ABSTRACT OF DISSERTATION

Nicholas Mattei

The Graduate School
University of Kentucky
2012

DECISION MAKING UNDER UNCERTAINTY: THEORETICAL AND EMPIRICAL RESULTS ON SOCIAL CHOICE, MANIPULATION, AND BRIBERY

ABSTRACT OF DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for the
degree of Doctor of Philosophy in the
College of Engineering at the
University of Kentucky

By
Nicholas Mattei
Lexington, Kentucky

Director: Dr. Judy Goldsmith, Professor of Computer Science
Lexington, Kentucky 2012

ABSTRACT OF DISSERTATION

DECISION MAKING UNDER UNCERTAINTY: THEORETICAL AND EMPIRICAL RESULTS ON SOCIAL CHOICE, MANIPULATION, AND BRIBERY

This dissertation focuses on voting as a means of preference aggregation. Specifically, empirically testing various properties of voting rules and theoretically analyzing how much information it takes to make tampering with an election computationally hard.

Groups of individuals have always struggled to come to consistent and fair group decisions and entire fields of study have emerged in economics, psychology, political science, and computer science to deal with the myriad problems that arise in these settings. In my research I have sought to gain a deeper understanding of the practical and theoretical issues that surround voting rules. This dissertation lies within the field of computational social choice, a subfield of artificial intelligence. This cross disciplinary area has broader impacts within the fields of economics, computer science, and political science.

My theoretical work focuses on the computational complexity of the bribery and manipulation problems. The bribery problem asks if an outside agent can affect the results of a voting scenario given some budget constraints, while the manipulation problem asks if one or more voting agents can strategically misrepresent their votes to induce a more preferred outcome. These questions seem to hinge on the amount of information an agent has. In this work I investigate the situations where the agents have access to perfect information, uncertain information, and structured preference information. I find that, depending on the structure and type of information, the complexity of the bribery and manipulation problems can range from computationally easy to computationally intractable.

Equally critical to the theoretical aspects of voting are empirical tests of existing assumptions. I have identified a large, sincere source of data with which to test many assumptions in the social choice and voting theory literature. A dearth of accurate data has led many studies of the properties of voting rules to take place in the theoretical domain. With the new dataset I have been able to test many theoretical voting paradoxes with orders of magnitude more data than previously available. This work shows that many of the irregularities or paradoxes associated with voting occur very rarely in practice.

KEYWORDS: Artificial Intelligence, Computational Social Choice,
Voting Theory, Bribery, CP-nets

Author's signature:_____

Date:_____

DECISION MAKING UNDER UNCERTAINTY: THEORETICAL AND EMPIRICAL RESULTS ON SOCIAL CHOICE, MANIPULATION, AND BRIBERY

By
Nicholas Mattei

Director of Dissertation:_____

Director of Graduate Studies:_____

Date:_____

RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the dissertation in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this dissertation for use by its patrons is expected to secure the signature of each user.

Name	Date

DISSERTATION

Nicholas Mattei

The Graduate School
University of Kentucky
2012

DECISION MAKING UNDER UNCERTAINTY: THEORETICAL AND EMPIRICAL RESULTS ON SOCIAL CHOICE, MANIPULATION, AND BRIBERY

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for the
degree of Doctor of Philosophy in the
College of Engineering at the
University of Kentucky

By
Nicholas Mattei
Lexington, Kentucky

Director: Dr. Judy Goldsmith, Professor of Computer Science
Lexington, Kentucky 2012

ACKNOWLEDGMENTS

This document and all the research supporting it would not exist without the support of many people over the years. No matter what I write here I will forget someone who has helped me along the way. I'll try my best.

My adviser Judy Goldsmith managed to put up with me for almost 6 years, including the time I tried to leave and go get a real job. She has been engaged, and in her own way, motivated me and kept me working on (mostly) one thing for the last several years. I would never have made it to the end without her guidance, feedback, and supervision, for this I will be forever grateful.

The rest of my committee, Andrew Klapper, Mirosław Truszczyński, and Stephen Voss, has provided invaluable feedback over the years. I would not have such a strong breadth and depth of knowledge if not for their insightful questions, comments, and suggestions.

I am eternally grateful to Francesca Rossi, who was an amazing host for a summer of research in Italy and a great outside committee member. I had a great time during my visit; I learned a lot, saw a lot, and returned to Kentucky with a new appreciation for the world and a new passion for research which carried me through the last year.

I have had the opportunity to work and publish papers with many different people over the years. I want to specifically thank all my coauthors for showing me the ropes of academic publishing and doing good quality research: Daniel Binkele-Raible, Gábor Erdélyi, Henning Fernau, Judy Goldsmith, Andrew Klapper, Maria Silvia Pini, Francesca Rossi, Jörg Rothe, and K. Brent Venable.

I wouldn't be the researcher, scholar, or person that I am today without the help of more people than I can name. In addition to my supervisors, friends and family, I have had great guidance in my professional and personal life from many mentors who have taken an interest in my development and helped me along the way. Specifically Kim Wagenbach,

Bill Caldwell, David Bui, Charlie Friedericks, Kuok Ling, and Nghia Mai at NASA Ames Research Center; Debbie Keen for supervising me as a TA and showing me how to be a good instructor; and a host of researchers including Craig Boutilier, Vincent Conitzer, Piotr Faliszewski, Jérôme Lang, Victor Marek, Patrice Perny, Florenz Plassmann, Michel Regenwetter, Nic Tideman, Joel Uckelman, and Lirong Xia, who (sometimes unbeknownst to them) have shown me the way to success.

I have been honored to receive funding from many sources during my graduate studies. Without the support I would not have been able to spend as much time researching the things that interested me. Thanks to the National Science Foundation, specifically grants CCF-1049360, NSF ITR-0325063, NSF IIS-1107011 as part of the IJCAI 2011 Doctoral Consortium, the Northern Kentucky Alumni Club for their fellowship, the Myrle E. and Verle D. Nietzel Visiting Distinguished Faculty Program for their sponsorship, and the Department of Computer Science at the University of Kentucky for several years of teaching assistantships.

Thanks to all the members of the AI-Lab (past and present) at the University of Kentucky. Their presence and feedback has been invaluable over the years (as well as sitting through many practice talks): Peng Dia, Gayathri Namasivayam, Liangrong Yi, Tom Allen, Robert Crawford, Tom Dodson, Joshua Guerin, Daniel Michler, Paul Mihail, James Forshee, Josiah Hanna, Libby Knouse, and Matt Spradling.

Thanks to all my friends who listened to me over the years. Keeping me distracted in my off hours and keeping me grounded (whether I was above or below it at the time); I couldn't have done any of it without you and your constant, unwavering support: Michael Dillion, Alex Zerga, Bo Padgett, Mike Karounos, Zach Rosen, Aaron Kemper, Will Carraco, Joshua Slayton, Brian Vincent, Eren Turgay, Peter Arnberg, Ben Potash, Sarah Peters, Stephanie Franxman, Armir Bujari, Meredith Gaffield, Aaron Schooley, Keith Peterson, Aaron Swank, K. Alison Brotzge, Craig Kannapel and many many more. I wouldn't be here and it wouldn't be worth it without all of you.

Thanks to my immediate family, Theresa, Mike and Eric, without their constant love and support I wouldn't have tried (and all those science camps helped too). I need to thank my extended family who have loved and supported me no matter what, especially my maternal grandparents Mary and Ernest Hillenmeyer; my paternal grandparents Mary Della and Innocente Mattei; my cousins Joe Hellebusch, Sara Hillenmeyer, and Liz Gillespie (among many); and all the rest of my extended family. For the years of nurturing, love, support, and for showing me that anything worth doing is worth doing well, I will be forever grateful.

Last and most to Liz, for everything.

For Liz and Mom, and for finally catching Dad.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Motivation

This dissertation focuses on voting as a means of preference aggregation. Specifically, empirically testing various properties of voting rules and theoretically analyzing how much information it takes to make tampering with an election computationally hard.

Democratic societies and fair minded individuals have used voting procedures to come to group decisions since at least 508 B.C.E. [107], and, since at least 105 A.D., there have been concerns over the honesty and security of voting [92]. The disconcerting and ever present threat is that one person within the group, a coalition within the group, some outside actor(s), or the individuals counting the ballots, would, could, or do express undue influence on the result of the vote. These threats are so significant that many laws and methods of voting have been devised to intentionally dissuade individuals from attempting to tamper with a vote.

A prime example of a society constructing a voting procedure to prevent tampering is the selection process for the Doge of Venice. The city-state of Venice (in present day Italy) began selecting its leaders in the following, somewhat convoluted, manner in 1172 A.D. The procedure remained mostly unchanged for over 600 years (about 75 iterations), until the fall of the Venetian Republic in 1797. The election proceeds in 10 rounds over the course of several days. Each round created a college, either by lottery or by election, for the next round. In the first round, every member of the Great Council age 30 or more (and only one member per family) convened in a college, and 30 of them were selected by lottery for the next round. Round 2 saw these 30 reduced to 9, as selected by lottery. In the third round, the 9 elected 40 for representation in the next round and each of the 40 had to be approved by at least 7 of the 9 members. The fourth round saw the college of 40 narrowed to 12 by lottery draw. The fifth round had the 12 elect a college of 25, each

requiring 9 of the 12 votes. In the sixth round the 25 was reduced to 9 by lottery and the seventh round had these 9 elect a college of 45, each by a 7 of 9 majority. In the eighth round the 45 were again pared down to 11 by lottery. The 11, in the ninth round, elected a final college of 41 with each member of the college requiring a 9 of 11 majority. The tenth and final college of 41, with a majority vote of at least 25 of the 41, elected the Doge of Venice [21].

While this procedure seems insane by modern standards, a comprehensive study of its security properties by Mowbray and Gollmann show that it was extremely robust to tampering [94]. It turns out that, with so many lotteries and rounds, it becomes very hard to manipulate or bribe voters; one never knows who exactly will be voting in the next round. This alternating of the type of selection round provides representation opportunities to minority candidates (egalitarianism) while still ensuring that more popular candidates (majoritarian) have a higher probability of winning.

This dissertation falls in the area of computational social choice (ComSoc), an emerging and rapidly evolving subfield of artificial intelligence. My work in ComSoc is focused on how groups of agents make collective decisions. social choice, an established research field at the intersection of mathematics and political science, has long studied the implications of group decisions in human systems.

The growth of multi-agent systems in computer science has created many situations where individual agents, be they robotic, software, or human, need to come together to make a group decision. These decisions can take the form of a recommendation on a shopping website, rankings of search results from the web, or coordinated robot behavior. This growth and proliferation of multi-agent systems research in the AI community had made it necessary to closely investigate how agents can work together and make group decisions [119]. ComSoc and social choice are related by two main bridges: bringing a computational perspective to decision systems already in use and/or studied by social choice, and bringing systems and processes developed through years of social choice research to bear

on multi-agent systems. Social choice has broad application in computer science including: multi-agent systems, intelligent systems, and human-computer interaction [23].

This dissertation contains both theoretical and empirical work. The main focus of the theoretical work is on questions of manipulation and bribery in social choice settings. The study of manipulation in social choice is about security. The central question of this dissertation is: how much information does it take to make tampering with an election computationally hard? To this end, I investigate the bribery and manipulation problems under two information assumptions: uncertain information and structured information. Voting rules are subject to multiple forms of attack, and the classical and most current literature studies these issues in a perfect information world: every agent knows exactly how every other agent will vote. I feel that this model is lacking since manipulation is trivially easy for many common voting rules under perfect information. Typically, agents have uncertain or probabilistic information. Pollsters have an idea of how you will vote, just as we have an expectation of what our friends will want to eat for dinner. It turns out that the question of security is much different when we take uncertain information into account. Another way to frame manipulation is in terms of resource allocation. Consider the process of electing or gaining support for a particular alternative. If we have some resources to achieve a goal, canvassers in elections or concessions with friends about where to eat next week, then how can we best distribute our influence or resource in order to achieve consensus?

1.2 Main Contributions and Related Publications

This dissertation is supported by several publications with partially disjoint groups of researchers. Most of the chapters detail work that was done jointly and, therefore, I use "we" when describing this work. While I have had a principle role in defining, performing, and writing the work detailed in these chapters, I would not have been able to do anything (and wouldn't be writing this dissertation) if it wasn't for the help of my coauthors. I am deeply indebted to my coauthors on work that has directly supported this dissertation: Daniel

Binkele-Raible, Gábor Erdélyi, Henning Fernau, Judy Goldsmith, Andrew Klapper, Maria Silvia Pini, Francesca Rossi, Jörg Rothe, and K. Brent Venable.

My work has been directly supported by NSF-EAGER grant CCF-1049360: Changing Minds, Changing Probabilities, NSF-ITR grant 0325063: Decision-Theoretic Planning with Constraints, NSF IIS-1107011: IJCAI 2011 Doctoral Consortium and International Experience, and the 2010 Northern Kentucky Alumni Association Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or the Northern Kentucky Alumni Association.

Bribery and Manipulation with Uncertain Information

In the work on bribery under uncertain information we proposed a novel and flexible model to represent majority voting with uncertain information. Given a set of voters, their individual preferences represented as probability distributions over a set of issues; their prices for changing their preferences; and a budget, we classified the complexity of finding efficient bribery schemes. We showed that, depending on the particular combination of evaluation and bribery models chosen, the problems range in complexity from polynomial time to NP-complete. This difference reveals that modeling choices can have significant effects on the complexity of calculating efficient bribery schemes. This work has resulted in two joint publications.

- Daniel Binkele-Raible,Gábor Erdélyi, Henning Fernau, Judy Goldsmith, Nicholas Mattei and Jörg Rothe, "The Complexity of Probabilistic Lobbying." Technical Report arXiv:0906.4431v3 [cs.CC], *ACM Computing Research Repository (CoRR)*, June 2009. Revised, February 2011.

- Gábor Erdélyi, Henning Fernau, Judy Goldsmith, Nicholas Mattei, Daniel Raible and Jörg Rothe, "The Complexity of Probabilistic Lobbying." *Proc. 1st International Conference on Algorithmic Decision Theory (ADT-09)*, October, 2009.

In addition to the work on multiple referenda, sports tournaments represent a domain where it is natural to express winners and losers in terms of probabilities of outcomes. We use this as a motivating example to study the related problems of manipulation in elimination and round-robin tournaments. These results can also be mapped to their corresponding voting rules. Given a set of teams and their probabilities of each possible win, along with prices for decreasing their competitive output (purposefully losing or underperforming in a match), we classified the complexity of finding efficient bribery schemes for three common types of sports tournaments. The evaluation complexity of these problems range over a variety of complexity classes from the easy to the very hard. Our results show that in some cases the added uncertainty increases the complexity of manipulating sports tournaments while in other cases it does not. While this increase in complexity is not uniform across all tournament types, the change shows strong evidence that reasoning in domains with uncertain data leads to an increase in reasoning complexity. This work has resulted in one joint conference publication.

- Nicholas Mattei, Judy Goldsmith, and Andrew Klapper, "On the Complexity of Bribery and Manipulation in Tournaments with Uncertain Information." *Proc. 25th Intl. Florida Artificial Intelligence Research Society Conference (FLAIRS 2012)*, June 2012.

Bribery and Manipulation in Combinatorial Domains

When looking at voting it is often natural to express group decision problems as the combination of a sequence of decisions. This method is used in many settings, from the United States Congress (specifically, votes for amendments to a bill) to a group of friends deciding what appetizer, main course, desert, and wine should be served for a group meal. In all these cases, agents express preferences and vote on parts of the overall decision to be taken. Agents may also have dependent preferences within this construction: the choice of wine may depend on the choice of main course for the meal. We consider a scenario

where agents use the CP-net formalism ("Conditional Preference" or *"Ceteris Paribus"* networks) which allows us to compactly model these conditional dependencies [15]. We investigated the computational complexity of bribery and manipulation schemes in combinatorial voting domains where voters' preferences are expressed as CP-nets. To do this, we generalized the traditional bribery problem to encompass these domains and found that, for most of the combinations of these parameters, bribery in this domain is computationally easy. This indicates that either CP-net preferences lead to highly manipulable aggregation schemes, or that we have over-constrained the problem. As CP-nets have become more ingrained in preference and social choice research we hope to continue this line of research into the security of CP-nets. This work was supported by one conference publication and one invited paper.

- Nicholas Mattei, Maria Silvia Pini, Francesca Rossi, K. Brent Venable, "Bribery in Voting Over Combinatorial Domains Is Easy." *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-12, short paper)*, June 2012.

- Nicholas Mattei, Maria Silvia Pini, Francesca Rossi, K. Brent Venable, "Bribery in Voting Over Combinatorial Domains Is Easy." *12th International Symposium on Artificial Intelligence and Mathematics (ISAIM-12), Special Session on Computational Social Choice*, January 2012.

Empirical Analysis of Voting Rules and Election Paradoxes

In addition to the theoretical work, I investigated the behavior of voting rules and occurrences of voting paradoxes using empirical data. To facilitate an empirical study, I mined a large dataset of "elections" from real preference data. This dataset has several million individual elections with tens to tens of thousands of voters. This represents orders of magnitude more election data than previously available, and I used it to analyze the behavior

of voting rules. Analysis of this dataset has provided useful insight into voting methods – including the surprising conclusion that, in contrast to much of the theoretical work, voting rules declare the same winner a majority of the time. I recently began separate research collaborations with colleagues in political science and psychology in order to more extensively study the data from the Netflix Prize. This work is supported by one conference publication.

- Nicholas Mattei, "Empirical Evaluation of Voting Rules with Strictly Ordered Preference Data" *Proc. 2nd International Conference on Algorithmic Decision Theory (ADT-11)*, October, 2011.

1.3 Structure of the Dissertation

This dissertation attempts to walk the reader through the study of how access to different types of information changes the reasoning complexity of various security problems in social choice systems and how we can test these computational properties.

Chapter 2, Preliminaries: In Section 2.1 I survey the mathematical and algorithmic fundamentals that one would need to understand the discussion in later chapters. This section is focused for readers not from computer science or mathematics. I expect the reader of this section to be a mathematically competent individual from another discipline.

In Section 2.2 I provide a comprehensive background of social choice and preference aggregation. I give an overview of the field of Computational Social Choice; survey research in voting systems including bribery and manipulation; and detail the specifics of several voting rules. I frame the work presented in this dissertation in the overall stream of research about bribery and manipulation in ComSoc.

