hand, in round $r+2$ the score of $\hat{a}$ is $2(r+1)$ while no other candidate has more than $r+1$ points. Therefore the Bucklin winning round is $r+2$ and $\hat{a}$ is the (unique) winner.

Conversely, suppose that M is not a perfect matching. Then, there exists at least one order s_i that is voted more than one time by voters of X^+. Therefore, there is at least one candidate $c \in C$ with a score of at least $r+2$ in round $r+1$. Then, the Bucklin winning round is less than or equals $r+1$. On the other hand, $\hat{a}$'s score in every round j, $1 \le j \le r+1$, is exactly $r+1$. Thus, $\hat{a}$ cannot win the elections.

The probability that the voters of X elect any specific perfect matching is r^{-r}. Thus, $\Pr[\hat{a} \text{ wins the election}] = r^{-r} \cdot \mathrm{PM}(G)$. Hence, the answer to the Evaluation problem also gives us one for the number of perfect matchings. $\square$

Note that all our proofs use equal weights for the voters, so the results hold for the weighted voters case with un-bounded or bounded weights too.

4.3.2 Chance-Evaluation Problem

Our original definition of the Evaluation problem yields a problem that is hard to compute for some common voting rules. Surprisingly, the weaker question, Chance-Evaluation, is hard even for the simplest voting rule – Plurality – when voters do not all have equal weights.

Theorem 4.12. *If n and m are not constant, the* Chance-Evaluation *problem is NP-complete for the Plurality voting rule when the voters do not all have equal weights.*

Proof. The problem is clearly in NP – given one voting scenario where c^* wins, we can check that indeed c^* is the winner in polynomial time. The NP-hardness reduction is from the NP-complete Bin-Packing problem: given a finite set U of items, an integer size $s(u)$ for each $u \in U$, a positive integer bin capacity B and a positive integer k, is there a partition of U into disjoint sets $U_1, U_2, \dots, U_k$ such that the sum of the sizes of the items in each U_i is B or less? The instance for the Chance-Evaluation problem is as follows. Every item is represented by a voter, where the item size is the voter's weight. We add another voter, v_z with the weight $B + 1$. Every bin is represented by a candidate, and we add another candidate z. v_z has a probability of 1 to vote for z, and all the other voters have an equal probability to vote for each one of the remaining candidates. We look for the possibility of z to be a winner. Note that every voting scenario corresponds to a packing and vice versa; a voter with weight x which votes for candidate y is like placing an item with size x in bin y. One item can not be in more than one bin and every voter can not vote for more than one candidate. Now suppose the tie-breaking rule is lexicographic and z is the minimal candidate with respect to the ordering. z is the winner if and only if all the other candidates get B or less votes. So there is a packing if and only if there is a voting scenario where z is the winner. The proof for random tie-breaking is similar, only that the weight of v_z is set to B. $\square$

This problem is NP-complete in the strong sense [54], meaning that even if the weights are bounded by $Poly(n)$ the problem remains hard (unlike the case with the constant number of candidates, as shown before).

Fortunately, for Plurality, if all voters have equal weights the problem can be solved in polynomial time.

Theorem 4.13. *Even if n and m are not constant, the* Chance-Evaluation *problem is in P for the Plurality voting rule where all voters have equal weights.*

Proof. We give a polynomial time algorithm to answer the Chance-Evaluation problem, assuming a random tie-breaking is used. The idea is very similar to the technique presented by West [109, p.176], and we also refer to Faliszewski et al. [44,45] for a different use of network flow techniques in the context of voting problems. Let c^* be the candidate for whom we are trying to determine whether it has any chance of winning. Count the number of voters that vote for c^* with non-zero probability, and denote this number by b. Then build a flow network $G = (V, E)$ which contains a bipartite graph $G' = (V1' \cup V2', E')$ and two additional nodes s and t, $V = V1' \cup V2' \cup \{s, t\}$. $V1'$ has a node for every voter which has a zero probability to vote for c^*, and $V2'$ has a node for every candidate but c^*. For every $i \in V_1'$, if voter i has a non-zero probability to vote for candidate j then $(i, j) \in E'$. In E, s has an edge with capacity 1 to all the nodes of $V1'$, t has an edge with capacity b from all the nodes of $V2'$, and if $(i, j) \in E'$, $(i, j) \in E$ too, with capacity 1. Now find a maximum flow and check that every edge from s to a node of $V1'$ has a residual capacity of zero. If such a flow exists, it represents a voting scenario where c^* gets b votes and all the other candidates get b or less votes so the algorithm returns “yes”. If not, then in every voting scenario, c^* can get at most b votes and there is at least one candidate who gets more than b votes so the algorithm returns “no”. The construction of the flow network and all the stages of the algorithm can be done in polynomial time, therefore the Chance-Evaluation problem for Plurality is in P where all the voters have equal weights.

The algorithm for lexicographic tie-breaking is similar. Let $Top(c^*)$ be the set of all candidates that are more favored than c^* according to the lexicographic tie-breaking. For every edge $(v, t) \in E$, such that v corresponds to a candidate in $Top(c^*)$, set the capacity to $b - 1$. The rest of the construction remains the same. If the required flow exists, it represents a voting scenario where c^* gets b votes, all the candidates that are more favored than c^* get $b - 1$ or less votes, and all the other candidates get b or less votes so the algorithm returns “yes”. If not, then in every voting scenario, c^* can get at most b votes and there is at least one candidate which is more favored than

c^* who gets more than $b-1$ votes, or there is other candidate who gets more than b votes. Thus, the algorithm returns "no". □

Figure 4.7 shows how the algorithm builds a flow network from the set of preferences in Figure 4.7a. In this example we seek a voting scenario where candidate D has a chance to win, and we use random tie-breaking. We remove voters V_1 and V_5 which have a non-zero probability of voting for D, and build a flow network as described in Figure 4.7b. In this example, a possible maximal flow is to assign 1 to all the outgoing edges of s, to the edges $(V_2, A), (V_3, B), (V_4, B), (V_6, C), (V_7, C)$ and (A, t), and to assign 2 to the edges from B and C to t. Therefore, D has a chance to win; if V_1 and V_5 vote for D, V_2 votes for A, V_3 and V_4 vote for B, and V_6 and V_7 vote for C.

For other voting rules, we get NP-hardness results as a corollaries of Xia and Conitzer's work [115](extended version). They considered the possible-winner problem, where they assumed to have a correct but incomplete model of each voter's preference order. The input to our problem is different; we have for each voter a collection of complete preference orders, with associated probabilities. Nevertheless, since the CHANCE-EVALUATION problem ignores the exact values of the probabilities, if the partial orders considered in possible-winner problem have a polynomial number of extensions, then possible-winner becomes a subproblem of CHANCE-EVALUATION. Thus we get:

Proposition 4.14. *If n and m are not constant, the* CHANCE-EVALUATION *problem is NP-complete for k-approval, Borda, Copeland, Bucklin, and Maximin voting rules, even if the voters are unweighted* [2].

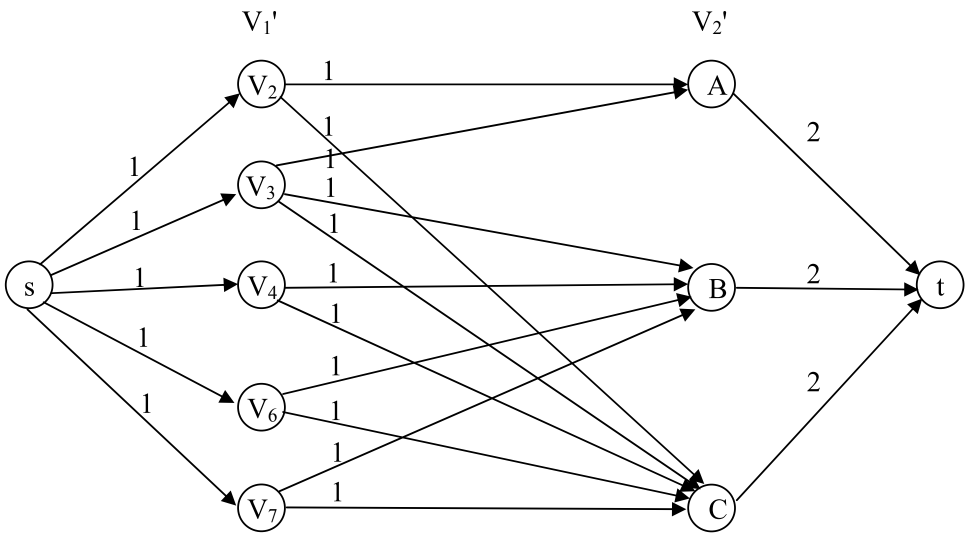
4.3.3 Monte Carlo Approximation

Computing the exact answer for EVALUATION and CHANCE-EVALUATION problems seems to be hard in many cases. However, we can utilize the underlying probabilities to achieve an approximate solution even for the EVALUATION problem. The idea is to use a statistical approach, in which we sample according to the given probabilities in order to estimate the real winning probability.

[2]This result also holds for Ranked Pairs and Voting tress, but we do not discuss these rules in this part of our work.

V_1	V_2	V_3	V_4	V_5	V_6	V_7
$\frac{1}{4}$ A	$\frac{1}{2}$ A	$\frac{1}{3}$ A	$\frac{1}{3}$ B	$\frac{1}{3}$ A	$\frac{1}{3}$ B	$\frac{1}{3}$ B
$\frac{1}{2}$ B	$\frac{1}{2}$ C	$\frac{1}{2}$ B	$\frac{2}{3}$ C	$\frac{2}{3}$ D	$\frac{2}{3}$ C	$\frac{2}{3}$ C
$\frac{1}{4}$ D		$\frac{1}{6}$ C				

(a) A set of preferences.



(b) The corresponding flow network for candidate D.

Figure 4.7: An example of how to build a flow network from a given set of preferences.

The algorithm is as follows. For each voter, we sample one preference order according to the given distribution, thus obtaining a voting scenario. Since the voters' choices are independent, this process is equivalent to sampling one voting scenario according to the voting scenarios' distribution (which we do not know). We then calculate the winner from this voting scenario using the given voting rule, and repeat the whole process t times. Given a specific candidate, c^*, we are interested in his winning probability, denoted p. This probability is approximately the number of sampled voting scenarios where c^* wins divided by t, denote this ratio by $\hat{p}$.

From the perspective of c^*, each iteration has two possible outcomes: where c^* wins or when another candidate wins. The winning probability of c^*, p, is the same in each iteration, and the iterations are statistically independent. Therefore, the distribution of p is a binomial distribution, and the maximum likelihood estimator for p is $\hat{p}$. This estimator is also known to be unbiased for the binomial distribution. We can build a binomial confidence interval which relies on approximating the binomial distribution with a normal distribution, by the following formula:

$$P\left(\hat{p} - \sqrt{\frac{\hat{p}(1-\hat{p})}{t}} Z_{1-\frac{\alpha}{2}} \leq \ p \ \leq \hat{p} + \sqrt{\frac{\hat{p}(1-\hat{p})}{t}} Z_{1-\frac{\alpha}{2}}\right) = 1 - \alpha \qquad (4.1)$$

where $Z_{1-\frac{\alpha}{2}}$ is the $1-\frac{\alpha}{2}$ percentile of a standard normal distribution, and α is our chosen probability of error. For bounding the distance from the real winning probability, we require that given an ϵ,

$$|p - \hat{p}| \leq \epsilon \qquad (4.2)$$

Combining (1) and (2) above we get that the number of required iterations is:

$$t \geq \left(\frac{\sqrt{\hat{p}(1-\hat{p})} Z_{1-\frac{\alpha}{2}}}{\epsilon}\right)^2 \qquad (4.3)$$

i.e, the winning probability $\hat{p}$ that we have found after such t iterations is, with a probability of $1-\alpha$, within ϵ-environment of the real probability p. Table 4.5 shows the required number of iterations (t) as a function of ϵ and α, assuming that $\hat{p}(1-\hat{p})$ is maximal, i.e. $\hat{p} = 0.5$.

α	ϵ	t
0.05	0.05	271
0.05	0.01	6,764
0.05	0.001	676,386
0.01	0.05	542
0.01	0.01	13,530
0.01	0.001	1,352,974
0.001	0.05	955
0.001	0.01	23,874
0.001	0.001	2,387,384
0.0001	0.05	1,384
0.0001	0.01	34,578
0.0001	0.001	3,457,771

Table 4.5: Number of iterations as a function of ϵ and α.

Chapter 5

How to Rig Elections and Competitions

In this chapter we consider the evaluation and control of elections, with the presence of uncertainty. We assume that we only know the probability that a candidate will be preferred over another. We first formally define the underlying assumptions and problems in Section 5.1. In Section 5.2 we give a polynomial time algorithm for evaluating an agenda with any voting tree, and show an optimized version of this algorithm for balanced voting trees. We then show that rigging an agenda for balanced voting trees is provably hard (the complete proof is due to [104, 105]). In Section 5.3 we analyze the linear order case. We first show how to improve the general agenda evaluation algorithm for linear orders, and prove the unfairness of the linear order rule; a candidate can only benefit by going late in a voting order. Thus, the election officer can try to increase a candidate's chance of winning by placing it last in the voting order. We then show that a relaxed version of the original rigging agenda problem, is hard to solve. However, it is also natural to ask if there is any agenda which would make a specific candidate the winner with a non-zero probability. With linear order, we show that this problem can be solved in polynomial time. Our hardness results may lead us to conjuncture that a designer cannot benefit from having the probabilistic information, since it is hard to rig an election agenda even with this input. However, in practice, a worst-case analysis is not enough. We thus present heuristics for agenda rigging in Section 5.4. We investigate the performance of these heuristics for both randomly generated data sets and real-world data sets from tennis and basketball competitions. Our heuristics achieved over

96% of the optimal solution on average for the randomly generated and the basketball data set, and performed reasonably well for the tennis data set.

5.1 Model and Problem Definitions

In our probabilistic model, we assume that for any pairwise election, 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, except the fundamental properties. Thus there might be no transitivity between the winning probabilities, e.g., candidate i has more than a 50% chance of beating candidate j, candidate j has more than a 50% chance of beating candidate k, but candidate k also has more than a 50% chance of beating candidate i. We summarize this information on an *imperfect information ballot matrix* M, which is a $C \times C$ matrix of probabilities, such that if $M[c_i, c_j] = p$, then in a pairwise election between c_i and c_j, candidate c_i will win with a probability of p. We require that $0 \leq M[c_i, c_j] = 1 - M[c_j, c_i] \leq 1$. If all probabilities are 0 or 1 then we say the scenario is one of *perfect information*, and the ballot matrix represents the adjacency matrix of the majority graph, G (see Chapter 3 for the definition of G).

The most obvious way to organize a series of pairwise elections is in a *fair tree order*. In Figure 5.1(b), we see how pairwise elections between candidates A, B, C and D may be organized into such a tree. The idea is that candidates A and B face each other in a pairwise election, while candidates C and D face each other in another pairwise election. This association of all the candidates to the leaves is the *agenda*, α. The winner of the first pairwise election (A in this case) then faces the winner of the second (D), and the winner of this third pairwise election (D) is declared the overall winner. This process of elimination is simulated by the *labeling* function, and the overall winner is the candidate labeled at the root of the tree (see Chapter 3 for a formal definition of the labeling function). The voting tree in Figure 5.1(b) is said to be *fair* because it is *balanced*, and as a consequence every possible overall winner would have to win the same number of pairwise elections.

The task of the election officer may thus be perceived as generating such a binary tree, T, with candidates C allocated to leaves of the tree. Since only partial (probabilistic) information is known, we first need to consider the *evaluation* problem, which requires computation of the winning probabilities

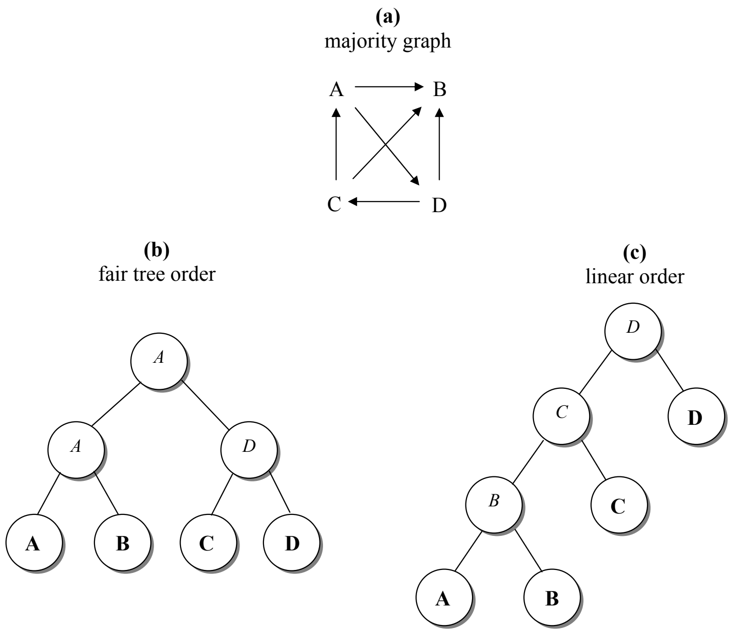


Figure 5.1: Majority graph (a), and two possible voting trees: linear order (b) and fair tree order (c). The bold font represents the agenda, while the italic font represents the labeling of the tree.

of the candidates. Formally, given $\langle T, \alpha, M\rangle$ the *evaluation* of T with respect to α and M is a mapping $\eta : V \rightarrow [0,1]^n$ such that for any $v \in V$, $\eta(v) = \langle p_1, \ldots, p_n\rangle$ if and only if $Pr[\ c_i$ is the winner at $v\ ] = p_i$. Thus at index i $\eta(r(T))$ will contain the probability that c_i will be the overall winner of T.

The opportunity for manipulation by the election officer is possible in such settings , for example, by placing a favored candidate against candidates it is likely to beat. If we relax the fairness constraint, then the possibilities for an election officer to manipulate the election increase. Figure 5.1(c) shows a rather unfair voting tree; in fact, it defines a *linear order* (A, B, C, D) for the candidates, with the first ballot taking place between A and B, the winner competing against C, and so on, until the winner of the final ballot (D in this case) is the overall winner. The unfairness arises due to the possibility of a candidate winning the overall election despite only participating in one ballot (as is the case depicted in Figure 5.1(c)). We will denote linear voting orders (i.e., permutations of C) by $\pi, \pi', \ldots$.

In a scenario of perfect information, a successful manipulation by the election officer is one that guarantees that a specific candidate will win. In our setting, as we deal with probabilities, the election officer's goal can be interpreted in two different concrete formulations:

- Imperfect information agenda rigging (IIAR): Given a set of candidates C, an imperfect information ballot matrix M, a favored candidate $c^* \in C$ and a probability p, does an agenda α exist such that c^* will win in this setting with a probability of at least p?

- Imperfect information weak agenda rigging (IIWAR): Given a set of candidates C, an imperfect information ballot matrix M and a favored candidate $c^* \in C$, does an agenda α which would make c^* the winner with a non-zero probability exist?

In the following sections we analyze the complexity of the IIAR and IIWAR problems with fair tree and linear orders. We will refer to these problems in the fair tree order setting as IIAR^f and IIWAR^f , respectively, and in the linear order setting as IIAR^l and IIWAR^l.

5.2 Voting with a Fair Tree Order

We begin by considering the evaluation problem, which requires computation of the candidates' winning probabilities. It is not obvious that even the evaluation problem is easy in our setting, since to compute the probability that a given candidate will win the overall election, we must consider every possible ordering of wins emerging from a given tree structure: in any given ballot, there are two outcomes, in contrast to the perfect information case. However, the following result implies the problem is in P for **every** voting tree.

Theorem 5.1. *Given $\langle T, \alpha, M\rangle$, where T is any m leaf binary tree, the* evaluation *of T with respect to α and M is computable in $O(m^3)$ arithmetic operations.*

Proof. Consider the following algorithm.

```
1. $\eta(x) = unlabelled$ for each $x \in V(T)$
2. For each leaf, $x$, of $T$, $\eta(x) = \langle x_1, \ldots, x_m\rangle$ with $x_i = 1$ if $\alpha(x) = c_i$ and
   $x_i = 0$ otherwise.
3. repeat
      a. Let $z$ be any node of $T$ with children $x$ and $y$ such that
         $\eta(x) \neq unlabelled$ and $\eta(y) \neq unlabelled$. Let $\langle x_1, \ldots, x_m\rangle$
         denote $\eta(x)$ and, similarly, $\langle y_1, \ldots, y_m\rangle$ denote $\eta(y)$. Compute
         $\eta(z) = \langle z_1, \ldots, z_m\rangle$ using
         $$z_i \quad := \quad \begin{cases} x_i \ \sum_{j=1}^{m} \ y_j \ M[c_i, c_j] & \text{if} \quad x_i > 0 \\ y_i \ \sum_{j=1}^{m} \ x_j \ M[c_i, c_j] & \text{if} \quad y_i > 0 \\ 0 & \text{if} \quad x_i = y_i = 0 \end{cases}$$
4. until every $v$ has $\eta(v) \neq unlabelled$
```

Step (2) correctly assigns $\eta(v)$ for each leaf v of T. Inductively assuming that z with children x and y has both $\eta(x)$ and $\eta(y)$ correctly assigned, consider the computation of $\eta(z)$ in 3(a). It cannot be the case that both $x_i > 0$ and $y_i > 0$ since the candidate c_i is a leaf in at most one of the sub-trees with roots x and y. If c_i is a leaf of neither sub-tree then $x_i = y_i = 0$ leading to $z_i = 0$, i.e., c_i cannot be a possible winner at z. Without loss of generality suppose $x_i > 0$. In order for c_i to be a winner at node z it must first be successful at node x (probability x_i by the inductive assumption) and defeat the winner, y_j, at node y ($y_j M[c_i, c_j]$). Summing over the contributing terms yields the expression given in 3(a). To complete the argument it suffices to note that $O(m^2)$ operations are carried out to compute $\eta(z)$, giving a worst-case overall number of $O(m^3)$ steps. □

Vu et al. [104, 105] show a similar algorithm that calculates the winning probability of a *specific* candidate in $O(m^2)$. However, if we consider the more common setting, in which ballots are organized according to a fair tree order, we can use an optimized version of our algorithm . In this manner we reduce the overall time complexity of calculating the winning probabilities of *all* the candidates to $O(m^2)$. The proof is given in the appendix. We obtain:

Theorem 5.2. *Given $\langle T, \alpha, M \rangle$, where T is a fair tree order, the* evaluation *of T with respect to α and M is computable in $O(m^2)$ arithmetic operations.*

We now continue with the control problem, which is interpreted in our setting as the problem of finding an agenda that gives a named candidate at least a certain probability of winning (IIAR^f problem). Due to Theorem 5.2, with a fair tree order the problem is in NP. Fortunately, it is also provably hard.

Theorem 5.3. *IIAR^f is NP-complete.*

The proof of this theorem is due to Theorem 5 in [104, 105]. However the complexity of the weaker version, IIWAR^f, is still an open problem.

5.3 Voting with a Linear Order of Ballots

We now focus on the linear order case. Given such an order $\pi = (c_1, \ldots, c_m)$ and candidate $c^* \in C$, the probability of c^* being the overall winner of π is denoted by $Pr[w(\pi) = c^* \mid M]$, and is given as follows. For a voting order $\pi = (c_{i_1}, c_{i_2}, \ldots, c_{i_m})$ with $c^* = c_{i_k}$, (i.e., the preferred winner occurs k'th in the order π),

$$P[w(\pi) = c^* \mid M] = \phi \times \psi$$

where

$$\phi = \left(\prod_{j=k+1}^{m} M[c_{i_k}, c_{i_j}] \right)$$
$$\psi = \left(\sum_{j=1}^{k-1} P[w(c_{i_1} c_{i_2} \ldots c_{i_{k-1}}) = c_{i_j} \mid M] \times M[c_{i_k}, c_{i_j}] \right)$$

That is, in order for c^* to emerge as the winning candidate, c^* must defeat every candidate put forward *later* in the voting order, *and* succeed against the eventual winner of the voting order formed by the *earlier* candidates. Theorem 5.1 showed that the probability of a candidate wining in any given voting tree can be computed in time $O(m^3)$; for linear orders, we can improve this to $O(m^2)$.

Theorem 5.4. *Given $\langle \pi, M \rangle$ and any $c^* \in C$, $P[w(\pi) = c^* \mid M]$ can be computed in $O(m^2)$ arithmetic operations.*

Proof. Let $\pi = (c_{i_1}, c_{i_2}, \ldots, c_{i_m})$. Consider the $m \times m$ matrix, $T^{(\pi)}$ whose entries, $T^{(\pi)}[j,k]$ are,

$$T^{(\pi)}[j,k] \quad = \quad P[w(c_{i_1} \ldots c_{i_j}) = c_k \mid M]$$

Informally, $T^{(\pi)}[j,k]$ is the probability that candidate c_k is the (current) winner, immediately prior to the k'th ballot being held. To prove the lemma it is sufficient to prove that $T^{(\pi)}$ may be constructed in polynomial time given M and π. This, however, is an easy consequence of the following,

$$T^{(\pi)}[j,k] \quad = \quad \begin{cases} 1 & \text{if} \quad j = 1 \text{ and } c_k = c_{i_1} \\ 0 & \text{if} \quad c_k \in \{c_{i_{j+1}}, c_{i_{j+2}}, \ldots, c_{i_m}\} \\ \sum_{r=1}^{j-1} M[c_k, c_{i_r}] \times T^{(\pi)}[j-1, i_r] & \text{if} \quad c_k = c_{i_j} \\ T^{(\pi)}[j-1, k] \times M[c_k, c_{i_j}] & \text{if} \quad c_k \in \{c_{i_1}, \ldots, c_{i_{j-1}}\} \end{cases}$$

Thus $[T^{(\pi)}[1,1], \ldots, T^{(\pi)}[1,m]]$ can be determined directly from π and each subsequent row of $T^{(\pi)}$ is computable from its predecessor, π, and M using $O(m)$ arithmetic operations. It follows that $T^{(\pi)}$ can be computed with $O(m^2)$ arithmetic operations. □

What else can be said about voting with linear orders? First, we can make precise, and prove correct, the intuition that there is no benefit to going early; a candidate can only benefit by going late in a voting order. While this seems intuitive, the proof is surprisingly involved.

Lemma 5.5. *Given $\langle M, C\rangle$ let c_i, c_j be distinct members of C. For every voting order $\pi_1\pi_2$ of $C \setminus \{c_i, c_j\}$, it holds that $Pr[w(\pi_1 c_j c_i \pi_2) = c_i] \;\geq\; Pr[w(\pi_1 c_i c_j \pi_2) = c_i]$.*

Proof. Without loss of generality, let $c_i = c_m$, $c_j = c_{m-1}$, and

$$\begin{array}{rcl} \pi & = & \pi_1 c_m c_{m-1} \pi_2 \\ \pi' & = & \pi_1 c_{m-1} c_m \pi_2 \end{array}$$

First observe that we may assume $|\pi_2| = 0$ and thus it is sufficient to prove for all choices of π_1,

$$P[win(\pi_1 c_{m-1} c_m) = c_m] \quad\geq\quad P[win(\pi_1 c_m c_{m-1}) = c_m]$$

The probability, P_{LHS} on the left-hand side of this expression is,

$$\begin{array}{l} \sum_{k=1}^{m-2} P[win(\pi_1) = c_k]\; M[c_{m-1}, c_k]\; M[c_m, c_{m-1}] \;+ \\ \sum_{k=1}^{m-2} P[win(\pi_1) = c_k]\; M[c_k, c_{m-1}]\; M[c_m, c_k] \end{array}$$

whereas that on the right-hand side, P_{RHS} is,

$$\sum_{k-1}^{m-2} P[win(\pi_1) = c_k]\; M[c_m, c_k]\; M[c_m, c_{m-1}]$$

In order to simplify the notational complexity, for $1 \leq k \leq m-2$, let

$$\begin{array}{rcl} x_k & = & P[win(\pi_1) = c_k] \\ y_k & = & M[c_{m-1}, c_k] \\ z_k & = & M[c_m, c_k] \\ \alpha & = & M[c_m, c_{m-1}] \end{array}$$

Recall that

$$M[c_k, c_{m-1}] \;=\; 1 \;-\; M[c_{m-1}, c_k] \;=\; 1 \;-\; y_k$$

So that $P_{\mathrm{LHS}} \geq P_{\mathrm{RHS}}$ holds only if

$$\sum_{k=1}^{m-2} x_k y_k \alpha \;+\; \sum_{k=1}^{m-2} x_k(1-y_k)z_k \quad\geq\quad \sum_{k=1}^{m-2} x_k z_k \alpha$$

which, after rearranging terms, holds only if

$$\alpha \sum_{k=1}^{m-2} x_k y_k \;+\; \sum_{k=1}^{m-2} x_k z_k \;\geq\; \alpha \sum_{k=1}^{m-2} x_k z_k \;+\; \sum_{k=1}^{m-2} x_k y_k z_k$$

That is,

$$\sum_{k=1}^{m-2} x_k(\alpha y_k \;+\; z_k) \;\geq\; \sum_{k=1}^{m-2} x_k z_k(\alpha + y_k) \tag{5.1}$$

The lemma follows only if this final inequality holds for every choice of α, x_k, y_k, and z_k for which $0 \leq x_k, y_k, z_k \leq 1$: these simply state that the individual terms are *probabilities*.

Now if it is the case that for each k, $x_k(\alpha y_k + z_k) \geq x_k z_k(\alpha + y_k)$, then the inequality in (5.1) is true. Thus, consider a typical terms $x(\alpha y + z)$, $xz(\alpha + y)$. We wish to show:

$$\forall\ 0 \;\leq\; \alpha, x, y, z \leq\; 1 \quad x(\alpha y + z) \geq xz(\alpha + y)$$

Since this is obviously true when $x = 0$, is sufficient to prove $\alpha y + z \geq z(\alpha + y)$. If $\alpha \in \{0, 1\}$ then $\alpha y + z \in \{z, y + z\}$ whereas $z(\alpha + y) \in \{zy, zy + z\}$ so that the required inequality is immediate: $z \geq zy$ and $y + z \geq zy + z$. We have two cases: $y \geq z$ and $y < z$. In the first of these $\alpha y + z \geq z(\alpha + y)$ follows by rearranging to obtain the inequality $\alpha(y - z) \;\geq\; z(y - 1)$: since $y - z \geq 0$ and $y - 1 \leq 0$ this inequality always holds. We are left with the case of $0 \leq y < z \leq 1$, $0 < \alpha < 1$. In this case, $\alpha y + z \geq z(\alpha + y)$ may be re-written as

$$y + (z - y) \;\;\geq\;\; yz + \alpha(z - y)$$

thus completing the proof of lemma: $(z - y) > 0$, $y \geq yz$ and $(z - y) \geq \alpha(z - y)$. □

The immediate corollary is as follows.

Corollary 5.6. *For any candidate $c^* \in C$, if there is a voting order, π such that $Pr[w(\pi) = c^*] \geq p]$, then there is a voting order π' such that $Pr[w(\pi') = c^*] \geq p]$ and in which c^* is the final candidate to run.*

Proof. Given π with $Pr[w(\pi) = c] \geq p$, apply Lemma 5.5 repeatedly to move c later in the voting order until it is the final candidate. □

Vu et al. [104,105] generalized this result, showing that in any voting tree, the biased tree that maximizes the winning probability of c^* has the biased structure in which c^* has to play only the final match.

The general problem of determining whether there exists any agenda which gives a named candidate at least a certain probability of winning is hard to classify in the linear order setting, and so we will analyze a restricted version of the problem, as follows. When we think informally about rigging an agenda, we tend not to think just in terms of the agenda, but also in terms of the *specific outcomes* that we want the agenda to lead to. So, we might think in terms of "if I put A up against B, then B wins and goes up against C, and C wins..." and so on. Here, we have not just the agenda (ABC) but also the outcomes of the ballots (B wins the first; C the second; ...). Here, we call these structures – which include the agenda for the ballots together with the intended outcomes – a *run*. A run has the form $r : c_1, c_2 \xrightarrow{c_2} c_3 \xrightarrow{c_3} \cdots \xrightarrow{c_{k-1}} c^* \xrightarrow{c^*}$ where c_1 and c_2 are the candidates up against each other in the first ballot, c_2 is the intended winner of this ballot, and so on, until the final ballot is between c_{k-1} and c^*, in which we intend the winner – and hence overall winner – to be c^*. Computing the probability that this run will result in our desired candidate c^* winning is simple – it is the value: $Pr[w(c_1, c_2) = c_2] \times Pr[w(c_2, c_3) = c_3] \times \cdots \times Pr[w(c^{k-1}, c^*) = c^*]$. We denote this value for a run r by $Pr[r \mid M]$. So, in the *relaxed imperfect information agenda rigging* (RIIAR) problem, we are given a set of candidates C, an imperfect information ballot matrix M, a favored candidate $c^* \in C$, and a probability p. We are asked whether there exists a run r, in which the overall winner is c^*, such that $Pr[r \mid M] \geq p$.