Chapter 3, Bribery and Manipulation with Uncertain Information: In Section 3.1 I detail the work on bribery in majority elections when the outside actor has access

to uncertain information. This section details the development of a new model of reasoning in voting domains with uncertain information and presents a complete analysis of the complexity of reasoning in the uncertain information setting.

In Section 3.2 I detail the work on bribery in sports tournaments when the outside actor has access to uncertain information. While the discussion deals primarily with sports tournaments, the model can be extended to certain voting rules which have the same structure as sports tournaments. This section details the extension of the model presented in Section 3.1 to the domain of sports tournaments and presents a complexity analysis of reasoning in this extended model.

Chapter 4, Bribery and Manipulation in Combinatorial Domains: In this chapter I detail the work on bribery in elections where each decision is described by the combination of a set of smaller decisions. I present an overview of CP-nets and their use in structured preference representation. I extend the traditional bribery problem and apply it to combinatorial domains. I then present complete complexity analysis of these new models of reasoning.

Chapter 5, Empirical Analysis of Voting Rules and Election Paradoxes: In this chapter I detail the work on empirically verifying some of the underlying assumptions in social choice research. I survey existing data sets and their properties and then identify and study a novel set of data that approximates real data from real elections. This chapter looks at why so few empirical studies occur in ComSoc and begins to close the gap between the theoretical and empirical.

Chapter 6, Conclusions and Future Directions: In the final chapter we look back at the research presented in this dissertation and attempt to gain a better perspective on the role that the type of information has in reasoning about election systems. We also detail directions future research that will help us better understand how to protect the way we choose.

Chapter 2 Preliminaries

In this chapter we provide an overview of the mathematics and literature necessary to frame the results we present in the later chapters. We provide a survey of computational complexity and a discussion of flow network problems in Section 2.1. In Section 2.2 we provide details about computational social choice, voting rules, and a survey of bribery and manipulation in computational social choice.

2.1 Mathematical Background

In this section we give an overview of the different mathematical and computational constructs that we will use throughout the rest of this document. We review complexity theory in Section 2.1.1 and provide details of network flow problems in Section 2.1.2. This review is not intended to be a complete treatment of these ideas. Suggestions for further reading are included for each topic.

2.1.1 Computational Complexity

Much of the research presented in this dissertation deals with the computational complexity of reasoning in a variety of domains. We spend most of our time devising models that approximate some form of reality. Once we formally define these models we want to quantify how hard the problems are in a computational sense. In this section we hope to give the reader enough understanding about the area of complexity theory in order to follow our discussion in the later chapters. Complete treatments of these topics can be found in the books by Papadimitriou [99]; Garey and Johnson [65]; and Hemaspaandra and Ogihara [74]. This section is aimed at researchers from disciplines other than computer science and mathematics. In some cases we use less than precise explanations in order to give the reader a feel for the issues and we include mathematically precise definitions

for those who want to understand the material more exactly. Our hope is that we convey enough understanding of complexity theory for a non-technical reader to appreciate the technical discussion in later chapters. For these reasons, a reader familiar with complexity theory can safely skip this section.

Complexity theory seeks to quantify how hard a problem is in regards to a measurable quantity, typically, time or space. Time, in the sense of: "How many operations does it take to compute an answer?" Space, in the sense of: "How much memory is necessary to compute the answer?" These notions of time and space complexity are well formed ideas in the area of computational complexity and we give an overview here of what it means for a problem to be **hard**.

The first object that we must formalize is a computer. We will use a Turing Machine (TM) as our model of computation. A TM is a simple model of a computer. Even though a TM may seem like a toy model, it is powerful enough to capture the computational power of any modern computer [99, 120].

Definition 2.1.1 *A **Turing Machine (TM)** is a 4-tuple, $\langle Q, \Sigma, \delta, s_0 \rangle$, where:*

Q**:** *A finite set of states including s_0, s_{ACCEPT}, and s_{REJECT}.*

Σ**:** *The finite alphabet recognized by the TM. We assume that there is no difference between the input and tape alphabets. This alphabet must include $\sqcup$, the blank symbol.*

δ**:** *The transition function or program of the TM. δ is a mapping: $Q \times \Sigma \to Q \times \Sigma \times \{L, R, N\}$, where $\{L, R, N\}$ are Left, Right, and No movement of the tape head, respectively.*

A TM is a simple computer with a semi-infinite tape of symbols. These symbols are the alphabet used by the TM and must contain the "blank" symbol, $\sqcup$. This tape has a starting point on one end and is infinite in the other direction. One can imagine the computation process as involving a tape head which can read, write, and move on this tape (in either

direction) one symbol at a time. A TM starts its computation at the beginning of tape in state s_0 and reads a single symbol off of the tape. The TM uses its δ function, decides what symbol to write back to the tape and which direction to move the tape head (Left, Right, or No Movement). The TM, based on the symbols on the tape and the position of its tape head, moves through its computation, processing the symbols one at a time. Each symbol/movement pair describes a state, and every possible state is enumerated in the set Q. The TM will eventually reach s_{ACCEPT}, s_{REJECT}, or it will compute forever. A TM **halts** if it completes computation in either s_{ACCEPT} or s_{REJECT}. We say that a TM **accepts** a given input if the TM halts in s_{ACCEPT} and a TM **rejects** a given input if the TM halts in s_{REJECT}.

We use TMs to define *decision problems*. A decision problem is any problem that is answered with either a "yes" or a "no". For instance, the question, "Is the average of a set $U = \{u_1, \ldots, u_n\} \subset \mathbb{Z}^+ > t$?" is an example of a decision problem. We define decision problems in terms of free variables and relationships over those variables. The question, "Is the average of $U = \{3,4,5,6,7\} > 9$?" is an *instance* of the decision problem, "Is the average of a set $U = \{u_1, \ldots, u_n\} \subset \mathbb{Z}^+ > t$?" When we fix the free variables (the set U and t) to specific values ($\{3,4,5,6,7\}$ and 9) we create an instance of the decision problem. In this example the answer is "no" and we say the particular instance is not in the set of "yes" instances of our decision problem. Any specific decision problem (where we are given variables and relationships) defines a set of problem instances.

Definition 2.1.2 *Let Σ be a finite alphabet and Σ^* be the set of finite strings over alphabet Σ, including the empty string (ε). A **language** L over Σ is a subset of Σ^* ($L \subseteq \Sigma^*$).*

Definition 2.1.3 *If L is a language over Σ^* and M is a TM, then M* decides *L if and only if, for all $x \in \Sigma^*$, $M(x)$ halts, and $M(x)$ halts in s_{accept} if and only if $x \in L$ (else it halts in s_{reject}).*

Generally, we describe a decision problem as a specific language [65] and we create TMs to decide this specific language. We say that a given TM decides a language if it

accepts exactly those inputs that are in the language. Informally, we judge the power of our TMs by the types of languages they can decide given certain restrictions over the resources of the TM. We count how many operations it takes for the TM to decide instances of the language. We measure time complexity for a given TM as a function from the size of the input to the number of steps. This function gives us a relationship between the number of operations the TM must perform and the size of the input, e.g. the size of the encoding of the problem instance [99]. Note that we can construct an arbitrarily convoluted TM which takes many extra steps and, for this reason, we define the complexity of a language as the minimum complexity over all of its deciders.

We define **classes** of languages to be sets of languages with common properties. By a class, we mean a set of languages that can be decided with TMs that are constrained in similar ways (e.g. time complexity). By restricting the resources available to a TM, we gain an understanding of how much of some resource any computer would require to decide a language within a certain class. This idea provides a way to separate problems into classes based on how computationally hard they are to decide. The two most important classes we study are P, Definition 2.1.4, and NP, Definition 2.1.5 [74].

The difference between the classes P and NP is the notion of determinism. Informally, the class P is the set of languages that can be decided in time polynomial in the size of the encoding of the problem instance with a deterministic TM, while NP is the class of languages that can be decided in polynomial time with a nondeterministic TM. A nondeterministic TM is one that can select from a set of possible transitions at each computational step. It accepts an input if and only if there is a set of "good guesses" that allow it to reach s_{ACCEPT} when processing. Formally, in a nondeterministic TM the transition function δ becomes a relation instead of a mapping. This means that for any state and symbol pair there is zero, one, or many possible next states and the TM can choose any of these when performing a transition [99]. On the other hand, a deterministic TM is one that computes in the "normal way" most readers will be familiar with. By this we mean it computes one

thing after another and never guesses or deviates from its programming.

The resources we define our classes by are "TIME", in the sense of number of operations, and "SPACE", in the sense of number of bits of memory. In what follows we will use $f(n)$ to denote a "proper complexity function" as defined by Papadimitriou [99]. Formally, f is a function that maps $\mathbb{Z}^+ \to \mathbb{Z}^+$, a TM M that, for all x in $M(x)$, M takes exactly $f(|x|)$ units of some specified resource to complete its computation. Let DTIME(M) be the amount of time used by deterministic TM M to run an algorithm on a given instance given infinite space. Likewise, we let NTIME(M) be the amount of time used by nondeterministic TM M to run an algorithm on a given instance with no bounds on the amount of memory used [99]. We can similarly define DSPACE(M) and NSPACE(M) for the case of space resources. In the following definitions, we replace $f(n)$ with a specific family of functions parameterized by some integer $k > 0$. We define the complexity classes as the union of all the complexity functions using a certain amount of the specified resource.

Definition 2.1.4 *We define the class P as:*

$$P = \bigcup_k DTIME(n^k).$$

P is the class of languages L such that there is a polynomial time, deterministic Turing Machine that accepts L.

Definition 2.1.5 *We define the class NP as:*

$$NP = \bigcup_k NTIME(n^k).$$

NP is the class of languages L such that there is a polynomial time, nondeterministic Turing Machine that accepts L.

Definition 2.1.6 *We define the class EXP as:*

$$EXP = \bigcup_k DTIME(2^{n^k}).$$

EXP is the class of languages L such that there is a deterministic Turing Machine that accepts L in an exponential amount of time.

We can also define classes in terms of the amount of space they require.

Definition 2.1.7 *We define the class PSPACE as:*

$$PSPACE = \bigcup_k DSPACE(n^k).$$

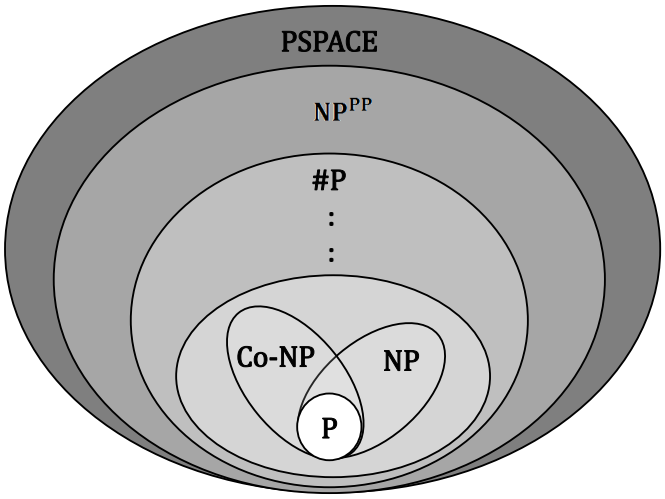
PSPACE is the class of languages L such that there is a polynomial space, deterministic Turing Machine that accepts L.

While we will not prove any theorems about EXP-time algorithms, they are a looming danger in several of the problems we study in later chapters. Many combinatoric problems may require brute force algorithms which take time exponential in the size of the problem input to compute an answer [100]. Any bribery scenario admits a simple algorithm: attempt every possible combination of bribes for every voter. However, this is not an efficient algorithm and would require EXP-time in the worst case. Our job is to figure out if there are better ways to compute the answers to questions about our model and avoid EXP-time algorithms.

Definition 2.1.8 *A language L is* $\in$ *coNP if and only if* $\overline{L} \in NP$.

The class of complements of NP languages is called coNP (Definition 2.1.8). For a given nondeterministic language L, determining if $x \in L$ requires only finding one accepting sequence of guesses (computation). However, if we want to show that $x \notin L$, we must check that no accepting sequence of guesses exists. This is in stark contrast to the class P, which is closed under complement. The question of whether or not nondeterministic TMs are closed under complement is one of the most important open questions in theoretical computer science [99].

Figure 2.1: An illustration of the ordering over the complexity classes. P is the computationally easiest class shown and PSPACE is the most computationally difficult class shown.



Generally, we say that a problem is **hard** if it falls into a class requiring more computational resources than problems in the class P. We can see the relationship among the complexity classes we use in this dissertation illustrated in Figure 2.1.1. We note that it is an open question as to whether P = NP. While the difference between P and NP seems obvious, it has not been directly proven that the two classes are different. It is also an open question as to whether or not P =?PSPACE. If P = PSPACE then the entire class hierarchy shown in Figure 2.1.1 would collapse (as all classes would be equal) [99].

In later chapters we consider several problems which may be in the function class #P introduced by Valiant [126]. Informally, #P is the counting analog of the class NP. For a given decision problem, rather than asking the question, "Does a solution exist?" the functional class #P asks, "How many solutions exist?"

Definition 2.1.9 *A function g is in #P if there is a nondeterministic TM M, such that $\forall x : g(x)$ is the number of accepting computations of $M(x)$.*

We can define a class of decision problems that are closely related to the functional class #P.

Definition 2.1.10 *A language L is in the class PP if there is a nondeterministic TM M such that, for all x, $x \in L$ if and only if $M(x)$ accepts on more than half of its computations.*

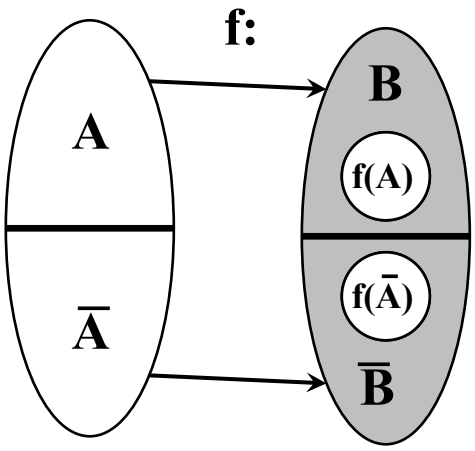
The class #P seems much harder than NP since a #P function requires that we count all the possible solutions to an NP-complete problem. We use Figure 2.1.1 to graphically illustrate the relationship between complexity classes. The class PP is the decision version of #P: we have to know all the possible accepting computation paths before we can decide if a majority of the computations end up accepting [99].

Above #P we include the complexity class $\mathrm{NP}^{\mathrm{PP}}$. This notation uses the concept of an **oracle**. An oracle A, in complexity, can be thought of as a unit cost sub-function. So, if we had sub-function A then the complexity class P^A is the set of TMs that decide in deterministic polynomial time with the use of A as an oracle. The class $\mathrm{NP}^{\mathrm{PP}}$ is the class of nondeterministic TMs that require access to a PP (or #P) oracle to check their work [74]. Additionally, we omit many (possibly infinitely many) classes in our diagram [99].

In order to classify a given problem we can leverage our knowledge of other problems that have already been classified. To make use of this knowledge we use reductions. A **reduction** is a technique whereby we use previously solved problems and algorithms to solve a new problem or use a known hard problem to prove hardness of a new problem. If we are given a problem, A, a reduction from A to B is a function, $f : A \to B$, that provides a transformation of an instance of A to an instance of B. This allows us to use algorithms for B in order to solve instances of problem A. This idea is illustrated in Figure 2.1.1. In this work we will focus on a specific type of reduction, the many-one, polynomial time transformation [74].

Definition 2.1.11 *Given two languages, L_1 and L_2, we say that $L_1 \leq_m^p L_2$, L_1 is **many-one, polynomial time reducible** to L_2, if there is a function f such that $x \in L_1$ if and only if $f(x) \in L_2$ and f is computable in polynomial time.*

Figure 2.2: An illustration of a reduction. Given an instance of problem A, we say f reduces A to B if all "yes" instances of A are transformed into "yes" instances of B and likewise for "no" instances.



Completeness of languages is the final property that we will use to understand our complexity hierarchy. Completeness is how we precisely place languages into classes.

Definition 2.1.12 *A language L is **hard** for a class C if and only if for all $L' \in C$, $L' \leq_m^p L$.*

Definition 2.1.13 *A language L is **complete** for a class C if and only if $L \in C$ and L is hard for C.*

Completeness (Definition 2.1.13) consists of both an upper and lower bound. Hardness (Definition 2.1.12) establishes a lower bound for completeness while inclusion defines an upper bound. Showing that a language L is hard for class C shows that it is at least as hard as any other language in C (up to a polynomial increase in complexity). This gives a lower bound on the complexity of L. We must establish both lower and upper bounds to establish completeness of a language [65]. The notion of completeness allows us to indicate the complexity of deciding a language not only directly but also in relation to other languages.

With the tools we have described in this section we can now classify problems into several complexity classes and determine if the problems are included in, hard for, or complete for some given complexity class. These tools will allow us to analyze our models of

voting, bribery, and manipulation, in terms of the difficulty of finding answers to decision problems.

Parameterized Complexity

We mention several results that relate to the theory of parameterized complexity. The field of parameterized complexity was introduced by Downey and Fellows as a complement to classical complexity theory [40]. The central notion of parameterized complexity is that one can determine what the effects of particular **parameters** are on the complexity of a problem. There is a very large difference between algorithmic running times of $\mathcal{O}(n^k)$ and $\mathcal{O}(2^k \cdot n)$, however, if k is small then the running times are closer together (asymptotically the same if $k = 1$) [62].

Parameterized complexity can also be expressed as hardness and completeness for levels of the $\mathrm{W}[t], t \geq 1$ hierarchy:

$$\mathrm{FPT} = \mathrm{W}[0] \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots.$$

Languages in FPT are also in P (for a fixed value of the parameter). In some of our later results we provide reductions from a problem that is known to be $\mathrm{W}[2]$-hard [62]. The class $\mathrm{W}[2]$ contains what are referred to as the "hardest" NP-complete problems that, in general, are not closely approximable such as k-DOMINATING SET [62] There is also a special notion of parameterized reductions which varies slightly from classical complexity theoretic reductions. We refer the reader to [62] for a more complete treatment of the topic.

2.1.2 Flow Networks

Several of our proofs in later chapters will make use of algorithms for flow networks. Flow networks are extremely important in many areas and are used for a variety of problems including maximum bipartite graph matching [35], modeling shipping networks, and modeling the flow of electricity in circuits [1]. The study of flow networks was introduced by

Ford and Fulkerson [64] and is important in computer science as evidenced by the chapter long treatment in the book by Corman et al. [35], one of the standard introductory algorithm textbooks in computer science.

Let $G = (V, E)$ be a graph. A graph contains a set of nodes or vertices V and a set of edges E connecting the vertices. Let $e = (u, v)$ and $e' = (v, u)$. In an undirected graph e and e' are the same. Informally, edges on a directed graph can only be traversed in one direction while in an undirected graph the edges are two-way streets.

A flow network is a directed, acyclic graph modeling a system in which we want to move some amount of resource from a source to a destination (sink). Each edge in the network has a finite capacity and a finite cost associated with it and we want to find an assignment of flow to edges such that certain constraints are not violated. Flow networks and the questions about them come in a variety of styles and flavors and each has a wide array of algorithms associated with them. We will not survey all the methods and flavors here; we point the reader to the book by Ahuja et al. [1] for a more complete discussion.

In this dissertation we will only be using the MINIMUM COST FEASIBLE FLOW problem.

Name: MINIMUM COST FEASIBLE FLOW

Given: A flow network $G(V, E)$ which is an undirected graph with source $s \in V$ and sink $t \in V$ where edge $(u, v) \in E$ has capacity $c(u, v)$, flow $f(u, v)$, cost $a(u, v)$, and minimum flow $d(u, v)$. The cost of sending flow across an edge is $f(u, v) \cdot a(u, v)$.

Question: Minimize the total cost of the flow: $\Sigma_{u,v \in V} f(u, v) \cdot a(u, v)$ under the constraints:

Capacity constraints: $\forall (u, v) : f(u, v) \leq c(u, v)$

Required flow: $\forall (u, v) : f(u, v) \geq d(u, v)$

Skew symmetry: $\forall (u, v) : f(u, v) = -f(u, v)$

Flow conservation: $\Sigma_{w \in V} f(u, w) = 0$ for all $u \neq s, t$.

In general, a flow is a function $f : V \times V \to \mathbb{R}$ and the MINIMUM COST FEASIBLE FLOW problem has a polynomial time algorithm. However, if we restrict our costs, capacities, and minimum flow constraints to be integers, the MINIMUM COST FEASIBLE FLOW will have an integer solution, if any solution exists, and this solution is computable in polynomial time [1]. This is good news for us, as we will see in later chapters, as we can model certain forms of bribery as flow problems and the guarantee of an integer solution provides us with a powerful tool.

2.2 Social Choice and Preference Aggregation

For thousands of years societies have not only had to work together but also make decisions together. The field of social choice is largely considered to be first established by the Marquis de Condorcet [27, 106] with a book published in 1785. In this work, Condorcet begins to carefully outline what makes good voting procedures and what properties are important to individuals who have to vote. This principled approach laid the ground work for the study of social choice. In this section we survey voting rules, their properties and how computer science interacts with voting and social choice.

Computational Social Choice (ComSoc) is still a relatively young field in computer science. In this section we attempt to provide the reader with an understanding of the basic concepts in the area and references to relevant literature. A more focused literature review accompanies each chapter of this dissertation. There have already been many great dissertations in ComSoc, and many of these provide a good introduction to the major research in the field related to voting and preference aggregation: Conitzer [28], Faliszewski [50], Pini [101], Procaccia [104], Uckleman [125] and Xia [134]. There are also some excellent review articles over specific sub areas of ComSoc including the current state of bribery and manipulation by Faliszewski [59], an overview of all the major areas of ComSoc by Chevaleyre et al. [23], and an introduction to existing work in ComSoc as it relates to the use of CP-nets by Conitzer et al. [20].

ComSoc and social choice are related by two main bridges: taking results from social choice and using them in computational systems (for decision making; recommendation; coordination and control of multi-agent systems; and other uses [119]) and looking at classical social choice research from a computational point of view (the complexity of determining winners in voting rules, computational properties of voting rules, computational aspects of other algorithms [23]). This two-way exchange of information has resulted in an explosion of scholarship both in the computer science and social choice communities. While we focu sspecifically on voting in this dissertation we mention that other major topics in ComSoc are cake cutting and fair division algorithms; coalition formation; resource allocation; belief merging; and judgment aggregation [23]. The unifying factor in all these areas is that we are asking a group of agents (human or automated) to work together in some meaningful way.

2.2.1 Voting and Common Voting Rules

The variety of voting rules and election models that have been implemented or "improved" over time is astounding. For a comprehensive history and survey of voting rules see Nurmi [97], Tideman [123], Taylor [122], or Arrow et al. [3]. Any of these great references provides a more complete survey of the history, development, and axiomatic properties of voting rules. In this section we discuss several voting rules that we will use in later chapters, some axiomatic properties that are considered with respect to voting rules, and a brief remark about tie-breaking in voting rules.

Throughout this dissertation we use several notation conventions. We refer to the set of candidates as C with size m; the set of voters is V of size n. We assume generally, and note specifically in the following chapters, that the set of voters is represented in some reasonable way by their preferences. This can either be a strict linear order, a CP-net, or some other way of compactly representing how voter v_i feels about the candidates in C. We are also provided a voting rule E that will aggregate the preferences of the voters and return

a winner (or a set of winners) from the set C. Occasionally, we will require the definition of a **social welfare function** [122].

Definition 2.2.1 *Given a set of candidates C and a set V of preferences over the elements of C, a **voting rule** (or social choice function) E returns a non-empty subset of C.*

Definition 2.2.2 *Given a set of candidates C and a set V of preferences over the elements of C, a **social welfare function** W returns a linear ordering over the elements of C.*[1]

We now survey some of the more common voting rules.

Positional Scoring Rules: The family of voting protocols that fall under the framework of positional scoring rules include plurality, veto, k-approval, and Borda (among others). In these methods, a scoring vector $\vec{S}$ of length m is associated with the rule. Each entry in $\vec{S}$ is an integer and $\vec{S}$ is non-increasing. Each voter ranks the candidates in C in some position within this vector. We then, for each candidate, sum the points assigned to that candidate by all voters. The winner of the election is the candidate who has the highest total score. For a multi-winner election, the set of candidates who tie with the highest score are the winners.

Plurality: Plurality is the most widely used voting rule [97] and, to many Americans, synonymous with the term "voting"). The Plurality score of a candidate is the sum of all the first place votes for that candidate. No other candidates in the vote are considered besides the first place vote. Thus, the scoring vector is $\vec{S} = [1,0,0,\ldots,0]$. The winner is the candidate with the highest score.

Veto: Veto is often referred to as anti-plurality. In a veto election, all candidates but the last placed candidate receive a point. Thus, the scoring vector is $\vec{S} = [1,1,1,\ldots,0]$. The winner(s) is the candidate with the highest score.

[1] This is sometimes called a strict social welfare function [122].

k-Approval: Under k-Approval voting, when a voter casts a vote, the first k candidates each receive a point. In a 2-Approval scheme, for example, the first 2 candidates of every voter's preference order receive a point. Thus, for some fixed k, the scoring vector is $\vec{S} = [1, 1, \ldots, 1_k, 0, \ldots, 0]$. The winner of a k-Approval election is the candidate with the highest total score.

Borda: Borda's System of Marks involves assigning a numerical score to each position. In most implementations [97] the first place candidate receives $m-1$ points, with each candidate later in the ranking receiving 1 less point down to 0 points for the last ranked candidate. Thus the scoring vector is $\vec{S} = [m-1, m-2, \ldots, m-(m-1), 0]$. The winner is the candidate with the highest total score.

Condorcet's Rule: While not technically a voting rule, the Marquis de Condorcet proposed that the winner of an election should be the alternative that is preferred, pairwise, to all other alternatives [27, 97]. This method, called the Condorcet Method, compares all pairs of alternatives and selects the one that wins, by majority, in *all* pairwise elections. This method does not necessarily have a winner.

Copeland: In a Copeland election each pairwise contest between candidates is considered. If candidate a defeats candidate b in a head-to-head comparison of first place votes then candidate a receives 1 point; a loss is -1 and a tie is worth 0 points. The particular number of points assigned can vary depending on the particular implementation and many different versions exist [3]. After all head-to-head comparisons are considered, the candidate with the highest total score is the winner of the election.

Voting Trees: Here we use the term "tree" fairly liberally: we define a voting tree to be any voting system that includes an ordered set of comparisons between the candidates. These voting trees can be in the form of a complete binary tree (the cup rule) or as a linear system of comparisons (linear balloting). The tree structure defines the order of comparisons between the candidates when each candidate is assigned to

a leaf node of the tree structure. We perform a majority comparison between the candidates, promoting the winner, until we have a single candidate at the top of the tree.

Repeated Alternative Vote: Repeated Alternative Vote (RAV) is an extension of the Alternative Vote (AV) [97] into a rule which returns a complete order over all the candidates [61]. For the selection of a single candidate there is no difference between RAV and AV. Scores are computed for each candidate as in Plurality. If no candidate has a strict majority of the votes the candidate receiving the fewest first place votes is dropped from all ballots and the votes are re-counted. If any candidate now has a strict majority, they are the winner. This process is repeated up to $m-1$ times [61]. In RAV this procedure is repeated, removing the winning candidate from all votes in the election after they have won, until no candidates remain. The order in which the winning candidates were removed is the total ordering of all the candidates.

With the selection of a voting rule it is almost as important to consider the selection of the tie-breaking method. Many times, candidates will tie with the same number of votes but we require a single winner. Oftentimes, in this dissertation, we speak of voting rules as returning a winning set of candidates and thus, we do not require a tie-breaking method. However, sometimes we will need a single or unique winner. In these cases we will define the tie-breaking method that we employ for the particular problem under consideration. There are a variety of tie-breaking methods used in the literature including lexicographic linearization, randomized tie-breaking, partial re-voting, and, in the state of New Mexico, "any reasonable game of chance such as poker or craps [77]." The effect of particular tie-breaking methods on the reasoning complexity of problems studied in later chapters can be significant and, often, tie-breaking is a completely separate line of investigation when studying voting rules [58, 98].

Axiomatic Properties of Voting Rules

Axiomatic characterizations of voting rules is one way that social choice theorists have, over the years, attempted to answer the question, "which voting rule is the best?" We will only scratch the surface of the field in this section. The references in the last section provide a more complete characterization of voting rule properties. Some properties that will be important to us are the following:

Condorcet Criterion: A voting rule that always returns the Condorcet winner, when one exists, is said to obey the Condorcet criteria or be **Condorcet consistent**.

Majority Criterion: The majority criteria is a slightly weaker version of the Condorcet criteria. A voting rule obeys the majority criteria when it always selects the alternative that wins a majority of its head-to-head comparisons by a majority vote.

Resoluteness: A voting rule is resolute if it always picks exactly one winner out of the set of alternatives. Similarly, a social welfare function is resolute if it returns a linear ordering of all the alternatives.

Non-Dictatorship: A voting rule is non-dictatorial if there is no voter v_i such that the result of the election always matches the preferences of v_i. This means that the voting rule considers all voters and does not just return the preference profile of some special voter.

Pareto Optimality: A voting rule is Pareto optimal if, for all pairs of alternatives x and y, if all voters prefer $x > y$ in their individual preferences, then the result of the voting rule will have $x > y$.

Independence of Irrelevant Alternatives (IIA): A rule satisfies IIA if, for any pair of alternatives x and y, if x ranked ahead of y in the social welfare ordering and some set of voters changes their votes, but no voter changes the relative ordering of x and y in their preference list, x should still win the election. For voting rules, we modify

the definition slightly to say that if x is in the winner set, then moving y without interchanging the relative ordering x and y should not remove x from the winner set.

These axioms, along with many others defined in the social choice literature, provide us some means of talking about voting rules with respect to the properties of individual voting rules. We provide additional discussion about the evaluation of voting rules and the study of the particular issues related to certain voting rules in Chapter 5. The properties we have discussed so far do not tell us anything about the security of voting rules against attacks of various forms, which is the focus of this dissertation.

2.2.2 Affecting Elections: Bribery, Manipulation and Control

Making group decisions is hard, and a primary concern is the fairness of these group decisions. One aspect that concerns many voters is the fairness of the process: did someone or something get chosen as a winner that should not have been? Did this particular candidate win the election through some form of cheating or manipulation? These questions form the basis for evaluating the safety and security of voting rules.

The field of preference aggregation manipulation originally stems from the field of social choice. The cornerstone text, Arrow's *Social Choice and Individual Values*, shows that any preference aggregation scheme for more than three alternatives cannot simultaneously satisfy Pareto optimality, non-dictatorship, and be independent of irrelevant alternatives [2]. Arrow's principled look at choice procedures led to a cascade of work in the field of social choice and raised new questions about the fairness of voting rules. Building on Arrow's work, the Gibbard–Satterthwaite Theorem shows that any aggregation system, meeting the same set of properties as Arrow's Theorem, can be manipulated by non-truthful voting [67, 115]. The Duggan–Schwartz Theorem extends the Gibbard-Satterthwaite Theorem to an even larger set of aggregation methods by removing the requirement of resoluteness [41].

Taking the results of Arrow and Duggan–Schwartz, we are left with the result that we cannot devise a fair and resolute preference aggregation scheme that is immune to manipulation. This seems unsettling on many levels because it implies that groups can never come to fair, non-manipulated agreements. However, in the late 1980's and early 1990's Bartholdi et al. proposed the idea of protecting the aggregation schemes through computational complexity [5, 7]. The idea, much like the central idea of cryptography, is that if it is difficult to compute a manipulation scheme then it is unlikely that there will be manipulation. With this idea of security in mind, much of the work in the ComSoc community seeks to classify aggregation systems in terms of their susceptibility to manipulation.

There is an ongoing discussion in the ComSoc literature about the quality of the protection afforded by computational complexity [32, 105]. In many real world cases, the number of candidates or voters is extremely limited. In these cases, even NP-completeness is not enough protection since it only tells us about the worst case, not the average case. However, in cases where we have a very large set of alternatives NP-completeness may be enough. The ComSoc community is actively investigating the sufficiency of NP-completeness as a computational barrier for protecting elections and enforcing honesty by the participants [56].

In the study of election systems we generally talk about three different ways to affect elections and social choice systems: bribery (influence), manipulation, and control. In bribery, each voter has a preference over the possible outcomes; we ask, can an outside actor change individual agents' votes in order to make some preferred outcome a winner? In manipulation, we are given a set of voters among all the possible voters; we ask, can we change the votes of only the given set of voters to make a preferred outcome a winner? In control, we are given all voters and their votes; we ask, can we change the requirements of the election (through adding or deleting voters or outcomes; or by changing the order of the comparisons in a voting tree) in order to make a preferred outcome a winner?

From this framework of ideas there stems a large collection of literature on the compu-

tational complexity of affecting elections. A survey by Faliszewski et al. gives a strong overview of the current work in the area of bribery and control [57]. There has also been work on the worst-case complexity of manipulation of voting mechanisms by voters [6, 30, 33] and average case complexity of manipulations [32, 47]. We provide a more focused look at the literature on bribery and manipulation as they relate to our models in Chapter 3 and Chapter 4.

In this dissertation we primarily focus on the bribery and manipulation problems. There has been a volume of research in computer science that address bribery and lobbying in deterministic domains [24, 49, 50, 52]. However, the study of manipulation is not the sole purview of computer science. There is a rich overlap between computer science and other disciplines including applied mathematics, operations research, and game theory. The search for a "better" preference aggregation function is an ongoing area of research. See the books by Taylor and Arrow for a current overview [3, 122].

Game theory [84, 127] has made, and continues to make, significant impacts on the study of voting and lobbying behavior. Shapley and Shubik studied the power of members in a committee system with voting represented as a simple game [118]. Operations and economic researchers have studied the lobbying process for both the US and European systems [36, 110]. One of the earliest formal models we can find of the lobbying process, as it is performed in the United States, is an operations research paper by Reinganum [110]. Baye et al. use the economic idea of "rent seeking" to study the lobbying process [8, 9]. While these studies focused only on the game theoretic aspects of behavior equilibriums, we focus on the complexity of finding computationally efficient manipulation schemes.

We also mention the increasing overlap between the economics research and the computer science research. It is important to understand that these disciplines pursue their research ends for different reasons but that significant overlap and cross-pollination exists. A good example of these inter-related ideas is a paper by Sandholm et al. in which auction generalizations are used to manipulate robot agents [114]. In addition, the book by Nisan

et al. provides an overview of game-theoretic aspects in use within multi-agent artificial intelligence systems [96].

Politics, Bribery and Real Life

There has been a move in the ComSoc community to reframe the discussion of the bribery problem in a more positive light. Initially the idea was to investigate the robustness of voting rules to various attacks [52]. However, the bribery problem is really a problem of resource allocation: any resource that can be distributed unevenly among the entrants can be used to affect the result. These resources could be referees or home fields in sports; volunteers and canvassers in political elections; money spent on targeted advertising and product placement; concessions made to friends; or subsidies in a spending bill. The ComSoc community uses the term "bribery" when unequal distribution of resources is a more general viewpoint. Due to recent developments on the bribery problem presented by Schlotter et al. [116], Elkind and Faliszewski [42], and some in this dissertation and supporting publications, the ComSoc community has started to look at these problems in a more positive light.

In this dissertation we focus on bribery and manipulation of preference aggregation functions. Arguably, the most important use of these functions in modern life are elections. Elections attempt to aggregate societal preferences (about politicians or pop songs) into winners and losers. The winners of these elections become presidents or albums of the year. Some elections are more important than others, and here we consider how these problems relate to political elections. Of course, when we speak of manipulation and bribery we are not advocating these procedures, however, it is important that we understand how susceptible our current procedures are to non-truthful and manipulative practices so that we can better secure them against these malicious actions.