Theorem 5.7. *RIIAR is NP-complete.*

Proof. A standard "guess and check" algorithm gives membership in NP. For hardness, we reduce the k-HCA problem on tournaments [12, p.46]: we are given a tournament $G = (V, E)$ (i.e., a complete digraph such that $(c, c') \in E$ iff $(c', c) \notin E$) and a subset $E' \subseteq E$, and we are asked whether G contains a Hamiltonian cycle containing all edges E'. We create an instance of RIIAR as follows. The outcomes will be the vertices of G together with a new vertex, $v_\perp$. Given a tournament $G = (V, E)$ and required edge set E', we create a probability matrix so that G contains a cycle with E' iff we can create an ordering of vertices satisfying the property given above. For each $(u, v) \in E'$ we set $M[u, v] = 1$. For each $(u, v) \in E \setminus E'$ we set $M[u, v] = 1 - \frac{1}{10|V|}$ (i.e., a "high" probability). We then set the target probability to be $(1-\frac{1}{10^{|V|}})^{|V|-|E'|}$. The intuition is that in any ordering which satisfies these properties, we can only visit at most $|E \setminus E'|$ edges not in E', and so we such an ordering must visit all edges in E'. The difficulty is that we are after a cycle, so we need to select a vertex (call it $v_\top$) to act as the source of the cycle and our new vertex, $v_\perp$, will act as the sink in the cycle; if we have an arc $(u, v_\top) \in E'$ then we define $M[u, v_\perp] = 1$, while if $(u, v_\top) \in E \setminus E'$ then we define $M[u, v_\perp] = 1 - \frac{1}{10^{|V|}}$; if for any vertex $u \in V$ we have not yet defined a value for $M[u, v_\perp]$ then define $M[u, v_\perp] = 0.5$. We then ask whether we can rig the agenda for $v^\top$ to win with probability greater than the target. □

It is also natural to ask if there is *some* permutation of C which would make c^* the winner with a non-zero probability. This is the IIWAR^l problem, which can be solved efficiently.

Theorem 5.8. *IIWAR^l is in P.*

Proof. In order to solve this IIWAR^l, we convert every non-zero probability to 1, i.e., $\forall(i,j)$, if $M[i,j] > 0$, assign $M[i,j] = 1$. We then use the algorithm introduced by Lang et al. [79] to find the voting tree structure that allows our favorite candidate c^* to win or decide that it is impossible for c^* to win. Using induction, it is easy to see that this voting tree can be built as a caterpillar tree, i.e. with a linear order of ballots. □

5.4 Heuristics and Experimental Evaluation

The hardness results of theorems 5.3 and 5.7 may lead us to conjuncture that a designer cannot benefit from having the probability matrix M, since it is hard to rig an election agenda even with this input. However, in practice, a worst-case analysis is not enough. The situation is similar to that in cryptography, where a secure protocol is not one that is hard to break in the worst case, but one that can be broken only with negligible probability. Bartholdi et al., who were in many ways pioneers of the complexity-theoretic approach to understanding election manipulation first voiced the concern that NP-hardness results are not enough:

> *Concern:* It might be that there are effective heuristics to manipulate an election even though manipulation is NP-complete.
> *Discussion:* True. The existence of effective heuristics would

> weaken any practical import of our idea. It would be very interesting to find such heuristics. [15]

This motivated us to consider heuristics for our problems, and to test their performance in different scenarios. We used a bespoke simulation program (written in C) to evaluate the heuristics. Our experiments were in two categories: first, randomly generated data, and second, public domain form data from sports competitions. For each of these settings, we evaluated the heuristics for both linear and fair tree orders. For the randomly generated data sets, we first generated random values for the probability matrix M from a uniform distribution in the range $[0, 1]$, and completed the matrix to preserve probability constraints. We then ran 100 iterations, and during each iteration a winner candidate, c^*, was randomly chosen and each heuristic was used in an attempt to generate an optimal order for this player. The second scenario was the same, except the random values for the matrix were taken from a normal distribution with an average of 0.5 and a standard deviation of 0.2 while preserving probability constraints. For the real-world data sets, we based our experiments on data from basketball and tennis competitions. For the basketball experiments, we took 29 teams from the NBA, and computed the probability matrix M from public domain form data. Here, there were no iterations, but for every team we used each heuristic to generate a playing order that would give this team the best chance of winning. For the tennis experiments, we used the 13 players from the top of the ATP ranking, again computing the probability matrix from public domain form tables.

5.4.1 Heuristics for Linear Order Voting Tree

We start with the case which is more vulnerable to manipulation by the election officer, where the elections are organized according to a linear order. The heuristics which we developed are as follows.

- *Optimal*: In those cases where it was computationally feasible to do so, we exhaustively evaluated every permutation in order to find the real optimal agenda as a comparison.

- *Far adversary*: The idea was to minimize the probability that our favorite candidate c^* would face candidates that he had a high probability of losing to. Thus, the candidate who had the highest probability to beat c^* was assigned to the leftmost leaf, the one that has the second highest probability to beat c^* was assigned to the second leaf (from the left), and so on; c^* was chosen to be at the rightmost leaf (cf. Corollary 5.6).

- *Best win*: The favorite candidate c^* was chosen to be at the rightmost leaf, and the candidate that c^* had the best chance of beating was chosen to be before it, the next being the one that this candidate was most likely to beat, and so on.

- *Simple convert*: The idea was to convert the probability matrix M into a binary matrix, and then simply apply the algorithm from Theorem 5.8. To create the binary matrix, every probability which was greater than or equal to 0.5 was converted to 1, and the others were converted to 0. If no agenda could be found, (which is sometimes the case when the number of candidates is small) a random agenda was generated.

- *Threshold convert*: This was a more sophisticated attempt to convert M into a binary matrix. We searched for the maximum threshold above which, if we convert all the probabilities above it to 1 and below it to 0, there was still an agenda that enabled c^* to win (on the converted matrix). We used a binary search, stopping when the difference between the low/high limit and the threshold was less than 0.005. As before, if no agenda could be found a random agenda was generated.

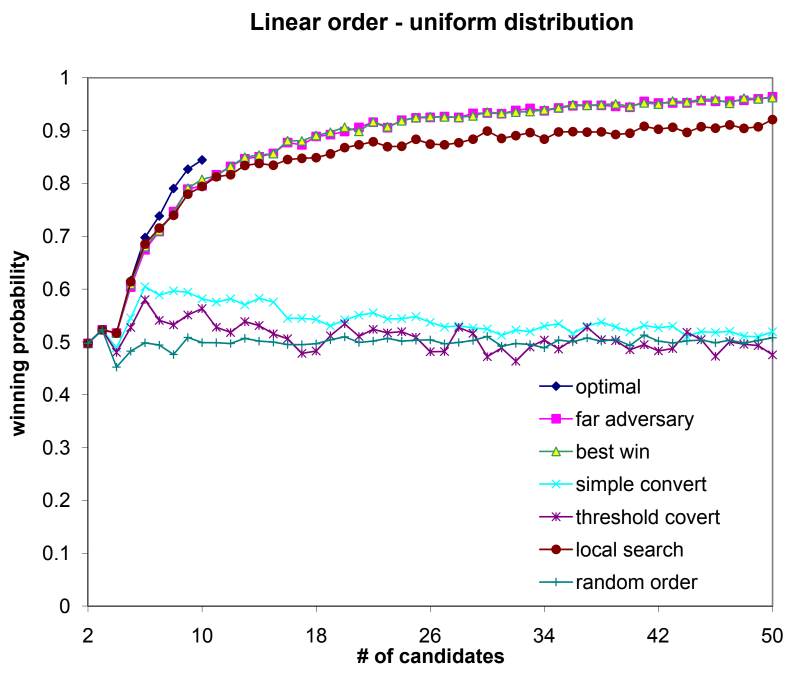
- *Local search*: c^* was chosen to be at the rightmost leaf. For the other places, in every iteration a random permutation was chosen. Then $0.5 * |N|$ random swaps were tested to find an agenda with maximum winning probability ($|N|$ iterations were done).

- *Random order*: As a control, a random agenda was also generated.

The results for heuristics on linear order are shown in Figures 5.2 and 5.3.

First, note that the overall performance of the heuristics does not vary significantly between uniform and normal distributions (Figure 5.2, first and second graphs, respectively). In these graphs, the x-axis is the number of candidates and the y-axis is the winning probability that was found using our heuristics. Every point in the graph represents the winning probability that was averaged over 100 iterations.

In the uniform distribution experiments, it seems that best-win and far adversary seem to perform similarly well (marginally better than local



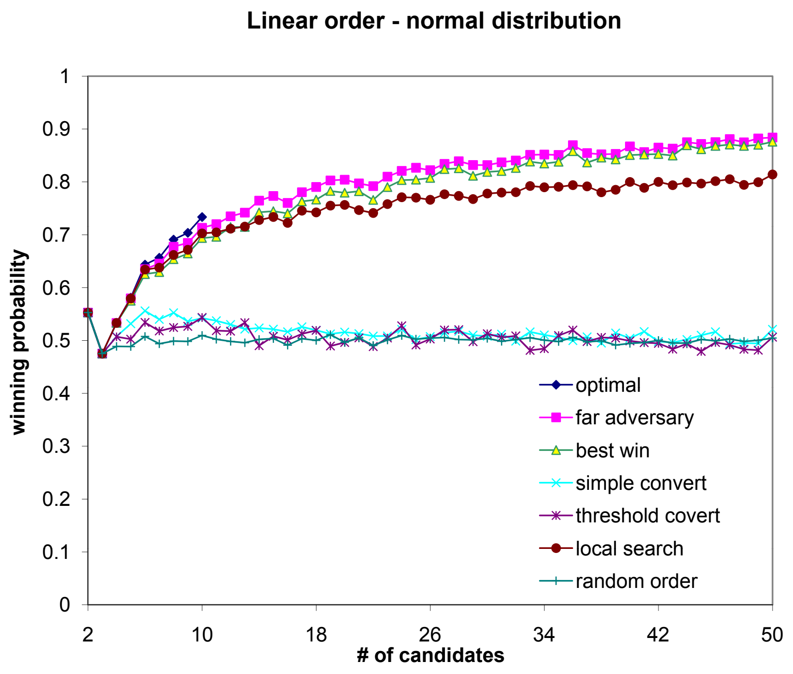
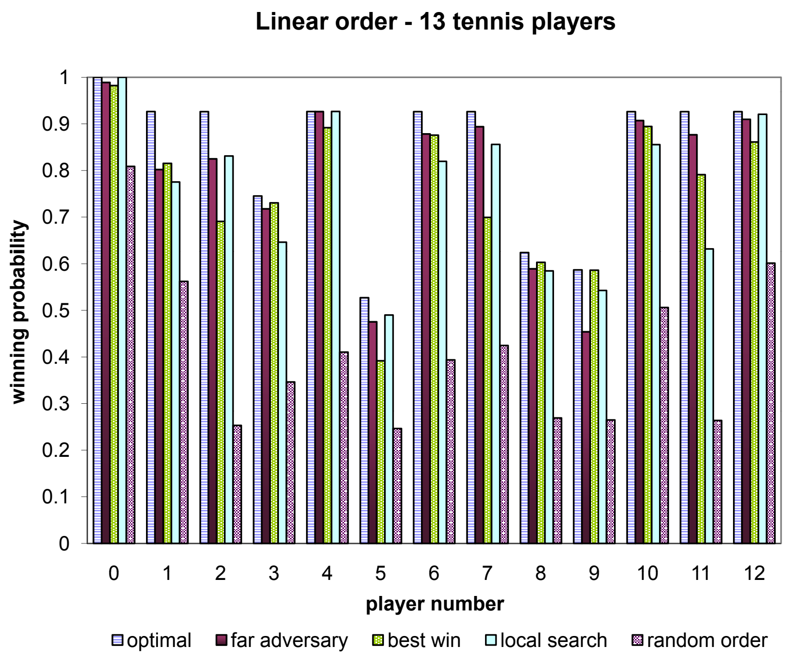


Figure 5.2: Performance of heuristics for linear order for randomly generated probability matrices using uniform and normal probability distributions.



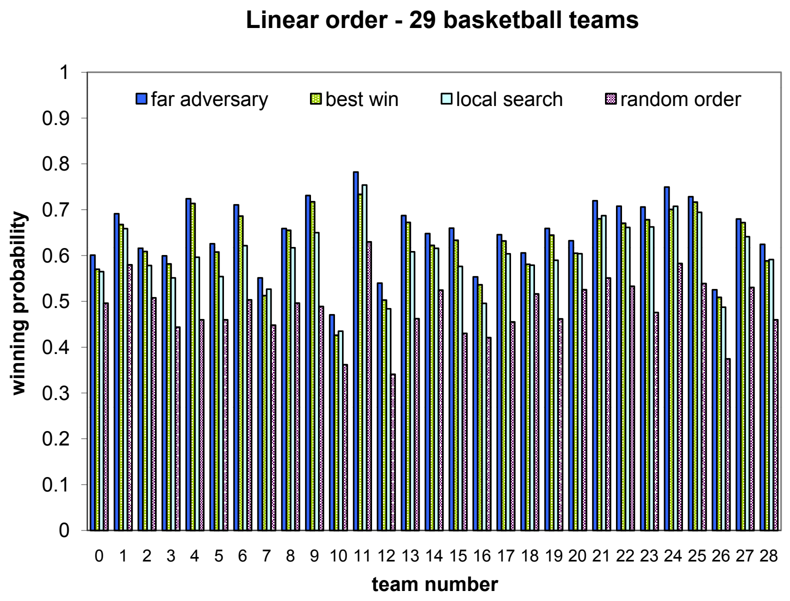


Figure 5.3: Performance of heuristics for linear order for real-world data from the domain of professional tennis and basketball.

search), while for a normal distribution they are slightly differentiated, but again performed better than local search. Both heuristics reached a very high winning probability, almost 0.9, and they performed better as the number of candidates increased. These heuristics also performed well in comparison to the optimal solution – they gave a winning probability which on average was only 98% from the optimal solution. Note that both of the "convert" heuristics (simple convert, threshold convert) performed very poorly, both for uniform and normal distributions, and thus we omitted them in subsequent experiments.

With the tennis players experiments (Figure 5.3, first graph), the x-axis is the player's number that was chosen to be the winning player and the y-axis as before. Here, there was no heuristic that performed significantly better than the others in general, but when choosing from the heuristics, the best solution for each player performed very well compared to the optimal solution. They gave a winning probability which was only 96% from the optimal solution on average, and the winning probability on average was more than twice as high as the random order. Player number 0, (Roger Federer, currently the world's number one player), even succeeded in obtaining a winning probability of 1 from local search.

We conclude that there is no one heuristic that performs significantly better than the others for all cases. We suggest the best thing to do here is to run all the heuristics and order the candidates according the heuristic which gives the best results for candidate c^* since they all run quite fast. Note that local search takes much more time than other heuristics, but still demonstrates acceptable time performance.

5.4.2 Heuristics for Fair Voting Tree

For this tree structure we investigated the following heuristics.

- *Optimal*, *Far adversary*, *Local search*: We organized the leaves of the balanced binary tree as a linear order from left to right and applied these heuristics as above. (Note that when the number of candidates is not a power of 2, some of the rightmost candidates may face one less ballot than the other candidates.)

- *Best win*: Because of the tree structure, we had to use a modified version of the previous best win heuristic, but the principle remained the same. We tried to maximize the probability that c^* would compete against candidates that he had a high probability of beating, and to maximize the probability that they would reach the point where they compete against him. We first assigned c^* at the rightmost leaf of the tree, and for each competition along its path to the root we assigned candidates that c^* had a high probability to of beating. In this manner, for each candidate we defined a sub-tree for which we wanted the candidate to be its overall winner (unless the candidate had been assigned to a leaf) so we could repeat this assignment procedure recursively.

Perhaps it will be easiest to understand this heuristics by a pseudo-code description. The first call to this algorithm is BestWin(the entire tree, c^*).

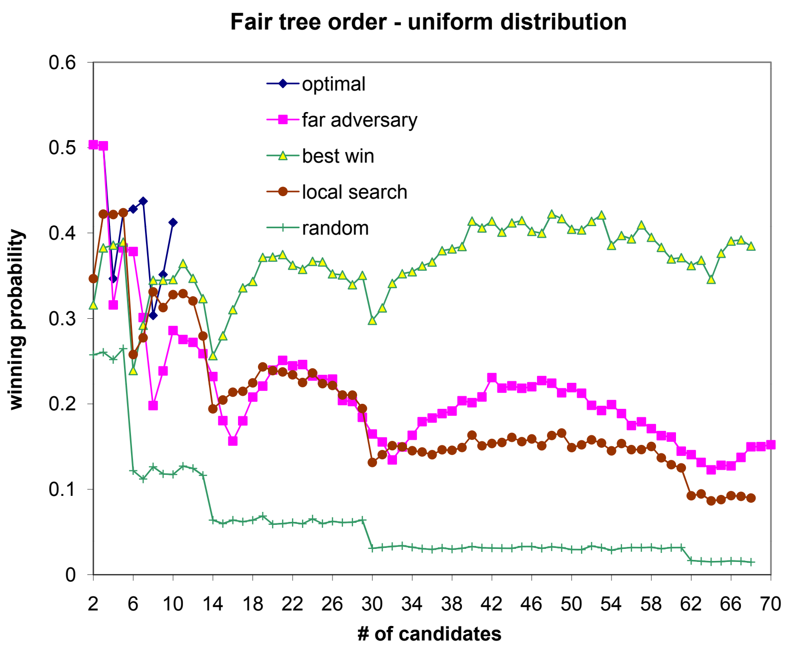
Algorithm 2 BestWin(Sub tree T, winner candidate c^*)

```
1: if height of $T$ is 0 then
2:     put $c^*$ in the root of $T$
3:     return
4: else
5:     put $c^*$ in the rightmost leaf of $T$
6: for $i \leftarrow 1$ to the height of $T$ do
7:     $temp \leftarrow$ a candidate that $c^*$ has the best probability to win and was not
       chosen yet
8:     push $temp$ into queue
9: for $i \leftarrow 1$ to the height of $T$ do
10:    pop $temp$ from queue
11:    $T' \leftarrow$ rightmost subtree of $T$, with height $i-1$ and unoccupied leaves
12:    BestWin($T', temp$)
```

- *Random order*: As a control, a random agenda was also generated.

The results of our experiments are shown in 5.4 and 5.5.

Generally, one can note that the overall wining probability is lower than in the linear order structure, which seems to be a direct consequence of the relative fairness of the procedure. Nevertheless, if we compare the best heuristic for each case to the random order, we get a winning probability which is on average 4.31 times higher than the random order winning probability and which is on average only 96% from the optimal solution, with the randomly generated data. We also note that with this randomly generated data the



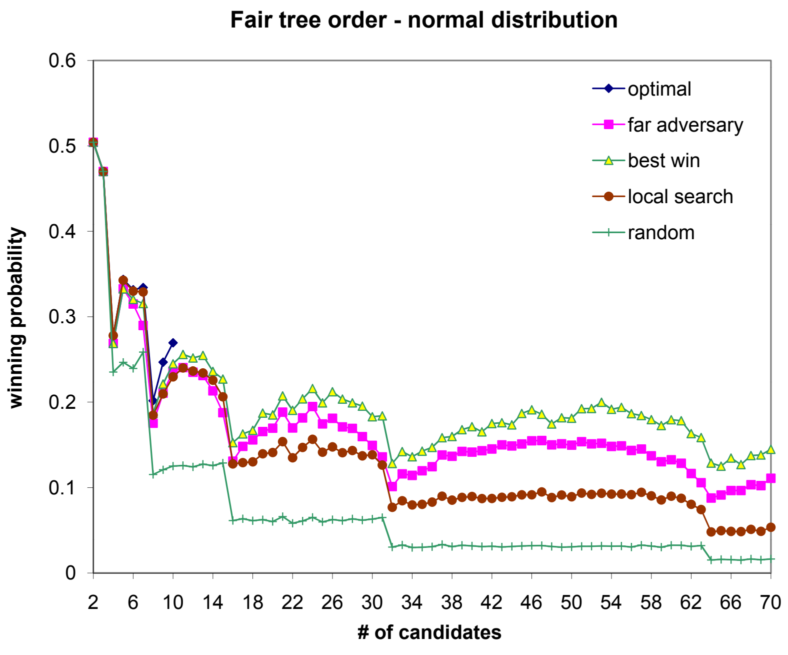
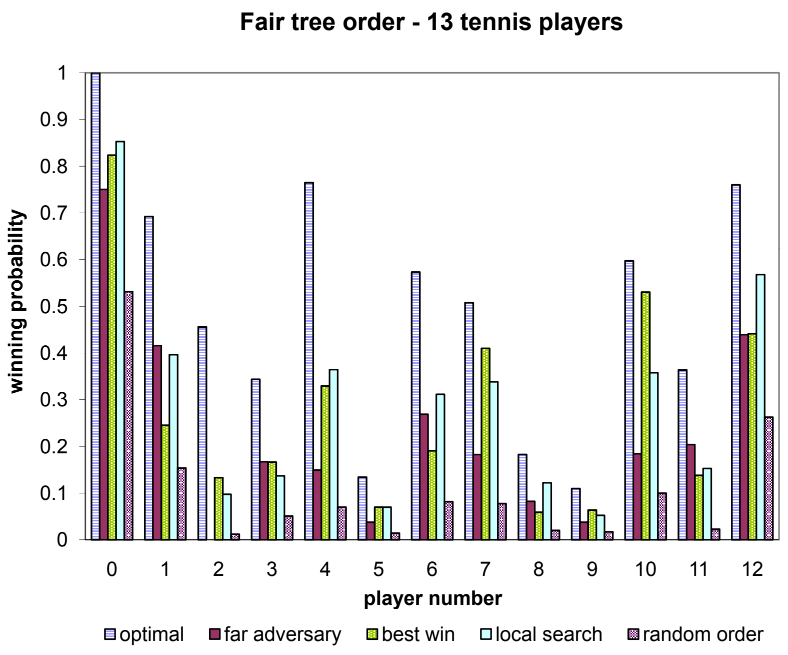


Figure 5.4: Performance of heuristics for fair tree order for randomly generated probability matrices using uniform and normal probability distributions.



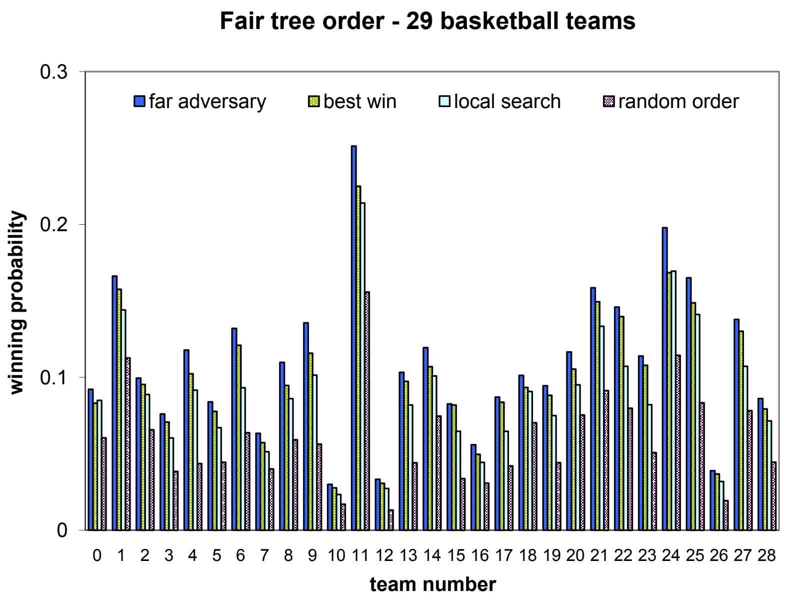


Figure 5.5: Performance of heuristics for fair tree order for real-world data from the domain of professional tennis and basketball.

graphs have a wave-like shape: the lowest points in these waves are when the number of candidates is exactly a power of 2 at which time the tree is perfectly balanced. This is simply because when the number of candidates is exactly a power of 2, every candidate must compete in exactly the same number of ballots. In all other cases, there are some candidates (including our favorite, c^*) that face one less ballot. This effect can help c^*, by forcing strong competitors to undergo one more ballot. It is also apparent that in contrast to the linear order with random normal distribution, the best win heuristic performs better than the others: 1.25 times better than far adversary, and 1.66 times better than local search on average.

With the NBA basketball teams experiments, (Figure 5.5, second graph), the far adversary method was the winning heuristic. It performed better than the others with a winning probability, which, on average was 1.24 times better than local search and 1.08 times better than best win. The highest winning probability was generated for team 11, the LA Lakers. It was not computationally feasible to calculate the optimal solution for 29 teams, so we ran another scenario with only the first 13 teams to check the performance of our heuristics against the optimal solution. The best heuristic in each case gave a winning probability which on average was only 99% of the optimal solution!

In the tennis players scenario ((Figure 5.5(b), first graph) there was no heuristic that performed significantly better than the others. But, when we chose the best solution from the heuristics for each case the winning probability on average was 61% of the optimal solution. Perhaps the performance difference between this case and the other cases was a result of the type of the distribution, since the probability matrix in the tennis case contains many high probabilities. Another indicator is the performance of our best heuristic in each case in comparison to the random order. In the tennis scenario it performed almost 5 times better, while in the basketball scenario it performed only about 1.5 times better.

Chapter 6

Complexity of Safe Strategic Voting

In this chapter we focus on algorithmic complexity of safe manipulation, as defined by Slinko and White [101]. We first define our problem and formalize the relevant computational questions (Section 6.1). We then study the complexity of these questions for several classic voting rules, for both weighted and unweighted voters. In Section 6.2 we analyze Plurality, Veto and k-approval voting rules, and in Section 6.3 we handle Bucklin and Borda voting rules. We then explore whether it is possible to extend the model of safe manipulation to settings where the manipulator may be joined by voters whose preferences differ from his own. In Section 6.4 we propose two ways of formalizing this idea, which differ in their approach to defining the set of a voter's potential followers, and provide initial results on the complexity of safe manipulation in these models.

6.1 Problem Definition and Computational Problems: First Observations

In our work we analyze the algorithmic complexity of safe manipulation, as defined by Slinko and White in [101]. For the purposes of our presentation, we can simplify their definitions considerably. The details follow. Recall that the voters' true preferences are given by a preference profile $\mathcal{R} = (R_1, \ldots, R_n)$.

Definition 6.1. We say that a vote L is an *incentive to vote strategically*, or a *strategic vote* for i at $\mathcal{R}$ under $\mathcal{F}$, if $L \neq R_i$ and for some $U \subseteq V_i$ we have $\mathcal{F}(\mathcal{R}_{-U}(L)) \succ_i \mathcal{F}(\mathcal{R})$. Further, we say that L is a *safe strategic vote* for a voter i at $\mathcal{R}$ under $\mathcal{F}$ if L is a strategic vote at $\mathcal{R}$, and for any $U \subseteq V_i$ either $\mathcal{F}(\mathcal{R}_{-U}(L)) \succ_i \mathcal{F}(\mathcal{R})$ or $\mathcal{F}(\mathcal{R}_{-U}(L)) = \mathcal{F}(\mathcal{R})$.

Unless specified otherwise, in this part of our work we assume that the tie-breaking rule is *lexicographic*, i.e., given a set of tied alternatives, it selects one that is maximal with respect to a fixed ordering $\succ$. To build intuition for our notions, consider the following example.

Example 6.2. *Suppose $C = \{a, b, c, d\}$, $V = \{1, 2, 3, 4\}$, the first three voters have preference $b \succ a \succ c \succ d$, and the last voter has preference $c \succ d \succ a \succ b$. Suppose also that the voting rule is 2-approval. Under truthful voting, a and b get 3 points, and c and d get 1 point each. Since ties are broken lexicographically, a wins. Now, if voter 1 changes his vote to $L = b \succ c \succ a \succ d$, b gets 3 points, a gets 2 points, and c gets 2 points, so b wins. As $b \succ_1 a$, L is a strategic vote for 1. However, it is not a safe strategic vote: if players in $V_1 = \{1, 2, 3\}$ all switch to voting L, then c gets 4 points, while b still gets 3 points, so in this case c wins and $a \succ_1 c$.*

The definition of safe strategic voting gives rise to two natural algorithmic questions. In the definitions below, $\mathcal{F}$ is a given voting rule and the voters are assumed to be unweighted.

- IsSafe($\mathcal{F}$): Given a voting domain, a voter i and a linear order L, is L a safe strategic vote for i under $\mathcal{F}$?
- ExistSafe($\mathcal{F}$): Given a voting domain and a voter i, can voter i make a safe strategic vote under $\mathcal{F}$?

The variants of these problems for weighted voters will be denoted, respectively, by wIsSafe($\mathcal{F}$) and wExistSafe($\mathcal{F}$). Note that, in general, it is not clear if an efficient algorithm for (w)ExistSafe($\mathcal{F}$) can be used to solve (w)IsSafe($\mathcal{F}$), or vice versa. However, if the number of candidates is constant, (w)ExistSafe($\mathcal{F}$) reduces to (w)IsSafe($\mathcal{F}$). We formulate the following two results for weighted voters; clearly, they also apply to unweighted voters.

Proposition 6.3. *Consider any voting rule $\mathcal{F}$. For any constant k, if $|C| \leq k$, then a polynomial-time algorithm for* wIsSafe($\mathcal{F}$) *can be used to solve* wExistSafe($\mathcal{F}$) *in polynomial time.*

Proof. In this case i has at most $k! = O(1)$ different votes, so he can try all of them. $\square$

A similar reduction exists when each voter only has polynomially many "essentially different" votes.

Proposition 6.4. *Consider any scoring rule $\mathcal{F}_\alpha$ that satisfies either (i) $\alpha_j = 0$ for all $j > k$ or (ii) $\alpha_j = 1$ for all $j \le m - k$, where k is a given constant. For any such rule, a polynomial-time algorithm for* wIsSafe($\mathcal{F}_\alpha$) *can be used to solve* wExistSafe($\mathcal{F}_\alpha$) *is polynomial time.*

Proof. We consider case (i); case (ii) is similar. There are at most $n^k = \mathrm{poly}(n)$ different ways to fill the top k positions in a vote. Further, if two votes only differ in positions $k+1, \dots, m$, they result in the same outcome. Thus, to solve wExistSafe($\mathcal{F}_\alpha$), it suffices to run wIsSafe($\mathcal{F}_\alpha$) on $\mathrm{poly}(n)$ instances. $\square$

Observe that the class of rules considered in Proposition 6.4 includes Plurality and Veto, as well as k-approval and k-veto when k is bounded by a constant.

Further, we note that for unweighted voters it is easy to check if a given manipulation is safe.

Proposition 6.5. *The problem* IsSafe($\mathcal{F}$) *is in* P *for any voting rule $\mathcal{F}$.*

Proof. Set $V_i = \{i_1, \dots, i_s\}$. Since our voting rule is anonymous, it suffices to check the conditions of Definition 6.1 for $U \in \{\{i_1\}, \{i_1, i_2\}, \dots, \{i_1, \dots, i_s\}\}$, i.e., for $s \le n$ sets of voters. $\square$

Together with Propositions 6.3 and 6.4, Proposition 6.5 implies that the problem ExistSafe($\mathcal{F}$) is in P for Plurality, Veto, k-veto and k-approval for constant k, as well as for any voting rule with a constant number of candidates.

Note that when voters are weighted, the conclusion of Proposition 6.5 no longer holds. Indeed, in this case the number of subsets of V_i that have different weights (and thus may have a different effect on the outcome) may be exponential in n. However, the problem remains easy when all weights are small (polynomially bounded).

Proposition 6.6. *For any voting rule $\mathcal{F}$,* wIsSafe($\mathcal{F}$) *can be solved in time* $\mathrm{poly}(n, w_{\max})$*, where* $w_{\max} = \max_{i=1,\dots,n} w_i$.

Proof. Let i be the manipulating voter; we have $w(V_i) \leq nw_{\max}$. Given a vote L, we use the standard dynamic programming algorithm to check for all $w = 1, \ldots, nw_{\max}$ if there exists a $U_w \subseteq V_i$ with $w(U_w) = w$. For each such w, we compare $\mathcal{F}(\mathcal{R}_{-U_w}(L))$ and $\mathcal{F}(\mathcal{R})$; clearly, the outcome is independent of the choice of U_w. Thus, we can check if L satisfies the conditions of Definition 6.1 in time $\text{poly}(n, w_{\max})$. □

As in the case of unweighted voters, Proposition 6.6 leads to a pseudopolynomial algorithm for wExistSafe($\mathcal{F}$) for a constant number of candidates, as well as for Veto and Plurality.