Political scientists study voting behavior of congressional members under many different lights. Poole and Rosenthal provide a review and models for modern spatial voting

theory [102]. This method classifies representatives in a two-dimensional voting space and uses this classification to predict and understand voting behavior. Thanks to computers and the Internet it is now possible to perform powerful statistical analysis on both roll call voting data [133] and political contributions [63]. There is a rich literature which attempts to address issues of representation and influence in the US political system. While the literature does not show a direct correlation between moneyed contributions and roll call votes, it allows for the conclusion that all constituents are not necessarily represented equally [69, 83]. We focus our discussion on determining what factors exert the most influence on these decision makers and how these factors can contribute to our models of influence and manipulation.

We cannot conclusively show that money directly buys roll call votes. In a paper by Hall and Wayman money is tied to a "participation metric" [69]. The model developed by the authors assumes that money buys access to a representative. The author tracks contributions to see what effects the contributions have, if any, on the behavior of the representatives. The conclusion is that time and information are the most important resources to a congressional member. Therefore, money and resources have some effect on congressional behavior but not necessarily a direct effect [69].

The findings by Hall and Wayman are reinforced in a paper by Clinton which develops a model to determine who is represented by congressional members [26]. Clinton formulates novel metrics to detail how responsive a congressional member is to their constituents. Clinton uses local survey data from constituents to determine a district's true voting preference and compares this with observable representative votes. He concludes that constituent preferences do not (necessarily) determine how the representative votes. This conclusion holds both for a large set of 800 roll call votes (within one congress) and 25 "key votes" on issues the author identifies as "highly salient", including health care, war, and abortion. The paper shows large systematic differences between what constituents desire and how representatives vote. Clinton speculates that the discrepancy could be due to "party influ-

ence, presidential influence, or other outside actors [26]." The conclusion that there are hidden influence structures within the US congressional system is supported by a different model developed by Levitt [83]. Clinton also studies voting patterns using Bayesian models for estimation and inference of congressional members' ideologies to account for these ideological discrepancies [25].

An important distinction to keep in mind is that not all votes in the US Congress are roll call votes. Other types of votes include quorum calls, committee votes, and subcommittee votes [102]. Therefore, looking at just roll call votes may not be enough to determine all the influence factors that come into a representatives' decision. In fact, most of the content of bills is written by committees long before a bill ever goes up for a general vote. Most lobbying occurs during this process and we must temper our expectations of how much we can learn from an analysis of roll call data [48].

We use the political science research as a tool to inform our models where appropriate. By looking at others' research to determine what influence factors come in to play during voting decisions, we hope to better construct our models. The complexity of the particular voting aggregation method is only one factor which goes into manipulation. It is important to see if, given varying influence factors, the voting procedures retain their easy manipulation results.

Chapter 3 Bribery and Manipulation with Uncertain Information

This chapter details work on probabilistic models of lobbying in environments with multiple referenda and sports tournaments. This work includes model building, aggregation methods, and complexity results for the given models and methods.

Section 3.1 develops a novel and flexible model to represent majority voting with uncertain information. In this section we provide a classification of the complexity of finding efficient bribery schemes given a set of voters. Their individual preferences are represented as probability distributions over a set of issues, their prices for changing their preferences, and a budget. Much of this material is available both as a conference publication [45] and a longer version is available as a technical report [46]. This first section also develops three different ways in which an outside actor can channel money to the individual voters and establishes three different criteria to evaluate the outcome of a vote in settings with uncertain information. We show that, depending on the particular combination of evaluation and bribery models chosen, the nine problems range in complexity from polynomial time to NP-complete; these results are summarized in Table 3.1 and Table 3.2. This difference reveals that modeling choices can have significant effects on the complexity of calculating efficient bribery schemes.

Section 3.2 develops a novel and flexible model to represent sports tournaments and competitions. In this section we show a classification of the complexity of finding efficient bribery schemes for three common types of sports tournaments over five distinct problem variants given a set of teams, their probabilities of each possible win, and prices for decreasing their competitive output (purposefully losing or underperforming in a match). This work has been previously published and is available as a conference publication [87]. The evaluation complexity of these problems range from polynomial time to $\mathrm{NP}^{\mathrm{PP}}$ and is summarized in Table 3.3. The results show that in some cases the added uncertainty in-

creases the complexity of manipulating sports tournaments, while in other cases it does not. While this increase in complexity is not uniform across all tournament types, the change shows strong evidence that reasoning in domains with uncertain data leads to an increase in reasoning complexity.

3.1 Majority Voting and Multiple Referenda

In most democratic political systems, laws are passed by elected officials who are supposed to represent their constituencies. Individual entities such as citizens or corporations are not supposed to have undue influence in the wording or passage of a law. However, they are allowed to make contributions to representatives, and it is common to include an indication that the contribution carries an expectation that the representative will vote a certain way on a particular issue.

Many factors can affect a representative's vote on a particular issue. There are the representative's personal beliefs about the issue, which presumably were part of the reason that the constituency elected them. There are also the campaign contributions, communications from constituents, communications from potential donors, and the representative's own expectations of further contributions and political support [83].

It is a complicated process to reason about. Earlier work (see the references given in Chapter 2) considered the problem of meting out contributions to representatives in order to pass a set of laws or influence a set of votes. However, the earlier computational complexity work by Christian et al. [24] and others [52, 110] on this problem made the assumption that a politician who accepts a contribution will in fact—if the contribution meets a given threshold—vote according to the wishes of the donor.

It is said that "an honest politician is one who stays bought," but that does not take into account the ongoing pressures from personal convictions and opposing lobbyists and donors. We consider the problem of influencing a set of votes under the assumption that we can influence only the *probability* that the politician votes as we desire.

There are several axes along which the picture is complicated in realistic scenarios. To describe these formally, we introduce various evaluation criteria and bribery methods. The first is the notion of sufficiency: What does it mean to say we have donated enough to influence the vote? Does it mean that the probability that a single vote will go our way is greater than some threshold, or that the probability that all the votes go our way is greater than that threshold? We formally define and discuss these and other criteria in the section on evaluation criteria (Section 3.1.3). In particular, we consider three methods for evaluating the outcome of a vote given voters' probability of voting "yes" on a particular issue.

Strict Majority (SM): A vote on an issue is won by a strict majority of voters having a probability of accepting this issue that exceeds a given threshold.

Average Majority (AM): A vote on an issue is won exactly when the voters' average probability of accepting this issue exceeds a given threshold.

Probabilistic Majority (PM): A vote on an issue is won exactly when the sum of the probabilities of possible futures (i.e., of possible scenarios in which a strict majority of voters accepts this issue) exceeds a given threshold.

How does one donate money to a campaign? In the United States there are several laws that influence how, when, and how much a particular person or organization can donate to a particular candidate. We examine ways in which money can be channeled into the political process in the section on bribery methods (Section 3.1.2). In particular, we consider three methods that an actor (called "The Lobby") can use to influence the voters' preferences of voting for or against multiple issues.

Microbribery (MB): The Lobby may choose which voter to bribe and on which issue in order to influence the outcome of the vote.

Issue Bribery (IB): The Lobby may choose which issues to support, and, for each issue supported, the funds are equally distributed over all the voters.

Voter Bribery (VB): The Lobby may choose which voters to bribe, and, for each voter bribed, the funds are equally distributed over all the issues.

The voter bribery method is due to Christian et al. [24], who were the first to study lobbying in the context of direct democracy where voters vote on multiple referenda. Their "Optimal Lobbying" problem (denoted OL) is a deterministic and unweighted variant of the lobbying problems that we present in this chapter. A generalized problem described by Christian et al., the "Optimal Weighted Lobbying" (OWL) problem, which allows different voters to have different prices and so generalizes OL, can be expressed as and solved via the "binary multi-unit combinatorial reverse auction winner-determination problem"(see [114] for its definition).

The microbribery method in the context of lobbying—though inspired by the different notion of microbribery than Faliszewski et al. [53–55] introduced in the context of bribery in voting—is new to this paper. As described in this section, microbribery more closely resembles the bribery methods discussed in Faliszewski [51] and Christian et al. [24] than any other existing model as we allow for individually priced voters and the novel addition of allowing voters prices to range based on the amount of change.

In Section 3.1.1 we formally describe our model of reasoning about bribery in voting with multiple referenda where voters express their probabilities of voting for or against each issue. In Section 3.1.2 we formally describe the three bribery methods at the disposal of The Lobby. In Section 3.1.3 we then describe three different evaluation criteria for this scenario. Section 3.1.4 formally states the nine decision problems we study in this domain while Section 3.1.5 formally states an instance of the problem where The Lobby expresses their preferences as weights over individual issues. Section 3.1.6 details our complexity results and ends with a few observations about the change in reasoning complexity when

models move from deterministic setting to models of uncertain information.

3.1.1 Initial Model

We begin with a simplistic version of the PROBABILISTIC LOBBYING PROBLEM (PLP, for short), in which voters start with initial probabilities of voting for an issue and are assigned known costs for increasing their probabilities of voting according to The Lobby's[1] agenda by each of a finite set of increments. The question, for this class of problems, is: given the above information, along with an agenda and a fixed budget B, can The Lobby target its bribes in order to achieve its agenda?

The complexity of the problem seems to hinge on the evaluation criterion for what it means to "win a vote" or "achieve an agenda." We discuss the possible interpretations of evaluation and bribery later in this section. First, however, we will formalize the problem by defining data objects needed to represent the problem instances. A similar model was first discussed by Reinganum [110] in the continuous case and we translate it here to the discrete case. This will allow us to present algorithms for, and a complexity analysis of, the problem.

Let $\mathbb{Q}_{[0,1]}^{m\times n}$ denote the set of $m\times n$ matrices over $\mathbb{Q}_{[0,1]}$ (the rational numbers in the interval $[0,1]$). We say $P \in \mathbb{Q}_{[0,1]}^{m\times n}$ is a probability matrix (of size $m\times n$), where each entry $p_{i,j}$ of P gives the probability that voter v_i will vote "yes" for referendum (synonymously, for issue) r_j. The result of a vote can be either a "yes" (represented by 1) or a "no" (represented by 0). Thus, we can represent the result of any vote on all issues as a $0/1$ vector $\vec{X} = (x_1, x_2, \ldots, x_n)$, which is sometimes also denoted as a string in $\{0,1\}^n$.

We associate with each voter/issue pair (v_i, r_j) a discrete price function $c_{i,j}$ for changing v_i's probability of voting "yes" for issue r_j. Intuitively, $c_{i,j}$ gives the cost for The Lobby of raising or lowering (in discrete steps) the ith voter's probability of voting "yes" on the jth issue. A formal description is as follows.

[1] In this chapter we use The Lobby to represent any outside agent attempting to manipulate the vote. This is in keeping with other work in the ComSoc community [52].

Given the entries $p_{i,j} = {}^{a_{i,j}}/_{b_{i,j}}$ of a probability matrix $P \in \mathbb{Q}_{[0,1]}^{m \times n}$, where $a_{i,j} \in \mathbb{N} = \{0,1,\ldots\}$, $b_{i,j} \in \mathbb{N}_+ = \{1,2,\ldots\}$, and $a_{i,j} \leq b_{i,j}$, choose some $k \in \mathbb{N}$ such that $k+1$ is a common multiple of all $b_{i,j}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, and partition the probability interval $[0,1]$ into $k+1$ steps of size ${}^1/_{(k+1)}$ each.[2] The integer k will be called the discretization level of the problem. For each $i \in \{1,2,\ldots,m\}$ and $j \in \{1,2,\ldots,n\}$, $c_{i,j} : \{0, {}^1/_{(k+1)}, {}^2/_{(k+1)}, \ldots, {}^k/_{(k+1)}, 1\} \to \mathbb{N}$ is the *(discrete) price function for* $p_{i,j}$, i.e., $c_{i,j}({}^\ell/_{(k+1)})$ is the price for changing the probability of the ith voter voting "yes" on the jth issue from $p_{i,j}$ to ${}^\ell/_{(k+1)}$, where $0 \leq \ell \leq k+1$. Note that the domain of $c_{i,j}$ consists of $k+2$ elements of $\mathbb{Q}_{[0,1]}$ including 0, $p_{i,j}$, and 1. In particular, we require $c_{i,j}(p_{i,j}) = 0$, i.e., a cost of zero is associated with leaving the initial probability of voter v_i voting on issue r_j unchanged. Note that $k = 0$ means $p_{i,j} \in \{0,1\}$, i.e., in this case each voter either accepts or rejects each issue with certainty and The Lobby can only flip these results. This special case in our problem definition encompasses the Optimal Lobbying problem of Christian et al. [24]. The image of $c_{i,j}$ consists of $k+2$ nonnegative integers including 0, and we require that, for any two elements a, b in the domain of $c_{i,j}$, if $p_{i,j} \leq a \leq b$ or $p_{i,j} \geq a \geq b$, then $c_{i,j}(a) \leq c_{i,j}(b)$. This guarantees monotonicity on the prices.

We represent the list of price functions associated with a probability matrix P as a table C_P, called the cost matrix, whose $m \cdot n$ rows give the price functions $c_{i,j}$ and whose $k+2$ columns give the costs $c_{i,j}({}^\ell/_{(k+1)})$, where $0 \leq \ell \leq k+1$. Note that we choose the same k for each $c_{i,j}$, so we have the same number of columns in each row of C_P. The entries of C_P can be thought of as "price tags" indicating what The Lobby must pay in order to change the probabilities of voting.

The Lobby also has an integer-valued budget B and an "agenda," which we will denote as a vector $\vec{Z} \in \{0,1\}^n$ for n issues, containing the outcomes The Lobby would like to see on these issues. For The Lobby, the prices for a bribery that moves the outcomes of a

[2]There is some arbitrariness in this choice of k. One might think of more flexible ways of partitioning $[0,1]$. We have chosen this way for the sake of simplifying the representation, but we mention that all that matters is that for each i and j, the discrete price function $c_{i,j}$ is defined on the value $p_{i,j}$, and is set to zero for this value.

referendum in the wrong direction do not matter. Hence, if $\vec{Z}$ is zero at position j, then we can set $c_{i,j}(a) = --$ (indicating an unimportant entry) for $a > p_{i,j}$, and if $\vec{Z}$ is one at position j, then we can set $c_{i,j}(a) = --$ (indicating an unimportant entry) for $a < p_{i,j}$. Without loss of generality, we can also assume that $c_{i,j}(a) = 0$ if and only if $a = p_{i,j}$.

For simplicity, we may assume that The Lobby's agenda is all "yes" votes, so the target vector is $\vec{Z} = 1^n$. This assumption can be made without loss of generality, since if there is a zero in $\vec{Z}$ at position j, we can flip this zero to one and also change the corresponding probabilities $p_{1,j}, p_{2,j}, \ldots, p_{m,j}$ in the jth column of P to $1 - p_{1,j}, 1 - p_{2,j}, \ldots, 1 - p_{m,j}$. (See the evaluation criteria in Section 3.1.3 for how to determine the result of voting on a referendum.) Moreover, the rows of the cost matrix C_P that correspond to issue j have to be mirrored, that is, the prices have been flipped so we are attempting to achieve all yes' instead of a mix of yes and no votes.

Example 3.1.1 *Consider the following problem instance with $k = 9$ (so there are $k+1 = 10$ steps), $m = 2$ voters, and $n = 3$ issues. We will use this as a running example for the rest of this section. In addition to the above definitions for k, m, and n, we give the following probability matrix P and cost matrix C_P for P. This example is normalized for an agenda of $\vec{Z} = 1^3$, which is why The Lobby has no incentive for lowering the acceptance probabilities, so those costs are omitted below.*

Our example consists of a probability matrix P:

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4

and the corresponding cost matrix C_P:

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

In Section 3.1.2, we describe three bribery methods which are three specific ways in which The Lobby can influence the voters. These will be referred to as *microbribery* (MB), *issue bribery* (IB), and *voter bribery* (VB). In addition to the three bribery methods described in Section 3.1.2, we define three ways of evaluating a set of votes. These evaluation criteria are defined in Section 3.1.3 and will be referred to as *strict majority* (SM), *average majority* (AM), and *probabilistic majority* (PM). It is important to formalize the notion of winning in this problem due to our modeling of uncertainty. Given different types of information agents can choose to optimize over different realizations of systems that contain uncertainty. This method of examining different decision strategies is in keeping with other works on game theory and decision making under uncertainty, see Luce and Raiffa for a more complete treatment [84]. The nine basic probabilistic lobbying problems we will study (each a combination of MB/IB/VB bribery under SM/AM/PM evaluation) are defined in Section 3.1.4, and a modification of these basic problems with additional issue weighting is introduced in Section 3.1.5.

3.1.2 Bribery Methods

We begin by formalizing the bribery methods by which The Lobby can influence votes on issues. We will define three methods for donating this money.

Microbribery (MB)

The first method at the disposal of The Lobby is *microbribery*. Though our notion of microbribery was inspired by the work of Faliszewski et al. [53–55], it should not be confused with their definition of the term "microbribery," used in the context of bribing "irrational" voters in Llull/Copeland elections. In the Llull/Copeland elections, voters are represented via binary preference relations that may or may not be transitive. Microbribery in the model defined by Faliszewski et al. [53–55] allows the briber to flip single entries in the voters' preference tables possibly making each voter irrational (a voter with non-transitive preferences). Microbribery is the editing of individual elements of the P matrix according to the costs in the C_P matrix. Thus The Lobby picks both which voter to influence and on which issue to influence that voter. This bribery method allows the most flexible version of bribery defined in this document. It generally models private donations made to candidates in support of specific issues from either Political Action Committees (PACs) or interested individual parties. This method of contribution, in some cases, has an impact on an individual's platform or voting beliefs [48].

More formally, if voter i is bribed with d dollars on issue j, then all entries $c_{i,j}[\ell]$ are updated as follows:

$$c_{i,j}[\ell] := \begin{cases} -- & \text{if}(c_{i,j}[\ell] = --) \vee ((c_{i,j}[\ell] - d) \leq 0) \\ c_{i,j}[\ell] - d & \text{if}(c_{i,j}[\ell] - d) > 0. \end{cases}$$

We also need to update P with the correct discrete price step. To do this we determine the maximum "$--$" entry in $c_{i,j}$, $U = \max\{x | c_{i,j}[x] = --\}$. We then update $c_{i,j}[x] = 0$ and set $p_{i,j} = {}^{x}\!/_{(k+1)}$.

Example 3.1.2 *Take our running example. In order to see the effect of MB, imagine The Lobby were to donate* \$100 *to voter* v_1 *on issue* r_2*. This would raise the probability of* v_1 *voting yes on* r_2 *to* 0.6*. The updated P matrix* P' *and updated* C_P *matrix* C'_P *are:*

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.6	0.5
v_2	0.4	0.7	0.4

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	−−	−−	−−	0	40	210	420	500
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

Issue Bribery (IB)

The second method at the disposal of The Lobby is *issue bribery*. We can see from the P matrix that each column represents how all voters think about a particular issue. In this method of bribery, The Lobby can pick a column of the matrix and edit it according to some budget. The money will be equally distributed among all the voters and the voter probabilities will move accordingly. So, for d dollars donated, each voter receives a fraction of d/m and his or her probability of voting "yes" changes accordingly. This can be thought of as special-interest group donations. Special-interest groups such as PETA[3] focus on issues and dispense their funds across an issue rather than by voter. The bribery could be funneled through such groups.

This method of influence is demonstrated in the US political system through the use of Super PACs in light of the US Supreme Court ruling, *Citizen United v. Federal Election Commission*. This controversial decision allows for almost unlimited campaign contribu-

[3]People for the Ethical Treatment of Animals, a narrow-focus group that protests animal testing of food and drugs, and the swatting of flies.

tions from interested parties to Super PACs; which are political action committees designed to generally support a narrow set of issues through the use of direct lobbying efforts and media campaigns.

We require the elements of C_P to be discrete, so we must place some restrictions on this method of bribery in order to avoid fractional dollars. Specifically, we assume that bribery will be donated in multiples of m, the number of candidates. In this way only integer numbers of dollars will be donated per voter. Example 3.1.3 illustrates the process.

Example 3.1.3 *Take our running example. In order to see the effect of IB, imagine The Lobby were to donate* \$140 *to issue* r_1. *Since there are two voters this means that* \$70 *is donated to each of* v_1 *and* v_2 *on issue* r_1. *The updated P matrix* P' *and updated* C_P *matrix* C'_P *are:*

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	0.7	0.4

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	30	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	−−	−−	−−	0	50	130	200
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

Voter Bribery (VB)

The third method at the disposal of The Lobby is *voter bribery*. In the P matrix, each row represents how an individual voter will cast his ballot for all issues on the docket. In this

method of bribery, The Lobby picks a voter and then pays to edit the entire row at once with the funds being equally distributed over all the issues. So, for d dollars a fraction of d/n is spent on each issue, which moves accordingly. The cost of moving the voter is given by the C_P matrix as before. This method of bribery is analogous to "buying" or pushing a single politician or voter. The Lobby seeks to donate so much money to some individual voters that they have no choice but to move all of their votes toward The Lobby's agenda.

This method of general lobbying occurs in the US political system [48], though not always in the form of monetary exchange. As discussed by Hall and Wayman [69], oftentimes an interested party will provide education or information about a broad set of issues to a voter. This information, in many cases, is biased in support of the interested parties position on the legislation. This method of intervention can change a voter's opinion across a large set of issues.

Again, we require the elements of C_P to be discrete so we must place some restrictions on this method of bribery in order to avoid fractional dollars. Specifically, we assume that bribery will be donated in multiples of n, the number of issues. In this way only integer numbers of dollars will be donated per voter. Example 3.1.4 illustrates the process.

Example 3.1.4 *Take our running example. In order to see the effect of VB, imagine The Lobby were to donate* \$270 *to voter* v_2*. Since there are three issues this means that* \$90 *is donated to each of the three issues for* v_2*. The updated P matrix* P' *and updated* C_P *matrix* C'_P *are:*

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	1.0	0.6

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	−−	−−	−−	−−	30	110	160
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	−−	−−	−−	0
$c_{2,3}$	−−	−−	−−	−−	−−	−−	0	10	90	210	360

Observe that microbribery is equivalent to issue bribery if there is only one voter. Similarly, microbribery is equivalent to voter bribery if there is only one referendum.

3.1.3 Evaluation Criteria

Defining criteria for how an issue is won is the next important step in formalizing our models. Here we define three methods that one could use to evaluate the eventual outcome of a vote. Since we are focusing on problems that are probabilistic in nature, it is important to note that no evaluation criterion will guarantee a win. The criteria below yield different outcomes depending on the model and problem instance.

Strict Majority (SM)

For each issue, a strict majority of the individual voters have probability greater than some threshold, t, of voting according to the agenda.

In Example 3.1.1, with $t = 0.5$ we would have the following result:

$P =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4
SM	0	0	0

None of the issues has a strict majority of voters with above a 0.5 probability of voting "yes" in this setting and thus, The Lobby has not achieved its agenda. However, if we look at the result for Example 3.1.4, with $t = 0.5$ after the illustrated round of VB we have:

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	1.0	0.6
SM	1	0	0

While The Lobby has still not achieved its agenda, it has moved closer to its desired result with the selected bribery action since there is now one issue, r_1, which has a strict majority of voters with $P_{i,1} > 0.5$ probability of voting in accordance with The Lobby.

Average Majority (AM)

For each issue r_j of a given probability matrix P, we define the average probability $\overline{p_j} = (\Sigma_{i=1}^{m} p_{i,j})/m$ of voting "yes" for r_j. We can now evaluate the vote to say that r_j is accepted if and only if $\overline{p_j} > t$ where t is some threshold.

In Example 3.1.1, with $t = 0.5$ we would have the following result:

$P =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4
$\overline{p_j}$	0.6	0.5	0.45
AM	1	0	0

This table is augmented with the resultant probability ($\overline{p_j}$) and the AM result for $t = 0.5$. The Lobby has achieved exactly one of its desired results in this example. However, if we investigate the voting result from Example 3.1.4 with $t = 0.5$ we have the following result after a round of VB:

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	1.0	0.6
$\overline{p_j}$	0.75	0.65	0.55
AM	1	1	1

In this example the round of VB has been successful, The Lobby has fully achieved its agenda through bribery.

Probabilistic Majority (PM)

The third criterion takes into account the probabilities of possible *scenarios*, or possible futures. Each possible scenario, in which voters commit to "yes" or "no" votes for each issue, has a probability. Under this criterion, the probability that The Lobby's agenda passes is the sum of probabilities of those scenarios in which the agenda passes. We call these majority scenarios and denote their issue-wise probability with $\vec{S} \in \mathbb{Q}_{[0,1]}$.

More formally, for each issue r_j of a given probability matrix P, we want to find the sum of the probabilities of those futures in which a strict majority of voters vote "yes" for r_j. We say that there is a probabilistic majority for r_j if the probability of receiving a majority of "yes" votes exceeds a threshold t. A similar possible futures evaluation metric is used by [71].

In our running example there are only two voters and, therefore, only one scenario, for each issue, in which a strict majority is reached. This is the situation where both voters cast "yes" ballots for both issues. We can compute these values by multiplying the probabilities of "yes" votes by each other ($0.8 \cdot 0.4$ for r_1 etc.). In Example 3.1.1, with $t = 0.2$ we would have the following result:

$P =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4
S_i	0.32	0.21	0.20
PM	1	1	0

This table is augmented with the resultant probability of a majority scenario (S_i) and the PM result for $t > 0.2$. The Lobby has achieved its desired effect on r_1 and r_2. However, if we investigate the voting result from Example 3.1.4 with $t = 0.5$ we have the following result after a round of VB:

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	1.0	0.6
S_i	0.56	0.30	0.30
PM	1	1	1

In this example the round of VB has been successful, The Lobby has fully achieved its agenda through bribery.

The examples shown here contain only two voters and, therefore, only one majority scenario for each issue. When there are more than two voters the computation of the probability of a majority scenario is not so straightforward.

Observe that all three evaluation criteria coincide if there is only one voter or if the discretization level equals zero, so the problem is deterministic.

3.1.4 Basic Probabilistic Lobbying Problem

We can now introduce the nine basic problems that we will study. For $\mathrm{X} \in \{\mathrm{MB}, \mathrm{IB}, \mathrm{VB}\}$ and $\mathrm{Y} \in \{\mathrm{SM}, \mathrm{AM}, \mathrm{PM}\}$, we define the following problem.

Name: X-Y PROBABILISTIC LOBBYING PROBLEM.

Given: A probability matrix $P \in \mathbb{Q}_{[0,1]}^{m \times n}$ with a cost matrix C_P (with integer entries), a budget B, and some threshold $t \in \mathbb{Q}_{[0,1]}$.

Question: Is there a way for The Lobby to influence P using bribery method X and evaluation criterion Y, without exceeding budget B, such that the result of the votes on all issues equals 1^n?

We abbreviate this problem name as X-Y-PLP. Observe that the discretization level is an implicit, unary parameter of the problem. It is indirectly specified through the given cost matrix C_P for a problem instance.

Example 3.1.5 *Recall our running Example 3.1.1. We have the following matrices for P and C_P:*

$P =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4

$C_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

If we have microbribery with the average majority criterion and $t = 0.5$ with the matrices considered above, then The Lobby needs to select some voters to bribe in order to achieve its agenda. If the budget is set for \$50*, then we have an instance of* MB-AM-PLP *with $(P, C_P, 50, 0.5)$ with P and C_P as shown above and target vector $\vec{Z} = 1^n$. The Lobby*

needs to bribe v_1 *on* r_2 *with a payment of* \$10 *and* v_1 *on* r_3 *with a payment of* \$25. *The updated matrices are:*

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	−−	0	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	−−	−−	0	45	65	125
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.4	0.7
v_2	0.4	0.7	0.4
$\overline{p_j}$	0.6	0.55	0.55
AM	1	1	1

Each referendum passes according to the AM evaluation criteria and therefore $(P, C_P, 50, 0.5)$ *is in* MB-AM-PLP.

3.1.5 Issue Weighting

We augment the model to include the concept of issue weighting. It is reasonable that certain issues will be of more importance than others. For this reason we will allow The Lobby to assign higher weights to the issues that they deem more important. The lobby no longer states an agenda, rather they have weights over the issues and a target total weight. These positive integer weights will be defined for each issue.

We will specify these weights as a vector $\vec{W} \in \mathbb{N}_+^n$ of length n, the total number of issues in our problem instance. The higher the weight, the more important that particular

issue is to The Lobby. Along with the weights for the issues we are also given an objective value $V \in \mathbb{N}_+$, which is the minimum weight The Lobby wants to see passed. We allow this set to be a partial order (a reflexive, transitive, and antisymmetric ordering) over the weights. Therefore, it is possible for The Lobby to have an ordering such as $w_1 = w_2 = \cdots = w_n$. If this is the case, and $V = n$, we are left with an instance of X-Y-PLP, where $\mathrm{X} \in \{\mathrm{MB}, \mathrm{IB}, \mathrm{VB}\}$ and $\mathrm{Y} \in \{\mathrm{SM}, \mathrm{AM}, \mathrm{PM}\}$.

We now introduce the nine probabilistic lobbying problems with issue weighting. For $\mathrm{X} \in \{\mathrm{MB}, \mathrm{IB}, \mathrm{VB}\}$ and $\mathrm{Y} \in \{\mathrm{SM}, \mathrm{AM}, \mathrm{PM}\}$, we define the following problem.

Name: X-Y PROBABILISTIC LOBBYING PROBLEM WITH ISSUE WEIGHTING.

Given: A probability matrix $P \in \mathbb{Q}_{[0,1]}^{m \times n}$ with cost matrix C_P an issue weight vector $\vec{W} \in \mathbb{N}_+^n$, an objective value $V \in \mathbb{N}_+$, and a budget B.

Question: Is there a way for The Lobby to influence P using bribery method X and evaluation criterion Y, without exceeding budget B such that the total weight of all issues that pass is at least V?

We abbreviate this problem name as X-Y-PLP-WIW.

Example 3.1.6 *Consider our running example, now augmented with* $\vec{W} = \langle 5, 10, 10 \rangle$. *We now provide an augmented P matrix which includes our vector of weights.*

$P =$

	r_1	r_2	r_3
w_i	5	10	10
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4

$C_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

If we have microbribery with the average majority criterion and $t = 0.5$ *with the matrices considered above, then The Lobby needs to select some voters to bribe in order to achieve its objective value* $V = 25$*. If the budget is set for* \$75*, then we have an instance of* VB-AM-PLP-WIW *with* $(P, C_P, 75, 0.5, 25)$ *with* P *and* C_P *as shown above. The Lobby needs to bribe* v_1 *with a payment of* \$75*. This payment will be split evenly over all the issues for* v_1*. The updated matrices are:*

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	75	115
$c_{1,2}$	−−	−−	−−	−−	0	45	75	115	285	495	575
$c_{1,3}$	−−	−−	−−	−−	−−	−−	−−	0	45	65	125
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

Table 3.1: Complexity results for the X-Y PROBABILISTIC LOBBYING PROBLEM, where $X \in \{MB, IB, VB\}$ and $Y \in \{SM, AM, PM\}$

Problem	Classical Complexity	Theorem or Corollary
MB-SM-PLP	P	Thm. 3.1.8
MB-AM-PLP	P	Thm. 3.1.9
MB-PM-PLP	$\in$ NP	Thm. 3.1.10 and 3.1.11
IB-SM-PLP	P	Thm. 3.1.12
IB-AM-PLP	P	Thm. 3.1.12
IB-PM-PLP	$\in$ NP	Thm. 3.1.10
VB-SM-PLP	NP-complete	Thm. 3.1.13
VB-AM-PLP	NP-complete	Thm. 3.1.13
VB-PM-PLP	NP-complete	Thm. 3.1.14

$P' =$

	r_1	r_2	r_3
w_i	5	10	10
v_1	0.8	0.4	0.7
v_2	0.4	0.7	0.4
$\overline{p_j}$	0.6	0.55	0.55
AM	1	1	1

In this case all the referenda have passed and so The Lobby has achieved its objective value $V = 25$. Therefore, $(P, C_P, 50, 0.5, 25)$ is in VB-AM-PLP-WIW.

3.1.6 Results

In this section we report some results from G. Erdélyi et al. [45]. There are additional results from this paper which we do not discuss here as these results were produced by coauthors and, while interesting, do not belong in this dissertation. We refer the reader to the long version of G. Erdélyi et al. [46] for a complete treatment of these problems including their parameterized complexity and approximability results. All the results related to the PM evaluation method are unique to this dissertation and have not been previously published.

Table 3.1 summarizes the classical complexity results for the X-Y PROBABILISTIC LOBBYING PROBLEM, where $X \in \{MB, IB, VB\}$ and $Y \in \{SM, AM, PM\}$. Most of the results show containment and hardness for certain complexity classes. However, in some cases we have not been able to show hardness for certain problems. In these cases we note the containment ($\in$) of the problem.

Complexity of Evaluation Methods

To begin our results we need to understand the complexity of evaluating a vote under our defined evaluation criteria. If it is computationally hard to evaluate the outcome of a vote, then it will necessarily be computationally hard to find sufficient bribery schemes for that voting system. Two of our evaluation criteria are easily computable from observation. SM requires only the investigation of each entry in the P matrix while AM requires computing a simple average. However, the evaluation procedure for PM is not so straightforward. Theorem 3.1.7 provides a polynomial time algorithm for computing the result of an election evaluated with the PM procedure.

Theorem 3.1.7 *Evaluating a winning scenario probability for* PM *is in P.*

Proof. We define a dynamic programming algorithm[4] which runs in time $\mathcal{O}(m^2)$ for a single issue, where m is the number of voters. We compute the probability $E_{i,x}$ that exactly i of the first x voters have voted "yes" on an issue. We build a table consisting of i rows and x columns called E and start indexing at $E_{1,1}$. We will create one table for each issue, r_j. We assume that P_i is the probability that voter v_i will vote "yes."

$E_{1,1} := P_1$

for $x := 1$ to $m-1$ **do**

$E_{1,x+1} = E_{1,x} \cdot (1 - P_{x+1}) + P_{x+1} \cdot \Pi_{j=1}^{x}(1 - P_j)$

end for

[4]We note that a similar proof appears in [71].

for $i := 2$ to $m-1$ **do**

for $k := 1$ to $m-1$ **do**

$E_{i,x+1} = E_{i-1,x} \cdot P_{x+1} + E_{i,x} \cdot (1 - P_{x+1})$

end for

end for

To determine whether the final probability that r_j passes exceeds t, we need to sum all elements of E such that $i > m/2$ and $x = m$. We compute the winning probability for each issue r_j with $1 \leq j \leq n$; the probability that the entire agenda passes is the product of the probabilities that each issue passes. The running time of the algorithm is $\mathcal{O}(m^2 \cdot n)$. ❑

Basic Probabilistic Lobbying Problem

We proceed by investigating each bribery method in turn. In some cases, multiple bribery methods can leverage the same proof of complexity. The first case we investigate is the MB method.

Theorem 3.1.8 MB-SM-PLP *is in P.*

Proof. The aim is to win all referenda. For each voter v_i and referendum r_j, $1 \leq i \leq m$ and $1 \leq j \leq n$, we can compute in polynomial time the amount $b(v_i, r_j)$ The Lobby has to spend to turn the favor of v_i in the direction of The Lobby (beyond the given threshold t). In particular, set $b(v_i, r_j) = 0$ if voter v_i would already vote according to the agenda of The Lobby. For each issue r_j, sort $\{b(v_i, r_j) \mid 1 \leq i \leq m\}$ non-decreasingly, yielding a sequence $b_1(r_j), \ldots, b_m(r_j)$ such that $b_k(r_j) \leq b_\ell(r_j)$ for $k < \ell$. To win referendum r_j, The Lobby must spend at least $B(r_j) = \sum_{i=1}^{\lceil (m+1)/2 \rceil} b_i(r_j)$ dollars. Hence, all referenda can be won if and only if $\sum_{j=1}^{n} B(r_j)$ is $\leq B$, the given bribery budget. ❑

Note that the time needed to execute the algorithm given in the previous proof can be bounded by a polynomial of low order. More precisely, if the input consists of m voters,

n referenda, and discretization level k, then $\mathcal{O}(n \cdot m \cdot k)$ time is sufficient to compute each $b(v_i, r_j)$. Having these values, $\mathcal{O}(n \cdot m \cdot \log(m))$ time is sufficient for the sorting phase. The sums can be computed in time $\mathcal{O}(n \cdot m)$. (Note that the time analysis can still be improved; however, a rough estimate of the computation time needed is enough to establish Theorem 3.1.8.)