Corollary 6.7. *For any voting rule $\mathcal{F}$ with at most k candidates, as well as for any scoring rule that satisfies the conditions of Proposition 6.4,* wExistSafe($\mathcal{F}$) *can be solved in time* $\text{poly}(n^k, w_{\max})$.

6.2 Plurality, Veto, and k-approval

We will now show that the easiness results for k-approval and k-veto extend to arbitrary $k \leq m$ and weighted voters (note that the distinction between k-veto and $(m-k)$-approval only matters for constant k).

Theorem 6.8. *For k-approval, the problems* wIsSafe *and* wExistSafe *are in* P.

Proof. Fix a voter $v \in V$. To simplify notation, we renumber the candidates so that v's preference order is given by $c_1 \succ_v \ldots \succ_v c_m$. Denote v's truthful vote by R. Recall that V_v denotes the set of voters who have the same preferences as v. We call a candidate the *maximal manipulation winner* for vote L, if it wins when all the voters from V_v choose to vote L. Now suppose that under truthful voting the winner is c_j. For $i = 1, \ldots, m$, let $s_i(\mathcal{R}')$ denote the k-approval score of c_i given a profile $\mathcal{R}'$, and set $s_i = s_i(\mathcal{R})$.

We start by proving a useful characterization of safe strategic votes for k-approval.

Lemma 6.9. *A vote L is a safe strategic vote for v if and only if the winner in $\mathcal{R}_{-V_v}(L)$ is a candidate c_i with $i < j$.*

Proof. Suppose that L is a safe strategic vote for v. Then there exists an $i < j$ and a $U \subseteq V_v$ such that the winner in $\mathcal{R}_{-U}(L)$ is c_i. It must be the case that each switch from R to L increases c_i's score or decreases c_j's score:

otherwise c_i cannot beat c_j after the voters in U change their vote from R to L. Therefore, if c_i beats c_j when the preference profile is $\mathcal{R}_{-U}(L)$, it continues to beat c_j after the remaining voters in V_v switch, i.e., when the preference profile is $\mathcal{R}_{-V_v}(L)$. Hence, the winner in $\mathcal{R}_{-V_v}(L)$ is not c_j; since L is safe, this means that the winner in $\mathcal{R}_{-V_v}(L)$ is c_ℓ for some $\ell < j$.

For the opposite direction, suppose that the winner in $\mathcal{R}_{-V_v}(L)$ is c_i for some $i < j$. Note that if two candidates gain points when some subset of voters switches from R to L, they both gain the same number of points; the same holds if both of them lose points.

Now, if $j > k$, a switch from R to L does not lower the score of c_j, so it must increase the score of c_i for it to be the maximal manipulation winner. Further, if a switch from R to L grants points to some $c_\ell \neq c_i$, then either $s_\ell < s_i$ or $s_\ell = s_i$ and the tie-breaking rule favors c_i over c_ℓ: otherwise, c_i would not be the maximal manipulation winner.

Similarly, if $j \leq k$, a switch from R to L does not increase the score of c_i, so it must lower the score of c_j. Further, if some $c_\ell \neq c_i$ does not lose points from a switch from R to L, then either $s_\ell < s_i$ or $s_\ell = s_i$ and the tie-breaking rule favors c_i over c_ℓ: otherwise, c_i would not be the maximal manipulation winner.

Now, consider any $U \subseteq V_v$. If $s_j(\mathcal{R}_{-U}(L)) > s_i(\mathcal{R}_{-U}(L))$, then c_j is the winner. If $s_i(\mathcal{R}_{-U}(L)) > s_j(\mathcal{R}_{-U}(L))$, then c_i is the winner. Finally, suppose $s_i(\mathcal{R}_{-U}(L)) = s_j(\mathcal{R}_{-U}(L))$. By the argument above, no other candidate can have a higher score. So, suppose that $s_\ell(\mathcal{R}_{-U}(L)) = s_i(\mathcal{R}_{-U}(L))$, and the tie-breaking rule favors c_ℓ over c_i and c_j. Then this would imply that c_ℓ wins in $\mathcal{R}$ or $\mathcal{R}_{-V_v}(L)$ (depending on whether a switch from R to L causes c_ℓ to lose points), a contradiction. Thus, in this case, too, either c_i or c_j wins. □

Lemma 6.9 immediately implies an algorithm for wIsSafe: we simply need to check that the input vote satisfies the conditions of the lemma. We now show how to use it to construct an algorithm for wExistSafe. We need to consider two cases.

$\mathbf{j > k}$:
In this case, the voters in V_v already do not approve of c_j and approve of all c_i, $i \leq k$. Thus, no matter how they vote, they cannot ensure that some c_i, $i \leq k$, gets more points than c_j. Hence, the only way they can change the outcome is by approving of some candidate c_i, $k < i < j$. Further, they can only succeed if there exists an $i = k+1, \dots, j-1$ such that either $s_i + w(V_v) > s_j$ or $s_i + w(V_v) = s_j$ and the tie-breaking rule favors c_i over

c_j. If such an i exists, v has an incentive to manipulate by swapping c_1 and c_i in his vote. Furthermore, it is easy to see that any such manipulation is safe, as it only affects the scores of c_1 and c_i.

$\mathbf{j \leq k}$:
In this case, the voters in V_v already approve of all candidates they prefer to c_j, and therefore they cannot increase the scores of the first $j-1$ candidates. Thus, their only option is to try to lower the scores of c_j as well as those of all other candidates whose score currently matches or exceeds the best score among $s_1, \ldots, s_{j-1}$.

Set $C_g = \{c_1, \ldots, c_{j-1}\}$, $C_b = \{c_j, \ldots, c_m\}$. Let C_0 be the set of all candidates in C_g whose k-approval score is maximal, and let $s_{\max}$ be the k-approval score of the candidates in C_0. For any $c_\ell \in C_b$, let s'_ℓ denote the number of points that c_ℓ gets from all voters in $V \setminus V_v$; we have $s'_\ell = s_\ell$ for $k < \ell \leq m$ and $s'_\ell = s_\ell - w(V_v)$ for $\ell = j, \ldots, k$. Now, it is easy to see that v has a safe manipulation if and only if the following conditions hold:

- For all $c_\ell \in C_b$ either $s'_\ell < s_{\max}$, or $s'_\ell = s_{\max}$ and there exists a candidate $c \in C_0$ such that the tie-breaking rule favors c over c_ℓ;
- There exist a set $C_{\text{safe}} \subseteq C_b$, $|C_{\text{safe}}| = k - j + 1$, such that for all $c_\ell \in C_{\text{safe}}$ either $s'_\ell + w(V_v) < s_{\max}$ or $s'_\ell + w(V_v) = s_{\max}$ and there exists a candidate $c \in C_0$ such that the tie-breaking rule favors c over c_ℓ.

Note that these conditions can be easily checked in polynomial time by computing s_ℓ and s'_ℓ for all $\ell = 1, \ldots, m$.

Indeed, if such a set C_{safe} exists, voter v can place the candidates in C_{safe} in positions $j, \ldots, k$ in his vote; denote the resulting vote by L. Clearly, if all voters in V_v vote according to L, they succeed to elect some $c \in C_0$. Thus, by Lemma 6.9, L is safe. Conversely, if a set C_{safe} with these properties does not exist, then for any vote $L \neq R$ the winner in $\mathcal{R}_{-V_v}(L)$ is a candidate in C_b, and thus by Lemma 1 L is not safe. $\square$

We remark that Theorem 6.8 crucially relies on the fact that we break ties based on a fixed priority ordering over the candidates. Indeed, it can be shown that there exists a (non-lexicographic) tie-breaking rule such that finding a safe vote with respect to k-approval combined with this tie-breaking rule is computationally hard (assuming k is viewed as a part of the input). As the focus of this part of our work is on lexicographic tie-breaking, we omit the formal statement and the proof of this fact.

In contrast, we can show that any scoring rule with 3 candidates is easy to manipulate safely, even if the voters are weighted and arbitrary tie-breaking rules are allowed.

Theorem 6.10. wIsSafe($\mathcal{F}$) *is in* P *for any voting rule* $\mathcal{F}$ *obtained by combining a positional scoring rule with at most three candidates with an arbitrary tie-breaking rule.*

Proof. For one candidate, the statement is trivial. With two candidates, every positional scoring rule is equivalent to Plurality, and under Plurality with two candidates no voter has an incentive to vote strategically.

Now, suppose that $|C| = 3$. Consider a voter i and assume without loss of generality that $R_i = (c_1, c_2, c_3)$. If $\mathcal{F}(\mathcal{R}) = c_1$, then i has no incentive to vote strategically. We will now consider the cases $\mathcal{F}(\mathcal{R}) = c_2$ and $\mathcal{F}(\mathcal{R}) = c_3$ separately.

1. $\mathcal{F}(\mathcal{R}) = c_2$. Suppose that L is a strategic vote for i. Then L cannot rank c_2 in top two positions. Indeed, any such manipulation does not decrease c_2's score and does not increase c_1's score. Thus, if c_2 had a higher score than c_1, this would still be the case no matter how many voters in V_i switch to voting L. Further, if both c_2 and c_1 had top scores, then L could succeed only if it does not change the scores of either of them. But in this case the score of c_3 does not change either, so the outcome remains the same. Thus, it remains to consider two cases: $L = (c_1, c_3, c_2)$ and $L = (c_3, c_1, c_2)$. Now, let $c = \mathcal{F}(\mathcal{R}_{-V_i}(L))$. If $c = c_3$ (i.e., c_3 is the maximal manipulation winner), L is not safe. Further, if $c = c_2$, then we have $c_2 = \mathcal{F}(\mathcal{R}_{-U}(L))$ for any $U \subseteq V_i$, i.e., L is not a strategic vote for i. Finally, if $c = c_1$, then L is a safe strategic vote. Indeed, suppose that L is not safe, i.e., $\mathcal{F}(\mathcal{R}_{-U}(L)) = c_3$ for some $U \subset V_i$. Each switch from R_i to L does not decrease c_3's score, so in that case c_3 would be a full manipulation winner.

2. $\mathcal{F}(\mathcal{R}) = c_3$. It can be checked that if L is a strategic vote for i, then L has to rank c_2 first, i.e., $L \in \{(c_2, c_1, c_3), (c_2, c_3, c_1)\}$. If $\mathcal{F}(\mathcal{R}_{-V_i}(L)) = c_3$, by the same argument as above, there is no incentive for i to vote for L. Otherwise, L is a safe strategic vote, since c_3 is the least preferred candidate.

□

6.3 Bucklin and Borda

Bucklin rule is quite similar to k-approval, so we can use the ideas in the proof of Theorem 6.8 to design a polynomial-time algorithm for finding a safe manipulation with respect to Bucklin. However, the proof becomes significantly more complicated.

Theorem 6.11. *For the Bucklin rule,* wExistSafe *is in* P.

Proof. As in the proof of Theorem 6.8, we fix a voter $v \in V$ and renumber the candidates so that v's preference order is given by $c_1 \succ_v \ldots \succ_v c_m$. Denote v's truthful vote by R. Let V_v denote the set of voters who have the same preferences as v. Suppose that under truthful voting the Bucklin winner is c_j, and the winning round is k.

We have to consider two possibilities.

j > k:
In this case, no matter how the voters in V_v vote, c_j's k-approval score will be at least $\lceil n/2 \rceil$. Thus, the only part of v's vote that can affect the final outcome is the top k positions. Now, no matter how the voters in V_v vote, none of the candidates currently ranked in the top k positions by v can beat c_j. Thus, the only way v can succeed is by ranking some candidate c_i, $k < i < j$, in the top position. This will work if it makes c_i's k'-approval score become at least $\lceil n/2 \rceil$ for some $k' < k$, or if it makes c_i a k-approval winner (under the given tie-breaking rule). To check if such a c_i exists, v can try to swap c_1 and c_i in his vote for all $i = k+1, \ldots, j-1$. Moreover, any such manipulation is safe, as it only affects the scores of c_1 and c_i.

j ≤ k:
Set $C_g = \{c_1, \ldots, c_{j-1}\}$, $C_b = \{c_j, \ldots, c_m\}$. Let k' be the smallest value of ℓ such that under truthful voting some $c \in C_g$ gets at least $\lceil n/2 \rceil$ votes in round ℓ; clearly, we have $k' \geq k$. Let C_0 be the set of all candidates in C_g whose k'-approval score is maximal (and hence is at least $\lceil n/2 \rceil$), and let $s_{\max}$ be the k'-approval score of the candidates in C_0. Let $\widehat{c}$ be the candidate in C_0 that is most favored by the tie-breaking rule. Throughout the proof, for any $c, c' \in C$ and any $\ell \leq m$, we will say that c beats c' under ℓ-approval if c's ℓ-approval score is no lower than that of c', and if they are equal, then our tie-breaking rule favors c over c'. Further, we say that c beats c' if there exist some $\ell, \ell' \leq m$, such that c gets at least $\lceil n/2 \rceil$ votes under ℓ-approval, but not under $(\ell-1)$-approval, c' gets at least $\lceil n/2 \rceil$ votes under ℓ'-approval,

but not under $(\ell' - 1)$-approval, and either $\ell < \ell'$ or $\ell = \ell'$ and c beats c' under ℓ-approval.

No matter how the voters in V_v vote, they cannot ensure that a candidate in C_g gets at least $\lceil n/2 \rceil$ votes in round ℓ for $\ell < k'$. Therefore, to succeed, they need to ensure that no candidate in C_b gets at least $\lceil n/2 \rceil$ ℓ-approval votes for $\ell < k'$, and that no candidate in C_b beats all candidates in C_g under k'-approval.

Hence, v has an incentive to manipulate if and only if there is a vote L such that if some voters in V_v switch from R to L, then for any $c \in C_b$

(a) c's $(k' - 1)$-approval score is less than $\lceil n/2 \rceil$;

(b) c's k'-approval is at most $s_{\max}$, and if it is equal to $s_{\max}$, then the tie-breaking rule favors $\widehat{c}$ over c.

We will first prove that essentially any safe vote L makes c_j lose when all voters in V_v switch from R to L.

Lemma 6.12. *If there exists a safe vote for v, then there exists a safe vote for v that ranks all candidates in C_g in top $j-1$ positions. Moreover, for any safe vote L that ranks all candidates in C_g in top $j-1$ positions, it cannot be the case that c_j wins if all voters in V_v vote according to L.*

Proof. To prove the first part, note that if we are given a safe vote that does not rank some $c \in C_g$ in top $j-1$ positions, then we can swap the candidate from C_b that appears in top $j-1$ positions with c, and the resulting vote would still be safe. We can repeat this step until all candidates ranked in top $j-1$ positions belong to C_g.

For the second part, observe that L cannot rank c_j in position $k'-1$ or higher, since otherwise c_j gets at least $\lceil n/2 \rceil$ $(k'-1)$-approval votes no matter how many voters in V_v switch from R to L. Now, suppose that after some voters $U \subseteq V_v$ switch from R to L, c_j stops being the Bucklin winner, and some $c \in C_g$ becomes the Bucklin winner. Note that since we assume that c is ranked in top $j-1$ positions in L, the corresponding winning round is k'.

At this point, c_j's $(k'-1)$-approval score is less than $\lceil n/2 \rceil$. Therefore, as the voters in $V_v \setminus U$ switch from R to L, c_j's $(k'-1)$-approval score remains less than $\lceil n/2 \rceil$. Further, each switch from R to L does not increase c_j's k'-approval score, and, as argued above, c's k'-approval score remains the same after each switch. Thus, c_j' continues to lose to c under k'-approval after all voters in V_v switch from R to L. Therefore, c_j cannot be the Bucklin winner when all voters in V_v vote according to L. □

We will now construct a family of votes for v as follows. For $\ell = j, \dots, m$, let us say that a position $p \in \{j, \dots, m\}$ is *safe* for c_ℓ if c_ℓ satisfies conditions (a) and (b) above whenever all voters in V_v rank c_ℓ in position p. Let $P(\ell)$ denote the set of positions that are safe for c_ℓ. The sets $P(\ell)$ can be computed independently and efficiently for each $c_\ell \in C_b$. Consider now a bipartite graph G whose vertices are candidates in C_b and positions in $\{j, \dots, m\}$, and there is an edge from a candidate c_ℓ to a position i if and only if $i \in P(\ell)$. For any complete matching in this graph, we can construct a vote L in which all candidates in C_g are ranked in top $j-1$ positions and all candidates in C_b are ranked according to the matching. Denote the set of all such votes by $\mathcal{L}(G)$. Clearly, if all voters in V_v vote according to any $L \in \mathcal{L}(G)$, then some candidate in C_g wins. We will now prove two lemmas that characterize the relationship between the set $\mathcal{L}(G)$ and the set of all safe votes.

Lemma 6.13. *Any safe vote L that ranks all candidates in C_g in top $j-1$ positions ranks each $c_\ell \in C_b$ in a position in $P(\ell)$.*

Proof. Suppose that L ranks some $\ell \in C_b$ in a position that is not in $P(\ell)$. This means that no candidate in C_g can win when all voters in V_v vote according to L. Now, if in this situation some $c' \in C_b$, $c' \neq c_j$, wins, this means that L is unsafe. On the other hand, if c_j wins, then by Lemma 6.12 L is not safe. □

Thus, each safe vote can be transformed into a vote in $\mathcal{L}(G)$, i.e., if $\mathcal{L}(G) = \emptyset$, there are no safe votes for v.

Lemma 6.14. *If there exists a safe vote for v, then any vote in $\mathcal{L}(G)$ is safe.*

Proof. The proof proceeds by contraposition: we will argue that if some vote $L \in \mathcal{L}(G)$ is not safe, then no vote is safe for v.

Fix a vote $L \in \mathcal{L}(G)$, and denote by C_{safe} the set of candidates ranked in positions $j, \dots, k'$ in L. Suppose that initially all voters in V_v vote truthfully, and then they switch from R to L one by one. Since $L \in \mathcal{L}(G)$, after all voters switch, some $c \in C_g$ becomes the Bucklin winner, so at some point in this process c_j stops being the Bucklin winner. Suppose that this happens after some subset of voters $U \subseteq V_v$ switch. At this point, some other candidate c becomes the Bucklin winner; let k'' be the corresponding winning round.

Suppose first that $c \in C_g$; note that this implies $k'' = k'$. We claim that in this case L is safe. Indeed, this means that currently no candidate gets at least $\lceil n/2 \rceil$ votes in the first $k' - 1$ rounds, and all candidates in C_b lose to c under k'-approval. Now, for any $\ell \leq k'$, each switch from R to L can increase the ℓ-approval scores of the candidates in C_{safe} only. However, we know that even if all voters in V_v switch from R to L, the candidates in C_{safe} still do not win. Thus, after any number of additional switches from R to L, it is still the case that no candidate gets at least $\lceil n/2 \rceil$ votes in the first $k' - 1$ rounds, and all candidates in C_b lose to c under k'-approval. Thus, in this case L is safe.

Now, suppose that $c \in C_b$ (and hence L is not safe). We will now argue that in this case no vote is safe for v. Indeed, suppose that L' is a safe vote for v. By Lemma 6.12, we can assume that L' ranks the candidates in C_g in top $j - 1$ positions. We will now show that after the voters in U switch from R to L', c beats both c_j and all candidates in C_g. We split the rest of the proof of Lemma 6.14 into two lemmas.

Lemma 6.15. *In $\mathcal{R}_{-U}(L')$, c beats all candidates in C_g.*

Proof. Since c is the Bucklin winner in $\mathcal{R}_{-U}(L)$ its k''-approval score in $\mathcal{R}_{-U}(L)$ is at least $\lceil n/2 \rceil$.

We will now argue that each switch from R to L decreases c's k''-approval score. Indeed, otherwise in $\mathcal{R}_{-U}(L)$ candidate c's k''-approval score remains at least $\lceil n/2 \rceil$. If $k'' < k'$, this is a contradiction with $L \in \mathcal{L}(G)$. Now suppose that $k'' = k'$. Since each switch from R to L does not increase the k''-approval scores of all $c' \in C_g$, c continues to beat all candidates in C_g under k''-approval. Again, this is a contradiction with $L \in \mathcal{L}(G)$.

Thus, a switch from R to L' cannot lower c's k''-approval score more than a switch from R to L does. Hence, if $k'' < k'$, after the voters in U switch from R to L', c has at least $\lceil n/2 \rceil$ k'-approval votes, so no candidate in C_g can win in this case. Now suppose that $k' = k''$. Neither a switch from R to L nor a switch from R to L' changes the k'-approval scores on any $c' \in C_g$. Thus, for any $c' \in C_g$, if c beats c' under k'-approval in $\mathcal{R}_{-U}(L)$, he also beats c' under k'-approval in $\mathcal{R}_{-U}(L')$. Thus, in $\mathcal{R}_{-U}(L')$ no candidate in C_g can win the election. □

Lemma 6.16. *In $\mathcal{R}_{-U}(L')$, c beats c_j.*

Proof. As argued in Lemma 6.12, both L and L' rank c_j in position k' or lower. Since c beats c_j in $\mathcal{R}_{-U}(L)$, it follows that in $\mathcal{R}_{-U}(L)$, candidate c_j's

$(k''-1)$-approval score is less than $\lceil n/2 \rceil$. Now, since $k''-1 < k'$, a switch from R to L' has the same effect on c_j's $(k''-1)$-approval score as a switch from R to L. Thus, in $\mathcal{R}_{-U}(L')$ candidate c_j's $(k''-1)$-approval score is less than $\lceil n/2 \rceil$.

It remains to show that in $\mathcal{R}_{-U}(L')$ candidate c_j continues to lose to c under k''-approval. Suppose first that $k'' < k'$ or L' ranks c_j in position $k'+1$ or lower. When a voter u of weight w switches from R to L', c's k''-approval score decreases by at most w, and c_j's k''-approval score decreases by w. Further, when u switches from R to L, c's k''-approval score decreases by w. and c_j's k''-approval score decreases by at most w. Thus, relative to c_j, c does better under L' than under L. Therefore, in this case, in $\mathcal{R}_{-U}(L')$ c_j loses to c under k''-approval.

Finally, suppose that $k'' = k'$ and L' ranks c_j in position k'. Note that in this case c_j's k'-approval score is the same no matter how many voters in V_v switch from R to L'. Thus, by Lemma 6.12 c_j loses to $\hat{c}$ under k'-approval in $\mathcal{R}_{-U}(L')$. Furthermore, we have argued that in $\mathcal{R}_{-U}(L')$. all candidates in C_g lose to c under k'-approval. Thus, by transitivity, in $\mathcal{R}_{-U}(L')$ candidate c_j loses to c under k'-approval. □

This completes the proof of Lemma 6.14. □

Thus, we conclude that to check whether v has a safe vote (and to find one if it exists), it suffices to compute the set $\mathcal{L}(G)$, check if it is not empty, and, if it contains some $L \in \mathcal{L}(G)$, check if L is safe. Indeed, we have argued that if $\mathcal{L}(G) = \emptyset$ then no vote is safe, and if $\mathcal{L}(G) \neq \emptyset$ then v has a safe vote if and only if an arbitrary vote in $\mathcal{L}(G)$ is safe. □

Interestingly, despite the intuition that wIsSafe should be easier than wExistSafe, it turns out that wIsSafe for Bucklin is coNP-hard.

Theorem 6.17. *For the Bucklin rule,* wIsSafe *is* coNP*-hard, even for a constant number of candidates.*

Proof. We give a reduction from Subset Sum. Recall that an instance of Subset Sum is given by a set of positive integers $A = \{a_1, \dots, a_s\}$ and a positive integer K. It is a "yes"-instance if there is a subset of indices $I \subseteq \{1, \dots, s\}$ such that $\sum_{i \in I} a_i = K$ and a "no"-instance otherwise.

Given an instance (A, K) of Subset Sum with $|A| = s$ and $\sum_{i=1}^{s} a_i = S$, we construct an instance of wIsSafe as follows. Set $C = \{a, b, c, x, y, z, x', y', z'\}$, and let $V = \{v_1, \dots, v_s, u_1, u_2, u_3, u_4\}$. Table 6.1

shows the preferences and weights of each voter; observe that the total weight of all voters is $4S$. We ask if the vote $L = (a, c, b, x, y, z, x', y', z')$ is a safe

Voter	Preference order	Weight
v_i	$(x, y, z, a, b, c, x', y', z')$	a_i
u_1	$(a, c, b, x, y, z, x', y', z')$	$2S - K - 1$
u_2	$(x, c, b, a, y, z, x', y', z')$	1
u_3	$(y, z, b, a, c, x, x', y', z')$	K
u_4	$(x', y', z', a, b, c, x, y, z)$	S

Table 6.1: Instance of wIsSafe for the proof of Theorem 6.17.

strategic vote for v_1 under Bucklin; as we will see, the answer to this question does not depend on the tie-breaking rule.

If all voters vote sincerely, then b wins in round 3 with $2S$ points, and all other voters get less that $2S$ points in the first three rounds. Note also that the total weight of voters in $C \setminus V_{v_1}$ that rank a first is $2S - K - 1$, and the total weight of voters in $C \setminus V_{v_1}$ that rank c second is $2S - K$.

Suppose that a group of voters $U \subseteq V_{v_1}$ votes L. If $w(U) < K$, then b remains the winner, while if $w(U) > K$ then a becomes the winner, as it gets the majority of votes in the first round. Therefore, L is a strategic vote for v_1. However, if $w(U) = K$, a only gets $2S - 1$ points in any of the first three rounds, while c gets $2S$ points in the second round. Therefore, in this case c wins, i.e., L is not safe for v_1. Hence, L is a safe strategic vote for v_1 if and only if no subset of A sums to K. □

For Borda, unlike k-approval and Bucklin, both of our problems are hard when the voters are weighted. The proof of the following theorem is very similar to that of Theorem 6.17.

Theorem 6.18. *For the Borda rule,* wIsSafe *and* wExistSafe *are* coNP*-hard. The hardness result holds even if there are only* 5 *candidates.*

Proof. We give a reduction from Subset Sum (see the proof of Theorem 6.17). Given an instance (A, K) of Subset Sum with $|A| = s$, $\sum_{i=1}^{s} a_i = S$, we construct an instance of wIsSafe as follows. Set $C = \{c_1, \dots, c_5\}$, and let $V = \{v_1, \dots, v_s, u_1, u_2, u_3, u_4\}$. Table 6.2 shows the preferences and weights of each voter. We ask if the vote $L = (c_1, c_5, c_4, c_2, c_3)$ is a safe

Voter	Preference order	Weight
v_i	$(c_5, c_1, c_2, c_4, c_3)$	a_i
u_1	$(c_1, c_2, c_3, c_4, c_5)$	S
u_2	$(c_3, c_2, c_1, c_5, c_4)$	S
u_3	$(c_3, c_4, c_2, c_1, c_5)$	S
u_4	$(c_4, c_5, c_2, c_3, c_1)$	K

Table 6.2: Instance of wIsSafe for the proof of Theorem 6.18.

strategic vote for v_1 under Borda where the ties are broken according to the preference ordering $c_3 \succ c_4 \succ c_2 \succ c_1 \succ c_5$ (i.e., is adversarially to v_1).

If all voters vote sincerely, then c_2 is the winner with $10S + 2K$ points, c_1 gets $10S$ points, c_3 gets $10S + K$ points, c_4 gets $5S + 4K$ points, and c_5 gets $5S + 3K$ points. Suppose that a group of voters $U \subseteq V_{v_1}$ votes L. If $w(U) < K$, then c_2 remains the winner, while if $w(U) > K$ then c_1 becomes the winner. Therefore, L is a strategic vote for v_1. If $w(U) = K$ then c_3 beats c_1 and c_2 in a three-way tie. Therefore, L is a safe strategic vote for v_1 if and only if no subset of A sums to K.

The hardness of wExistSafe follows from the observation that L is the only strategic vote available to v_1. □

6.4 Extensions of the Safe Strategic Voting Model

So far, we followed Slinko and White's model [101] and assumed that the only voters who may change their votes are the ones whose preferences exactly coincide with those of the manipulator. Clearly, in real life this assumption does not always hold. Indeed, a voter may follow a suggestion to vote in a certain way as long as it comes from someone he trusts (e.g., a well-respected public figure), and this does not require that this person's preferences are completely identical to those of the voter. For example, if both the original manipulator v and his would-be follower u rank the current winner last, it is easy to see that following v's recommendation that leads to displacing the current winner is in u's best interests.

In this section, we will consider two approaches to extending the notion of safe strategic voting to scenarios where not all manipulators have identical preferences. In both cases, we define the set of potential followers for each voter (in our second model, this set may depend on the vote suggested), and define a vote L to be safe if, whenever a subset of potential followers votes L, the outcome of the election does not get worse (and sometimes get better) from the manipulator's perspective. However, our two models differ in the criteria they use to identify a voter's potential followers.

Preference-Based Extension Our first model identifies the followers of a given voter based on the similarities in voters' preferences.

Fix a preference profile $\mathcal{R}$ and a voting rule $\mathcal{F}$, and let c be the winner under truthful voting. For any $v \in V$, let $I(v, c)$ denote the set of candidates that v ranks strictly above c. We say that two voters u and v are *similar* if $I(u, c) = I(v, c)$. A *similar set* S_v of a voter v for a given preference profile $\mathcal{R}$ and a voting rule $\mathcal{F}$ is given by $S_v = \{u \mid I(u, c) = I(v, c)\}$. (The set S_v depends on $\mathcal{R}$ and $\mathcal{F}$; however, for readability we omit $\mathcal{R}$ and $\mathcal{F}$ from the notation).

Note that if u and v are similar, they rank c in the same position. Further, a change of outcome from c to another alternative is positive from u's perspective if and only if it is positive from v's perspective. Thus, intuitively, any manipulation that is profitable for u is also profitable for v. Observe also that similarity is an equivalence relation, and the sets S_v are the corresponding equivalence classes. In particular, this implies that for any $u, v \in V$ either $S_u = S_v$ or $S_u \cap S_v = \emptyset$.

We can now adapt Definition 6.1 to our setting by replacing V_v with S_v.

Definition 6.19. *A vote L is a* strategic vote in the preference-based extension *for v at $\mathcal{R}$ under $\mathcal{F}$ if for some $U \subseteq S_v$ we have $\mathcal{F}(\mathcal{R}_{-U}(L)) \succ_v \mathcal{F}(\mathcal{R})$. Further, we say that L is a* safe strategic vote in the preference-based extension *for a voter v at $\mathcal{R}$ under $\mathcal{F}$ if L is a strategic vote at $\mathcal{R}$ under $\mathcal{F}$, and for any $U \subseteq S_v$ either $\mathcal{F}(\mathcal{R}_{-U}(L)) \succ_v \mathcal{F}(\mathcal{R})$ or $\mathcal{F}(\mathcal{R}_{-U}(L)) = \mathcal{F}(\mathcal{R})$.*

Observe that if L is a (safe) strategic vote for v at $\mathcal{R}$ under $\mathcal{F}$, then it is also a (safe) strategic vote for any $u \in S_v$. Indeed, $u \in S_v$ implies $S_u = S_v$ and for any $a \in C$ we have $a \succ_u \mathcal{F}(\mathcal{R})$ if and only if $a \succ_v \mathcal{F}(\mathcal{R})$. Note also that we do not require $L \neq R_v$: indeed, in the preference-based extension $L = R_v$ may be a non-trivial manipulation, as it may induce voters in $S_v \setminus \{v\}$ to switch their preferences to R_v. That is, a voter may manipulate the election simply by asking other voters with similar preferences to vote like he does. Finally, it is easy to see that for any voter v, the set S_v of similar voters is easy to compute.

The two computational problems considered throughout this work, i.e., the safety of a given manipulation and the existence of a safe manipulation remain relevant for the preference-based model. We will refer to these problems in this setting as IsSafe^{pr} and ExistSafe^{pr}, respectively, and use prefix w to denote their weighted variants. The problems (w)IsSafe^{pr} and (w)ExistSafe^{pr} appear to be somewhat harder than their counterparts in the original model. Indeed, while voters in S_v have similar preferences, their truthful votes may be substantially different, so it now matters *which* of the voters in S_v decide to follow the manipulator (rather than just *how many* of them, as in the original model). In particular, it is not clear if IsSafe$^{pr}(\mathcal{F})$ is polynomial-time solvable for any voting rule $\mathcal{F}$. However, it turns out that both of our problems are easy for k-approval, even with weighted voters.

Theorem 6.20. *For k-approval, the problems* wIsSafe^{pr} *and* wExistSafe^{pr} *are in* P.

Proof. As before, we fix a voter $v \in V$ and renumber the candidates so that v's preference order is given by $c_1 \succ_v \ldots \succ_v c_m$. Suppose that under truthful voting the winner is c_j, and set $C_g = \{c_1, \ldots, c_{j-1}\}$, $C_b = \{c_j, \ldots, c_m\}$. Note that all voters in S_v rank c_j in position j, and the candidates in C_g in positions $1, \ldots, j-1$. For $i = 1, \ldots, m$, let $s_i(\mathcal{R}')$ denote the k-approval score of c_i given a profile $\mathcal{R}'$, and set $s_i = s_i(\mathcal{R})$. We say that c_i *beats* c_ℓ at a preference profile $\mathcal{R}'$ if $s_i(\mathcal{R}') > s_\ell(\mathcal{R}')$ or $s_i(\mathcal{R}') = s_\ell(\mathcal{R}')$ and the tie-breaking rule favors c_i over c_ℓ; note that this relation is transitive. For any $\ell = 1, \ldots, m$, let U_ℓ be the set of all voters in S_v that do not rank c_ℓ in the top k positions. Finally, let $Up(L)$ denote the set of candidates ranked in the top k positions in L.

We will first show how to solve wIsSafe^{pr}.

Lemma 6.21. *A vote L is a safe strategic vote for v if and only if $c_j \notin Up(L)$, the winner in $\mathcal{R}_{-S_v}(L)$ belongs to C_g, and for every $\ell = 1, \ldots, m$, the winner in $\mathcal{R}_{-U_\ell}(L)$ belongs to $C_g \cup \{c_j\}$.*

Proof. Suppose that L ranks c_j in the top k positions, and let the voters in S_v switch to voting L one by one. If $j > k$, each switch increases the score of c_j by at least as much as it increases the score of any other candidate, so c_j remains the winner throughout this process. On the other hand, if $j \leq k$, all voters in S_v already approve all candidates ranked above c_j, so whenever a voter switches to L, the scores of the candidates in C_g do not increase, and the score of c_j does not decrease, so c_j remains the winner. Hence, in this case L is not a strategic vote.

Now assume that L does not rank c_j in the top k positions. We will now argue that if L is a safe strategic vote, then the winner in $\mathcal{R}_{-S_v}(L)$ belongs to C_g. Clearly, if the winner in $\mathcal{R}_{-S_v}(L)$ is c_i with $i > j$, L is not a safe strategic vote. Now, suppose that the winner in $\mathcal{R}_{-S_v}(L)$ is c_j, but some $c_i \in C_g$ wins in $\mathcal{R}_{-U}(L)$ for some $U \subset S_v$. This implies that, in particular, c_i beats c_j at $\mathcal{R}_{-U}(L)$. We will now show that $c_i \in Up(L)$. Indeed, if $j \leq k$, all voters in S_v approve of c_i, so if $c_i \notin Up(L)$, we have $s_i(\mathcal{R}_{-U}(L)) = s_i - w(U)$ and $s_j(\mathcal{R}_{-U}(L)) = s_j - w(U)$. On the other hand, if $j > k$ and $c_i \notin Up(L)$, we have $s_i(\mathcal{R}_{-U}(L)) \leq s_i$ and $s_j(\mathcal{R}_{-U}(L)) = s_j$. In both cases, c_i beats c_j when everyone votes truthfully, a contradiction.

Now, suppose that the voters in $S_v \setminus U$ switch to voting L one by one. Since L ranks c_i, but not c_j, in the top k positions, after each switch c_j's score does not increase, and c_i's score does not decrease, so c_i would still beat c_j when all voters in S_v switch, a contradiction. Hence, if L is a safe strategic vote, the winner in $\mathcal{R}_{-S_v}(L)$ must belong to C_g.

To complete the proof for the "only if" direction, note that if for some $\ell = 1, \dots, m$ the winner in $\mathcal{R}_{-U_\ell}(L)$ is a candidate from $C_b \setminus \{c_j\}$, then obviously L is not a safe strategic vote.

To prove the "if" direction, consider a vote L such that $c_j \notin Up(L)$ and the winner in $\mathcal{R}_{-S_v}(L)$ belongs to C_g. We will show that if there exists at $\ell > j$ and a subset U such that c_ℓ is a winner in $\mathcal{R}_{-U}(L)$, then c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U_\ell}(L)$, i.e., the winner in $\mathcal{R}_{-U_\ell}(L)$ is either c_ℓ or some other candidate in $C_b \setminus \{c_j\}$. To this end, we will gradually transform U into U_ℓ so that c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at each stage of the transformation. We will consider two cases.

$\mathbf{j > k}$:
Suppose first that $U \setminus U_\ell \neq \emptyset$, let u be some voter in $U \setminus U_\ell$, and set $U^- = U \setminus \{u\}$. We claim that c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U^-}(L)$. Indeed, u ranks c_ℓ in the top k positions, so $s_\ell(\mathcal{R}_{-U^-}(L)) \geq s_\ell(\mathcal{R}_{-U}(L))$. Since $j > k$, $s_j(\mathcal{R}_{-U^-}(L)) = s_j(\mathcal{R}_{-U}(L))$. Thus, c_ℓ beats c_j at $\mathcal{R}_{-U^-}(L)$. Further, since $j > k$, c_j beats all candidates in $C \setminus Up(L)$ at $\mathcal{R}_{-S}(L)$ for any $S \subset S_v$. Therefore, by transitivity, c_ℓ beats all candidates in $C_g \setminus Up(L)$ at $\mathcal{R}_{-U^-}(L)$. Finally, for all $c_i \in Up(L)$ we have $s_i(\mathcal{R}_{-U^-}(L)) \leq s_i(\mathcal{R}_{-U}(L))$. Thus, c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U^-}(L)$.

By removing the voters from $U \setminus U_\ell$ one by one, we can assume that $U \subseteq U_\ell$. Now, if $U_\ell \setminus U$ is non-empty, pick a voter $u \in U_\ell \setminus U$, and set $U^+ = U \cup \{u\}$. We claim that c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U^+}(L)$. Indeed, u

does not rank c_ℓ in the top k positions, so $s_\ell(\mathcal{R}_{-U^+}(L)) = s_\ell(\mathcal{R}_{-U}(L))+w(u)$. On the other hand, $s_i(\mathcal{R}_{-U^+}(L)) \leq s_i(\mathcal{R}_{-U}(L)) + w(u)$ for all $c_i \in C$. Thus, c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U^+}(L)$. As we can add voters from $U_\ell \setminus U$ one by one, this also holds at $\mathcal{R}_{-U_\ell}(L)$.

$\mathbf{j \leq k}$**:**
Let c_i, $i < j$, be the winner in $\mathcal{R}_{-S_v}(L)$. Note that all voters in S_v rank c_i in the top k positions. This implies $c_i \in Up(L)$: otherwise, we would have $s_i(\mathcal{R}_{-S_v}(L)) = s_i - w(S_v)$, $s_j(\mathcal{R}_{-S_v}(L)) = s_j - w(S_v)$, so c_j would beat c_i at $\mathcal{R}_{-S_v}(L)$.

Suppose that there exists an $\ell > j$ and a $U \subset S_v$ such that c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U}(L)$. We have $c_\ell \notin Up(L)$: otherwise, we would have $s_\ell(\mathcal{R}_{-S_v}(L)) \geq s_\ell(\mathcal{R}_{-U}(L))$, $s_i(\mathcal{R}_{-S_v}(L)) = s_i(\mathcal{R}_{-U}(L))$, so c_ℓ would beat c_i at $\mathcal{R}_{-S_v}(L)$.

As in the case $j > k$, suppose first that $U \setminus U_\ell \neq \emptyset$, let u be some voter in $U \setminus U_\ell$, and set $U^- = U \setminus \{u\}$. We claim that c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U^-}(L)$. Indeed, u ranks c_ℓ in the top k positions, so $s_\ell(\mathcal{R}_{-U^-}(L)) = s_\ell(\mathcal{R}_{-U}(L)) + w(u)$. On the other hand, $s_r(\mathcal{R}_{-U^-}(L)) \leq s_r(\mathcal{R}_{-U}(L)) + w(u)$ for all $r \in C$. Thus, c_ℓ is the winner in $\mathcal{R}_{-U^-}(L)$, and therefore we can assume that $U \subseteq U_\ell$.

Now, suppose that $U_\ell \setminus U \neq \emptyset$, let u be some voter in $U_\ell \setminus U$, and set $U^+ = U \cup \{u\}$. Since all voters in S_v rank all candidates in C_g in top k positions, we have $s_r(\mathcal{R}_{-U^+}(L)) \leq s_r(\mathcal{R}_{-U}(L))$ for all $r \in C_g$. Moreover, since $c_j \notin Up(L)$, we have $s_j(\mathcal{R}_{-U^+}(L)) = s_j(\mathcal{R}_{-U}(L)) - w(u)$. Finally, since u does not rank c_ℓ in the top k positions, we have $s_\ell(\mathcal{R}_{-U^+}(L)) = s_\ell(\mathcal{R}_{-U}(L))$. Thus, c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U^+}(L)$. As in the case $j > k$, we conclude that c_ℓ beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U_\ell}(L)$. □

Given the characterization of safe strategic votes provided by Lemma 6.21, we can now solve wExistSafe^{pr} for k-approval. The argument is similar to that in the proof of Theorem 6.8. We consider two cases:

$\mathbf{j > k}$**:**
In this case, we cannot lower the score of c_j, so we need to increase the score of some c_i with $i < j$. Suppose there exists a safe strategic vote L such that the winner at $\mathcal{R}_{-S_v}(L)$ is c_i; as argued above, this implies $i < j$. Let $L(i)$ be a vote that ranks c_i first, ranks some candidates from C_g in positions $2, \ldots, k$ (since $j > k$, we have $|C_g| \geq k$), and the rest of the candidates in the remaining positions. It is not hard to see that $L(i)$ is also a safe strategic vote. Indeed, we have $s_i(\mathcal{R}_{-S_v}(L(i))) \geq s_i(\mathcal{R}_{-S_v}(L))$

and $s_\ell(\mathcal{R}_{-S_v}(L(i))) = s_\ell \leq s_\ell(\mathcal{R}_{-S_v}(L))$ for any $\ell \geq j$. Thus, c_i beats all candidates in C_b at $\mathcal{R}_{-S_v}(L(i))$, i.e., the winner in $\mathcal{R}_{-S_v}(L(i))$ must be some candidate from C_g (though not necessarily c_i). Therefore, $L(i)$ is also a strategic vote. To see that $L(i)$ is safe, observe that c_j beats any c_ℓ, $\ell > j$, under truthful voting, and we have $s_j(\mathcal{R}_{-U}(L(i))) = s_j$, $s_\ell(\mathcal{R}_{-U}(L(i))) = s_\ell$ for any $\ell > j$ and any $U \subseteq S_v$.

Thus, to check whether a profile $\mathcal{R}$ admits a safe strategic vote for v, it suffices to construct votes of the form $L(1), \dots, L(j-1)$, and check if any of them is a safe strategic vote for v using Lemma 6.21.

$\mathbf{j \leq k}$:
It is not hard to see that if v has a safe strategic vote, then she also has one that ranks all candidates in C_g in top $j-1$ positions. Thus, to construct a safe strategic vote, we need to fill the remaining $k-j+1$ positions with "safe" candidates.

Let $C_0 = \arg\max\{s_i \mid c_i \in C_g\}$, and let $s_{\max}$ be the k-approval score of the candidates in C_0. For any $c_i \in C$, let s'_i denote the number of points that c_i obtains from voters in $V \setminus S_v$. We have $s_i = s'_i + w(S_v)$ for all $i = 1, \dots, j-1$. We claim that v has a safe strategic vote if and only if the following conditions hold:

(1) For all $c_r \in C_b$, either $s'_r < s_{\max}$, or $s'_r = s_{\max}$ and there exists a candidate $c \in C_0$ such that the tie-breaking rule favors c over c_r;

(2) For all $c_r \in C_b \setminus \{c_j\}$ and all $\ell = 1, \dots, m$, at least one of the following conditions holds:

- $s'_r + w(S_v \setminus (U_r \cup U_\ell)) < s_{\max}$, or
- $s'_r + w(S_v \setminus (U_r \cup U_\ell)) = s_{\max}$ and the tie-breaking rule favors some candidate in C_0 over c_r, or
- $s'_r + w(S_v \setminus (U_r \cup U_\ell)) < s_j - w(U_\ell)$, or
- $s'_r + w(S_v \setminus (U_r \cup U_\ell)) = s_j - w(U_\ell)$ and the tie-breaking rule favors c_j over c_r,

and, moreover, there exists a set $C_{\text{safe}} \subset C_b \setminus \{c_j\}$ with $|C_{\text{safe}}| = k - j + 1$ such that

(3) For all $c_r \in C_{\text{safe}}$, either $s'_r + w(S_v) < s_{\max}$, or $s'_r + w(S_v) = s_{\max}$ and there exists a candidate $c \in C_0$ such that the tie-breaking rule favors c over c_r;

(4) For all $c_r \in C_{\text{safe}}$ and all $\ell = 1, \dots, m$, at least one of the following conditions holds:

- $s'_r + w(S_v \setminus (U_r \cup U_\ell)) + w(U_\ell) < s_{\max}$, or
- $s'_r + w(S_v \setminus (U_r \cup U_\ell)) + w(U_\ell) = s_{\max}$ and the tie-breaking rule favors some candidate in C_0 over c_r, or
- $s'_r + w(S_v \setminus (U_r \cup U_\ell)) + w(U_\ell) < s_j - w(U_\ell)$, or
- $s'_r + w(S_v \setminus (U_r \cup U_\ell)) + w(U_\ell) = s_j - w(U_\ell)$ and the tie-breaking rule favors c_j over c_r.

Note that these conditions can be easily checked in polynomial time; in particular, for each candidate $c_r \in C_b \setminus \{c_j\}$ we can independently check if it can be placed in C_{safe}, so we simply need to verify if there are sufficiently many candidates that satisfy (3) and (4).

Indeed, suppose that these conditions are satisfied, and consider a vote L that ranks the candidates in C_g in the first $j-1$ positions, followed by the candidates in C_{safe}. Condition (1) ensures that any candidate not ranked in the top k positions in L is not the winner in $\mathcal{R}_{-S_v}(L)$, and condition (3) ensures that any candidate ranked in positions $j, \dots, k$ in L is not the winner in $\mathcal{R}_{-S_v}(L)$. Thus, together, these two conditions ensure that the winner in $\mathcal{R}_{-S_v}(L)$ is a candidate from C_g. Condition (2) ensures that no candidate $c_r \in (C_b \setminus \{c_j\}) \setminus Up(L)$ can be the winner in $\mathcal{R}_{-U_\ell}(L)$ for any $\ell = 1, \dots, m$. Similarly, condition (4) ensures that no candidate $c_r \in (C_b \setminus \{c_j\}) \cap Up(L)$ can be the winner in $\mathcal{R}_{-U_\ell}(L)$ for any $\ell = 1, \dots, m$. Thus, together, conditions (2) and (4) imply that for any $\ell = 1, \dots, m$, the winner in $\mathcal{R}_{-U_\ell}(L)$ is a candidate from $C_g \cup \{c_j\}$. Therefore, by Lemma 6.21, L is a safe strategic vote.

Conversely, if condition (1) is violated by some $c_r \in C_b$, then for any vote L the candidate c_r beats all candidates in C_g at $\mathcal{R}_{-S_v}(L)$. Further, if condition (2) is violated by some $c_r \in C_b \setminus \{c_j\}$ and some $\ell \le m$, then for any vote L the candidate c_r beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U_\ell}(L)$. In both cases, by Lemma 6.21, v does not have a safe strategic vote. Now, suppose that there is no set $C_{\text{safe}} \subset C_b \setminus \{c_j\}$ of size $k-j+1$ that satisfies conditions (3) and (4). Then for any vote L there is a candidate $c_r \in C_b \setminus \{c_j\}$ that is ranked in top k positions in L and fails (3) or (4). In the former case, c_r beats all candidates from C_g at $\mathcal{R}_{-S_v}(L)$. In the latter case, if (4) is violated for some $\ell = 1, \dots, m$, it follows that c_r beats all candidates in $C_g \cup \{c_j\}$ at $\mathcal{R}_{-U_\ell}(L)$. Thus, by Lemma 6.21 no vote L can be a safe strategic vote for v. $\square$

In the preference-based model, a voter v follows a recommendation to vote in a particular way if it comes from a voter whose preferences are similar to those of v. However, this approach does not describe settings where a voter follows a recommendation not so much because he trusts the recommender, but for pragmatic purposes, i.e., because the proposed manipulation advances her own goals. Clearly, this may happen even if the overall preferences of the original manipulator and the follower are substantially different. We will now propose a model that aims to capture this type of scenarios.

Goal-Based Extension If the potential follower's preferences are different from those of the manipulator, his decision to join the manipulating coalition is likely to depend on the specific manipulation that is being proposed. Thus, in this subsection we will define the set of potential followers F in a way that depends both on the original manipulator's identity i and his proposed vote L, i.e., we have $F = F_i(L)$. Note, however, that it is not immediately obvious how to decide whether a voter j can benefit from following i's suggestion to vote L, and thus should be included in the set $F_i(L)$. Indeed, the benefit to j depends on which other voters are in the set $F_i(L)$, which indicates that the definition of the set $F_i(L)$ has to be self-referential.

In more detail, for a given voting rule $\mathcal{F}$, an election (C, V) with a preference profile $\mathcal{R}$, a voter $i \in V$ and a vote L, we say that a voter j is *pivotal for a set* $U \subseteq V$ *with respect to* (i, L) if $j \notin U$, $R_j \neq L$ and $\mathcal{F}(\mathcal{R}_{-(U\cup\{j\})}(L)) \succ_j \mathcal{F}(\mathcal{R}_{-U}(L))$. That is, a voter j is pivotal for a set U if when the voters in U vote according to L, it is profitable for j to join them. Now, it might appear natural to define the follower set for (i, L) as the set that consists of i and all voters $j \in V$ that are pivotal with respect to (i, L) for some set $U \subseteq V$. However, this definition is too broad: a voter is included as long as it is pivotal for some subset $U \subseteq V$, even if the voters in U cannot possibly benefit from voting L. To exclude such scenarios, we need to require that U itself is also drawn from the follower set. Formally, we say that $F_i(L)$ is a *follower set* for (i, L) if it is a maximal set F that satisfies the following condition:

$$\forall j \in F\ [\ (j = i)\ \vee\ (\exists\ U \subseteq F \text{ s. t. } j \text{ is pivotal for } U \text{ with respect to } (i, L))] \qquad (*)$$

Observe that this means that $F_i(L)$ is a fixed point of a mapping from 2^V to 2^V, i.e., this definition is indeed self-referential. To see that the follower set is uniquely defined for any $i \in V$ and any vote L, note that the union of any two sets that satisfy condition (*) also satisfies (*); note also that we always have $i \in F_i(L)$.

We can now define what it means for L to be a *strategic vote in the goal-based extension* and a *safe strategic vote in the goal-based extension* by replacing the condition $U \subseteq S_i$ with $U \subseteq F_i(L)$ in Definition 6.19. We will denote the computational problems of checking whether a given vote is a safe strategic vote for a given voter in the goal-based extension and whether a given voter has a safe strategic vote in the goal-based extension by $\textsc{IsSafe}^{gl}$ and $\textsc{ExistSafe}^{gl}$, respectively, and use the prefix W to refer to weighted versions of these problems.

Two remarks are in order. First, it may be the case that even though i benefits from proposing to vote L, he is never pivotal with respect to (i, L) (this can happen, e.g., if i's weight is much smaller that that of the other voters). Thus, we need to explicitly include i in the set $F_i(L)$, to avoid the paradoxical situation where $i \notin F_i(L)$. Second, our definition of a safe vote only guarantees safety to the original manipulator, but not to her followers. In contrast, in the preference-based extension, any vote that is safe for the original manipulator is also safe for all similar voters.

The definition of a safe strategic vote in the goal-based extension captures a number of situations not accounted for by the definition of a safe strategic vote in the preference-based extension. To see this, consider the following example.

Example 6.22. *Consider an election with the set of candidates $C = \{a, b, c, d, e\}$, and three voters 1, 2, and 3, whose preferences are given by $a \succ_1 b \succ_1 c \succ_1 d \succ_1 e$, $e \succ_2 b \succ_2 a \succ_2 d \succ_2 c$, and $d \succ_3 a \succ_3 b \succ_3 c \succ_3 e$. Suppose that the voting rule is Plurality, and the ties are broken according to the priority order $d \succ b \succ c \succ e \succ a$.*

Under truthful voting, d is the winner, so we have $S_1 \neq S_2$. Thus, in the preference-based extension, a vote that ranks a first is a safe strategic vote for voter 2, but a vote than ranks b first is not. On the other hand, let L be any vote that ranks b first. Then $F_1(L) = F_2(L) = \{1, 2\}$. Indeed, if voter 1 switches to voting L, the winner is still d, but it becomes profitable for voter 2 to join her, and vice versa. On the other hand, it is easy to see that voter 3 cannot profit by voting L. It follows that in the goal-based extension L is a safe strategic vote for voter 1.

From a practical perspective, it is plausible that in Example 6.22 voters 1 and 2 would be able to reconcile their differences (even though they are substantial—voter 1 ranks voter 2's favorite candidate last) and jointly vote for b, as this is beneficial for both of them. Thus, at least in some situations the model provided by the goal-based extension is intuitively more appealing. However, computationally it is considerably harder to deal with than the preference-based extension.

Indeed, it is not immediately clear how to compute the set $F_i(L)$, as its definition is non-algorithmic in nature. While one can consider all subsets of V and check whether they satisfy condition (*), this approach is obviously inefficient. We can avoid full enumeration if have access to a procedure $\mathcal{A}(i, L, j, W)$ that for each pair (i, L), each voter $j \in V$ and each set $W \subseteq V$ can check if $j = i$ or there is a set $U \subseteq W$ such that j is pivotal for U with respect to (i, L). Indeed, if this is the case, we can compute $F_i(L)$ as follows. We start with $W = V$, run $\mathcal{A}(i, L, j, W)$ for all $j \in W$, and let W' to be the set of all voters for which $\mathcal{A}(i, L, j, W)$ outputs "yes". We then set $W = W'$, and iterate this step until $W = W'$. In the end, we set $F_i(L) = W$. The correctness of this procedure can be proven by induction on the number of iterations and follows from the fact that if a set W contains no subset U that is pivotal for j, then no smaller set $W' \subset W$ can contain such a subset. Moreover, since each iteration reduces the size of W, the process converges after at most n iterations. Thus, this algorithm runs in polynomial time if the procedure $\mathcal{A}(i, L, j, W)$ is efficiently implementable. We will now show that this is indeed the case for Plurality (with unweighted voters).

Theorem 6.23. *Given an election (C, V) with a preference profile $\mathcal{R}$ and unweighted voters, a manipulator i, and a vote L, we can compute the set $F_i(L)$ with respect to Plurality in time polynomial in the input size.*

Proof. As argued above, it suffices to show that the procedure $\mathcal{A}(i, L, j, W)$ can be implemented in polynomial time. We will now show how to implement it for given values of i, L, j and W. Let $\succ_j$ be j's preference order, and let $\succ$ be the preference order associated with the tie-breaking rule T. Let a be the top-ranked candidate in j's truthful vote, and let c be the top candidate in L.

Suppose that some subset of voters $U \subseteq W$ switches to L, while j votes truthfully, let S be the set of top-scoring candidates at this point, and let x be the score of all candidates in S. When j switches from R_j to L, this decreases by one the number of points a has, and increases by one the number of points that c has.

Observe first if c has at most $x - 2$ points, switching to L cannot be beneficial to j. Indeed, if $S = \{a\}$, or if a is the top-ranked alternative in S

with respect to $\succ$, j already obtains its most preferred outcome. Otherwise, the winner is some $d \in S$, $d \neq a$, and this remains so after the switch.

Now, suppose that c has $x-1$ points. Suppose that the winner is d; this implies that d is the top-ranked candidate in S with respect to $\succ$. Then switching from R_j to L is beneficial for j if and only if $c \succ d$ and $c \succ_j d$. Indeed, if $d \succ c$, d would beat c even after the switch, and if $c \succ d$, but $d \succ_j c$, voter j prefers the current outcome (i.e., d) to the outcome after the switch.

Finally, suppose that c has x points, and let d be the election winner. Then switching from R_j to L is beneficial for j if and only if $c \succ_j d$.

Our algorithm proceeds as follows. For each $d \in W$ such that $c \succ_j d$ and each $x = 1, \dots, n$, we check whether there exists a set $U \subseteq W$ such that when the voters in U vote L and j votes truthfully, all candidates have at most x points, d is the election winner with x points, and either (a) c has $x-1$ point and $c \succ d$, or (b) c has x points. It outputs "yes" if and only if it finds a pair (d, x) that satisfies this condition. The correctness of this algorithm follows from the case analysis above. We will now show how to check whether such a set U exists for a given pair (d, x).

First, we check whether $c \succ_j d$ and the current score of d is at least x, and reject if this is not the case. Next, for each candidate $e \neq c$ whose score is at least x, if $e \succ d$, we add to U all but $x-1$ voters that rank e first, and if $e = d$ or $d \succ e$, we add to U all but x voters that rank e first. If at this point $|U| > x$, we reject. Otherwise, we have ensured that all candidates have at most x points and d is the election winner with x points. It remains to check whether we can implement condition (a) or (b); for that, we may need to add to U some voters who currently vote for candidates in $C \setminus \{d, c\}$. Let y denote the current score of c, and let z be the number of voters that now vote for candidates in $C \setminus \{d, e\}$. We accept if and only if $z \geq x - y$ (i.e., we can satisfy condition (b)) or $z \geq x - y - 1$ and $c \succ d$ (i.e., we can satisfy condition (a)).

Clearly, for each pair (d, x) this check can be implemented in polynomial time. As there are polynomially many such pairs, we can implement the procedure $\mathcal{A}$ (and hence compute the set $F_i(L)$) in polynomial time. □

We can use Theorem 6.23 to show that under Plurality one can determine in polynomial time whether a given vote L is safe for a player i, as well as find a safe strategic vote for i if one exists, as long as the voters are unweighted.

Theorem 6.24. *The problems* IsSafe^{gl} *and* ExistSafe^{gl} *are polynomial-time solvable for Plurality.*

Proof. Suppose that we are given an election (C, V) with a preference profile $\mathcal{R}$ and unweighted voters, a manipulator i, and a vote L. Let $\succ$ denote the preference order used by the tie-breaking rule. Given a profile $\mathcal{R}'$ and a candidate $c \in C$, let $n(c, \mathcal{R}')$ denote the number of voters in $\mathcal{R}'$ that rank c first. Let a be the top-ranked candidate in L, and let x be the winner under truthful voting. We observe that if the outcome of an election $\mathcal{R}'$ changes after some voter k that ranks a candidate t first switches from R'_k to L, then either the outcome was t before the switch, or it becomes a after the change.

At the high level, our algorithm (1) computes the set $F_i(L)$ and (2) for each $c \in C$, determines whether there is a set $S \subseteq F_i(L)$ such that the winner in $\mathcal{R}_{-S}(L)$ is c. Clearly, L is a safe strategic vote for i if and only if the answer is "yes" for some $c \succ_i x$, and "no" for all $c \prec_i x$.

To get the answer for a specific $c \in C$, for each $k = 1, \dots, m$ we check if there is a set $S \subseteq F_i(L)$ such that $n(c, \mathcal{R}_{-S}(L)) = k$, $n(c', \mathcal{R}_{-S}(L)) < k$ for all $c' \succ c$, and $n(c', \mathcal{R}_{-S}(L)) \leq k$ for all $c' \prec c$, and output "yes" if any of these checks produces a positive answer.

We will now show how to implement this check for a given pair (c, k). Consider all $c' \neq a, c$ one by one. Suppose first that $c' \succ c$. Then if at least k voters in $V \setminus F_i(L)$ rank c' first, we output "no" and stop. Otherwise we ask some of the voters in $F_i(L)$ that rank c' first to switch to L so that in the resulting profile exactly $k - 1$ voters rank c' first. The case $c' \prec c$ is similar: if at least $k + 1$ voters in $V \setminus F_i(L)$ rank c' first, we output "no" and stop, and otherwise we ask some of the voters in $F_i(L)$ that rank c' first to switch to L so that in the resulting profile exactly k voters rank c' first.

Now, suppose that we have successfully completed the previous steps. Denote the resulting profile by $\mathcal{R}'$. The rest of the procedure depends on whether $c = a$. Suppose first that $c = a$. In this case, if c is ranked first by at least $k+1$ voters in $\mathcal{R}'$, we output "no". Otherwise, we ask all voters in $F_i(L)$ who have not been asked to change their vote so far to switch their vote to L, and output "yes" if c gets at least k first-place votes in the resulting election. Now, suppose that $c \neq a$. In this case, we cannot increase the score of c, and we cannot lower the score of a. Therefore, if c is ranked first by at most $k - 1$ voters in $\mathcal{R}'$, or $a \succ c$ and a is ranked first by at least k voters in $\mathcal{R}'$, or $a \prec c$ and a is ranked first by at least $k + 1$ voters in $\mathcal{R}'$, we output "no". Otherwise, c beats all other candidates in $\mathcal{R}'$; however, it may have more than k votes. If this is the case, we check if $F_i(L)$ contains $n(c, \mathcal{R}') - k$ voters that rank c first. If not, we output "no"; otherwise, we ask these voters to switch to voting L and output "yes" if and only if c remains the winner in the resulting profile.

To see that ExistSafe^{gl} for Plurality is also polynomial-time solvable, we observe that an analogue of Proposition 6.4 remains true in the goal-based extension. In other words, we can try all m substantially different votes, and run IsSafe^{gl} on each of them. □

For weighted voters, computing the follower set is computationally hard even for Plurality. While this result does not immediately imply that wIsSafe^{gl} and wExistSafe^{gl} are also hard for Plurality, it indicates that these problems are unlikely to be easily solvable.

Theorem 6.25. *Given an instance $(C, V, \mathbf{w}, \mathcal{R})$ of Plurality elections, voters $i, j \in V$ and a vote L, it is* NP*-hard to decide whether $j \in F_i(L)$.*

Proof. We give a reduction from Subset Sum. Given an instance (A, K) of Subset Sum with $|A| = s$, $\sum_{i=1}^{s} a_i = S$, we construct an instance of our problem as follows. We let $C = \{a, b, c\}$, and create $s + 3$ voters $v_1, \ldots, v_s, u_1, u_2, u_3$ with the preferences given by Table 6.3.

Voter	Preference order	Weight
v_i	(a, b, c)	$3a_i$
u_1	(a, b, c)	2
u_2	(b, a, c)	$3S$
u_3	(c, a, b)	$3S + 3K + 1$

Table 6.3: Preferences and weights of voters, in the proof of Theorem 6.25.

Let $\mathcal{R}$ denote the resulting preference profile. Clearly, under truthful voting a gets $3S + 2$ votes, b gets $3S$ votes and c gets $3S + 3K + 1$ votes, so c wins.

Let $L = b \succ a \succ c$. We claim that $u_1 \in F_{u_2}(L)$ if and only if the given instance of Subset Sum is a "yes"-instance. Indeed, suppose first that there exists a set $I \subseteq \{1, \ldots, s\}$ such that $\sum_{i \in I} a_i = K$. Set $U = \{v_i \mid i \in I\}$. In the preference profile $\mathcal{R}_{-U}(L)$ candidate c gets $3S + 3K + 1$ votes, candidate b gets $3S + 3K$ votes, and candidate a gets $3S - 3K + 2$ votes. If, in addition,

u_1 switches to voting L, b becomes the winner. This implies that u_1 is pivotal for U with respect to (u_2, L).

Conversely, suppose that u_1 is pivotal for some set U of voters with respect to (u_2, L). It is not hard to see that it must be the case that c wins in $\mathcal{R}_{-U}(L)$ and b wins in $\mathcal{R}_{-U\cup\{u_1\}}(L)$, and, furthermore, $u_3 \notin F_{u_2}(L)$. Therefore, the score of c is both profiles is $3S + 3K + 1$. Thus, it must be the case that before u_1 switches, b's score is at most $3S + 3K + 1$, and after u_1 switches, b's score is at least $3S + 3K + 1$. Further, if u_1 votes truthfully, the score of b is a multiple of 3. It follows that $w(U \setminus \{u_{s+2}\}) = 3K$, i.e., we started with a "yes"-instance of Subset Sum. □

Just a little further afield, checking whether a given vote is safe with respect to 3-approval is computationally hard even for unweighted voters. This is in contrast with the standard model and the preference-based extension, where safely manipulating k-approval is easy for arbitrary k.