Theorem 3.1.9 MB-AM-PLP *is in P.*

Proof. Let (P, C_P, B, t) be a given MB-AM-PLP instance, where $P \in \mathbb{Q}_{[0,1]}^{m \times n}$, C_P is a cost matrix, B is The Lobby's budget, and t is a given threshold. Let k be the discretization level of P, i.e., the interval is divided into $k+1$ steps of size $1/(k+1)$ each. For $j \in \{1, 2, \ldots, n\}$, let d_j be the minimum cost for The Lobby to bring referendum r_j into line with the jth entry of its target vector 1^n. If $\sum_{j=1}^{n} d_j \leq B$, then The Lobby can achieve its goal that the votes on all issues pass.

We show that, for a fixed j, we can, in polynomial time, compute d_j. Therefore, the decision problem of whether The Lobby can afford to bring all referenda into line with its target vector is also in P. We compute d_j by dynamic programming. The aim is to have $\sum_{i=1}^{m} p_{i,j}/m \geq t$, i.e., $\sum_{i=1}^{m} p_{i,j} \geq mt$. Recall that $p_{i,j} \cdot (k+1)$ always gives an integer. Define $c_j = (k+1)(mt - \sum_{i=1}^{m} p_{i,j})$. This is the overall number of confidence steps The Lobby has to buy to win referendum r_j. Note that c_j is polynomial in the size of the input, as is mtk. We define $T[l, s]$ to be the minimum cost of raising $(k+1)(\sum_{i=1}^{m} p_{i,j})$ by value $s \leq c_j$ using only microbribes to the first l voters. Notice, $T[l, 0] = 0$. Let $q_{i,j}$ be the integer with $c_{i,j}[q_{i,j}] = 0$. Hence, for κ with $0 \leq \kappa < q_{i,j}$, we find $c_{i,j}[\kappa] = --$, and for κ with $q_{i,j} < \kappa \leq k+1$, we have $c_{i,j}[\kappa] > 0$. This means that $q_{i,j}/(k+1) = p_{i,j}$. As the maximum number of confidence steps we can gain by bribing voter i is $k+1-q_{i,j}$, we obtain

$$T[1, s] = \begin{cases} \infty & \text{if } s > k+1-q_{1,j}, \\ c_{1,j}[q_{1,j}+s] & \text{otherwise.} \end{cases}$$

Based on this initialization, we can compute:

$$T[l,s] = \min\left\{T[l-1,s-q] + c_{l,j}[q_{l,j}+q] \mid 0 \leq q \leq \min\{s, k+1-q_{l,j}\}\right\}.$$

In particular, $q = 0$ covers the case when no money is spent on voter l, as $c_{l,j}[q_{l,j}] = 0$. Thus, each of the polynomially many entries, $T[l,s]$, can be computed in polynomial time. In particular, $T[m,c_j] = d_j$ can be computed in polynomial time. ❑

We have not been able to prove exact lower bounds for some bribery methods in conjunction with the PM method. However, we can establish upper bounds on MB and IB under the PM criteria.

Theorem 3.1.10 MB-PM-PLP *and* IB-PM-PLP *are in NP.*

Proof. Using a standard guess and check algorithm we can show that MB-PM-PLP and IB-PM-PLP are in NP. Given a set of bribery actions, we can verify if the resulting P matrix is sufficient to pass The Lobby's agenda using the algorithm shown in Theorem 3.1.7. ❑

Though we do not know the exact lower bound, we can show an algorithm that is pseudo-polynomial with respect to the size of the budget.

Theorem 3.1.11 *There is a bribery algorithm for* MB-PM-PLP *that runs in polynomial time with respect to B, n, and m.*

Proof. Since the evaluation method PM evaluates each referendum separately we will show a dynamic programming algorithm for a single issue. We can then apply this algorithm for all m issues.

Our goal is to raise above t the probability that a given referendum r will pass as cheaply as possible when only a subset of the voters is voting. Given bribes to the subset we can define a dynamic programming algorithm that incorporates each voter $x \in \{v_1, \ldots, v_n\}$ in

turn until we find the maximum probability of r passing using the least amount of our budget.

Consider an instance of MB-PM-PLP. We compute $F_x = \Pi_{j=1}^{x} p_j$, the probability that the single referendum will pass assuming that the voters $v_{x+1}, \ldots, v_n$ deterministically vote for the referendum. Without loss of generality, we assume that B is bounded by the sum of all possible bribes.

Our goal is to find the minimum cost to make $F_n > t$. We do this by finding the maximum value we can raise each F_x to, given each value $b \leq B$. This search is achieved using dynamic programming to build a $(n+1) \times B$ table D. Filling in each entry in the table takes polynomial time, so the problem is in P if the budget, B, is polynomial in the size of the input (e.g., B is input in unary or bribes are paid in dollar bills). Otherwise, the algorithm is pseudo-polynomial.

The table entry $D[n,b]$ is the maximum we can make F_n via bribes to voters $v_1, \ldots, v_n$, within budget b. Initialize $D[0,b] = 1$ for all b.

Intuitively, we bring each voter $x \in \{v_1, \ldots, v_n\}$ online one at a time. When we incorporate a new voter v_{x+1} we may or may not require a bribe to this new voter. The maximum probability (of r passing) obtainable either involves a bribe to v_{x+1}, and whatever bribes are needed with the remaining money, or does not require a bribe to v_{x+1}. This gives us the dynamic programming update.

A bribe to voter v_{x+1} (to vote for r) changes p_{x+1} to some new value p'_{x+1}. We can compute F_{x+1} for the given bribe by $F_{x+1} = F_x \cdot p'_{x+1}$.

Let $h = \max\{D[x, b-d] \cdot p'_{x+1}\}$ where d is the cost of increasing p_{x+1} to p'_{x+1}. We set $D[x+1,b] = \max\{D[x,b] \cdot p_{x+1}, h\}$. Since there are at most $k+1$ bribes to consider for each v_x, computing $\{D[x+1,b] : 0 \leq b \leq B\}$ takes $\mathcal{O}(k \cdot n \cdot B)$ operations. This is polynomial in k, n, and B. The algorithm runs in polynomial time with respect to $|B|$ and n.

We can apply this algorithm independently to all m issues. We can do this because we must pass all issues and this dynamic programming algorithm finds the minimum cost for

each referendum. Since spending any less than the minimum on a per issue basis will not achieve The Lobby's agenda we can consider each issue independently. Therefore, for any number of issues, the algorithm runs in $\mathcal{O}(k \cdot n \cdot B \cdot m)$ operations. ❑

If we define PM slightly differently, so that a win happens if the probability of the scenario in which **all** referenda pass is $> t$ (instead of currently defined as each individual referenda is $> t$) we can still use the algorithm in Theorem 3.1.11. We create an equivalent instance with only one issue and add $n \times m$ unique voters. We label the voters as $v_i r_j$ with $1 \leq i \leq n$ and $1 \leq j \leq m$ and combine them all into one overarching issue. The Lobby must pass this overarching issue in order for a win.

Theorem 3.1.12 IB-SM-PLP *and* IB-AM-PLP *are in P.*

Proof. We prove that IB-SM-PLP is in P; the proof for IB-AM-PLP is analogous.

In IB-SM-PLP we are required to influence issues, not individual voters. For each issue r_j, we determine if it will pass by counting the number of voters whose probability is above the threshold t. If this number of voters is $> \lfloor n/2 \rfloor$ then this issue will pass and we do not need to determine any bribery actions.

Otherwise, we must determine the minimum cost to bring r_j to passing (bribing enough voters). For issue r_j that is not currently passing, we count the minimum number s of voters that need to be bribed. We split the voters into two groups: Y are the voters whose probability of voting "yes" is $> t$ and X is the set of voters whose probability of voting "yes" is $\leq t$. We then number the set X from cheapest to most expensive according to how much it would cost to bring the probability that x_i votes "yes" above t.

We then select voter x_s and investigate their bribery price to elevate our referendum to a "yes." Since the bribe will be evenly split across all voters we need to spend n times the cost of bribery for voter x_s in order to have a majority on the issue. We repeat this process for every issue.

After we have computed this value for all issues we compute the total amount needed, and compare to B. If the amount to spend is $\leq B$ then we accept, otherwise we reject. ❑

In order to determine the complexity of variants of the VB method, we need to formally introduce the Optimal Lobbying Problem (OL) from Christian et al. [24]. We state this problem in the standard format for parameterized complexity:

Name: OPTIMAL LOBBYING (OL)

Given: An $m \times n$ matrix E and a $0/1$ vector $\vec{Z}$ of length n. Where each row of E represents a voter, each column represents an issue, and $\vec{Z}$ represents The Lobby's target outcome.

Parameter: A positive integer b (representing the number of voters to be influenced).

Question: Is there a choice of b rows of the matrix (i.e., of b voters) that can be changed such that in each column of the resulting matrix (i.e., for each issue) a majority vote yields the outcome targeted by The Lobby?

Christian et al. [24] proved that this problem is W[2]-complete by a reduction from k-DOMINATING SET to OL (showing the lower bound) and from OL to INDEPENDENT-k-DOMINATING SET (showing the upper bound). In particular, this implies NP-hardness of OL. The following result focuses on the classical complexity of VB-SM-PLP and VB-AM-PLP.

To employ Christian et al.'s W[2]-hardness result [24], we show that OL is a special case of VB-SM-PLP and thus (parameterized) polynomial-time reduces to VB-SM-PLP. This reduction is parameter preserving for k and shows that VB-SM-PLP is W[2]-hard. Since the original reduction for OL was from INDEPENDENT-k-DOMINATING SET to show an upper bound, OL is also NP-hard. It is this NP-hardness that we make use of in these proofs. Analogous arguments apply to VB-AM-PLP.

Theorem 3.1.13 VB-SM-PLP *and* VB-AM-PLP *are NP-complete.*

Proof. Membership in NP is easy to see for both VB-SM-PLP and VB-AM-PLP.

We prove that VB-SM-PLP is NP-hard by reducing OL to VB-SM-PLP. We are given an instance $(E,\vec{Z},b)$ of OL, where E is a $m\times n$ $0/1$ matrix, b is the number of votes to be edited, and $\vec{Z}$ is the agenda for The Lobby. Without loss of generality, we may assume that $\vec{Z}=1^n$ (see Section 3.1.1).

We construct an instance of VB-SM-PLP consisting of the given matrix $P=E$ (a "degenerate" probability matrix with only the probabilities 0 and 1), a corresponding cost matrix C_P, a target vector $\vec{Z}=1^n$, and a budget B. C_P has two columns (i.e., we have $k=0$, since the problem instance is deterministic, see Section 3.1.1), one column for probability 0 and one for probability 1. All entries of C_P are set to unit cost.

The cost of increasing any value in P is n, since bribes are distributed evenly across issues for a given voter. We want to know whether there is a set of bribes of cost at most $b\cdot n=B$ such that The Lobby's agenda passes. This holds if and only if there are b voters that can be bribed so that they vote uniformly according to The Lobby's agenda and that is sufficient to pass all the issues. Thus, the given instance $(E,\vec{Z},b)$ is in OL if and only if the constructed instance $(P,C_P,\vec{Z},B)$ is in VB-SM-PLP, which shows that OL is a polynomial-time recognizable special case of VB-SM-PLP, and thus VB-SM-PLP is NP-hard.

Note that for the construction above it does not matter whether we use the strict-majority criterion (SM) or the average-majority criterion (AM). Since the entries of P are 0 or 1, we have $\overline{p_j}>0.5$ if and only if we have a strict majority of ones in the jth column. Thus, VB-AM-PLP is NP-hard too. ❑

We can also extend this proof for VB with the PM evaluation criteria.

Corollary 3.1.14 VB-PM-PLP *is NP-complete.*

Proof. The proof of Theorem 3.1.13 shows a reduction of OL to VB-AM-PLP-WIW and VB-SM-PLP-WIW. This reduction is independent of the evaluation criterion since

Table 3.2: Complexity results for X-Y PROBABILISTIC LOBBYING PROBLEM WITH ISSUE WEIGHTING, where $X \in \{MB, IB, VB\}$ and $Y \in \{SM, AM, PM\}$

Problem	Classical Complexity	Theorem or Corollary
MB-SM-PLP-WIW	NP-complete	Thm. 3.1.15
MB-AM-PLP-WIW	NP-complete	Thm. 3.1.15
MB-PM-PLP-WIW	NP-complete	Thm. 3.1.15
IB-SM-PLP-WIW	NP-complete	Thm. 3.1.15
IB-AM-PLP-WIW	NP-complete	Thm. 3.1.15
IB-PM-PLP-WIW	NP-complete	Thm. 3.1.15
VB-SM-PLP-WIW	NP-complete	Thm. 3.1.16
VB-AM-PLP-WIW	NP-complete	Thm. 3.1.16
VB-PM-PLP-WIW	NP-complete	Thm. 3.1.16

we create deterministic instances of OL and therefore we can extend it to an instance of VB-PM-PLP.

This shows that VB-PM-PLP is NP-complete. ❑

Probabilistic Lobbying with Issue Weighting

Table 3.2 summarizes our results for X-Y-PLP-WIW, where $X \in \{MB, IB, VB\}$ and $Y \in \{SM, AM, PM\}$. The most interesting observation is that introducing issue weights raises the complexity from P to NP-completeness for all cases of microbribery and issue bribery (though it remains the same for voter bribery). Additional results shown by G. Erdélyi [46] indicate that these NP-complete problems are fixed-parameter tractable and, in some cases, admit a FPTAS.

To begin we first need to introduce the well known NP-complete problem KNAPSACK [65].

Name: KNAPSACK

Given: given a set of objects $U = \{o_1, \ldots, o_n\}$ with weights $w : U \to \mathbb{N}$ and profits $p : U \to \mathbb{N}$, and $W, P \in \mathbb{N}$.

Question: Is there a subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} w(o_i) \leq W$ and $\sum_{i \in I} p(o_i) \geq P$.

Theorem 3.1.15 *The problems* MB-SM-PLP-WIW*,* MB-AM-PLP-WIW*,* MB-PM-PLP-WIW*,* IB-SM-PLP-WIW*,* IB-AM-PLP-WIW*, and* IB-PM-PLP-WIW *are NP-complete.*

Proof. Membership in NP can be seen for each problem X-Y-PLP-WIW, $X \in \{MB, IB\}$ and $Y \in \{SM, AM, PM\}$ through a guess and check algorithm.

To prove that MB-SM-PLP-WIW is NP-hard, we give a reduction from KNAPSACK. Given a KNAPSACK instance (U, w, p, W, P), create a MB-SM-PLP-WIW instance with $k = 0$ and only one voter, v_1, where for each issue, v_1's acceptance probability is either zero or one. For each object $o_j \in U$, create an issue r_j such that the acceptance probability of v_1 is zero. Let the cost of raising this probability on r_j be $c_{1,j}(1) = w(o_j)$ and let the weight of issue r_j be $w_j = p(o_j)$. Let The Lobby's budget be W and its objective value be $V = P$. By construction, there is a subset $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} w(o_i) \leq W$ and $\sum_{i \in I} p(o_i) \geq P$ if and only if there is a subset $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} c_{1,i} \leq W$ and $\sum_{i \in I} w_i \geq V$.

As the reduction introduces only one voter, there is no difference between the bribery methods MB and IB, and no difference either between the evaluation criteria SM, AM, and PM. Hence, the above reduction works for all six problems. ❑

Turning to voter bribery with issue weighting an immediate consequence of Theorem 3.1.13 is that VB-SM-PLP-WIW, VB-AM-PLP-WIW, and VB-PM-PLP-WIW are NP-hard, since they are generalizations of VB-SM-PLP, VB-AM-PLP, and VB-PM-PLP-WIW. Again, membership in NP can also be seen for the issue weighted problems

Corollary 3.1.16 VB-SM-PLP-WIW*,* VB-AM-PLP-WIW*, and* VB-PM-PLP-WIW *are NP-complete.*

3.1.7 Observations

We have studied four lobbying scenarios in a probabilistic setting, both with and without issue weights. Among these, we identified problems that can be solved in polynomial time and problems that are NP-complete. For the case of weighted issues we find that all problem variants are, in fact, strongly NP-hard; their hardness does not depend on whether the numbers in their inputs are encoded in unary or in binary. In some cases not discussed in this presentation of the results, we find problems that are fixed-parameter tractable problems, and problems that are hard (namely, W[2]-complete or W[2]-hard) in terms of their parameterized complexity with suitable parameters. The additional results also investigate the approximability of hard probabilistic lobbying problems (without issue weights) and obtain both approximation and inapproximability results. A complete treatment of these results can be found in the paper by G. Erdélyi et al. [46].

As a general statement, this section shows the addition of uncertainty into the reasoning process increases the computational complexity. While this is not true in all cases, a direct comparison is difficult to classify. Unweighted, deterministic bribery is computationally tractable, while almost all weighted variants are computationally hard [52]. We obtain mixed results in this section. This can be attributed, in large part, to the particular modeling choices. We have provided a mix of models in an attempt to identify where, exactly, the problem becomes hard. We continue this quest for a clear dividing line in the next section.

3.2 Sports Tournaments and Ranking Problems

Sports competitions are common forms of entertainment and recreation around the world. In most sports contests both observers and players have some notion of which competitors are favored over others. Many individuals, including some players, wager vast sums of money on the outcomes of particular games and tournaments. A quick Google search reveals dozens of players, coaches, referees, and judges convicted of manipulating the outcome of sports competitions through match fixing, point shaving, and outright cheat-

ing. Additionally many websites (such as `www.kenpom.com`) produce and publish in depth statistics for not only overall team win/loss predications, but also predictions for individual player stats on a per game basis. It is a world of probabilities and manipulation.

We use sports tournaments as a motivating example of other domains in which bribery [52] and coalitional manipulation [33] can undermine the integrity of competition. Tournaments and single winner elections, when the set of candidates and the set of voters are equivalent, are used in many domains including self-organization of ad-hoc wireless sensor networks, where leaders are elected to delegate work or act as central routing nodes [121], and multi-criteria decision making, where page rankings are sometimes determined by links from the set of pages under consideration [16]. In addition to these important applications of tournaments, there has been recent empirical research in political science and sociology revealing that, in the United States, voter preferences in political elections can be significantly affected by apparently irrelevant events, specifically sports tournaments [73].