Theorem 6.26. IsSafe^{gl} *is* coNP*-hard for* 3*-approval.*

Proof. We reduce from Exact Cover by 3-Sets (X3C). Recall that an instance of X3C is given by a ground set $G = \{g_1, \dots, g_s\}$ and a collection $\mathcal{X} = \{X_1, \dots, X_t\}$ of subsets of G with $|X_j| = 3$ for $j = 1, \dots, t$. It is a "yes"-instance if G can be covered with exactly $\frac{s}{3}$ sets from $\mathcal{X}$, and a "no"-instance otherwise.

Suppose that we are given an instance $(G, \mathcal{X})$ of X3C with $|G| = s$, $|\mathcal{X}| = t$. We can assume without loss of generality that $t > \frac{s}{3} + 3$: otherwise, the instance is easily solvable by checking all $O(t^3)$ triples of sets to be deleted. We will now construct an instance of our problem as follows. The set of candidates consists of G, three extra candidates $\{a, u, w\}$, and a set $D = \{d_1, \dots, d_6\}$ of dummy candidates.

For each set $X_i \in \mathcal{X}$, we construct a voter i with preferences

$$X_i \succ_i a \succ_i G \setminus X_i \succ_i u \succ_i D \succ_i w.$$

Let V_1 denote the set of all such voters; we have $V_1 = \{1, \dots, t\}$. We then construct a set V_2 whose size is polynomial in s and t. For each voter $j \in V_2$, his preferences satisfy

$$G \succ_j u \succ_j D \succ_j w \succ_j a.$$

In addition, these votes are constructed so that after the votes in $V_1 \cup V_2$ are counted, all candidates in G have the same number of points (and, by construction, any other candidate has zero points). Let f denote the 3-approval score of all alternatives in G based on $V_1 \cup V_2$. We can assume that $f > 2\frac{s}{3} + 2$; if this is not the case, we can add the required number of voters with suitable preferences to V_2.

We now construct three more voter sets: the set V_3 consists of $\frac{s}{3}$ voters, each of which ranks a first, u second and d_1 third, the set V_4 consists of f voters, each of which ranks u first, w second, d_2 third and a last, and the set V_5 consists of $f - 2\frac{s}{3} - 1$ voters, each of which ranks a first, d_3 second and d_4 third. We set $V = \cup_{\ell=1}^{5} V_\ell$.

Let T be a tie-breaking rule based on any order that is consistent with

$$a \succ G \succ w \succ u \succ D;$$

we do not specify how T orders candidates in G or D.

We have $s_3(g) = f$ for all $g \in G$, $s_3(u) = f + \frac{s}{3}$, $s_3(a) = f - \frac{s}{3} - 1$, $s_3(w) = s_3(d_2) = f$, $s_3(d_1) = \frac{s}{3}$ and $s_3(d_3) = s_3(d_4) = f - 2\frac{s}{3} - 1$, where $s_3(c)$ denotes the 3-approval score of a candidate $c \in C$. Thus, under truthful voting the winner is u.

Let L be a vote that ranks a, d_5, and d_6 in the top three positions. We claim that $F_1(L) = V_1 \cup V_3$.

Observe first that no matter which voters choose to vote L, no candidate in D can win. Indeed, d_2 cannot get more points than w, no candidate in $D \setminus \{d_2\}$ can get more points than a, and the tie-breaking rule ranks the candidates in D below all other candidates.

Now, if $\frac{s}{3} - 1$ voters in V_3 and $\frac{s}{3} + 1$ in V_1 vote L, then a gets f points, u gets $f + 1$ points, and u remains the winner. However, if any additional voter from $V_1 \cup V_3$ votes L then a wins, and, moreover, all voters in $V_1 \cup V_3$ prefer a to u.

Next, consider a voter $j \in V_2 \cup V_4$, and suppose that some subset of voters $U \subseteq V$ votes L. Let H denote the set of top-scoring candidates when the voters in U vote L and j votes truthfully. Let T_3 denote the set of three candidate ranked in the top three positions in j's truthful vote. If $a \in H$ or a trails the candidates in H by one vote, j makes a the winner by voting L, which is not an improvement from j's perspective. Now, suppose that this is not the case. If $T(H) = g'$ for some $g' \in T_3$, by changing his vote to L, j either does not change the outcome at all, or changes it from g' to another candidate not in T_3, which is obviously not an improvement from j's perspective. In any other case, j does not have any effect on determining the winning candidate. Clearly, in any of these cases the outcome does not improve from j's point of view, i.e., j is not pivotal for any $U \subseteq V$ with respect to $(1, L)$.

Further, the voters in V_5 cannot increase a's score by switching their vote to L, since their truthful vote already gives a point to a. So, by switching to L they simply reduce the scores of d_3 and d_4 and increase the scores of d_5 and d_6, which does not affect the final outcome.

Now, suppose that the given instance of X3C is a "yes"-instance and let $\mathcal{X}'$ be the corresponding cover. Let $U = \{i \in V_1 \mid X_i \in \mathcal{X}'\} \cup V_3$. Under the profile $\mathcal{R}_{-U}(L)$, all candidates in G get $f-1$ points, u and w get f points, a gets $f-1$ points, and all other candidates get at most f points. Therefore, in this case w wins, showing that L is not safe for voter 1.

Conversely, suppose that the given instance of X3C is a "no"-instance, and consider a set $U \subseteq V_1 \cup V_3$. If $V_3 \not\subseteq U$ then under the profile $\mathcal{R}_{-U}(L)$, u gets at least $f+1$ points, so the winner is either u or a (this depends on the number of voters from V_1 that change their vote to L). Otherwise, under the profile $\mathcal{R}_{-U}(L)$, u and w get f points. Now, if $|U \setminus V_3| > \frac{s}{3}$, a gets at least f points, so it wins. Finally, if $|U \setminus V_3| \leq \frac{s}{3}$, then by our assumption the set $\mathcal{X}' = \{X_i \in \mathcal{X} \mid i \in V_1\}$ is not a cover, so at least one candidate from G gets at least f points. Thus, the winner is either a or some candidate from G. In any of the above cases, the vote is safe from voter 1 perspective. Hence, L is unsafe for 1 (and, by the same argument, for all the other voters in V_1) if and only if G can be covered by exactly $\frac{s}{3}$ sets in $\mathcal{X}$. $\square$

Thus, while the preference-based extension appears to be similar to Slinko and White's original model [101] from the computational perspective, the goal-based extension is considerably more difficult to work with.

Part II

Physical Search Problems with Uncertain Knowledge

Chapter 7

Single Agent

In this chapter we investigate single agent physical search problems with uncertain knowledge. We begin by defining three problems: *Min-Expected-Cost* (minimizing the total expected cost),*Max-Probability* (maximizing the success probability given an initial budget), and *Min-Budget* (minimizing the budget necessary to obtain a given success probability). We then analyze the general metric space case, with any distance function. Unfortunately, in these settings, all three problems are NP-hard. In Section 7.2 we show the hardness of *Min-Expected-Cost*, and in Section 7.3 we show the hardness of *Min-Budget* and *Max-Probability*, which remain hard even if the metric space is a tree. Thus, we focus on the path setting. For this case we provide a polynomial algorithm for the *Min-Expected-Cost* problem (Section 7.2). We show the other two problems (*Min-Budget* and *Max-Probability*) to be NP-complete even for the path setting in Section 7.3. Thus, we consider further restrictions and also provide an approximation scheme. We show that both problems are polynomial if the number of possible prices is constant. For the *Min-Budget* problem, we provide an FPTAS (fully-polynomial-time-approximation-scheme), such that for any $\epsilon > 0$, the *Min-Budget* problem can be approximated with a $(1+\epsilon)$ factor in $O(n\epsilon^{-6})$ time, where n is the size of the input.

7.1 Terminology and Definitions

We are provided with m points - $S = \{u_1, \ldots, u_m\}$, which represent the store locations, together with a distance function $dis : S \times S \to R^+$ - determining the travel costs between any two stores. We are also provided with the agents' initial locations, which are assumed WLOG (without loss of generality) to be at one of the stores (the product's price at this store may be ∞). With a single agent, there is one initial location, u_s; with k agents there is a vector of initial locations $(u_s^{(1)}, \ldots, u_s^{(k)})$. WLOG, we may assume that the agents are ordered from left-to-right, i.e. $u_s^{(1)} < u_s^{(2)} < \cdots < u_s^{(k)}$. In addition, we are provided with a cost probability function $p^i(c)$ - stating the probability that the cost of obtaining the item at store i is c. Let D be the set of distinct prices with non-zero probability, and $d = |D|$. We assume that the actual price at a store is only revealed once an agent reaches the store. Given these inputs, the goal is roughly to obtain the product at the minimal total cost, including both travel costs and purchase price. Since we are dealing with probabilities, this rough goal can be interpreted in three different concrete formulations:

1. *Min-Expected-Cost*: minimize the expected cost of purchasing the product.

2. *Min-Budget*: given a success probability, p_{succ}, minimize the initial budget necessary to guarantee purchase with probability at least p_{succ}.

3. *Max-Probability*: given a total budget B, maximize the probability to purchase the product.

In all the above problems, the optimization problem entails determining the strategy (order) in which to visit the different stores, and if and when to terminate the search. For the *Min-Expected-Cost* problem we assume that an agent can purchase the product even after leaving the store (say by phone). Technically, it is sometimes easy to work with the failure probability instead of the success probability. Therefore, instead of maximizing p_{succ} we may phrase our objective as minimizing the failure probability. Unfortunately, for general distance functions, all three of the above mentioned problems are NP-hard. Thus, we focus on the case that the stores are all located on a single path. We denote these problems *Min-Budget* (path), *Max-Probability* (path), and *Min-Expected-Cost* (path), respectively. In this case we can assume that, WLOG all points are on the line, and do away with the distance function *dis*. Rather, the distance between u_i and u_j is simply $|u_i - u_j|$. Furthermore, WLOG we may assume that the stores are ordered from left-to-right, i.e. $u_1 < u_2 < \cdots < u_m$.

7.2 Minimize-Expected-Cost

We prove that the *Min-Expected-Cost* variant is hard for general metric spaces. To prove this we first convert the problem into its decision version. In *Min-Expected-Cost-Decide* we are given a set of points S, a distance function $dis : S \times S \to R^{+}$, an agent's initial location u_s, a price-probability function $p^{\cdot}(\cdot)$, and a maximum expected cost M. We have to decide whether there is a policy with an expected cost at most M.

7.2.1 Hardness in General Metric Spaces

Theorem 7.1. *For general metric spaces* Min-Expected-Cost-Decide *is NP-hard.*

Proof. The proof is by reduction from Hamiltonian path, defined as follows. Given a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, decide whether there is a simple path $(v_{i_1}, v_{i_2}, ..., v_{i_n})$ in G covering all nodes of V. The reduction is as follows. Given a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, set S (the set of stores) to be $S = \{u_s\} \cup \{u_1, \ldots, u_n\}$, where u_s is the designated start location, and $\{u_1, \ldots, u_n\}$ correspond to $\{v_1, \ldots, v_n\}$. The distances are defined as follows. For all $i, j = 1, \ldots, n$, $dis(u_s, u_i) = 2n$, and $dis(u_i, u_j)$ is the length of the shortest path between v_i and v_j in G. For all i, $p^i(0) = 0.5$, and $p^i(\infty) = 0.5$, and for u_s, $p^s(n!) = 1$. Finally, set $M = 2n + \sum_{j=1}^{n} 2^{-j}(j - 1) + 2^{-n}(n! + n - 1)$.

Suppose that there is a Hamiltonian path $H = (v_{i_1}, v_{i_2}, ..., v_{i_n})$ in G. Then, the following policy achieves an expected cost of exactly M. Starting in u_s move to u_{i_1} and continue traversing according to the Hamiltonian path. If at any point u_i along the way the price is 0, purchase and stop. Otherwise continue to the node in the path. If at all points along the path the price is ∞, purchase from store u_s, where the price is $n!$. The expected cost of this policy is as follows. The price of the initial step (from u_s to u_{i_1}) is a fixed $2n$. For each j, the probability to obtain price 0 at u_{i_j} but not before is 2^{-j}. The cost of reaching u_{i_j} from u_{i_1} is $j - 1$. The probability that no u_j has a price of 0 is 2^{-n}, in which case the purchase price is $n!$, plus $n - 1$ wasted steps. The total expected cost is thus exactly M.

Conversely, suppose that there is no Hamiltonian path in G. Clearly, since the price at u_s is so large, any optimal strategy must check all nodes/stores $\{u_1, \ldots, u_n\}$ before purchasing at u_s. Since there is no Hamiltonian path in G, any such exploration would be strictly more expensive than one with a Hamiltonian path. Thus, the expected cost would be strictly more than M. □

Note that in this proof the number of possible prices, d, is 3. Thus, for general metric spaces *Min-Expected-Cost-Decide* is hard even if d is bounded.

7.2.2 Solution for the Path

When all stores are located on a path, the *Min-Expected-Cost* problem can be modeled as a finite-horizon Markov decision process (MDP), as follows. Note that on the path, at any point in time the points/stores visited by the agent constitute a contiguous interval, which we call the *visited interval.* Clearly, the algorithm need only make decisions at store locations. Furthermore, decisions can be limited to times when the agent is at one of the two stores edges of the *visited interval.* At each such location, the agent has only three possible actions: "go right" - extending the visited-interval one store to the right, "go left" - extending the visited-interval one store to the left, or "stop" - stopping the search and buying the product at the best price so far. Also note that *after* the agent has already visited the interval $[u_\ell, u_r]$, how exactly it covered this interval does not matter for any future decision; the costs have already been incurred. Accordingly, the states of the MDP are quadruplets $[\ell, r, e, c]$, such that $\ell \leq s \leq r$, $e \in \{\ell, r\}$, and $c \in D$, representing the situation that the agents visited stores u_ℓ through u_r, it is currently at location u_e, and the best price encountered so far is c. The terminal states are *Buy(c)* and all states of the form $[1, m, e, c]$, and the terminal cost is c. For all other states there are two or three possible actions - "go right" (provided that $r < m$), "go left" (provided that $1 < \ell$), or "stop". The cost of "go right" on the state $[\ell, r, e, c]$ is $(u_{r+1} - u_e)$, while the cost of "go-left" is $(u_e - u_{\ell-1})$. The cost of "stop" is always 0. Given the state $[\ell, r, e, c]$ and move "go-right", there is probability $p^{r+1}(c')$ to transition to state $[\ell, r+1, r+1, c']$, for $c' < c$. With the remaining probability, the transition is to state $[\ell, r+1, r+1, c]$. Transition to all other states has zero probability. Transitions for the "go left" action are analogous. Given the state $[\ell, r, e, c]$ and the action "stop", there is probability 1 to transition to state *Buy(c).* This fully defines the MDP. The optimal strategy for finite-horizon MDPs can be determined using dynamic programming (see [90, Ch.4]). In our case, the complexity can be brought down to $O(d^2m^2)$ steps (using $O(dm^2)$ space).

7.3 Min-Budget and Max-Probability

7.3.1 NP Completeness

Unlike the *Min-Expected-Cost* problem, the other two problems are NP-complete even on a path. To prove this we again convert the problems into their decision versions. In the *Min-Budget-Decide* problem, we are given a set of points S, a distance function $dis : S \times S \to R^{+}$, an agent's initial location u_s, a price-probability function $p^{\cdot}(\cdot)$, a minimum success probability p_{succ} and maximum budget B. We have to decide whether a success probability of at least p_{succ} can be obtained with a budget of at most B. The exact same formulation also constitutes the decision version of the *Max-Probability* problem.

Theorem 7.2. *The* Min-Budget-Decide *problem is NP-complete even on a path.*

Proof. Given an optimal policy it is easy to compute its total cost and success probability in $O(n)$ steps, therefore *Min-Budget-Decide* is in NP. The proof of NP-hardness is by reduction from the KNAPSACK problem, defined as follows. Given a knapsack of capacity $C > 0$ and N items, where each item has value $v_i \in \mathbb{Z}^{+}$ and size $s_i \in \mathbb{Z}^{+}$, determine whether there is a selection of items ($\delta_i = 1$ if selected, 0 if not) that fits into the knapsack, i.e. $\sum_{i=1}^{N} \delta_i s_i \leq C$, and the total value, $\sum_{i=1}^{N} \delta_i v_i$, is at least V.

Given an instance of KNAPSACK we build an instance for the *Min-Budget-Decide* problem as follows. We assume WLOG that all the points are on the line. Our line consists of $2N+2$ stores. N stores correspond to the knapsack items, denoted by $u_{k_1}, ..., u_{k_N}$. The other $N+2$ stores are denoted $u_{g_0}, u_{g_1}, ..., u_{g_{N+1}}$, where u_{g_0} is the agent's initial location. Let $T = 2 \cdot \sum_{i=1}^{N} s_i$ and $maxV = N \cdot \max_i v_i$. For each odd i, u_{g_i} is to the right of u_{g_0} and $u_{g_{i+2}}$ is to the right of u_{g_i}. For each even i ($i \neq 0$), u_{g_i} is to the left of u_{g_0} and $u_{g_{i+2}}$ is to the left of u_{g_i}. We set $|u_0 - u_1| = |u_0 - u_2| = T$ and for each $i > 0$ also, $|u_{g_i} - u_{g_{i+2}}| = T$. If N is odd (even) u_{k_N} is on the right (left) side of u_{g_i} and it is the rightmost (leftmost) point. As for the other u_{k_i} points, u_{k_i} is located between u_{g_i} and $u_{g_{i+2}}$, if i is odd, and between $u_{g_{i+2}}$ and u_{g_i} otherwise. For both cases, $|u_{g_i} - u_{k_i}| = s_i$. See figure 7.1 for an illustration.

We set $B = T \cdot \sum_{j=1}^{N+1} j + 2C + 1$ and for each i set $X^i = T \cdot \sum_{j=1}^{i} j + 2 \cdot \sum_{j=1}^{i-1} s_j$. At store $u_{g_{N+1}}$ either the product is available at the price of 1 with probability $1 - 2^{-maxV}$, or not available at any price. On any other

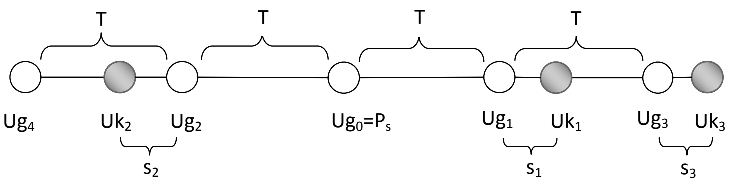


Figure 7.1: Reduction of KNAPSACK to *Min-Budget-Decide* problem used in proof of Theorem 7.2, for N=3.

store u_{g_i}, either the product is available at the price of $B - X^i$ with the same probability, or not available at all. At any store u_{k_i}, either the product is available at the price of $B - X^i - s_i$, with probability $1 - 2^{-v_i}$, or not available at any price. Finally, we set $p_{succ} = 1 - 2^{-maxV \cdot (N+1)} \cdot 2^{-V}$.

Suppose there is a selection of items that fit the knapsack with a total value of at least V, and consider the following policy: go right from u_{g_0} to u_{g_1}. Then for each $i = 1, 2, .., N$, if $\delta_i = 0$ (item i was not selected) change direction and go to the other side to $u_{g_{i+1}}$. Otherwise, continue in the current direction to u_{k_i} and only then change direction to $u_{g_{i+1}}$. This policy's total travel cost is $\sum_{i=1}^{N}(i \cdot T + \delta_i \cdot 2s_i) + (N+1) \cdot T = T \cdot \sum_{i=1}^{N+1} i + 2C = B - 1$, thus the agent has enough budget to reach all u_{g_i}, and u_{k_i} with $\delta_i = 1$. When the agent reaches u_{g_i}, $i < N + 1$ it has already spent on traveling cost exactly $T \cdot \sum_{j=1}^{i} j + 2 \cdot \sum_{j=1}^{i-1}(\delta_j \cdot s_j) \leq X^i$ so the agent has a probability of $1 - 2^{-maxV}$ to purchase the product at this store. When it reaches $u_{g_{N+1}}$ its on the end of its tour and since the agent's total traveling cost is $B - 1$, here it also has a probability of $1 - 2^{-maxV}$ to purchase the product. When it reaches u_{k_i} it has already spent exactly $T \cdot \sum_{j=1}^{i} j + 2 \cdot \sum_{j=1}^{i-1}(\delta_j \cdot s_j) + s_i \leq X^i + s_i$ so the agent has a probability of $1 - 2^{-v_i}$ to purchase the product in this store. In total, the success probability is $1 - (2^{-maxV \cdot (N+1)} \cdot \prod_{i=1}^{N} 2^{-v_i \cdot \delta_i}) \geq 1 - (2^{-maxV \cdot (N+1)} \cdot 2^{-V}) = p_{succ}$ as required.

Suppose there is a policy, *plc* with a total travel cost that is less than or equal to B, and its success probability is at least p_{succ}. Hence, *plc*'s failure probability is at most $1 - p_{succ} = 2^{-maxV \cdot (N+1)} \cdot 2^{-V}$. Since $maxV = N \cdot \max_i v_i$, *plc* must reach all the $N + 1$ stores u_{g_i} with enough budget. Hence, *plc* must go right from u_{g_0} to u_{g_1} and then to each other u_{g_i} before $u_{g_{i+1}}$. Therefore *plc* goes in a zigzag movement from one side of u_s to the other side and so on repeatedly. *plc* also has to select some u_{k_i} to reach with enough budget. Thus, *plc* has to reach these u_{k_i} right after the corresponding store u_{g_i}. We use $\gamma_i = 1$ to indicate the event in which *plc* selects to reach u_{k_i} right after u_{g_i}, and $\gamma_i = 0$ to denote the complementary event. *plc*'s total traveling cost is less than or equal to $B-1$ to be able to purchase the product also at the last store, $u_{g_{N+1}}$, so $T \cdot \sum_{j=1}^{N+1} j + 2 \cdot \sum_{j=1}^{N} \gamma_j \cdot s_j \leq T \cdot \sum_{j=1}^{N+1} j + 2C$. Thus, $\sum_{j=1}^{N} \gamma_j \cdot s_j \leq C$. Also, $p_{succ} = 1 - 2^{-maxV \cdot (N+1)} \cdot 2^{-V} \leq 1 - 2^{-maxV \cdot (N+1)} \cdot \prod_{i=1}^{N} 2^{-v_i \cdot \gamma_i} \Rightarrow 2^{-V} \leq \prod_{i=1}^{N} 2^{-v_i \cdot \gamma_i} \Rightarrow V \geq \sum_{i=1}^{N} v_i \cdot \gamma_i$. Setting $\delta_i = \gamma_i$ gives a selection of items that fit the knapsack. □

Thus, we either need to consider restricted instances or consider approximations. We do both.

7.3.2 Restricted Case: Bounded Number of Prices

We consider the restricted case when the number of possible prices, d, is bounded. For brevity, we focus on the *Min-Budget* (path) problem. The same algorithm and similar analysis work also for the *Max-Probability* (path) problem. Consider first the case where there is only one possible price c_0. At any store i, either the product is available at this price, with probability $p_i = p^i(c_0)$, or not available at any price. In this setting we show that the problem can be solved in $O(m)$ steps. This is based on the following lemma, stating that in this case, at most one direction change is necessary.

Lemma 7.3. *Consider a price c_0 and suppose that in the optimal strategy starting at point u_s the area covered while the remaining budget is at least c_0 is the interval $[u_\ell, u_r]$. Then, WLOG we may assume that the optimal strategy is either $(u_s \rightarrowtail u_r \rightarrowtail u_\ell)$ or $(u_s \rightarrowtail u_\ell \rightarrowtail u_r)$.*

Proof. Any other route would yield a higher cost to cover the same interval. □

Using this observation, we immediately obtain an $O(m^3)$ algorithm for the single price case: consider both possible options for each interval $[u_\ell, u_r]$, and for each compute the total cost and the resulting probability. Choose the option which requires the lowest budget but still has a success probability of at least p_{succ}. With a little more care, the complexity can be reduced to $O(m)$. First note that since there is only a single price c_0, we can add c_0 to the budget at the end, and assume that the product will be provided at stores for free, provided that it is available. Now, consider the strategy of first moving right and then switching to the left. In this case, we need only consider the *minimal* intervals that provide the desired success probability, and for each compute the necessary budget. This can be performed incrementally, in a total of $O(m)$ operations for all such minimal intervals, since at most one point can be added and one deleted at any given time. Similarly for the strategy of first moving left and then switching to the right. The details are provided in Algorithm 3.

Algorithm 3 OptimalPolicyForSinglePrice(Success probability p_{succ}, single price c_0)

```
1: $u_r \leftarrow$ leftmost point on right of $u_s$ s.t. $1 - \prod_{i=s}^{r} 1 - p_i \geq p_{succ}$
2: $\ell \leftarrow s$
3: $B_{\min}^{RL} \leftarrow |u_r - u_s|$
4: **while** $\ell \geq 0$ and $r \geq s$ **do**
5:     $B \leftarrow 2|u_r - u_s| + |u_s - u_\ell|$
6:     **if** $B < B_{\min}^{RL}$ **then**
7:         $B_{\min}^{RL} \leftarrow B$
8:     $r \leftarrow r - 1$
9:     **while** $\ell \geq 0$ and $1 - \prod_{i=\ell}^{r} 1 - p_i < p_{succ}$ **do**
10:        $\ell \leftarrow \ell - 1$
11: $u_\ell \leftarrow$ rightmost point to left of $u_s$ s.t. $1 - \prod_{i=\ell}^{s} 1 - p_i \geq p_{succ}$
12: $r \leftarrow s$
13: $B_{\min}^{LR} \leftarrow |u_s - u_l|$
14: **while** $r \leq m$ and $\ell \leq s$ **do**
15:     $B \leftarrow 2|u_s - u_\ell| + |u_r - u_s|$
16:     **if** $B < B_{\min}^{LR}$ **then**
17:         $B_{\min}^{LR} \leftarrow B$
18:     $\ell \leftarrow \ell + 1$
19:     **while** $r \leq m$ and $1 - \prod_{i=\ell}^{r} 1 - p_i < p_{succ}$ **do**
20:         $r \leftarrow r + 1$
21: **return** $\min\{B_{\min}^{RL}, B_{\min}^{LR}\} + c_0$
```

Next, consider the case that there may be several different available prices, but their number, d, is fixed. We provide a polynomial algorithm for this case (though exponential in d). First note that in the *Min-Budget* problem, we seek to minimize the initial budget B necessary so as to guarantee a success probability of at least p_{succ} given this initial budget. Once the budget has been allocated, however, there is no requirement to minimize the actual expenditure. Thus, at any store, if the product is available for a price no greater than the remaining budget, it is purchased immediately and the search is over. If the product has a price beyond the current available budget, the product will not be purchased at this store under any circumstances. Denote $D = \{c_1, c_2, \ldots, c_d\}$, with $c_1 > c_2 > \cdots > c_d$. For each c_i there is an interval $I_i = [u_\ell, u_r]$ of points covered while the remaining budget was at least c_i. Furthermore, for all i, $I_i \subseteq I_{i+1}$. Thus, consider the *incremental* area covered with remaining budget c_i, $\Delta_i = I_i - I_{i-1}$ (with $\Delta_1 = I_1$). Each Δ_i is a union of an interval at left of u_s and an interval at the right of u_s (both possibly empty). The next lemma, which is the multi-price analogue of Lemma 7.3, states that there are only two possible optimal strategies to cover each Δ_i:

Lemma 7.4. *Consider the optimal strategy and the incremental areas Δ_i $(i = 1, \ldots, d)$ defined by this strategy. For $c_i \in D$, let u_{ℓ_i} be the leftmost point in Δ_i and u_{r_i} the rightmost point. Suppose that in the optimal strategy the covering of Δ_i starts at point u_{s_i}. Then, WLOG we may assume that the optimal strategy is either $(u_{s_i} \rightarrowtail u_{r_i} \rightarrowtail u_{\ell_i})$ or $(u_{s_i} \rightarrowtail u_{\ell_i} \rightarrowtail u_{r_i})$. Furthermore, the starting point for covering Δ_{i+1} is the ending point of covering Δ_i.*

Proof. The areas Δ_i fully determine the success probability of the strategy. Any strategy other than the ones specified in the lemma would require more travel budget, without enlarging any Δ_i. □

Thus, the optimal strategy is fully determined by the leftmost and rightmost points of each Δ_i, together with the choice for the ending points of covering each area. We can thus consider all possible cases and choose the one with the lowest budget which provides the necessary success probability. There are $\frac{m^{2d}}{(2d)!} \leq (\frac{em}{2d})^{2d}$ ways for choosing the external points of the Δ_i's, and there are a total of 2^d options to consider for the covering of each. For each option, computing the budget and probability takes $O(m)$ steps. Thus, the total time is $O(m2^d(\frac{em}{2d})^{2d})$. Similar algorithms can also be applied for the *Max-Probability* (path) problem. In all, we obtain:

Theorem 7.5. Min-Budget *(path) and* Max-Probability *(path) can be solved in $O(m)$ steps for a single price and $O(m2^d(\frac{em}{2d})^{2d})$ for d prices.*

Unfortunately, even with a bounded number of possible prices, *Min-Budget-Decide* is still hard, even on a tree.

Theorem 7.6. *The* Min-Budget-Decide *problem is NP-complete on a tree, even with a bounded number of prices.*

Proof. Membership in NP is immediate as in the proof of Theorem 7.2. The proof of NP-hardness is by reduction from KNAPSACK problem.

Given an instance of KNAPSACK we build an instance for the *Min-Budget-Decide* problem as follows. We have $N+2$ stores. N stores corresponds to the knapsack items, denoted by $u_{k_1}, ..., u_{k_N}$. The other 2 stores are u_0 and u_e, where u_0 is the agent's initial location. The stores are placed on a star, which is a tree with one internal node, u_0, and $N+1$ leaves. The distance to any u_{k_i} is defined according to the item value, $dis(u_0, u_{k_i}) = s_i/2$, and $dis(u_0, u_e) = C$. At any store u_{k_i}, either the product is available at the price of 0 with probability $1 - 2^{-v_i}$, or not available at any price. At store u_0 the product is not available, and at store u_e either the product is available at the price of 0 with probability $1 - 2^{-maxV}$, $maxV = N \cdot \max_i v_i$, or not available at any price. Finally, we set $p_{succ} = 1 - 2^{-maxV} \cdot 2^{-V}$, and $B = 2 \cdot C$.

Suppose there is a selection of items that fit the knapsack with a total value of at least V, and consider the following policy: for each $i = 1, 2, .., N$, if $\delta_i = 1$ (item i was selected) go from u_0 to u_{k_i} and then back to u_0. Finally, go from u_0 to u_e. This policy's travel cost is $\sum_{i=1}^{N} (\delta_i \cdot s_i) + C \leq 2 \cdot C = B$. If the product is available at any store, its price is 0. Thus, the success probability of this policy is $1 - (2^{-maxV} \cdot \prod_{i=1}^{N} 2^{-v_i \cdot \delta_i}) \geq 1 - (2^{-maxV} \cdot 2^{-V}) = p_{succ}$ as required.

Suppose there is a policy, *plc* with a total travel cost that is less than or equal to B, and its success probability is at least p_{succ}. Hence, *plc*'s failure probability is at most $1 - p_{succ} = 2^{-maxV} \cdot 2^{-V}$. Since $maxV = N \cdot \max_i v_i$, *plc* must reach store u_e. *plc* also has to select some u_{k_i} to reach, but since $dis(u_0, u_e) = C$ and $B = 2 \cdot C$, *plc* must reach these u_{k_i} before reaching u_e. We use $\gamma_i = 1$ to indicate the event in which *plc* selects to reach u_{k_i}, and $\gamma_i = 0$ to denote the complementary event. *plc*'s traveling cost before going to u_e is less than or equal C, to be able reach u_e, so $\sum_{j=1}^{N} \gamma_j \cdot s_j \leq C$. Also, $p_{succ} = 1 - 2^{-maxV} \cdot 2^{-V} \leq 1 - 2^{-maxV} \cdot \prod_{i=1}^{N} 2^{-v_i \cdot \gamma_i} \Rightarrow 2^{-V} \leq \prod_{i=1}^{N} 2^{-v_i \cdot \gamma_i} \Rightarrow V \geq \sum_{i=1}^{N} v_i \cdot \gamma_i$. Setting $\delta_i = \gamma_i$ gives a selection of items that fit the knapsack. □

7.3.3 Min-Budget Approximation

Next, we provide an FPTAS (fully-polynomial-time-approximation-scheme) for the *Min-Budget* (path) problem. The idea is to force the agent to move in quantum steps of some fixed size δ. In this case the tour taken by the agent can be divided into *segments*, each of size δ. Furthermore, the agent's decision points are restricted to the ends of these segments, except for the case where along the way the agent has sufficient budget to purchase the product at a store, in which case it does so and stops. We call such a movement of the agent a *δ-resolution tour*. Note that the larger δ the less decision points there are, and the complexity of the problem decreases. Given $0 < \epsilon < 1$, we show that with a proper choice of δ we can guarantee a $(1 + \epsilon)$ approximation to the optimum, while maintaining a complexity of $O(n\text{poly}(1/\epsilon))$, where n is the size of the input.

Our algorithm is based on computing for (essentially) each initial possible budget B, the maximal achievable success probability, and then pick the minimum budget with probability at least p_{succ}. Note that once the interval $[\ell, r]$ has been covered without purchasing the product, the only information that matters for any future decision is (i) the remaining budget, and (ii) the current location. The exact (fruitless) way in which this interval was covered is, at this point, immaterial. This, "memoryless" nature calls, again, for a dynamic programming approach. We now provide a dynamic programming algorithm to compute the optimal δ-resolution tour. WLOG assume that $u_s = 0$ (the initial location is at the origin). For integral i, let $w_i = i\delta$. The points w_i, which we call the *resolution points*, are the only decision points for the algorithm. Set L and R to be such that w_L is the rightmost w_i to the left of all the stores and w_R the leftmost w_i to the right of all stores. We define two tables, $\mathsf{fail}[\cdot,\cdot,\cdot,\cdot]$ and $\mathsf{act}[\cdot,\cdot,\cdot,\cdot]$, such that for all ℓ, r, $L \le \ell \le 0 \le r \le R$, $e \in \{\ell, r\}$ (one of the end points), and budget B, $\mathsf{fail}[\ell, r, e, B]$ is the minimal failure probability achievable for purchasing at the stores *outside* $[w_\ell, w_r]$, assuming a remaining budget of B, and starting at location w_e. Similarly, $\mathsf{act}[\ell, r, e, B]$ is the best act to perform in this situation ("left", "right", or "stop"). Given an initial budget B, the best achievable success probability is $(1 - \mathsf{fail}[0, 0, 0, B])$ and the first move is $\mathsf{act}[0, 0, 0, B]$. It remains to show how to compute the tables. The computation of the tables is performed from the outside in, by induction on the number of remaining points. For $\ell = L$ and $r = R$, there are no more stores to search and $\mathsf{fail}[L, R, e, B] = 1$ for any e and B. Assume that the values are known for i remaining points, we show how to compute for $i + 1$ remaining points. Consider $\mathsf{cost}[\ell, r, e, B]$ with $i + 1$ remaining points. Then, the least failure probability obtainable by a decision to move right (to w_{r+1}) is:

$$F_R = \left(1 - \sum_{c \leq B-\delta} p^{r+1}(c)\right) \mathsf{fail}[\ell, r+1, r+1, B-\delta]$$

Similarly, the least failure probability obtainable by a decision to move left (to $w_{\ell-1}$) is:

$$F_L = \left(1 - \sum_{c \leq B-\delta} p^{\ell-1}(c)\right) \mathsf{fail}[\ell-1, r, \ell-1, B-\delta]$$

Thus, we can choose the act providing the least failure probability, determining both $\mathsf{act}[\ell, r, e, B]$ and $\mathsf{fail}[\ell, r, e, B]$. In practice, we compute the table only for B's in integral multiples of δ. This can add at most δ to the optimum. Also, we may place a bound $B^{\delta}_{\max}$ on the maximal B we consider in the table. In this case, we start filling the table with $w_L = -B^{\delta}_{\max}/\delta$ and $w_R = B^{\delta}_{\max}/\delta$, the furthest point reachable with budget $B^{\delta}_{\max}$.

Next, we show how to choose δ and prove the approximation ratio. Set $\lambda = \epsilon/9$. Let $\alpha = \min\{|u_s - u_{s+1}|, |u_s - u_{s-1}|\}$ - the minimum budget necessary to move away from the starting point, and $\beta = m^2|u_m - u_1| + \max\{c : \exists i, p^i(c) > 0\}$ - an upper bound on the total usable budget. We start by setting $\delta = \lambda^2\alpha$ and double it until $\delta > \lambda^2\beta$, performing the computation for all such values of δ. For such value of δ, we fill the tables (from scratch) for all values of B's in integral multiples of δ up to $B^{\delta}_{\max} = 2\lambda^{-2}\delta$. We now prove that for at least one of the choices of δ we obtain a $(1+\epsilon)$ approximation.

Consider a success probability p_{succ} and suppose that optimally this success probability can be obtained with budget B_{opt} using the tour T_{opt}. By *tour* we mean a list of actions ("right", "left" or "stop") at each decision point (which, in this case, are all store locations). We convert T_{opt} to a δ-resolution tour, T^{δ}_{opt}, as follows. For any $i \geq 0$, when T_{opt} moves for the first time to the right of w_i then T^{δ}_{opt} moves all the way to w_{i+1}. Similarly, for $i \leq 0$, when T_{opt} moves for the first time to the left of w_i then T^{δ}_{opt} moves all the way to w_{i-1}.

Note that T^{δ}_{opt} requires additional travel costs only when it "overshoots", i.e. when it goes all the way to the resolution point while T_{opt}would not. This can either happen (i) in the last step, or (ii) when T_{opt} makes a direction change. Type (i) can happen only once and costs at most δ. Type (ii) can happen at most once for each resolution point, and costs at most 2δ. Suppose that T^{δ}_{opt} makes t *turns* (i.e. t directions changes). Then, the total additional travel cost of the tour T^{δ}_{opt} over T_{opt} is at most $(2t{+}1)\delta$. Furthermore, if we use T_{opt} with budget B_{opt} and T^{δ}_{opt} with budget $B_{opt}+(2t+1)\delta$ then at any store, the available budget under T^{δ}_{opt} is at least that available with T_{opt}. Thus, T^{δ}_{opt} is a δ-resolution tour that with budget at most $B_{opt}+(2t+1)\delta$ succeeds with probability $\geq p_{succ}$. Hence, our dynamic algorithm, which finds the optimal such δ-resolution tour will find a tour with budget $B^{\delta}_{opt} \leq B_{opt} + (2t+2)\delta$ obtaining at least the same success probability (one additional δ for the integral multiples of δ in the tables).

Since T^{δ}_{opt} has t-turns, T_{opt} must also have t-turns, with targets at t *distinct* resolution segments. For any i, the i-th such turn (of T_{opt}) necessarily means that T_{opt}moves to a point at least $(i-1)$ segments away, i.e. a distance of at least $(i-1)\delta$. Thus, for B_{opt}, which is at least the travel cost of T_{opt}, we have:[1]

$$B_{opt} \geq \sum_{i=1}^{t}(i-1)\delta = \frac{(t-1)(t)}{2}\delta \geq \frac{t^2}{4}\delta \tag{7.1}$$

On the other hand, since we consider all options for δ in multiples of 2, there must be a $\hat{\delta}$ such that:

$$\lambda^{-2}\hat{\delta} \geq B_{opt} \geq \frac{\lambda^{-2}}{2}\hat{\delta} \tag{7.2}$$

Combining (7.1) and (7.2) we get that $t \leq 2\lambda^{-1}$. Thus, the approximation ratio is:

$$\frac{B^{\hat{\delta}}_{opt}}{B_{opt}} \leq \frac{B_{opt+2(t+1)\hat{\delta}}}{B_{opt}} \leq 1 + \frac{2(t+1)\hat{\delta}}{\lambda^{-2}\hat{\delta}/2} \tag{7.3}$$

$$\leq 1 + (8\lambda + 4\lambda^2) \leq 1 + \epsilon \tag{7.4}$$

Also, combining (7.2) and (7.4) we get that

$$B^{\hat{\delta}}_{opt} \leq B_{opt}(1+\epsilon) \leq 2\lambda^{-2}\hat{\delta} = B^{\hat{\delta}}_{\max}$$

Hence, the tables with resolution $\hat{\delta}$ consider this budget, and $B^{\hat{\delta}}_{opt}$ will be found.

[1]Assuming that $t > 1$. If $t = 0, 1$ the additional cost is small by (7.2).

It remains to analyze the complexity of the algorithm. For any given δ there are $B^{\delta}_{\max}/\delta = 2\lambda^{-2}$ budgets we consider and at most this number of resolution points at each side of u_s, for each, there are two entries in the table. Thus, the size of the table is $\leq 8\lambda^{-6} = O(\epsilon^{-6})$. The computation of each entry takes $O(1)$ steps. We consider δ in powers of 2 up to $\beta \leq 2^n$, where n is the size of the input. Thus, the total computation time is $O(n\epsilon^{-6})$. We obtain:

Theorem 7.7. *For any $\epsilon > 0$, the* Min-Budget *(path) problem can be approximated with a $(1+\epsilon)$ factor in $O(n\epsilon^{-6})$ steps.*

Chapter 8

Multi-Agent

In this chapter we investigate multi-agent physical search problems with uncertain knowledge. In these settings, we analyze two models for handling resources, the shared and the private budget models. We present polynomial algorithms that work for any fixed number of agents, both for the shared budget model (Section 8.1) and for the private budget model (Subsection 8.2.2). For non-communicating agents in the private budget model, we present a polynomial algorithm that is suitable for any number of agents (Subsection 8.2.1). We also analyze the difference between homogeneous and heterogeneous agents in Section 8.4, both with respect to their allotted resources and with respect to their capabilities. Finally, we define our variants in an environment with self-interested agents in Section 8.3. We show how to find a Nash Equilibrium in polynomial time, and prove that the bound on the performance of our algorithms, with respect to the social welfare, is tight. We conclude this chapter in Section 8.5, with a discussion on the assumptions we have made, and we suggest ways to extend our current results.

8.1 Shared Budget

Since even the single agent case is hard for general metric spaces, with the multi-agent case we focus solely on situations in which all the stores are on a single path. We assume k agents, operating in the same underlying physical setting as in the single agent case, i.e. a set of *stores* S and a price probability function for each store. We assume that the goal is not individualized; the agents seek to obtain only one item and having multiple goods is not beneficial. Furthermore, since the agents are fully collaborative, they do not care which agent will obtain the item.

We begin by analyzing the *shared budget* multi-agent model, where all the resources and costs are shared among all the agents. In theory, agents may move in parallel, but since minimizing time is not an objective, we may assume WLOG that at any given time only one agent moves. When an agent reaches a store and finds the price at this location, the optimal strategy should tell whether to purchase the product (and where) and if not what agent should move next and to where. Therefore, in the *k-Shared-Min-Expected-Cost* problem the agents try to minimize the expected total cost, which includes the travel costs of all agents plus the final purchase price (which is one of the prices that the agents have sampled). In *k-Shared-Min-Budget* and *k-Shared-Max-Probability*, the initial budget is for the use of all the agents, and the success probability is for any of the agents to purchase, at any location. Since all the agents use the same budget in this model, inter alia, for traveling costs, we assume the agents can communicate with each other to coordinate their moves. In *k-Shared-Min-Budget* and *k-Shared-Max-Probability* the agents only need to announce to the other agents when they reach a specific store. In *k-Shared-Min-Expected-Cost* the agents also need to communicate the price they find at the location they have reached.

In general, the algorithms for the single-agent case (for the path) can be extended to the multi-agent case, with the additional complexity of coordinating between the agents. We present the proofs in the appendix since they are very similar to proofs in the single agent case.

Theorem 8.1. *With k agents,* k-Shared-Min-Expected-Cost *can be solved in $O(d^2 2^k (\frac{m}{k})^{2k})$.*

Theorem 8.2. *With k agents,* k-Shared-Min-Budget *and* k-Shared-Max-Probability *with d possible prices can be solved in $O(m 2^{kd} (\frac{em}{2kd})^{2kd})$.*

Theorem 8.3. *With k agents, for any $\epsilon > 0$,* k-Shared-Min-Budget *can be approximated to within a factor of $(1+k\epsilon)$ in $O(n\epsilon^{-6k})$ steps (for an arbitrary number of prices).*

While the complexity in the multi-agent case grows exponentially in the number of agents, in most physical environments where several agents cooperate in exploration and search, the number of agents is relatively moderate.

In these cases the computation of the agents' strategies is efficiently facilitated by the principles of the algorithmic approach presented in this work.

If the number of agents is not fixed (i.e. part of the input) then, the complexity of all three variants grows exponentially. Most striking perhaps is that *k-Shared-Min-Budget* and *k-Shared-Max-Probability* are NP-complete even on the path with a *single price*. To prove this we again formulate the problems into a decision version- *k-Shared-Min-Budget-Decide* - given a set of points S on the path, initial locations for all agents $(u_s^{(1)}, \ldots, u_s^{(k)})$, a price-probability function $p^{\cdot}(\cdot)$, a minimum success probability p_{succ} and a maximum budget B, decide if success probability of at least p_{succ} can be achieved with a maximum budget B.

Theorem 8.4. k-Shared-Min-Budget-Decide *is NP-complete even on the path with a* single price.

Proof. An optimal policy defines for each time step which agent should move and in which direction. Since there are at most $2m$ time steps, it is easy to compute the success probability and the total cost in $O(m)$ steps, therefore the problem is in NP. The NP-hard reduction is from the KNAPSACK problem.

We assume WLOG that all the points are on the line. We use N agents and our line consists of $2N$ stores. N stores correspond to the knapsack items, denoted $u_{k_1}, ..., u_{k_N}$. The other N points are the starting point of the agents, $\{u_s^{(i)}\}_{i=1,..,N}$. We set the left most point to $u_s^{(1)}$ and the right most point to u_{k_N}. For all $1 \leq i \leq N-1$ set u_{k_i} right after $u_s^{(i)}$ and $u_s^{(i+1)}$ right after u_{k_i}. Set $|u_s^{(i)} - u_{k_i}| = s_i$ and $|u_{k_i} - u_s^{(i+1)}| = B+1$. See figure 8.1 for an illustration.

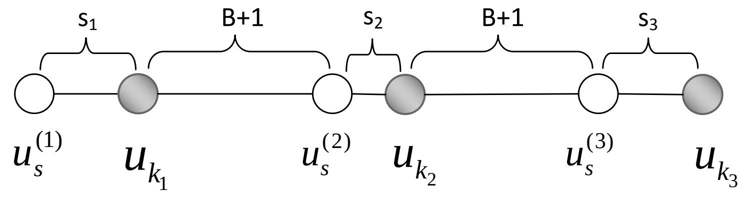


Figure 8.1: Reduction of the KNAPSACK problem to the Multi-Min-Budget-Decide problem used in the proof of Theorem 8.4, for N=3.

The price at all the nodes is $c_0 = 1$ and $p^{k_i}(1) = 1 - 2^{-v_i}$. Finally, set $B = C + 1$ and $p_{succ} = 1 - 2^{-V}$.

For every agent i, the only possible move is to node p_{k_i}, denote by $\gamma_i = 1$ if agent i moves to p_{k_i}, and 0 if not. Therefore, there is a selection of items that fit, i.e, $\sum_{i=1}^{N} \delta_i s_i \leq C$, and the total value, $\sum_{i=1}^{N} \delta_i v_i$, is at least V iff there is a selection of agents that move such that $\sum_{i=1}^{N} \gamma_i s_i \leq B$, and the total probability $1 - \prod_{i=1}^{N} \gamma_i 2^{-v_i}$, is at least $p_{succ} = 1 - 2^{-V}$. $\square$

This is in contrast to the single agent case where the single price case can be solved in $O(n)$ steps.

8.2 Private Budget

We now investigate a model of *private budgets*, whereby each agent j has its own initial budget B_j (unlike the previous shared budget model). If the objective is to minimize the total expected cost, the private budgets model is equal to the shared budget model since the agents are cooperative. Therefore in this case we have two concrete problem formulations:

1. *k-Private-Max-Probability*: given initial budgets B_j, for each agent j, maximize the probability of obtaining the item.

2. *k-Private-Min-Budget*: given a target success probability p_{succ}, minimize the agents' initial budgets necessary to guarantee acquisition of the item with a probability of at least p_{succ}.

Since the corresponding single-agent problems are hard even for the path, we again assume that the number of possible prices, d, is bounded. In the *k-Private-Min-Budget* problem it is also important to distinguish between two different agent models:

- *Identical budgets*: the initial budgets of all the agents must be the same. The problem is to minimize this initial budget, and we denote the problem as *k-Private-Min-Budget*$^{\text{identical}}$.

- *Distinct*: the agents' initial budgets may be different. In this case the problem is to minimize the average initial budget, and we denote the problem as *k-Private-Min-Budget*$^{\text{distinct}}$.

8.2.1 Non-Communicating Agents

We first consider the case where agents cannot communicate with each other. In this case agents cannot assist each other. Hence a solution is a *strategy* comprising of a set of ordered lists, one for each agent, determining the sequence of stores this agent must visit.

The success probability of a strategy is the probability that at least one of the agents will succeed in its task. Technically, in this case, it is easier to calculate the complementary failure probability: the probability that all the agents will not succeed in their tasks. For example, suppose that the stores and agents are located as illustrated in Figure 8.2, and consider the depicted strategy. This strategy fails if for both agents and each of the stores they visit the cost of the item is higher than their remaining budget. This will happen with probability $\frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{2} \cdot \frac{4}{5} = \frac{1}{20}$. Hence, the success probability of this strategy is $\frac{19}{20}$.

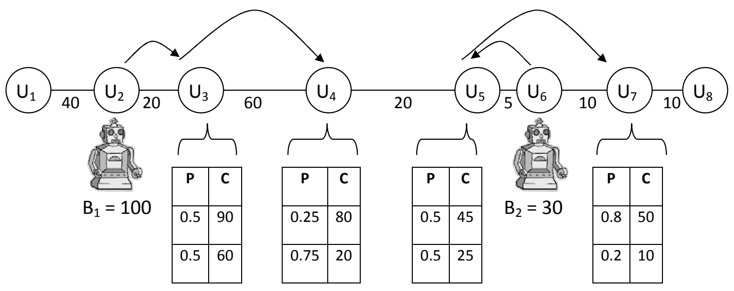


Figure 8.2: A possible input with a suggested strategy. The numbers on the edges represent traveling costs. The table at each store u_i represents the cost probability function $p_i(c)$. The strategy of each agent is illustrated by the arrows.

We begin by considering the *k-Private-Max-Probability* problem. We prove:

Theorem 8.5. *In the no communication case if the number of possible costs is constant then* k-Private-Max-Probability *can be solved in polynomial time for any number of agents.*

The proof is based on the following definitions and lemmata.

Note that multiple strategies may result in the same success probability. In this case we say that the strategies are *equivalent*. In particular there may be more than one optimal strategy.

Definition 8.6. *Let S be a strategy. Agents i and $\bar{i}$ are said to be* separated *by S if each store that is reached by i is not reached by $\bar{i}$.*

Lemma 8.7. *If agents i and $\bar{i}$ are not separated by any optimal strategy, then in at least one optimal strategy at least one of these agents must pass the initial location of the other.*

Proof. WLOG assume that i is on the right side of $\bar{i}$. Consider an optimal strategy S. Let r be the rightmost store that is reached by i and $\bar{l}$ the leftmost store that is reached by $\bar{i}$. Assume that none of the agents passes the initial location of the other in S. Thus, there is at least one store between their initial locations that is reached by both agents. WLOG assume that $\bar{i}$ reaches at least one store with a higher budget than i's remaining budget when reaching it, and denote by $\bar{r}^*$ the rightmost such store. Consider the following modified strategy: i goes according to S till the stage it has to reach $\bar{r}^*$. If i did not reach r yet then instead of reaching $\bar{r}^*$ it goes all the way straight to r. Otherwise, it stops just before reaching $\bar{r}^*$. $\bar{i}$ goes according to S till the stage it has to reach $\bar{r}^*$. If $\bar{i}$ did not reach $\bar{l}$ yet then after reaching $\bar{r}^*$ it goes all the way straight to $\bar{l}$. Otherwise, it stops after reaching $\bar{r}^*$. Agents i and $\bar{i}$ are separated by this strategy and it has at least the same success probability as S, in contradiction. □

Lemma 8.8. *Suppose that agents i and $\bar{i}$ are not separated by any optimal strategy. Let S be an optimal strategy. Suppose that in S agent i passes the initial location of agent $\bar{i}$ and agent $\bar{i}$ does not stay in its initial location. Then, there is an optimal strategy such that one of the following holds:*

- *$\bar{i}$ moves only in one direction which is opposite to the final movement's direction of i. Furthermore, if the final movement's direction of i is right(left) then $\bar{i}$ passes the leftmost(rightmost) store that is reached by i.*

- *either i or $\bar{i}$ does not move.*

Proof. WLOG assume that i is on the right side of $\bar{i}$. Let $[l, r]$ be the interval of stores covered by i. Since i passes the initial location of $\bar{i}$, l is located on the left of $u_s^{(\bar{i})}$ and r is located on the right of $u_s^{(\bar{i})}$.

First we show that we may assume that $\bar{i}$ reaches at least one store outside the interval $[l, r]$. If this is not the case, consider two cases. If i's remaining budget at each store is always as high as $\bar{i}$'s remaining budget then $\bar{i}$ does not have to move and the theorem holds. Otherwise, let $\bar{r}^*$ the rightmost store where $\bar{i}$'s remaining budget is higher than i's remaining budget. If $\bar{r}^*$ is on the left side of i's initial location, then as in the proof of Lemma 8.7, the agents can be separated. If $\bar{r}^*$ is on the right side of i's initial location and it equals r, there is no need for i to reach r since at each store in $[u_s^{(i)}, r]$, $\bar{i}$ has at least the same budget as i. Thus, there is an optimal strategy where either i does not move or it moves only to the left, so $\bar{i}$ passes the rightmost store that is reached by i. If $\bar{r}^*$ is on the right side of i but on the left side of r then there is no need for $\bar{i}$ to go beyond $\bar{r}^*$. Since it has more budget than i at this location, $\bar{i}$ can move to l while i moves to r. Thus, again, there is an optimal strategy where either i does not move or it moves only to the right, so $\bar{i}$ passes the leftmost store that is reached by i. Thus, we may assume that $\bar{i}$ reaches at least one store outside the interval $[l, r]$.

WLOG assume that i's final movement's direction is left and suppose that $\bar{i}$ reaches at least one store outside the interval $[l, r]$ to the left of l. If $\bar{i}$'s budget at l is higher than i's remaining budget there, then it is also higher at $u_s^{(\bar{i})}$, and again the agents can be separated. If $\bar{i}$'s budget at l is not higher than i's remaining budget, than $\bar{i}$ does not have to move since i can reach the same stores to the left of l.

Now suppose that $\bar{i}$ moves to the right (which is the opposite direction of i's final movement) and passes $u_s^{(i)}$, but it also changes its direction. The only reason for $\bar{i}$ to change directions is to reach a store on the left side of its initial location, with a higher budget than i has at this store, or to reach a store that i does not reach at all. In both cases $\bar{i}$ must reach each store in $[l, u_s^{(i)}]$ with at least the same budget as i has at the same location, so either S is not optimal, or we can modify S by letting only $\bar{i}$ to move while i does not move at all. □

Using these lemmata we observe that for any two agents, there is only a constant number of possible cases where the agents are not separated by the optimal strategies. Figure 8.3 illustrates the three core cases (the others are symmetrical). Here, agents 1 and 3 are non-separated agents. Note that every agent between them, like agent 2, does not have to move at all in the optimal strategy.

Therefore we can use a dynamic programming approach to find an optimal

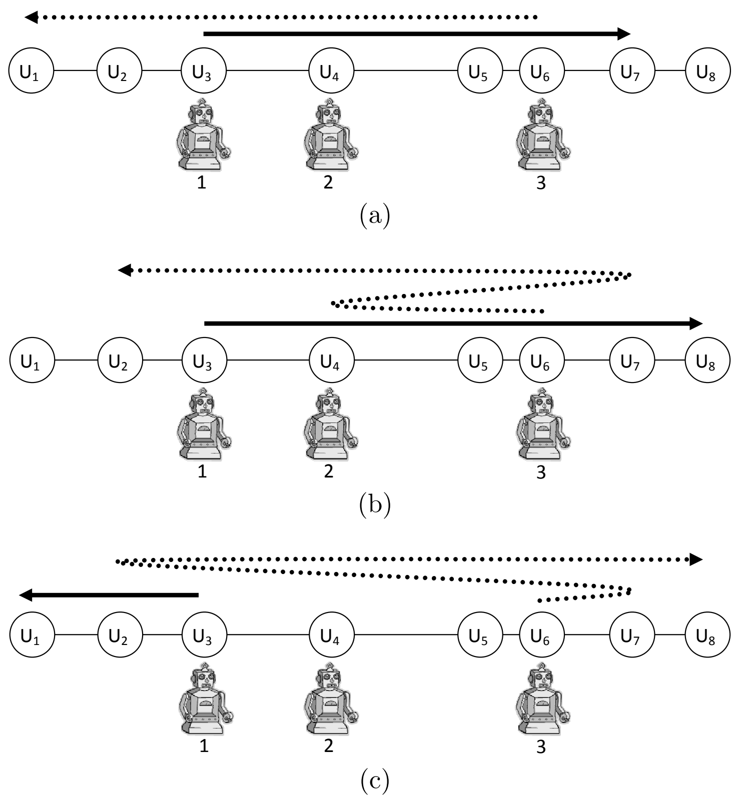


Figure 8.3: The only three cases where a pair of agents may not be separated.

strategy whereby all the agents are separated, but we also check the non-separated strategies individually.

Recall that in our problem the objective is to maximize the success probability, given the initial budgets. Technically, it is easy to work with the failure probability instead of the success probability.

Definition 8.9. *$\mathsf{fail}[u_i, j]$ is the minimal failure probability if the only reachable stores are in the interval $[u_1, u_i]$, and only agents $1, \cdots, j$ are allowed to move. $\mathsf{act}[u_i, j]$ is the optimal strategy achieving $\mathsf{fail}[u_i, j]$, under the same conditions* [1]*.*

Note that where $u_i < u_s^{(j)}$, $\mathsf{fail}[u_i, j]$ is not defined. Given $\mathsf{act}[u_i, j]$, $\mathsf{fail}[u_i, j]$ can be easily computed in $O(m)$ steps. For technical reasons we add another agent, 0, with a budget of zero and set its initial location to the leftmost store, i.e $u_s^{(0)} = u_1$. $\mathsf{fail}[u_i, 0]$=1 for all i, and this agent doesn't affect the failure probability of any policy.

We are now ready to prove Theorem 8.5

Proof of Theorem 8.5 We use dynamic programming to calculate $\mathsf{fail}[u_m, k]$ and $\mathsf{act}[u_m, k]$. For $\mathsf{fail}[u_i, 1]$ and $\mathsf{act}[u_i, 1]$, which is the single agent case, we employ the polynomial algorithm obtained from Theorem 7.5.

Given any agent $\bar{j}$ we first consider the case where $u_i = u_s^{(\bar{j})}$. In this case in the optimal strategy $\bar{j}$ moves only to the left, or not at all. Let $u_l^{(\bar{j})}$ be the leftmost store visited by $\bar{j}$ with the optimal strategy for the given interval, and agent l be the one such that $u_s^{(l)} \leq u_l^{(\bar{j})}$ (l may equal 0). Each agent t such that $l < t < \bar{j}$ does not move in the optimal strategy. Otherwise, agents t and $\bar{j}$ are not separated and according to Lemma 8.8 agent t must pass the rightmost store $u_s^{(\bar{j})}$, which is not possible. The same argument shows that each agent t such that $t \leq l$ does not reach $u_l^{(\bar{j})}$. Therefore $\mathsf{act}[u_i, \bar{j}]$ is composed of $\mathsf{act}[u_{l-1}^{(\bar{j})}, l]$, which are already known, together with the movement of agent $\bar{j}$ to $u_l^{(\bar{j})}$. Thus, computing $u_l^{(\bar{j})}$ takes $O(m)$ steps.

Next, consider the case where $u_i > u_s^{(\bar{j})}$. In this case, in the optimal strategy $\bar{j}$ may move in both directions, or not move at all. Let $u_l^{(\bar{j})}$ be

[1]There may be more than one strategy with the same failure probability, $\mathsf{act}[u_i, j]$ is one of them

the leftmost store visited by $\bar{j}$ with the optimal strategy for this interval, and agent l is the one such that $u_s^{(l)} \leq u_l^{(\bar{j})}$. First note that each agent t, $t \leq l$, and $\bar{j}$ are separated by the optimal policy, or $\bar{j}$ does not move. Otherwise, according to Lemma 8.7 t must pass the initial location of $\bar{j}$ but according to Lemma 8.8 $\bar{j}$ must reach a store outside the interval $[u_s^{(l)}, u_s^{(\bar{j})}]$ which does not occur. Since $\bar{j}$ passes the initial locations of every agent t, $l < t < \bar{j}$, if one of them moves it goes only in the opposite direction of the final movement direction of $\bar{j}$ according to Lemma 8.8 , and as illustrated in Figure 8.3. Since they all must move in the same direction, according to the same Lemma at most one of them moves in the optimal policy. Therefore, to compute $\mathsf{act}[u_{\bar{i}}, \bar{j}]$ we check only the following options, and choose the best one:

1. $\bar{j}$ does not move, and $\mathsf{act}[u_{\bar{i}}, \bar{j}] = \mathsf{act}[u_{\bar{i}}, \bar{j} - 1]$.

2. Each agent t, $l < t < \bar{j}$, does not move. Thus, $\mathsf{act}[u_{\bar{i}}, \bar{j}]$ is composed of $\mathsf{act}[u_{l-1}^{(\bar{j})}, l]$, with the optimal movement of agent $\bar{j}$ in the interval $[u_l^{(\bar{j})}, u_i]$.

The previous two options assume that $\bar{j}$ and every other agent are separated. Otherwise:

3. One agent t, $l < t < \bar{j}$, moves. Let $u_l^{(\bar{t})}$ be the leftmost store visited by either agent t or $\bar{j}$, with the optimal strategy, and agent l is the one such that $u_s^{(l)} \leq u_l^{(\bar{t})}$. $\mathsf{act}[u_{\bar{i}}, \bar{j}]$ is composed of $\mathsf{act}[u_{l-1}^{(\bar{t})}, l]$, with the optimal movement of the two agents $\bar{j}$ and t in the interval $[u_l^{(\bar{t})}, u_i]$.

There are at most m possible options for $u_l^{(\bar{j})}$. In each option we check for at most k agents m possible options for $u_l^{(\bar{t})}$. Therefore for each agent j and store u_i $\mathsf{act}[u_i, j]$ can be found in $O(m^2 k)$ steps, and $\mathsf{act}[u_m, k]$ can be found in $O(m^3 k^2)$ time steps using $O(mk)$ space.

For the *k-Private-Min-Budget* problem we obtain:

Theorem 8.10. *In the no communication setting, if the number of costs is constant,* k-Private-Min-Budget^{identical} *can be solved in polynomial time for any number of agents.*

Proof. By Theorem 8.5, given a budget $\bar{B}$, we can calculate the maximum achievable success probability. Thus we can run a binary search over the possible values of $\bar{B}$ to find the minimal one that still guarantees a success probability p_{succ}. The maximum required budget is $2 \cdot |u_1 - u_m|$, which is part of the input. Thus the binary search will require a polynomial number of steps. □

Theorem 8.11. *If the number of agents is a parameter,* k-Private-Min-Budget^{distinct} *with no communication is NP-complete even for a single possible cost.*

Proof. Membership in NP is trivial. Theorem 8.4 considered the shared budget case and showed that when the number of agents is not constant the *k-Shared-Min-Budget* problem is NP-complete. In the *k-Private-Min-Budget*$^{\text{distinct}}$ problem the objective is to minimize the average budget, which is the same as minimizing the total budget. Thus, the hardness of the problem follows from that of the *k-Shared-Min-Budget* problem. □

8.2.2 Communicating Agents

Once communication is added agents can call upon each other for assistance and the relative scheduling of the agents' moves must also be considered. In this case a *solution* is an ordered list of *moves*, where each move is a pair stating an agent and its next destination.

The success probability of a solution is now calculated according to the order of moves. For example, suppose that the stores and agents are located as illustrated in Figure 8.4.

Consider the following solution: agent 2 first goes to u_4 and then agent 1 goes to u_2. Agent 2 is the only one which can succeed at u_4, with a probability of 0.8. With probability of 0.2 it will not succeed and agent 1 has a probability of 0.2 to succeed at u_2. Hence, the success probability is $0.8 + 0.2 \cdot 0.2 = 0.84$. If we switch the order of the moves we get a probability of 0.9 to succeed at u_2 with the first move, since agent 2 will be called for assistance if the cost required is less than 100. If not, agent 2 will move to u_5 as before. Hence, this solution success probability is $0.9 + 0.1 \cdot 0.8 = 0.98$.