In this section we study three different types of sports tournaments.

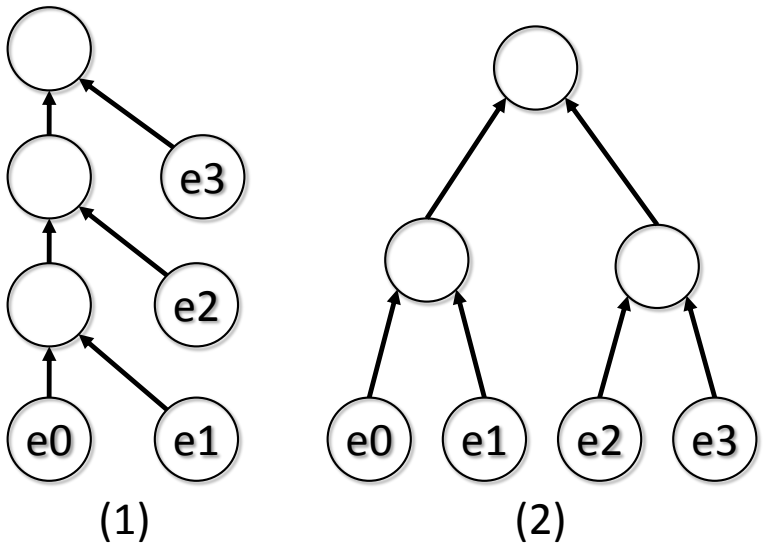
Cup Tournament: A single-elimination competition (or knockout tournament [128]) over a complete binary tree where each entrant[5] plays a sequence of matches head-to-head; the winner is the entrant that is left undefeated. The United States' men's and women's NCAA Basketball Tournaments and most tennis majors fall into this category.

Round Robin tournament: A competition where each entrant competes against every other entrant and earns a points for each victory; the winner is the entrant with the most points. The group play round of the FIFA World Cup falls into this category.

Challenge or Caterpillar Tournament: A series of matches where the winner of each match plays the next entrant in increasing order of rank; the winner is the entrant

[5]We use the term entrant in this section because we can imagine a tournament made up of individuals or teams.

Figure 3.1: Example of (1) A challenge tournament and (2) a cup tournament. The winner is the entrant who reaches the top node.



who wins the final match. Boxing titles and some PBA bowling competitions use this type of tournament.

Figure 3.2 illustrates the difference between cup and challenge tournaments. These types of sporting events correspond to the voting rules: cup for cup tournaments, Copeland for round robin tournaments, and linear balloting for challenge tournaments. We refer the reader to Section 2.2.1 or to Arrow et al. [3] for a more complete treatment of voting rules. In tournaments and sporting events there are several natural questions which arise, such as: "What are the odds my preferred entrant wins?" and "Does my preferred entrant have a chance of winning?"

The classical notation of manipulation, introduced by Bartholdi et al. [7], has been extensively studied in the deterministic case. Conitzer et al. [33] studied some manipulation problems in stochastic settings, however, many of their NP-completeness results break down in the setting under study here because the reductions require the ability to add voters that are not in the candidate set. There are also results relating to manipulation under deterministic information for cup [33] and Copeland [55]. Likewise, the bribery problem

for elections, introduced by Faliszewski et al. [52], has been well studied for voting rules under deterministic information. Again, many of these results do not transfer; the hardness reductions require the ability to introduce an unrestricted number of non-candidate voters.

There has been significant work on the schedule control problem for cup or knock-out tournaments: in the deterministic case by Lang et al. [81] (for the similar problem of sequential majority voting), Vu et al. [128], and Williams [132]; in the stochastic case by Hazon et al. [72]. In addition to the work on the control problem, there is work from Russell and Walsh about the complexity of coalitional manipulation in sports tournaments [113]. In Russell and Walsh's model, entrants may be a part of a coalition of manipulators that can choose to lose their matches. There is also an extensive body of work on the elimination problem for sports tournaments. In the elimination problem, the probability that team i beats team j is known, and the problem is to find the probability that a team is eliminated given a sports season or playoff schedule [68, 76].

There has been surprisingly little work on stochastic models of tournaments; perhaps the closest notion is that of a *possible winner* — a notion intrinsic to reasoning under uncertainty introduced by Konczak and Lang [78], with further study by Lang et al. [81], and others. Recent studies of the complexity of computing possible winners include those by Lang et al. [81], Faliszewski et al. [4], and Xia et al. [139]. These papers address complexity questions when voters are defined by their (possibly incomplete) preference profiles over a set of outcomes, but not probabilities of winning.

In Section 3.2.1 we formally describe our model of reasoning about bribery in tournaments where entrants have probabilities of winning and losing. Section 3.2.2 formally states the six decision problems we study in this domain for all three tournament types. Section 3.2.3 details our complexity results and ends with a few observations about the change in reasoning complexity when models move from deterministic tournaments to models of uncertain information.

3.2.1 Model Definition

We begin by defining a model with which to reason about sports tournaments and other head-to-head competitions. The details of the model closely follow the model proposed in Section 3.1.1 for plurality elections [45].

Consider a tournament with n entrants $\{e_1, \ldots, e_n\}$. Note that what distinguishes a tournament from a voting rule here is that, for a given election with separate sets of candidates and voters, we require the set of candidates to be exactly the set of voters. Let $\mathbb{Q}_{[0,1]}^{n \times n}$ be the set of $n \times n$ matrices over $[0,1] \cap \mathbb{Q}$. Let $p_{i,j}$ denote the probability that entrant i will defeat entrant j. Let $P = [p_{i,j}] \in \mathbb{Q}_{[0,1]}^{n \times n}$. We require that $p_{i,j} + p_{j,i} = 1$. We choose $k \in \mathbb{N}_+$ to discretize the interval [0,1] (we call this the *discretization level* of the problem). That is, we let each $p_{i,j}$ be in $\{0, 1/(k+1), 2/(k+1), \ldots, k/(k+1), 1\}; i, j \in \{1, \ldots, n\}$. Taking $k = 0$ gives us the deterministic case where each game is either won or lost with certainty.

We define C_P as the **discrete price table**. C_P has n^2 rows (indexed by pairs i, j) and $k+2$ columns. For each entry in the table we have a positive integer ($\mathbb{Z}^+$) value representing the cost to lower $p_{i,j}$ to a given probability. We assume that all entrants will compete to the best of their abilities and we cannot increase an entrant's probability of winning a particular match. Therefore we designate these entries as $--$. We also require that $c_{i,j}(p_{i,j}) = 0$. That is, it does not cost anything to have an entrant compete at its highest level. The entries for $c_{i,j}$ must also be monotone decreasing; payment of $c_{i,j}(l)$ changes the probability that i beats j to $l/(k+1)$. Bribery must be performed in one fell swoop, before the first game is played. Outside actors may not have access to the entrants after the start of the competition (i.e., the tournament is taking place in a remote or tightly controlled location) so all bribery must be done beforehand.

We are also given a threshold t and an integer-valued budget B. For cup and challenge tournaments we are also given a tree T with entrants labeled on the leaves to represent the order of the matches. Let $Pr[e_i|P,T]$ denote the probability that e_i wins the tournament defined by schedule T and probability matrix P.

Example 3.2.1 *Consider the following example with $k = 4$ (so there are $k+1 = 5$ possible probability configurations) and $n = 4$ entrants. We use this example setup as a running example for the rest of the chapter. These parameters for k and n are given implicitly by the probability matrix P and cost matrix C_P.*

$P =$

	e_1	e_2	e_3	e_4
e_1	$--$	0.75	0.50	0.25
e_2	0.25	$--$	0.25	0.25
e_3	0.50	0.75	$--$	0.25
e_4	0.75	0.75	0.75	$--$

$C_P =$

$c_{i,j}$	0.00	0.25	0.50	0.75	1.00
$c_{1,2}$	100	40	20	0	$--$
$c_{1,3}$	30	25	0	$--$	$--$
$c_{1,4}$	200	0	$--$	$--$	$--$
$c_{2,1}$	10	0	$--$	$--$	$--$
$c_{2,3}$	10	0	$--$	$--$	$--$
$c_{2,4}$	10	0	$--$	$--$	$--$
$c_{3,1}$	120	75	0	$--$	$--$
$c_{3,2}$	200	100	50	0	$--$
$c_{3,4}$	300	0	$--$	$--$	$--$
$c_{4,1}$	300	200	100	0	$--$
$c_{4,2}$	300	200	100	0	$--$
$c_{4,3}$	400	300	200	0	$--$

In this section we focus on the following three types of tournaments. In each case we assume that no game ends in a tie and we have a unique winner for every tournament. In the deterministic case a scoring model which includes ties or does not normalize to certain forms for round robin tournaments and Copeland elections can have significant effects on the complexity of evaluation, manipulation, and bribery [55, 76].

Challenge Tournament: In a challenge tournament the entrants are ordered $e_1, e_2, \ldots, e_n$. In the first match, e_1 plays e_2. In the second match, the winner plays e_3, and so on. The winner of the last match wins the tournament.

Cup: In a cup tournament we are given a complete binary tree T with entrants labeled on the leaves. Each internal node is decided by the competition between the two entrants on the level below. The winner is the entrant who reaches the top node of the tree.

Round Robin: In a round robin tournament (Copeland Scoring) each entrant plays every other entrant exactly once. In each match each entrant receives 1 point for a win and 0 points for a loss. The winner of the tournament is the entrant with the maximum number of points. If more than one entrant has the same maximum score, we employ a lexicographic tie-breaking scheme where e^* always comes first.

3.2.2 The Probabilistic Tournament Bribery Problem

With this model we can now define our problem and a set of related decision problems for $Y \in \{\text{Challenge Tournament (CT), Cup, Round-Robin (RR)}\}$.

Name: Y - PROBABILISTIC TOURNAMENT BRIBERY PROBLEM (Y-PTBP)

Given: A probability matrix $P \in \mathbb{Q}_{[0,1]}^{n \times n}$ describing the entrants e_i in the tournament with some preferred entrant e^* and (where necessary) cost matrix C_P, threshold t, budget B, set of manipulators $M \subseteq E$, and ordering T.

Questions: We define the following 6 decision problems on the above information:

Evaluation: Is the probability that e^* wins Y (the sum of the probabilities of the futures with e^* a winner) above t ?

Pos-Win: Is e^*'s probability of winning the tournament above 0?

Pos-Win-$: Can we raise e^*'s probability of winning Y above 0 by bribing specific entrants according to C_P and not exceeding the budget B?

Constructive Coalitional Manipulation (CCM): Can we raise e^*'s probability of winning Y above t by strategically setting all the probabilities associated with a subset M (a *coalition*) of the entrants.

Constructive Bribery: Can we raise e^*'s probability of winning Y above t by bribing specific entrants according to C_P and not exceeding the budget B?

Exact: Can we raise e^*'s probability of winning the tournament above t by bribing specific entrants according to C_P and spending exactly B?

Example 3.2.2 *Recall our running example matrices, shown below.*

$P =$

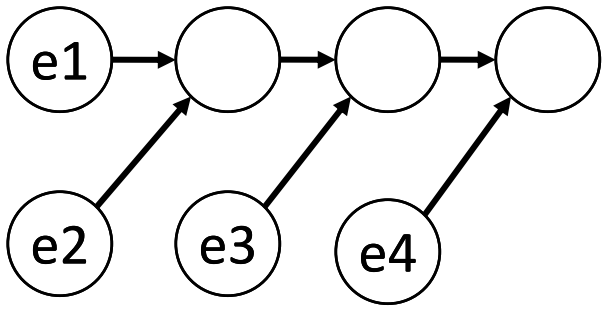
	e_1	e_2	e_3	e_4
e_1	$--$	0.75	0.50	0.25
e_2	0.25	$--$	0.25	0.25
e_3	0.50	0.75	$--$	0.25
e_4	0.75	0.75	0.75	$--$

$C_P =$

$c_{i,j}$	0.00	0.25	0.50	0.75	1.00
$c_{1,2}$	100	40	20	0	$--$
$c_{1,3}$	30	25	0	$--$	$--$
$c_{1,4}$	200	0	$--$	$--$	$--$
$c_{2,1}$	10	0	$--$	$--$	$--$
$c_{2,3}$	10	0	$--$	$--$	$--$
$c_{2,4}$	10	0	$--$	$--$	$--$
$c_{3,1}$	120	75	0	$--$	$--$
$c_{3,2}$	200	100	50	0	$--$
$c_{3,4}$	300	0	$--$	$--$	$--$
$c_{4,1}$	300	200	100	0	$--$
$c_{4,2}$	300	200	100	0	$--$
$c_{4,3}$	400	300	200	0	$--$

Consider a challenge tournament with the entrants ordered on T as illustrated in Figure 3.2.2

Figure 3.2: Tournament graph (T) for Example 3.2.2.



For this graph we can evaluate the six questions presented in above. We assume that $e^* = e^1$ *and we enumerate the other necessary parameters for the questions that require them.*

Evaluation: *For the decision problem we require a threshold, here we take* $t = 0.08$*. For a caterpillar tournament with* e^* *in the first position, computing the exact probability of winning is* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.75 \times 0.5 \times 0.25 = 0.09375$*. Since this is* $> t$ *this example is a "yes" instance.*

Pos-Win: *For the decision problem for pos-win we only need to examine the result of the evaluation procedure. Since* $Pr[e_1|P,T] > 0$ *then this example is a "yes" instance.*

Pos-Win-$: *For our running example* $Pr[e_1|P,T] > 0$ *without requiring any bribes and therefore is also a "yes" instance for pos-win-$.*

Constructive Coalitional Manipulation (CCM): *For this question let* $M = \{e_4\}$ *and* $t = 0.15$*. Since we are trying to raise* $Pr[e_1|P,T] > 0.15$ *the best thing to do is to set* $p_{1,4} = 0$*. This way the manipulator,* e_4*, loses to the preferred entrant with certainty. We can then evaluate* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.75 \times 0.5 \times 1.0 = 0.375$*. Since* $Pr[e_1|P,T] > 0.15$ *this is a "yes" instance.*

Table 3.3: Complexity results for the PROBABILISTIC TOURNAMENT BRIBERY PROBLEM. In some cases we have been unable to provide lower bounds, in these cases we note our upper bound results ($\in$).

	Challenge	Cup	Round Robin
Evaluation	P (Thm. 3.2.3)	P (Thm. 3.2.10)	$\in$ #P (Thm. 3.2.15)
Pos-Win	P (Thm. 3.2.4)	P (Thm. 3.2.11)	P (Cor. 3.2.18)
Pos-Win-$	P (Thm. 3.2.5)	P (Thm. 3.2.12)	P (Thm. 3.2.17)
CCM	P (Thm. 3.2.6)	$\in$ NP (Thm. 3.2.13)	$\in$ NP^{PP} (Thm. 3.2.16)
Cons. Bribery	$\in$ NP (Thm. 3.2.8)	$\in$ NP (Thm. 3.2.13)	$\in$ NP^{PP} (Thm. 3.2.16)
Exact	NP-C (Thm. 3.2.9)	NP-C (Thm. 3.2.14)	$\in$ NP^{PP} (Thm. 3.2.16)

Constructive Bribery: *For this question let* $t = 0.15$ *and* $B = 100$*. By investigation we see there are only two meaningful options for bribes. We can bribe* e_4 *so that* $p_{1,4} = 0.5$ *or we can bribe* e_2 *and* e_3 *so that* $p_{1,2} = 1.0$ *and* $p_{1,3} = 0.75$*. Using the evaluation formula again we see that with the first option of bribes we have* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.75 \times 0.5 \times 0.50 = 0.1875$ *with the second option of bribes we have* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 1.0 \times 0.75 \times 0.25 = 0.1875$*. Both of these options give us the same value for* $Pr[e_1|P,T]$ *and, since both options are* > 0.15*, this is a "yes" instance.*

Exact: *For this question let* $t = 0.15$ *and* $B = 100$*. By investigation we see there are only two meaningful options for bribes and only one option that exactly spends our budget. Therefore, we bribe* e_4 *so that* $p_{1,4} = 0.5$*. Using the evaluation formula again we see that* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.75 \times 0.5 \times 0.50 = 0.1875$*. Since* $Pr[e_1|P,T] > 0.15$ *and we have spent exactly our budget this is a "yes" instance.*

3.2.3 Results

We proceed through the results in order of tournament type. This allows us to present the results in a coherent fashion as, in many cases, the proofs and algorithms build on one another.

Challenge Tournament

We begin by investigating the evaluation problem for challenge tournaments. It is important to establish that we have a polynomial time winner determination scheme, otherwise, the bribery results would, by definition, be computationally hard.

Theorem 3.2.3 *Evaluation for* CT-PTBP *is in P.*

Proof. We provide a polynomial time dynamic programming algorithm to compute the winning probability of any entrant in an instance of CT-PTBP. Let the entrants be $\{e_1, \ldots, e_n\}$. Suppose $e^* = e_k$ and $1 \leq k \leq n$. Let $z(i, j)$ denote the probability that e_i wins match j.

Let $z(0,0) = 1$.
for $i = 1$ to k **do**
  $u = max(1, i-1)$ {The first match e_i plays.}
  $z(i,u) = \sum_{t=1}^{i-1} z(t, u-1) p_{i,t}$
  for $j = max(1, i-1)$ to $n-1$ **do**
    $z(i,j) = z(i, j-1) p_{i,j}$
  end for
end for

At the termination of the algorithm, the probability that e_i wins is equal to $z(i, n-1)$. ❑

Now that we have determined that it is computationally easy to evaluate the result of a given instance of CT-PTBP, we move our attention to the possible winner problem.

Theorem 3.2.4 *Pos-Win for* CT-PTBP *is in P.*

Proof. Run the evaluation algorithm given in Theorem 3.2.3 with $t = 0$. Note that e^* is a possible winner if and only if $Pr[e^*|P,T] > 0$. ❑

Theorem 3.2.5 *Pos-Win-$ for* CT-PTBP *is in P.*

Proof. We provide a polynomial time dynamic programming algorithm to compute the minimum cost such that e^* has a non-zero probability of winning the tournament.