When the number of agents is not fixed, *k-Private-Max-Probability*, *k-Private-Min-Budget*$^{\text{identical}}$ and *k-Private-Min-Budget*$^{\text{distinct}}$ are not

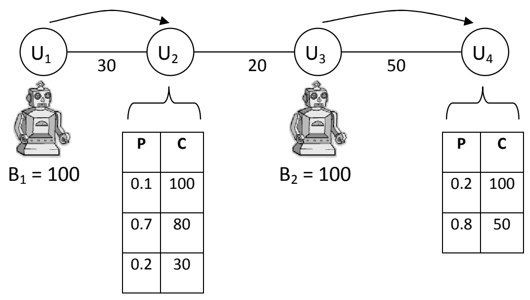


Figure 8.4: A possible input with suggested moves. The numbers on the edges represent traveling costs. The table at each store u_i represents the cost probability function $p_i(c)$. The moves are illustrated by the arrows.

known to be solvable in polynomial time. However, in many physical environments where several agents cooperate in exploration and search, the number of agents is relatively small. In this case we can show that all the three problems can be solved in polynomial time. We show:

Theorem 8.12. *In the setting of communicating agents, if the number of agents and the number of different costs is fixed then* k-Private-Max-Probability*,* k-Private-Min-Budget^{identical} *and* k-Private-Min-Budget^{distinct} *can be solved in polynomial time.*

For brevity, we focus on the *k-Private-Max-Probability* problem. The same algorithm and similar analysis work also for the other two problems.

First note that as in *Max-Probability*, in *k-Private-Max-Probability* we need to maximize the probability of obtaining the item given the initial budgets B_i, but there is no requirement to minimize the actual resources consumed (in contrast to *k-Shared-Max-Probability*). Thus, at any store, if agents can obtain the item for a cost no greater than its remaining budget, the search is over. Furthermore, if the cost is beyond the agent's available budget, but there is another agent with a sufficient budget to both travel from its current location and to obtain the item, then this agent is called upon and the search is also over. Otherwise, the item will not be obtained at this store under any circumstances. Thus, the basic strategy structure, which determines which agent goes where, remains the same. Unless the search has to be terminated, the decision of one agent where to go next is not affected by the knowledge gained by others. Using a similar argument as in the proof of Theorem 8.2, we get the following result. For brevity, we denote $\bar{d}$ instead of $d+1$.

Proposition 8.13. *For k agents, one needs only to consider $O(m2^{k\bar{d}}(\frac{em}{2k\bar{d}})^{2k\bar{d}})$ number of options for the set of moves of the agents.*

Proof. Let $c_1 > c_2 > \cdots > c_d$ be the set of costs. For each agent j and for each c_i there is an interval $I_i^{(j)} = [u_\ell, u_r]$ of points covered while *the agent's* remaining budget is at least c_i. Furthermore, for each j and for all i, $I_i^{(j)} \subseteq I_{i+1}^{(j)}$. Thus, consider for each agent the *incremental* area covered when *its* remaining budget is c_i but less than c_{i-1}, $\Delta_i^{(j)} = I_i^{(j)} - I_{i-1}^{(j)}$ (with $\Delta_1^{(j)} = I_1^{(j)}$). Each $\Delta_i^{(j)}$ is a union of an interval at left of $u_s^{(j)}$ and an interval at the right of $u_s^{(j)}$ (both possibly empty). Since there is communication, an agent may continue to reach a store even if it does not have any chance of obtaining the item there, in order to reveal the cost for the use of other agents. Thus, the optimal strategy may define also an interval $I_{\bar{d}}^{(j)} = [u_\ell, u_r]$ of points covered while the remaining budget of j is greater than 0. By Lemma C.1, the moves of each agent are fully determined by the leftmost and rightmost stores of each $\Delta_i^{(j)}$, together with the choice for the ending points of covering each area. For each two agents j_1, j_2, the intervals of covered points are disjoint, i.e. $I_{\bar{d}}^{(j_1)} \cap I_{\bar{d}}^{(j_2)} = \emptyset$. Therefore, for each j there are at most $\frac{(\frac{m}{k})^{2\bar{d}}}{(2\bar{d})!} \leq (\frac{em}{2k\bar{d}})^{2\bar{d}}$ possible choices for the external stores of the $\Delta_i^{(j)}$'s, and there are a total of $2^{\bar{d}}$ options to consider for the covering of each. Thus, the total number of options for the set of moves is $O(2^{k\bar{d}}(\frac{em}{2k\bar{d}})^{2k\bar{d}})$, which is polynomial (in m) $\square$

It thus remains to consider the scheduling between the moves, i.e. their *order*. Theoretically, with n moves there are $n!$ different possible orderings. We show, however, that for any given set of moves, we need only to consider a polynomial number of possible orderings.

Consider a given set of moves M, determining the sets $\Delta_i^{(j)}$. Note that for each agent, M fully determines the order of the moves of this agent. A subset M' of M is said to be a *prefix* of M, if for each agent the moves in M' are a prefix of the moves of this agent in M. A subset M' is a suffix of M if $M - M'$ is a prefix. We now inductively define the notion of a *cascading order*:

1. The trivial order on moves of a single agent is cascading.

2. Let M be a set of moves, and let c_{i_0} be the highest cost that any agent can pay. An order S on M is *cascading* if M and S can be decomposed $M = M_{pre} \cup M_{mid} \cup M_{post}$ and $S = S_{pre} \circ S_{mid} \circ S_{post}$, such that:
   - M_{pre} is a prefix of M consisting only of moves of agents with budget less than c_{i_0} and S_{pre} is a cascading order on M_{pre}.
   - There exists an agent j' with budget at least c_{i_0} such that M_{mid} consists of all the moves of j' in $\Delta_{i_0}^{(j')}$ and S_{mid} is the (one possible) order on these moves.
   - M_{post} are the remaining moves in M and S_{post} is a cascading order on them.

We prove (by induction) that cascading orders are optimal.

Lemma 8.14. *For any set of moves M there exists a cascading order with optimal success probability.*

Proof. The proof is by induction on the number of agents and the number of moves in M. If there is only one agent moving in M then the order is cascading. Otherwise, consider any other order S on M and let A_{i_0} be the set of agents with budget at least c_{i_0}. Let j' be the first agent in A_{i_0} to cover its $\Delta_{i_0}^{j'}$ and let t_0 be the time it completes covering it. M_{pre} includes all the moves taken by agents not in A_{i_0} prior to t_0; M_{mid} includes all the moves of j' in $\Delta_{i_0}^{(j')}$; and M_{post} the rest of the moves in M. We show that we do not decrease the success probability by first making all moves of M_{pre} then all those of M_{mid}, and finally those of M_{post}. By the inductive hypothesis S_{pre}, S_{mid} and S_{post} are optimal for M_{pre}, M_{mid} and M_{post}, respectively and the result follows.

Before t_0 all agents in A_{i_0} have a higher budget than any agent not in A_{i_0}. Thus, before t_0 agents of A_{i_0} will never call upon those not in A_{i_0}. Thus, it cannot decrease the success probability if we let the agents not in A_{i_0} take their moves first. Thus, we can allow to first perform all moves of M_{pre}.

Also, before t_0 no agents of A_{i_0} needs to call upon each other for assistance (since they are all in the same resource bracket). Thus, we may allow them to take their moves independently without decreasing the success probability. In particular, we can allow j' to complete its covering of $\Delta_{i_0}^{(j')}$ before any other member of A_{i_0} moves. Thus, we get that first having the moves of M_{pre} and then of M_{mid} does not decrease the success probability. The moves of M_{post} are the remaining moves. □

Finally we show that the number of cascading orders is polynomial:

Lemma 8.15. *For fixed k and d and any set of moves M there are a polynomial number of cascading orders on M.*

Proof. Set $f(n,k,d,\ell)$ be the number of cascading orders with k agents, n moves, d costs and ℓ agents in A_{i_0}. We prove by induction that f is a polynomial in n. Since $\ell \leq k$, the result follows. Clearly, for any ℓ, $f(n,k,0,\ell) = \ell!$ (all of which are useless). Then, by the definition of cascading orders $f(n,k,d,\ell) \leq \ell n^{k-\ell} f(n,k-\ell,d-1,k-\ell) f(n,k,d,\ell-1)$ (the n^k being for the choice of M_{pre}). By the inductive hypothesis $f(n,k-\ell,d-1,k-\ell)$ and $f(n,k,d,\ell-1)$ are polynomials in n. Thus, so is $f(n,k,d,\ell)$. $\square$

Together with Corollary 8.13 we get that the total number of options to consider is polynomial, proving the *k-Private-Max-Probability* part of Theorem 8.12. The proof for the other two problems is similar.

8.3 Self-interested Agents

In this section we consider the strategic behavior that may occur when the agents are self-interested. We assume k agents, operate in the same underlying physical setting as in the previous multi-agent case with private budgets, i.e. the stores are all on a single path, the number of possible prices, d, is bounded, and there is a fixed number of agents. However in the self-interested agents setting, the agents seek to obtain the item but do not want to spend their individual budgets on travel costs; we assume the purchase price is equally shared among all the agents. In this case we define two games, a simultaneous game, *Min-Budget-Game*, and a sequential game, *Min-Expected-Cost-Game*.

8.3.1 Min-Budget Game

In the *Min-Budget-Game* we are given a target success probability p_{succ}, and each agent's objective is to minimize its initial budget necessary to guarantee that the item will be acquired with a probability of at least p_{succ}. To avoid the case where each agent will set its initial budget at zero, we set the utility of not guaranteeing the success probability p_{succ} so low that it will always be worthwhile to attain it. We assume the game is a simultaneous game;

the agents can only choose their initial budgets. After this phase, the agents calculate the (collaborative) strategy that will maximize their success probability (given their chosen budgets) and follow it. The only decision point in this game is when an agent needs to choose its budget.

Since the number of agents and the number of different costs is fixed, the optimal solution for *k-Private-Min-Budget*$^{\text{distinct}}$ can be found in polynomial time, whether the agents can or cannot communicate (Theorem 8.12). Let B_i^{alg} be the initial budget that was assigned to agent i by the algorithm from Theorem 8.12. This solution of *k-Private-Min-Budget*$^{\text{distinct}}$, which is optimal, can be directly translated into a strategy, denote Opt_{soc}: each agent i should individually choose its initial budget to be B_i^{alg}. Obviously, Opt_{soc} maximizes the social welfare and it can be computed in polynomial time. Furthermore, Opt_{soc} is also a Nash Equilibrium [86, p.14]. Clearly, for each agent i, there is no incentive to deviate and to choose a budget for itself which is larger than B_i^{alg}, since with B_i^{alg} the success probability p_{succ} is already guaranteed (assuming the other agents will not deviate). On the other hand, since the algorithm of Theorem 8.12 is optimal, there is no incentive for each agent i to deviate and choose a budget for itself which is smaller than B_i^{alg}, as p_{succ} will not be achieved (recall that the utility of not guaranteeing the success probability is very low). We obtain:

Theorem 8.16. *In the* Min-Budget-Game*, the strategy that maximizes the social welfare,* Opt_{soc}*, can be found in polynomial time and it is also a Nash Equilibrium.*

8.3.2 Min-Expected-Cost Game

In the *Min-Expected-Cost-Game* each agent's objective is to minimize its total expected cost. As in the previous game, to avoid the case where each agent will not want to make any move, we set the utility of not obtaining the item so low that it is always worth traveling to at least one store to purchase the product. The *Min-Expected-Cost-Game* is a sequential game and the rules are as follows. At each time step, only one of the agents is allowed to move to the next store, but it can also decide not to move at all. Then there is a decision phase, where every agent is allowed to buy the product, to opt-out, or to do nothing. If at least on agent decides to buy the product, it is purchased and then the game is over (even if other agents decide to opt-out). No matter how many agents decide to buy the product, only one is purchased. If no agent decides to buy the product and at least one agent decides to opt-out then the game is over without buying the product. Otherwise, the decision phase ends and the game proceeds by allowing the next agent (according to a fixed, pre-defined cyclic order) to move. The pre-defined order of movement phases well-define the game, but it is has no essential meaning; the agents have the option not to move during their turns, so actually any order of movements may occur.

In order to find the strategy that will maximize the social welfare, Opt_{soc}, we need to run the algorithm from Theorem 8.1. In our setting, it will run in polynomial time. However, unlike in the *Min-Budget-Game*, the solution found cannot be directly translated into a strategy. First, we need to translate the movements. At any stage, if the algorithm for *k-Shared-Min-Expected-Cost* decides that a specific agent should move, for instance agent i, then the strategy for *Min-Expected-Cost-Game* defines that until it is agent i's turn to move, any other agent will not move during its movement phase, and all the agents will do nothing during the decision phase. We also need to handle the case where one agent does not move according to this strategy. For this purpose, we determine that in any case where one of the agents deviates from its determined policy in the movement phase, the other agents purchase the product during the decision phase that follows. If it is not possible, i.e. the product is not available yet, the other agents opt-out during the decision phase. The translation of the algorithm's decision to buy is straightforward; the strategy defines that in the corresponding decision phase all the agents decide to buy. We also do not need to handle the case where one agent deviates in a decision phase, since the game will be over in that case. In conclusion, Opt_{soc}, the strategy for *Min-Expected-Cost-Game* that maximizes the social welfare, can be found in polynomial time using the algorithm from Theorem 8.1. However, Opt_{soc} is not always a Nash Equilibrium, as will be shown in Example 8.17. For ease of notation, when describing Opt_{soc} or any other strategy we omit the movement and decision phases when the agents do nothing.

Example 8.17. *Suppose that the stores and agents are located as illustrated in Figure 8.5. The traveling costs between u_2 and u_3 and between u_4 and u_5 are so high, that the only reasonable moves are according to the illustrated arrows. Opt_{soc} for this example is that agent 1 will go to u_2. If the price is 6 the product will be purchased. Otherwise, agent 3 will go to u_6 and if the price is 6 the product will be purchased. Otherwise, agent 2 will go to*

u_4 and the product will be purchased at the minimal sampled price (which can be 12 or 27). The expected cost of this strategy is 14.375, but it is not a Nash Equilibrium. Clearly, if the product was not purchased after the moves of agents 1 and 3, then the minimal sampled price will be 27. At this stage, if agent 2 deviates and decides not to move, the product will be purchased and the private cost of agent 2 will be 9 (the purchase price is equally shared among all the agents). If agent 2 proceeds according to Opt_{soc}, its expected cost will be $4 + 0.5 \cdot 4 + 0.5 \cdot 9 = 10.5 > 9$. Therefore, agent 2 will have an incentive to deviate from Opt_{soc}.

If we switch the movement order of agents 2 and 3, the expected cost will be higher, 15.125, but this strategy is a Nash Equilibrium. Clearly, agent 1 will not deviate during its turn since the other agents will opt-out. Agent 2 will not deviate during its turn since its private cost will be 10, and if it will follow the strategy its expected cost will be $4+0.5\cdot4+0.5\cdot(0.5\cdot2+0.5\cdot9) = 8.75 < 10$ (assuming the other agents will not deviate). Agent 3 will not deviate during its turn either, since its private cost will be 10, and if it will follow the strategy its expected cost will be $4 + 0.5 \cdot 2 + 0.5 \cdot 9 = 9.5 < 10$.

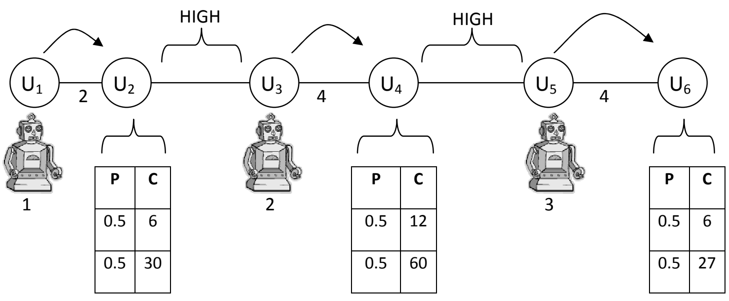


Figure 8.5: A possible input with suggested moves. The numbers on the edges represent traveling costs. The table at each store u_i represents the cost probability function $p_i(c)$. The reasonable moves are illustrated by the arrows.

Example 8.17 demonstrates that Opt_{soc} is not always a Nash Equilibrium. We now show a polynomial algorithm that always returns a strategy which is a Nash Equilibrium. Furthermore, we show an upper bound on the algorithm's performance, and prove that it is tight.

Theorem 8.18. *There is a polynomial algorithm for finding a Nash Equilibrium for the* Min-Expected-Cost-Game.

Proof. The algorithm works as follows. It divides all the buying costs by k, and then solves the finite-horizon MDP as in the proof of Theorem 8.1. The solution gained is then translated into a strategy for the *Min-Expected-Cost-Game*, in the same way we translated the optimal solution of *k-Shared-Min-Expected-Cost* to Opt_{soc}. Let denote this strategy by Opt_{Nash}. Since the pre-process takes $O(d)$ operations the algorithm for finding Opt_{Nash} is polynomial.

Any strategy S consists of traveling costs and buying costs, denoted by $\{t_i\}$ and $\{b_i\}$, respectively. We can then write the expected cost of S as $\mathrm{E[S]} = \sum\limits_i (p_i^t \cdot t_i) + \sum\limits_i (p_i^b \cdot b_i)$, where p_i^t, p_i^b are the associated probabilities. We also write $t_i \in j$ if the traveling cost t_i was credited to the movement of agent j.

We now analyze the steps of Opt_{Nash}. First note that if the product is not available yet, there is no incentive to deviate since the other agents will opt-out and the game will be over. We thus assume that the product is available. The last step of Opt_{Nash} is a decision step, where the product is purchased. By definition, there is an incentive to purchase the product in this step. In any other decision phase the strategy of Opt_{Nash} is not to purchase the product. However, if agent j deviates in a movement phase, the product will be purchased in the decision phase that follows. Therefore, we only need to consider the movement phases. Now, consider agent j and a movement phase r, and suppose that j needs to move in r. If j deviates (does not move), his expected cost will be c/k, the best price encountered so far divided by the number of agents. Since Opt_{Nash} is optimal (with respect to the modified buying costs),

$$c/k \geq \sum_{i \geq r} (p_i^t \cdot t_i) + \sum_{i \geq r} (p_i^b \cdot b_i/k) \tag{8.1}$$

In addition,

$$\sum_{i \geq r} (p_i^t \cdot t_i) \geq \sum_{i \geq r, t_i \in j} (p_i^t \cdot t_i) \tag{8.2}$$

Combining (8.1) and (8.2) we obtain,

$$c/k \geq \sum_{i \geq r, t_i \in j} (p_i^t \cdot t_i) + \sum_{i \geq r} (p_i^b \cdot b_i/k)$$

where the right term is the expected cost of agent j if it follows Opt_{Nash}.

Therefore, agent j has no incentive to deviate. The same analysis shows that j does not have an incentive to deviate if it does need to move in r. □

Opt_{Nash} is a Nash Equilibrium, but it does not maximize the social welfare. Furthermore, there may be another Nash Equilibrium which will yield a larger social welfare. For example, recall the settings in Example 8.17. In these settings, Opt_{Nash} policy is that agent 1 will go to u_2. If the price is 6 the product will be purchased. Otherwise, agent 3 will go to u_6 and buy the product at the minimal price (6 or 27). This is indeed a Nash Equilibrium with an expected cost of 15.25. However, we already showed a better Nash Equilibrium with an expected cost of 15.125. We now prove an upper bound on the performance of Opt_{Nash}; the expected cost of Opt_{Nash} is no more than k times worse than the expected cost of Opt_{soc}.

Theorem 8.19. $\mathrm{E}[\mathit{Opt}_{Nash}] \leq k \cdot \mathrm{E}[\mathit{Opt}_{soc}]$.

Proof. Suppose that $\mathrm{E}[\mathsf{Opt}_{Nash}] > k \cdot \mathrm{E}[\mathsf{Opt}_{soc}]$. Therefore,

$$\mathrm{E}[\mathsf{Opt}_{Nash}] = \sum_i (p_i^t \cdot t_i) + \sum_i (p_i^b \cdot b_i) > k \cdot [\sum_j (p_j^t \cdot t_j) + \sum_j (p_j^b \cdot b_j)] = \mathrm{E}[\mathsf{Opt}_{soc}]$$

Then,

$$\sum_i (p_i^t \cdot t_i/k) + \sum_i (p_i^b \cdot b_i/k) > \sum_j (p_j^t \cdot t_j) + \sum_j (p_j^b \cdot b_j)$$

Since $t_i > t_i/k$ and $b_j > b_j/k$ then,

$$\sum_i (p_i^t \cdot t_i) + \sum_i (p_i^b \cdot b_i/k) > \sum_j (p_j^t \cdot t_j) + \sum_j (p_j^b \cdot b_j/k)$$

The left and right terms are the expected costs of Opt_{Nash} and Opt_{soc}, respectively, where all the buying costs are divided by k. Therefore, Opt_{Nash} is not optimal in these settings. Contradiction. □

As for the lower bound, consider the following example.

Example 8.20. *For any $\epsilon > 0$, suppose that the price at $u_2 = u_s^{(1)}$ is k with a probability of 1, and the price at the leftmost store, u_1 is 0 with a probability of 1. The traveling cost from u_2 to u_1 is $1 + \epsilon$. In all other stores the price is always 0, and the traveling costs between any other store to u_2*

is very high, for instance k. Opt_{soc} for this example is that agent 1 will go left and buy the product at u_1. The cost of this strategy is $1+\epsilon$, but it is not a Nash Equilibrium. Clearly, agent 1 will prefer to buy the product in its initial location, u_2, since its own cost will be 1, instead of $1+\epsilon$ in Opt_{soc}. The total cost from this strategy will be k, and it is the only Nash Equilibrium. Therefore, for any algorithm that finds a strategy S *which is a Nash Equilibrium,* $\mathrm{E}[\mathrm{S}] \in \Omega(\frac{k}{1+\epsilon} \cdot \mathrm{E}[\mathsf{Opt}_{soc}])$, *and the bound from Theorem 8.19 is tight.*

8.4 Heterogenous Agents

The analysis so far assumes that all agents are of the same type, with identical capabilities. Specifically, the cost of obtaining the item at any given store is assumed to be the same for all agents. However, agents may be of different *types* and hence with different capabilities. For example, some agents may be equipped with a drilling arm, which allows them to consume less battery power while mining. In this section we consider situations of *heterogenous agents*, and show that the results can be extended to such settings.

While agents may have different capabilities, in many cases it is reasonable to assume that if one agent is more capable than the other at one location, it is also more capable at all other locations (or at least not less capable). Hence the following definition:

Definition 8.21. *We say that agents are* inconsistent *if there exist budgets B, B', agents j, j', and locations i, i', such that at location i with budget B*

$$\Pr[j \textit{ can obtain the item}] < \Pr[j' \textit{ can obtain the item}]$$

but at location i' with budget B'

$$\Pr[j \textit{ can obtain the item}] > \Pr[j' \textit{ can obtain the item}]$$

We now show that the results of Subsection 8.2.1 can be extended to heterogenous agents.

Theorem 8.22. *In the private budget and no communication setting, if the number of different costs for each agent is constant, then* k-Private-Max-Probability *and* k-Private-Min-Budget^{identical} *can be solved in polynomial time with any number of heterogenous agents, provided that the agents are consistent.*

The algorithm is essentially the same dynamic programming algorithm described in Subsection 8.2.1. The consistency assumption is necessary for lemmata 8.7 and 8.8 to remain true.

In any other case, we can do away with the consistency assumption. Clearly, however, we do need to assume that upon reaching a site, agents can assess the cost for obtaining the item for all other agents. Otherwise, communication would be meaningless. We obtain:

Theorem 8.23. *In the setting of communicating agents, with a constant number of agents, and a constant number of different costs for each agent,* k-Shared-Min-Expected-Cost*,* k-Shared-Max-Probability*,* k-Shared-Min-Budget*,* k-Private-Max-Probability*,* k-Private-Min-Budget^{identical} *and* k-Private-Min-Budget^{distinct} *can be solved in polynomial time even with inconsistent heterogenous agents.*

The algorithms and proofs remain essentially the same as those for the case of homogenous agents.

8.5 Extending our Results - Discussion

In this work we mainly analyzed the case where the stores are located along a path (either closed or non-closed). There are many settings where this assumption holds. For example, the assumption faithfully captures the setting of perimeter patrol applications (see [40, 112]). Also, as pointed out in the introduction, many coverage algorithms convert their complex environment into a simple path. However, numerous physical environments may only be represented by a planar graph. Theorems 7.1 and 7.6 show that physical search problems are hard even on planar graphs and trees, even with a single agent, but finding a heuristic is of practical interest nonetheless. It seems that the first steps in building such a heuristic will be to utilize our results. For example, one should try to avoid repeated coverage as much as possible and restrict the number of cases where such coverage is necessary, as we showed in theorem 8.5. Another idea is to convert the complex graph structure into a path, where each site on the path represents a region of strongly-connected nodes on the original graph. Many graphs which represent real physical environments consist of some regions with strongly-connected nodes, but few edges connect these regions (for example, cities, have many roads inside but are connected by only a few highways). A heuristic algorithm for these graphs may use our algorithm to construct a strategy for the sites along the path, and use an additional heuristic for visiting the sites inside a region.

We also considered the case where mining costs are rounded/estimated to one of a constant number of possible options. We believe that this assumption is appropriate since the given input for our problems includes prior probabilistic knowledge. Usually, this data comes from some sort of estimation so it is reasonable to assume that the number of options is fixed. Nevertheless, if the number of costs will not be a constant it can be rounded to a fixed number of costs, which yields a PTAS (polynomial-time approximation scheme) for our problems.

We also assumed that the agents seek only one item. As soon as more than one item is needed, our results do not hold, and seemingly the problems become NP-complete.

Chapter 9

Future Directions and Final Remarks

We summarize the key contributions of this thesis in Section 9.1, and suggest future directions for this research in Section 9.2.

9.1 Summary of Key Contributions

In the first part of this dissertation we focus on the computational aspects of voting procedures under uncertainty. We analyze computational aspects of three major problems within the computational voting theory: winner determination, control and manipulation. Our main contribution in this part of the work is as follows.

- When the number of candidates is a constant, we provide a polynomial time algorithm which computes the probability that a candidate will win an election, given probabilistic information (the EVALUATION problem). The algorithm can handle many voting rules whether voter weights are equal, or not. When the number of candidates is not bounded, we prove that the aforementioned EVALUATION problem is #P-hard to compute, for Plurality, k-approval, Borda, Copeland and Bucklin voting rules. We prove that even checking whether a candidate has any chance of winning (the CHANCE-EVALUATION problem) with the Plurality voting rule is NP-complete when the voters are weighted. For the unweighted voters case, we prove that the CHANCE-EVALUATION problem remains NP-complete for k-approval, Borda, Copeland, Bucklin and Maximin rules. However, for the Plurality protocol we propose a polynomial time algorithm. We also provide a simple Monte Carlo algorithm that is able to approximately compute the probability of a candidate to win under any setting, with an error as small as we would like.

- With voting trees, we show that calculating the probability of a candidate to be chosen is easy, even when the only given is the probability that a candidate will be preferred over another. We provide an optimized algorithm for this problem for linear order and fair tree order. We demonstrate the unfairness of the linear order rule, and prove that finding an agenda which would make a specific candidate the winner with a non-zero probability (weak agenda rigging) is easy with linear order. However, with fair tree order, we show that the agenda rigging problem is provably hard. We also demonstrate that, while it seems hard to control an election by rigging the agenda in theory, there are heuristics that perform well on this problem in practice.

- initiate the computational analysis of a new model for coalition manipulation, termed safe manipulation, that was recently introduced by Slinko and White [101]. We show that finding a safe manipulation is easy for k-approval for an arbitrary value of k and for Bucklin, even with weighted voters. We prove that checking whether a given manipulation is safe is polynomial-time solvable for k-approval, but is coNP-hard for Bucklin. For the Borda rule, we show that both checking whether a given manipulation is safe and identifying a safe manipulation is hard when the voters are weighted. We also propose two ways of extending the notion of safe manipulation to heterogeneous group of manipulators, and initiate the study of computational complexity of related questions.

In the second part of the dissertation, we consider collaborative physical search problems with uncertain knowledge. We analyze single and multi-agent settings, with various models and assumptions. The contribution of this part is summarized as follows.

- With a single agent, we define three variants of our general problem: *Min-Expected-Cost*, *Max-Probability* and *Min-Budget*. We prove that these problems are hard on a metric space, sometimes even if it is a tree. We thus focus on the path case, presenting a polynomial algorithm for the *Min-Expected-Cost* problem, and proving hardness for the *Max-Probability* and the *Min-Budget* problems. We provide an FPTAS for *Min-Budget* and show that both problems are polynomial if the number of possible prices is bounded.

- With multiple agents, we analyze shared and private budget models. With shared budget, we show how all of the single-agent algorithms extend to k agents, with the time bounds growing exponentially in k. We prove that this is also the case with the private budget model, if the agents can communicate. In the case of private budget with a setting of no communicating, we present a polynomial algorithm that is suitable for any number of agents. We also extend the analysis to heterogenous agents. Finally, we consider the self-interested agents setting, showing how to find a Nash Equilibrium for the simultaneous game (*Min-Budget-Game*) and the sequential game (*Min-Expected-Cost-Game*) in polynomial time. In both cases, we show an upper bound on the ratio between this solution and the optimal one (the one which maximizes social welfare) and prove that it is tight.

9.2 Future Directions

In the context of the first part of this dissertation, the following points have been left open for future work.

- **Evaluation of election outcomes under uncertainty** We would like to extend our current analysis to more voting rules, including multi-winner protocols. Even with the current protocols that we have considered there are still open questions, i.e. the complexity of EVALUATION with un-weighted voters under Maximin and the CHANCE-EVALUATION with un-weighted voters under STV. Another extension we would like to consider would be to define and analyze a general imperfect knowledge model, which would combine Konczak and Lang's model [75] of incomplete preferences with our model of probabilistic estimation of the voters' preferences. Moreover, we would like to improve our results for the current voting rules. It would be useful to have an approximation algorithm (or prove that one cannot be found) for the problems that are #P-hard or NP-complete to compute. Another promising direction would be to use the parameterized complexity paradigm [50] to analyze our problems. We have already showed that if we bind one of our parameters, namely, the number of candidates, the Evaluation problem becomes easy to solve (and thus our problem belongs to XP, but not to FPT). It would be interesting to check whether other restrictions could help, for example, if the number of different probability distributions is bounded by a constant.

- **How to rig elections and competitions** The complexity of finding an agenda which would make a specific candidate the winner with at least a specific probability, is still open for linear order (IIAR^l problem). The complexity of finding an agenda which would make a specific candidate the winner with a non-zero probability is still open for fair tree order (IIWAR^f).

- **Complexity of safe strategic voting** A natural question which has been left open is determining the complexity of finding a safe strategic vote for voting rules not considered in this work, such as Copeland, Ranked Pairs, or Maximin. Furthermore, the picture depicted in this work is incomplete for some of the voting rules we have investigated. In particular, it would be interesting to understand the computational complexity of finding a safe manipulation for Borda (and, more generally, for all scoring rules) for unweighted voters. The problem for Borda is particularly intriguing as this is perhaps the only widely studied voting rule for which the complexity of unweighted coalitional manipulation in the standard model is not known. Other exciting research directions include formalizing and investigating the problem of selecting the best safe manipulation (is it the one that succeeds more often, or one that achieves better results when it succeeds?), and extending our analysis to other types of tie-breaking rules, such as randomized tie-breaking rules. However, the latter question may require modifying the notion of a safe manipulation, as the outcome of a strategic vote becomes a probability distribution over the alternatives.

There are still many interesting open problems concerning the second part of this dissertation. With a single agent, the complexity of the *Min-Expected-Cost* problem on a tree remains open. This case is interesting since it can be shown that *Min-Expected-Cost* is easy for a specific tree, namely a star graph, where d is bounded. In the shared budget model, the complexity of the *k-Shared-Min-Expected-Cost* problem where k is part of the input is still unresolved. In the private budget model, the complexity of all the problems with a non-constant number of communicating agents is unsolved. In addition, there are interesting extensions to consider. We showed that most of our results can be extended to heterogenous agents, with different *buying* capabilities. The next step should be to analyze our results with heterogenous agents with different *traveling* capabilities. Another direction would be to add a time constraint, which would possibly result in completely different optimal strategies. Also, the *Min-Budget-Game* can be extended; instead of defining the utility of achieving p_{succ} as a step function ("high" if p_{succ} is achieved and "low" if not), it could be defined as a linear function of p_{succ}. Finally, metric spaces beyond the line remain a challenge. As we discussed in Section 8.5, it would be interesting to check the use of our techniques in building approximations and/or heuristics for the general metric space.

Appendix

Appendix A

Proofs for Chapter 4

A.1 Correctness Proof for Algorithm 1

Theorem A.1. *Given an imperfect information model of voters' preferences, as described in Section 4.1, Algorithm 1 enumerate all the possible voting scenarios in polynomial time, when the number of candidates is a constant.*

Proof. Let VR^{i} be the set of voting results, that the algorithm generates in iteration i. We prove that the algorithm enumerates all the relevant voting scenarios, by induction on the number of voters. If there is only one voter, the algorithm will generate at most l voting results from the voter's preferences. Clearly, these are all possible voting scenarios for this voter. Otherwise, consider the n-th iteration. Every voting scenario of n voters consist of a voting scenario of $n-1$ voters plus one preference order of the n-th voter. By the inductive hypothesis, all the possible voting scenarios of $n-1$ voters are summarized by $\mathsf{VR}^{\mathsf{n-1}}$. Thus, combining every voting results from $\mathsf{VR}^{\mathsf{n-1}}$ with every preference order of the n-th voter generates VR^{n}, which summarizes all the voting scenarios of n voters, as required.

As for the running time, the total number of voting results is polynomial in n (since m is a constant), and for each voting result there are $O(l)$ operations for generating other voting results. Generating voting results takes polynomial time, by definition. Thus, the algorithm's running time is polynomial in n and l. □

Appendix B

Proofs for Chapter 5

B.1 Proof of Theorem 5.2

Theorem 5.2. *Given $\langle T, \alpha, M\rangle$, where T is fair tree order, the* evaluation *of T with respect to α and M is computable in $O(m^2)$ arithmetic operations.*

Proof. Let T_k^j be a sub tree of T with height j which contains c_k in one of its leaves. Let $L(T)$ and $R(T)$ be the left and right sub trees of T, respectively, and denote by $(T)_1, (T)_2, ..., (T)_m$ the candidate on the first, second, ..., last leaf of T, respectively, when ordering the leaves from left to right.

Consider the $log(m) \times m$ matrix, F whose entries, $F[j,k]$ are,

$$F[j,k] = P[winner(T_k^j) = c_k | M)$$

Informally, $F[j,k]$ is the probability that candidate c_k is the winner of the sub tree T_k^j. F may be constructed in polynomial time given $\langle T, \alpha, M\rangle$ in the following way.

$F[1,1], F[1,2], ..., F[1,m]$ can be determined directly from M. Each subsequent row of F can be computed from the previous row according to the following rule,

$$F[j,k] = \begin{cases} F[j-1,k] \times \sum_{r=1}^{j}(M[c_k, c_{(R(T_k^j))_r}] \times F[j-1, (R(T_k^j))_r]) & \text{if } c_k \in L(T_k^j) \\ F[j-1,k] \times \sum_{r=1}^{j}(M[c_k, c_{(L(T_k^j))_r}] \times F[j-1, (L(T_k^j))_r]) & \text{if } c_k \in R(T_k^j) \end{cases}$$

Clearly, $\eta(r(T)) = F[log(m), \cdot]$. Furthermore, for each $2 \leq j \leq log(m)$ we use 2^j arithmetic operations for each candidate. So the overall complexity is: $m \times \sum_{i=2}^{log(m)} 2^i = m \times (\sum_{i=0}^{log(m)} 2^i - 3) = m \times (2^{log(m)+1} - 1 - 3) = O(m^2)$ $\square$

Appendix C

Proofs for Chapter 8

C.1 Proof of Theorem 8.1

Theorem 8.1. *With k agents,* k-Shared-Min-Expected-Cost *can be solved in $O(d^2 2^k (\frac{m}{k})^{2k})$.*

Proof. Since the stores are on the path, at any point in time the points/stores visited by the agents constitute a set of k disjoint contiguous intervals, which we call the *visited intervals.* Clearly, the algorithm need only make decisions at store locations. Furthermore, decisions can be limited to times when the agents are at one of the two stores edges of the *visited interval.* At each such location, each agent has only three possible actions: "go right" - extending its visited-interval one store to the right, "go left" - extending its visited-interval one store to the left, or "stop" - stopping the search and buying the product at the best price so far. Also note that *after* each agent i has already visited its interval $[u_\ell^{(i)}, u_r^{(i)}]$, how exactly it covered this interval does not matter for any future decision; the costs have already been incurred. Accordingly, the states of the MDP are quadruplets $[L, R, E, c]$, such that $L = (\ell^{(1)}, \ell^{(2)}, \ldots, \ell^{(k)})$, $R = (r^{(1)}, r^{(2)}, \ldots, r^{(k)})$, $E = (e^{(1)}, e^{(2)}, \ldots, e^{(k)})$ and $c \in D$. For each agent i, $\ell^{(i)} \leq s^{(i)} \leq r^{(i)}$ and $e^{(i)} \in \{\ell^{(i)}, r^{(i)}\}$. Every such state represents the situation that each agent i visited stores $u_{\ell^{(i)}}$ through $u_{r^{(i)}}$, it is currently at location $u_{e^{(i)}}$, and the best price encountered so far is c. Since the intervals are disjoint, $r^{(i)} < \ell^{(i+1)}$ for every i.

The terminal states are *Buy(c)* and all states such that $\bigcup_i [\ell^{(i)}, r^{(i)}] = [1, m]$. The terminal cost is c. For all other states there are at most $2k + 1$

possible actions - "agent i go right" (provided that $r^{(i)} < \ell^{(i+1)}$ and $r^{(i)} < m$), "agent i go left" (provided that $r^{(i-1)} < \ell^{(i)}$ and $1 < \ell^{(i)}$), or "stop". The cost of "agent i go right" is $(u_{r^{(i)}+1} - u_{e^{(i)}})$, while the cost of "agent i go-left" is $(u_{e^{(i)}} - u_{\ell^{(i)}-1})$. The cost of "stop" is always 0. Given a vector V, let $V^i(j)$ be the same vector but with value j at index i. Given the state $[L, R, E, c]$ and move "agent i go-right", there is probability $p^{r^{(i)}+1}(c')$ to transition to state $[L, R^i(r^{(i)}+1), E^i(r^{(i)}+1), c']$, for $c' < c$. With the remaining probability, the transition is to state $[L, R^i(r^{(i)}+1), E^i(r^{(i)}+1), c]$. Transition to all other states has zero probability. Transitions for the "agent i go left" actions are analogous, while with the action "stop" there is probability 1 to transition to state *Buy(c)*. This fully defines the MDP. The optimal strategy for finite-horizon MDPs can be determined using dynamic programming (see [90, Ch.4]). In our case, the complexity can be brought down to $O(d^2 2^k (\frac{m}{k})^{2k})$ steps (using $O(d2^k(\frac{m}{k})^{2k})$ space). $\square$

C.2 Proof of Theorem 8.2

Theorem 8.2. *With k agents,* k-Shared-Min-Budget *and* k-Shared-Max-Probability *with d possible prices can be solved in $O(m2^{kd}(\frac{em}{2kd})^{2kd})$.*

Proof. For brevity, we focus on the *k-Shared-Max-Probability* problem. The same algorithm and similar analysis work also *k-Shared-Min-Budget* problem. Let $c_1 > c_2 > \cdots > c_d$ be the set of costs. For each agent j and for each c_i there is an interval $I_i^{(j)} = [u_\ell, u_r]$ of points covered while the remaining budget is at least c_i. Furthermore, for each j and for all i, $I_i^{(j)} \subseteq I_{i+1}^{(j)}$. Thus, consider for each agent the *incremental* area covered with remaining budget c_i but less than c_{i-1}, $\Delta_i^{(j)} = I_i^{(j)} - I_{i-1}^{(j)}$ (with $\Delta_1^{(j)} = I_1^{(j)}$). Each $\Delta_i^{(j)}$ is a union of an interval at left of $u_s^{(j)}$ and an interval at the right of $u_s^{(j)}$ (both possibly empty). The next lemma, which is the multi-agent Max-Probability analogue of Lemma 7.4 states that there are only two possible optimal strategies to cover each $\Delta_i^{(j)}$:

Lemma C.1. *Consider the optimal solution and the incremental areas for each agent j, $\Delta_i^{(j)}$ $(i = 1, \ldots, d)$ defined by this solution. For $i \in 1, \ldots, d$, let $u_{\ell_i}^{(j)}$ be the leftmost store in $\Delta_i^{(j)}$ and $u_{r_i}^{(j)}$ the rightmost store. Suppose that in the optimal strategy the covering of $\Delta_i^{(j)}$ starts at location $u_{s_i}^{(j)}$. Then, WLOG* *we may assume that the optimal strategy for each j is either $(u_{s_i}^{(j)} \rightarrowtail u_{r_i}^{(j)} \rightarrowtail u_{\ell_i}^{(j)})$ or $(u_{s_i}^{(j)} \rightarrowtail u_{\ell_i}^{(j)} \rightarrowtail u_{r_i}^{(j)})$. Furthermore, the starting point for covering $\Delta_{i+1}^{(j)}$ is the ending point of covering $\Delta_i^{(j)}$.*

Proof. Any strategy other than the ones specified in the lemma would reach all the stores covered by the optimal solution with at most the same available budget. □

By the previous lemma, the moves of each agent are fully determined by the leftmost and rightmost stores of each $\Delta_i^{(j)}$, together with the choice for the ending points of covering each area. For each two agents j_1, j_2, the intervals of covered points are disjoint, i.e. $I_d^{(j_1)} \cap I_d^{(j_2)} = \emptyset$. Therefore, for each j there are at most $\frac{(\frac{m}{k})^{2d}}{(2d)!} \leq (\frac{em}{2kd})^{2d}$ possible choices for the external stores of the $\Delta_i^{(j)}$'s, and there are a total of 2^d options to consider for the covering of each. For each option, computing the budget and probability takes $O(m)$ steps. Thus, the total time is $O(m2^{kd}(\frac{em}{2kd})^{2kd})$, which is polynomial (in m). □

C.3 Proof of Theorem 8.3

Theorem 8.3. *With k agents, For any $\epsilon > 0$,* k-Shared-Min-Budget *can be approximated to within a factor of $(1 + k\epsilon)$ in $O(n\epsilon^{-6k})$ steps (for arbitrary number of prices).*

Proof. For k agents, we extend the dynamic programming algorithm, which calculates $\mathsf{fail}[\cdot,\cdot,\cdot,\cdot]$ and $\mathsf{act}[\cdot,\cdot,\cdot,\cdot]$ tables, in the same way we extended the single agent algorithm in the proof of Theorem 8.1. We now save k disjoint intervals, thus the tables size becomes $O(\epsilon^{-6k})$. The rest of the approximation algorithm remains essentially the same. We still consider δ in powers of 2 up to $\beta \leq 2^n$, where n is the size of the input. Thus, the total computation time is $O(n\epsilon^{-6k})$. Since the approximation ratio in each interval is guaranteed to be $(1 + \epsilon)$, we get a total ratio of $(1 + k\epsilon)$. □

Bibliography

[1] F. Afrati, S. Cosmadakis, C. Papadimitriou, G. Papageorgiou, and N. Papakonstantinou. The complexity of the traveling repairman problem. *Theoretical Informatics and Applications*, 20:79–87, 1986.

[2] E. Arkin, J. S. B.Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry (SCG-1998)*, pages 307–316, 1998.

[3] S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC-1999)*, pages 688–693, 1999.

[4] S. Arora and G. Karakostas. A $2+\epsilon$ approximation algorithm for the k-mst problem. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2000)*, pages 754–759, 2000.

[5] K. J. Arrow. *Social Choice and Individual Values.* Yale University Press, 1951.

[6] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*, volume 1. Elsevier Science Publishers B.V., 2002.

[7] Y. Aumann, N. Hazon, S. Kraus, and D. Sarne. Physical search problems applying economic search models. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008)*, pages 9–16, 2008.

[8] G. Ausiello, S. Leonardi, and A. Marchetti-Spaccamela. *Algorithms and Complexity*, chapter On Salesmen, Repairmen, Spiders, and Other Traveling Agents, pages 1–16. Springer Berlin / Heidelberg, 2000.

[9] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guarantees for minimum weight k-trees and prize-collecting salesmen. *SIAM Journal on Computing*, 28(1):254–262, 1999.

[10] Y. Bachrach, N. Betzler, and P. Faliszewski. Probabilistic possible winner determination. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010)*, pages 697–702, 2010.

[11] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[12] J. Bang-Jensen and G. Gutin. On the complexity of hamiltonian path and cycle problems in certain classes of digraphs. *Discrete Applied Mathematics*, 95(1-3):41–60, 1999.

[13] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing (STOC-2004)*, pages 166–174, 2004.

[14] J. J. Bartholdi and J. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

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

[16] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[17] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. How hard is it to control an election? *Mathematical and Computer Modelling*, 16:27–40, 1992.

[18] D. Baskins. Judy IV. http://judy.sourceforge.net/, 2001.

[19] N. Betzler and B. Dorn. Towards a dichotomy of finding possible winners in elections based on scoring rules. In *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science (MFCS-2009)*, volume 5734 of *Lecture Notes in Computer Science (LNCS)*, pages 124–136. Springer-Verlag, 2009.

[20] N. Betzler and B. Dorn. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *Journal of Computer and System Sciences*, 2010. To appear.

[21] N. Betzler, S. Hemmann, and R. Niedermeier. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009)*, pages 53–58, 2009.

[22] L. Bianco, A. Mingozzi, and S. Ricciardelli. The traveling salesman problem with cumulative costs. *Networks*, 23(2):81–91, 1993.

[23] D. Black. On the rationale of group decision-making. *Journal of Political Economy*, 56(1):23–34, 1948.

[24] D. Black, editor. *The theory of committees and elections.* Cambridge University Press, 1958.

[25] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing (STOC-1994)*, pages 163–171, 1994.

[26] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM Journal on Computing*, 37(2):653–670, 2007.

[27] A. Blum, R. Ravi, and S. Vempala. A constant-factor approximation algorithm for the k-mst problem. *Journal of Computer and System Sciences*, 58(1):101–108, 1999.

[28] S. J. Brams and P. C. Fishburn. Voting procedures. In K. J. Arrow, A. K. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare Volume 1*, chapter 4. Elsevier, 2002.

[29] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tour. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS-2003)*, pages 36–45, 2003.

[30] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2008)*, pages 661–670, 2008.

[31] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A short introduction to computational social choice. In *Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM-2007)*, volume 4362 of *Lecture Notes in Computer Science (LNCS)*, pages 51–69. Springer-Verlag, 2007.

[32] Y. Chevaleyre, J. Lang, N. Maudet, and G. Ravilly-Abadie. Compiling the votes of a subelectorate. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009)*, pages 97–102, 2009.

[33] S. P. M. Choi and J. Liu. Optimal time-constrained trading strategies for autonomous agents. In *Proceedings of the International ICSC Symposium on Multi-agents and Mobile Agents in Virtual Organizations and E-commerce (MAMA-2000)*, pages 11–13, 2000.

[34] V. Conitzer. *Computational Aspects of Preference Aggregation.* PhD thesis, Department of Computer Science, Carnegie Mellon University, 2006.

[35] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 314–319, 2002.

[36] V. Conitzer and T. Sandholm. Vote elicitation: complexity and strategy-proofness. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 392–397, 2002.

[37] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):1–33, 2007.

[38] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC-2005)*, volume 3827 of *Lecture Notes in Computer Science (LNCS)*, pages 206–215. Springer-Verlag, 2005.

[39] E. Elkind and H. Lipmaa. Small coalitions cannot manipulate voting. In *Financial Cryptography and Data Security*, volume 3570 of *Lecture Notes in Computer Science (LNCS)*, pages 285–297. Springer-Verlag, 2005.

[40] Y. Elmaliach, A. Shiloni, and G. A. Kaminka. A realistic model of frequency-based multi-robot fence patrolling. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008)*, pages 63–70, 2008.

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

[42] J. Fakcharoenphol, C. Harrelson, and S. Rao. The k-traveling repairman problem. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2003)*, pages 655–664, 2003.

[43] J. Fakcharoenphol, C. Harrelson, and S. Rao. The k-traveling repairmen problem. *ACM Transactions on Algorithms*, 3(4):40, 2007.

[44] P. Faliszewski. Nonuniform bribery. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008)*, pages 1569–1572, 2008.

[45] P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Llull and copeland voting broadly resist bribery and control. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-2007)*, pages 724–730, 2007.

[46] P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Copeland voting fully resists constructive control. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM-2008)*, pages 165–176, 2008.

[47] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: ties matter. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008)*, pages 983–990, 2008.

[48] T. S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4(3):282–289, 1989.

[49] M. Fischetti, G. Laporte, and S. Martello. The delivery man problem and cumulative matroids. *Operations Research*, 41:1065–1064, 1993.

[50] J. Flum and M. Grohe, editors. *Parameterized Complexity Theory.* Springer, 2006.

[51] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31:77–98, 2001.

[52] S. Gal. *Search Games.* Academic Press, 1980.

[53] A. García, P. Jodrá, and J. Tejel. A note on the traveling repairman problem. *Networks*, 40:27–31, 2002.

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

[55] N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS-1996)*, pages 302–309, 1996.

[56] N. Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing (STOC-2005)*, pages 396–402, 2005.

[57] A. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41:587–601, 1973.

[58] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-1996)*, pages 152–158, 1996.

[59] C. Groh, B. Moldovanu, A. Sela, and U. Sunde. Optimal seedlings in elimination tournaments. *Economic Theory*, 2008. To appear.

[60] N. Hazon. Social interaction under uncertainty in multi agent systems. In *Proceedings of the Thirteenth Annual AAAI/SIGART Doctoral Consortium (In association with AAAI-2008)*, pages 1851–1852, 2008.

[61] N. Hazon, Y. Aumann, and S. Kraus. Collaborative multi agent physical search with probabilistic knowledge. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009)*, pages 167–164, 2009.

[62] N. Hazon, Y. Aumann, S. Kraus, and M. Wooldridge. Evaluation of election outcomes under uncertainty. In *Proceedings of the DIMACS Workshop on the Boundary between Economic Theory and Computer Science*, 2007.

[63] N. Hazon, Y. Aumann, S. Kraus, and M. Wooldridge. Evaluation of election outcomes under uncertainty. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008)*, pages 959–966, 2008.

[64] N. Hazon, Y. Aumann, S. Kraus, and M. Wooldridge. On the evaluation of election outcomes under uncertainty. *Artificial Intelligence*, 2010. Under submission.

[65] N. Hazon, P. E. Dunne, S. Kraus, and M. Wooldridge. How to rig an election. In *Proceedings of the 9th Bar-Ilan Symposium on the Foundations of Artificial Intelligence (BISFAI-2007)*, 2007.

[66] N. Hazon, P. E. Dunne, S. Kraus, and M. Wooldridge. How to rig elections and competitions. In *Proceedings of the Second International Workshop on Computational Social Choice (COMSOC-2008)*, pages 301–312, 2008.

[67] N. Hazon and E. Elkind. Complexity of safe strategic voting. In *Proceedings of the 3rd International Symposium on Algorithmic Game Theory (SAGT-10)*, 2010.

[68] N. Hazon and E. Elkind. Complexity of safe strategic voting. In *Proceedings of the Third International Workshop on Computational Social Choice (COMSOC-2010)*, 2010.

[69] N. Hazon and E. Elkind. Complexity of safe strategic voting. In *Proceedings of the First Workshop on Cooperative Games in Multiagent Systems (CoopMAS-2010)*, 2010.

[70] N. Hazon and G. A. Kaminka. Redundancy, efficiency, and robustness in multi-robot coverage. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA-2005)*, pages 735–741, 2005.

[71] E. Hemaspaandra and L. A. Hemaspaandra. Dichotomy for voting systems. *Journal of Computer and System Sciences*, 73(1):73–83, 2007.

[72] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Exact analysis of dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.

[73] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Hybrid elections broaden complexity-theoretic resistance to control. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 1308–1314, 2007.

[74] O. Hudry. A note on banks winners in tournaments are difficult to recognize by g. j. woeginger. *Social Choice and Welfare*, 23(1):113–114, 2004.

[75] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proceedings of the Multidisciplinary IJCAI-05 Workshop on Advances in Preference Handling (M-PREF)*, 2005.

[76] B. O. Koopman. *Search and Screening: General Principles with Historical Applications*. Pergamon Press, 1980.

[77] E. Koutsoupias, C. H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming (ICALP-1996)*, pages 280–289, 1996.

[78] D. Lacy and E. M. S. Niou. A problem with referendums. *Journal of Theoretical Politics*, 12(1):5–31, 1998.

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

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

[81] S. Lippman and J. McCall. The economics of job search: A survey. *Economic Inquiry*, 14:155–189, 1976.

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

[83] A. Lucena. Time-dependent traveling salesman problemthe deliveryman case. *Networks*, 20(6):753–763, 1990.

[84] J. McMillan and M. Rothschild. Search. In R. Aumann and S. Amsterdam, editors, *Handbook of Game Theory with Economic Applications*, chapter 27, pages 905–927. Elsevier, 1994.

[85] E. Minieka. The delivery man problem on a tree network. *Annals of Operations Research*, 18(1–4):261–266, 1989.

[86] M. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, 1994.

[87] M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Incompleteness and incomparability in preference aggregation. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 1464–1469, 2007.

[88] M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Dealing with incomplete agents preferences and an uncertain agenda in group decision making via sequential majority voting. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR-2008)*, pages 571–578, 2008.

[89] 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.

[90] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

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

[92] J. Rothe, , H. Spakowski, and J. Vogel. Exact complexity of the winner problem for young elections. *Theory of Computing Systems*, 36(4):375–386, 2003.

[93] S. Sahni and T. Gonzales. P-complete problems and approximate solutions. In *Proceedings of the 15th Annual Symposium on Switching and Automata Theory (SWAT-1974)*, pages 28–32, 1974.

[94] T. Sandholm. Distributed rational decision making. In G. Weiß, editor, *Multiagent Systems*, pages 201–258. The MIT Press: Cambridge, MA, 1999.

[95] D. Sarne and S. Kraus. Cooperative exploration in the electronic marketplace. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-2005)*, pages 158–163, 2005.

[96] M. A. Satterthwaite. Strategy-proofness and arrows conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

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

[98] D. T. Searls. On the probability of winning with different tournament procedures. *Journal of the American Statistical Association*, 58(304):1064–1081, 1963.

[99] D. Simchi-Levi and O. Berman. Minimizing the total flow time of n jobs on a network. *IIE Transactions*, 23(3):236–244, 1991.

[100] R. Sitters. The minimum latency problem is np-hard for weighted trees. In *Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO-2002)*, pages 230–239, 2002.

[101] A. Slinko and S. White. Non-dictatorial social choice rules are safely manipulable. In *Proceedings of the Second International Workshop on Computational Social Choice (COMSOC-2008)*, pages 403–413, 2008.

[102] S. V. Spires and S. Y. Goldsmith. Exhaustive geographic search with mobile robots along space-filling curves. In *First International Workshop on Collective Robotics*, pages 1–12, 1998.

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

[104] T. Vu, A. Altman, and Y. Shoham. On the complexity of schedule control problems for knockout tournaments. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2009)*, pages 225–232, 2009.

[105] T. Vu, N. Hazon, A. Altman, Y. Shoham, S. Kraus, and M. Wooldridge. On the complexity of schedule control problems for knock-out tournaments. *Journal of Artificial Intelligence Research*, 2010. Under submission.

[106] T. Walsh. Uncertainty in preference elicitation and aggregation. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-2007)*, pages 3–8, 2007.

[107] T. Walsh. Where are the really hard manipulation problems? the phase transition in manipulating the veto rule. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009)*, pages 324–329, 2009.

[108] I. R. Webb. Depth-first solutions for the deliveryman problem on tree-like networks: an evaluation using a permutation model. *Transportation Science*, 30(2):134–147, 1996.

[109] D. B. West, editor. *Introduction to Graph Theory.* Prentice Hall, 2 edition, 2001.

[110] R. J. V. Wiel and N. V. Sahinidis. Heuristic bounds and test problem generation for the time dependent traveling salesman problem. *Transportation Science*, 29(2):167–183, 1995.

[111] T. Will. *Extremal Results and Algorithms for Degree Sequences of Graphs.* PhD thesis, University of Illinois at Urbana-Champaign, 1993.

[112] K. Williams and J. Burdick. Multi-robot boundary coverage with plan revision. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA-2006)*, pages 1716–1723, 2006.

[113] G. J. Woeginger. Banks winners in tournaments are difficult to recognize. *Social Choice and Welfare*, 20(3):523–528, 2003.

[114] M. Wooldridge. *An Introduction to Multiagent Systems.* John Wiley & Sons, 2002.

[115] L. Xia and V. Conitzer. Determining possible and necessary winners under common voting rules given partial orders. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008)*, pages 196–201, 2008.

[116] L. Xia and V. Conitzer. Compilation complexity of common voting rules. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010)*, pages 915–920, 2010.

[117] L. Xia, J. Lang, and M. Ying. Sequential voting rules and multiple elections paradoxes. In *The Eleventh Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2007)*, pages 279–288, 2007.

[118] L. Xia, M. Zuckerman, A. D. Procaccia, V. Conitzer, and J. S. Rosenschein. Complexity of unweighted coalitional manipulation. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009)*, pages 348–353, 2009.

[119] C. Yang. A dynamic programming algorithm for the travelling repairman problem. *Asia-Pacific Journal of Operations Research*, 6(10):192–2061, 1989.

[120] H. P. Young. Extending condorcets rule. *Journal of Economic Theory*, 16:335–353, 1977.

[121] M. Zuckerman, A. D. Procaccia, and J. S. Rosenschein. Algorithms for the coalitional manipulation problem. *Artificial Intelligence*, 173(2):392–412, 2009.

תוכן העניינים

החלק השני בעבודתי חוקר בעיות חיפוש שיתופי בסביבה פיזית, כאשר קיים מידע הסתברותי. במקרה זה סוכן יחיד או צוות של סוכנים מחפש מוצר כלשהו שיש סיכוי שנמצא במספר מקומות שונים בסביבה. העלות של השגת המוצר בכל נקודה אינה ידועה מראש- ההנחה היא שרק ההתפלגות על המחירים האפשריים בכל נקודה ידועה, וניתן לדעת מהי העלות האמיתית רק כאשר אחד הסוכנים מבקר פיזית באותו מקום. הגדרתי את הבעיות בסביבה כזו עבור סוכן יחיד וניתחתי אותן. הראיתי שחלק מהבעיות ניתנות לפתרון יעיל והוכחתי שקשה לתת פתרון מדויק לחלק האחר. לכן, הצגתי סכימה אלגוריתמית לקירוב יעיל עבור אחת הבעיות. לאחר מכן הרחבתי את התוצאות לסביבה מרובת סוכנים, וניתחתי שני מודלים לשיתוף המשאבים, מודל תקציב שיתופי ומודל תקציב פרטי. הצגתי אלגוריתם יעיל למספר קבוע של סוכנים, הן עבור מדל התקציב השיתופי והן עבור מודל התקציב הפרטי. בנוסף, הצגתי אלגוריתם יעיל שפועל עבור כל מספר של סוכנים במידה ואין תקשורת במודל התקציב הפרטי. לבסוף, הגדרתי את בעיות החיפוש השיתופי בסביבה בה הסוכנים אינם שיתופיים באופן מלא, והצגתי אלגוריתמים למציאת שיווי-משקל נאש בצורה יעילה. הוכחתי חסם עליון על ביצועי האלגוריתמים (ביחס לרווחה החברתית) והוכחתי שהוא הדוק.

תקציר

המחקר של מערכות מרובות סוכנים מתמודד עם סביבות בהן ישנם מספר סוכנים בעלי יחסי גומלין. חדירת טכנולוגית האינטרנט, כמו גם הפיתוחים האחרונים בתחום הרובוטים האוטונומיים, הינם הסיבה העיקרית למספר רב של מחקרים העוסקים בתחום של מערכות מרובות סוכנים. במערכות אלו, בהן יש מספר סוכנים המתקשרים זה עם זה, עולה ביתר שאת הצורך במחקר של האינטראקציות החברתיות בין הסוכנים, כגון תיאום, שיתוף פעולה וקבלת החלטות. הרבה עבודות בתחום של מערכות מרובות סוכנים חקרו אינטראקציות כאלו, בהנחה שכל המידע נגיש וקיים. ברם, סוכנים אמורים לקבל החלטות בצורה נבונה גם במקרים בהם ישנו מימד של חוסר וודאות. בעבודה זו, חקרתי את היסודות לבניית סוכנים כאלו. באופן ספציפי, חקרתי את המאפיינים החישוביים של שתי אינטראקציות חברתיות נפוצות, קבלת החלטות על ידי בחירות וחיפוש שיתופי, תוך כדי שימוש במודלים הסתברותיים המאירים באור חדש בעיות אלו.

החלק הראשון של עבודה זו עוסק במאפיינים חישוביים של בחירות, כאשר ישנו חוסר וודאות. הבעיה הראשונה אותה חקרתי היא בעיית קביעת המנצח, שבסביבה של מידע הסתברותי נקראת בעיית האבלואציה. בבעיה זו הקלט הוא ההתפלגות על העדפות הבוחרים וכלל בחירות מסוים, ויש צורך לחשב את ההסתברות של מועמד מסוים לזכות בבחירות. כאשר מספר המועמדים חסום, הצעתי אלגוריתם יעיל לפתרון הבעיה, והצגתי תוצאות ניסויים המדגימות את ביצועי האלגוריתם בסימולציות. אולם, הוכחתי שבעיית האבלואציה נעשית קשה עבור מספר נרחב של כללי בחירות כאשר מספר המועמדים אינו חסום. הראיתי גם שבהרבה מקרים קשה אפילו לבדוק האם למועמד מסוים יש איזשהו סיכוי לזכות, ולכן הצעתי אלגוריתם קירוב לבעיה זו ולבעיית האבלואציה שהוזכרה קודם.

בהמשך העבודה חקרתי מודל הסתברותי שונה, כאשר כל מה שידוע זו ההסתברות שמועמד מסוים יועדף על מועמד אחר כאשר הם מושווים זה מול זה. ניתן להשתמש במידע זה במשפחה של כללי בחירות שנקראת "עצי בחירות", שנפוצה בעיקר בתחרויות ספורט. במקרה זה חקרתי לא רק את בעיית חישוב הסתברות הזכייה של מועמד (בעיית האבלואציה) אלא גם את האפשרות למניפולציה על ידי מארגני הבחירות (בעיית השליטה). במסגרת של עצי בחירות, מניפולציה כזו יכולה לבוא לידי ביטוי בקביעת סדר ההתמודדויות בצורה מסוימת. הראיתי שניתן לפתור את בעיית האבלואציה בצורה יעילה, אך קשה עבור מקרים מסוימים לפתור את בעיית השליטה בה קובעים את סדר ההתמודדויות. הצעתי היוריסטיקות לקביעת סדר התמודדויות, וחקרתי את ביצועיהן על מידע הסתברותי סינתטי ומידע אמיתי שנלקח מתחרויות טניס וכדורסל.

לסיום החלק הראשון, חקרתי מאפיינים חישוביים של בעיית המניפולציה על ידי הבוחרים. במקרה זה לא הנחתי שיש איזשהו מידע הסתברותי, אלא שכלל לא ידוע מי מהבוחרים יחליט להצטרף לקואליציה של המניפולאטורים. מודל חדש זה להצבעה אסטרטגית, שנקרא מניפולציה בטוחה, הוצע לאחרונה ע"י סלינקו וואיט. חקרתי את הסיבוכיות של מציאת מניפולציה בטוחה עבור מספר כללי הצבעה, והראיתי מתי ניתן לפתור ומתי קשה לפתור בעיה זו. התייחסתי גם למקרה בו יש משקולות לבוחרים וגם למקרה בו הם לא ממושקלים. בנוסף הצעתי שתי דרכים להרחיב את המודל המקורי של מניפולציה בטוחה וחקרתי את המאפיינים החישוביים של הרחבות אלו.

עבודה זו נעשתה בהדרכתם של פרופסור יונתן אומן ופרופסור שרית קראוס

מן המחלקה למדעי המחשב של אוניברסיטת בר-אילן.

אינטראקציות חברתיות תחת אי-וודאות במערכות מרובות סוכנים

חיבור לשם קבלת התואר "דוקטור לפילוסופיה"

מאת:

נועם חזון

המחלקה למדעי המחשב

הוגש לסנאט של אוניברסיטת בר אילן

רמת גן

תשרי, תשע"א