In order for entrant e^* to have a non-zero probability of winning it must be able to beat at least one possible entrant in each of e^*'s matches in the tournament graph T. We compute the minimum cost to ensure that each entrant has a non-zero probability of reaching level L in the graph.

Given P and C_P we can compute the **minimum cost matrix**, $min\$$, an integer matrix of size n^2, as follows: $min\$_{ij} = 0$ if i starts with a non-zero probability to beat j and the minimum bribe cost $min\$_{ij} = c_{ij}(1/(k+1))$ otherwise. Let L be a level in the tournament graph T starting with 0 at the bottom level. We then create a vector for each entrant, V_{e_i}, of size n with all entries initialized to 0 (the cost to get to the 0th level). Let $V_{e_i}(L)$ be the minimum cost for e_i to be a winner at level L. Without loss of generality we assume that the entrants are numbered such that e_1 and e_2 compete in the first match with the winner competing against e_3 in the second match.

for $L = 1$ to $n-1$ **do**
  Let $G = e_1, ..., e_{L+1}$.
  for $e \in G$ **do**
    if $e = e_{L+1}$ **then**
      $V_e(L) = \min_{x \in G, x \neq e}(min\$_{j,x} + V_x(L-1))$
    end if
    if $e = e_j, j \leq L$ **then**
      $V_e(L) = min\$_{j,L+1} + V_e(L-1)$
    end if
  end for
end for

At the end of execution V_{e_i} will contain the minimum cost to promote entrant i to the top level of T with a non-zero winning probability. We can compare this cost with B and either accept if $V_{e^*}(n-1) \leq B$ else reject. ❑

Recall that in the coalitional manipulation problem we are given a set $M \subseteq E$ of members of the manipulating coalition. Observe that the coalitional manipulation problem is a special case of the bribery problem in our model. Specifically, in a coalitional manipulation instance, $B = |M|$ and the cost of bribing members of M is also 1 (unit priced). The entrants $E \setminus M$ have bribery costs equal to ∞ (so they cannot be changed in this scenario).

Theorem 3.2.6 *Constructive Coalitional Manipulation for* CT-PTBP *is in P.*

Proof. Since there are no budget-related resource bounds in this problem we can assume our problem is to maximize e^*'s probability of winning by setting the entries for each $m \in M$ in the probability matrix P. We refer to the setting of M as a strategy. In this instance, C_P gives the available options of probability values for each $m \in M$ even though there are only unit prices.

There are $|M| \cdot (n - |M|)$ potential contests between coalition and non-coalition members with 2 possible strategies for each, and there are $|M| \cdot \frac{|M|-1}{2}$ contests between coalition members, with 3 possible strategies for each (a loses, b loses, both try to win), for a total of $2^{|M| \cdot (n-|M|)} 3^{|M| \cdot \frac{|M|-1}{2}}$ strategies.

Any manipulators who enter the tournament after e^* need to deterministically lose to e^* in order for e^* to win the championship. Therefore, for the rest of the proof, we focus on strategies for manipulators who enter the tournament before e^*. We construct a strategy in a step by step fashion starting with the first game that e_{n-1} participates in. We proceed in reverse match order, one game at a time, setting the strategies for each $m \in M$ against each entrant as we go.

Step 0: For each $m \in M$, if m reaches e^* then m chooses to lose. This sets the strategy against e^*.

Repeat for: $j = 1$ to $n-2$

Step i: For each $m \in M$ that enters before e_{n-j}, suppose m reaches e_{n-j}. Set m's strategy against e_{n-j} to maximize the probability that e^* wins by evaluating the 2 or 3 possibilities. This is possible because strategies for all contests above e_{n-j} have been set. Also, if $e_{n-j} \in M$, then for each $e_i \notin M$, $i < n-j$, assuming e_i reaches e_{n-j}, pick e_{n-j}'s strategy against e_i to maximize e^*'s probability of winning.

After Step $j = n-2$ all strategies have been set for all M. Whatever the strategy is at lower levels, $Pr(e^* \text{ wins })$ is maximized by maximizing

$$q_{ij} = Pr(e^* \text{ wins } | e_i \text{ reaches } e_{n-j}). \tag{3.1}$$

Since the events "e_i reaches e_{n-j}" for different i are disjoint, the q_{ij} can be maximized independently.

We do 2 or 3 evaluations for each pair of contestants. For each we require a call to the $\mathcal{O}(n^2)$ evaluation procedure described in Theorem 3.2.3. Therefore we compute an optimal manipulation strategy in $\mathcal{O}(n^4)$. ❑

Theorem 3.2.3 gives us membership in NP for the constructive bribery problem since we can verify a bribery plan in polynomial time. Theorem 3.2.6 shows that in a situation of CCM we can achieve optimal manipulations in polynomial time. However, our results for CCM do not extend in a straightforward manner to the situation of priced bribery.

The simple answer would be to apply a *bang-for-the-buck* greedy algorithm to iteratively choose the cheapest available single bribe which results in the largest change in $Pr[e^*|P,T]/\text{COST OF BRIBE}$. However, this will not reach an optimal solution in all cases. This can be illustrated by an example of CT-PTBP. Consider the following example over four entrants.

Example 3.2.7 *Assume that $e^* = e_1$ and the tournament graph, T, is the same as shown in Figure 3.2.2 (where e_1 plays every match). Assume we have $k = 4$ and P and C_P given below with $B = 20$. For simplification, we omit the bribery cost for all matches that do not contain e^*.*

$P =$

	e_1	e_2	e_3	e_4
e_1	$--$	0.25	0.25	0.25

$C_P =$

$c_{i,j}$	0.00	0.25	0.50	0.75	1.00
$c_{2,1}$	13	13	13	0	$--$
$c_{3,1}$	25	10	10	0	$--$
$c_{4,1}$	25	10	10	0	$--$

We know that when e^* is involved in every match, the winning probability is given by: $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.25 \times 0.25 \times 0.25 = 0.015625$. We set the cheapest bribe of entrant e_2 to change $p_{1,2} = 1.0$ with cost 15. This would give $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 1.0 \times 0.25 \times 0.25 = 0.0625$, giving a change in probability of 0.046875 for a cost of 13 or a bang-for-the-buck ratio of 0.0036058. Looking at either a bribe of e_3 or e_4 we would change $p_{1,3} = 0.75$ with a cost of 10. This would give $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.25 \times 0.75 \times 0.25 = 0.046875$, giving a change in probability of 0.03125 for a cost of 10 or a bang-for-the-buck ratio of 0.003125. This would mean that according to the bang-for-the-buck heuristic, bribing e_2 is best single action.

However, looking at our budget we see that we can afford to bribe both $p_{1,3} = p_{1,4} = 0.75$ for a combined cost of 20. This would give $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.25 \times 0.75 \times 0.75 = 0.140625$. This is, by far, the best action and shows that a simple single action greedy algorithm would make non-optimal decisions in this case.

In order to eliminate as many complications of the tournament graph as possible, we turn our attention to the simplest case of a challenge tournament. This is a version where e^* is potentially involved in every game. In this version, CHALLENGE-THE-CHAMP-PTBP, e^* takes on all challengers and is a common sub-problem for both the general CT-PTBP and CUP-PTBP problems.

Theorem 3.2.8 *There is a Constructive Bribery algorithm for* CHALLENGE-THE-CHAMP-PTBP *that runs in polynomial time with respect to B and n.*

Proof. The Challenge-the-Champ problem assumes that $e^* = e_1$ and is in the first match. Our goal is to maximize the probability that e^* will win the tournament within budget through bribes to each of the other players in the tournament. The idea of this proof is to consider bribes on only a subset of the entrants. At each step we add an entrant $x \in \{e_2, \ldots, e_n\}$ and show a dynamic programming update that will compute the cheapest bribe for the new subset of entrants until we have considered the set of all the entrants.

In an instance of Challenge-the-Champ we compute the probability that e^* will win the first x games with $F_x = \Pi_{j=1}^{x} p_{1,j}$. Intuitively F_x tells us the probability that e^* wins the tournament considering only x of the n games. Without loss of generality, we assume that B is bounded by the sum of all possible bribes.

Our goal is to maximize F_x (within budget). We do this using dynamic programming to build a $(n+1) \times B$ table D. Filling in each entry in the table takes polynomial time, so the problem is in P if the budget, B, is polynomial in the size of the input (e.g., B is input in unary). Otherwise, the algorithm is pseudo-polynomial.

The table entry $D[n,b]$ is the maximum we can make F_n via bribes to entrants $e_1, \ldots, e_n$, within budget b. Initialize $D[0,b] = 1$ for all b.

Intuitively, we add games to the set one at time by considering each new entrant $x \in \{e_2, \ldots, x_n\}$. When we incorporate the new e_{x+1} we may or may not require a bribe to this new entrant. The maximum probability (of e^* winning) obtainable either involves a bribe to e_{x+1}, and whatever bribes are needed with the remaining money, or does not require a bribe to e_{x+1}. This gives us the dynamic programming update.

A bribe to entrant e_{x+1} (to lose to e^*) changes $p_{1,x+1}$ to some new value $p'_{1,x+1}$. We can compute F_{x+1} for the given bribe by $F_{x+1} = F_i \cdot p'_{1,x+1}$.

Let $h = \max\{D[x, b-d] \cdot p'_{1,x+1}\}$ where d is the cost of increasing $p_{1,x+1}$ to $p'_{1,x+1}$. We set $D[x+1,b] = \max\{D[i,b] \cdot p_{1,x+1}, h\}$. Since there are at most $k+1$ bribes for each e_i,

computing $\{D[i+1,b] : 0 \leq b \leq B\}$ takes $\mathcal{O}(k \cdot n \cdot B)$ operations. This is polynomial in k, n, and B. The algorithm runs in polynomial time with respect to B and n. ❑

While we have been unable to show NP-hardness for any version of the constructive bribery problem, Theorem 3.2.8, with the insights listed in this section, leads to the conjecture that the problem is hard. Many well known NP-hard problem like KNAPSACK admit pseudo-polynomial time dynamic programming algorithms [65].

If we require our outside manipulator to spend exactly the allocated budget then we can show NP-completeness for challenge tournaments. Situations which require an organization to spend a budget exactly occur implicitly in many large organizations and governments. For resources other than money that could be used, such as referees in sports tournaments or canvassers in political elections, there sometimes the requirement of exact allocations.

We must introduce the SUBSET SUM PROBLEM for our next proof. The statement of the problem is from Garey and Johnson's book [65].

Name: SUBSET SUM

Given: A set $W \in \mathbb{Z}_+$, $w_1, \ldots, w_m$, and a target $S \in \mathbb{Z}_+$.

Question: Does there exist a subset $I \subseteq \{1, \ldots, m\}$ such that $\sum_{i \in I} w_i = S$?

This problem was shown to be NP-complete via a transformation from PARTITION by Karp [75]. Using this problem we can exactly classify the complexity of the exact bribery problem for CT-PTBP.

Theorem 3.2.9 *Exact Bribery for* CT-PTBP *is NP-complete.*

Proof. Membership is an immediate consequence of Theorem 3.2.3 and a guess and check algorithm.

To show NP-hardness we provide a reduction from SUBSET SUM: For a given SUBSET SUM instance we set up a challenge tournament with S players such that e^* always wins (i.e., e^* has a 100% chance of winning all games against all players) and we let $t = 0$, $B = S$, and $k = 0$. We then create the bribery cost matrix C_P with prices equal to the weights of $w_1, \ldots, w_m$. In this situation bribery will have no effect and we must distribute the money exactly. We have now established a polynomial time mapping such that there will be an exact bribery if and only if there is a subset such that $\sum_{i \in I} w_i = S$. ❑

Cup Rule

We observe that many of the results for CUP-PTBP use reasoning similar to that used for CT-PTBP. We assume that each CUP-PTBP instance is a complete tree. Note that this does not limit our results; we can always pad the tree with entrants who lose to all other entrants and cannot be bribed.

Theorem 3.2.10 *Evaluation for* CUP-PTBP *is in P.*

Proof. This is a proof by construction of a polynomial time algorithm that computes $Pr[e^*|P,T]$ for all entrants. We construct a table of size $n \cdot \lceil \log_2(n) \rceil$ and use dynamic programming to compute for each $i : L_{r,i}$ where $L_{r,i}$ is the probability that entrant i advances to round r (with $r = 1$ being the first match e_i competes in). When we get to the last round, our table will contain the probability that each entrant won.

Let $G(r,i)$ be the entrants that e_i may face in round r.
for $i = 1$ to n **do**
  $L_{1,i} = 1$
end for
for $r = 2$ to $\lceil \log_2(n) \rceil$ **do**
  for $i = 1$ to n **do**
    $L_{r+1,i} = L_{r,i} \cdot \sum_{x \in G(r,i)} p_{i,x} \cdot L_{r,x}$

end for

end for

At each stage, r, the numerators and denominators of the probabilities are $\mathcal{O}(k^r) \subset \mathcal{O}(k^n)$, having sizes $n\log(k)$ (where k is the discretization level). Each e_i potentially competes against each x in exactly one round, so there are exactly $n(n-1) \in \mathcal{O}(n^2)$ multiplications in total so the total time is $O(n^2(n\log(k))^2) = O(n^4\log(k)^2)$. ❑

Turning again to the possible winner problem variants we see that evaluating whether a entrant has a chance of winning is easy for cup tournaments as well.

Theorem 3.2.11 *Pos-Win for* CUP-PTBP *is in P.*

Proof. Run the evaluation algorithm given in Theorem 3.2.10 with $t = 0$. ❑

Theorem 3.2.12 *Pos-Win-$ for* CUP-PTBP *is in P.*

Proof. We provide a polynomial time dynamic programming algorithm to compute the minimum cost such that e^* has a non-zero probability of winning the tournament.

Let the tournament graph T be a complete binary tree. Let L be the level of the binary tree starting with 0 at the bottom level. We then create a vector for each entrant, V_{e_i}, of size n with all entries initialized to 0 (the cost to get to the $0th$ level). Let $V_{e_i}(L)$ be the minimum cost for e_i to have a non-zero probability of winning at level L. We construct the **minimum cost matrix**, $min\$$, an integer matrix of size n^2, as follows: $min\$_{ij} = 0$ if i starts with a non-zero probability to beat j and the minimum bribe cost $min\$_{ij} = c_{ij}(1/(k+1))$ otherwise.

for $L = 1$ to $\log(n)$ **do**

for all $e_i \in T$ **do**

Let $G(e_i, L)$ be the set of entrants in the sub-tree at level L containing e_i.

$V_{e_i}(L) = V_{e_i}(L-1) + \min_{x \in G(e_i,L) - G(e_i,L-1)}(min\$_{i,x} + V_x(L-1))$

end for

end for

At the end of execution $V_{e_i}(\log_2(n))$ will contain the minimum cost to promote e_i to the top level of T with a non-zero winning probability. We can compare this cost with B and either accept if $V_{e^*}(\log_2 n) \leq B$ or reject. ❑

Theorem 3.2.10 immediately provides us with the following corollaries through standard guess and check algorithms.

Corollary 3.2.13 *Constructive Coalitional Manipulation and Constructive Bribery are in NP for* CUP-PTBP.

We have not been able to show lower bounds results for CCM or constructive bribery for CUP-PTBP. The result does not hold for cup tournaments because the winner within separate sub-trees can affect the winning probability of all entrants in a neighboring sub-tree. We can, however, straightforwardly transfer our results for the exact case.

Theorem 3.2.14 *Exact Bribery for* CUP-PTBP *is NP-complete.*

Proof. To show membership in NP we can easily check whether we have spent exactly our budget. We can nondeterministically guess a bribery scheme that utilizes the entire budget and check it using the algorithm developed in the proof of Theorem 3.2.10. Thus we have a polynomial time algorithm which requires a nondeterministic guess to come up with an exact bribery scheme.

To show NP-hardness we can use a similar reduction as the one used in Theorem 3.2.9. Recall that S is the target sum in a SUBSET SUM instance. In this construction we choose a number, j, such that $j^2 \geq |S|$. We build our cup instance with j players. We set the added players' bribery prices to be $> B$ so that these players can be safely ignored. We then use the same mapping presented in the proof of Theorem 3.2.9 to show Exact Bribery is NP-complete. ❑

Round Robin

Round Robin tournaments are of particular interest in the social choice community because of their close correspondence to the Copeland election system. Faliszewski et al. [55] provided a comprehensive study of bribery and manipulation of Copeland elections under deterministic scenarios.

In addition, many papers on sports elimination, including those by Gusfield and Martel [68] and Kerns and Paulusma [76], study sports round robin tournaments for particular competitions (FIFA, Major League Baseball etc.). Our model is more general: we do not restrict our model to deterministic settings or specific tournament formats.

We strongly believe the evaluation problem for round robin tournaments to be #P-hard though we have been unable to prove this statement. Hazon et al. showed that the problem is #P-hard for the voting system Copeland with an imperfect information model [71]. However, their results do not transfer in a straightforward manner, as their reduction requires the ability to add voters who are not candidates. Likewise Gusfield and Martel showed a similar problem of determining the probability that a team is eliminated from a round robin sports competition is #P-hard [68]. Their result does not hold in our context as their proof requires the ability to restrict the round robin schedule to a subset of the total number of games played.

Theorem 3.2.15 *Evaluation for* RR-PTBP *is in #P.*

Proof. Suppose we're given an evaluation problem with probabilities expressed with precision d. Each probability $p_{i,j}$ can be written as $n_{i,j}/d$, where $n_{i,j}$ is an integer.

We create a nondeterministic polynomial time algorithm, M, as follows:

- For each match $[i, j]$
  - M guesses $n \in [0, d)$

- If $n < p_{i,j}$ then i wins and increments a counter for i, otherwise it increments a counter for j.

- If e^*'s counter has the highest value, accept; otherwise, reject.

- M accepts this tournament if and only if e^* wins on $\geq 1/2$ of the computations. Note that the value of 1/2 can be replaced by t with no loss of generality.

The number of accepting computations divided by the number of computations is the probability that e^* wins the tournament. ❑

The upper bound on evaluation complexity given in Theorem 3.2.15 can be extended immediately to the bribery, CCM, and exact bribery problems as well.

Theorem 3.2.16 *CCM, Constructive Bribery, and Exact Bribery for* RR-PTBP *are in* NP^{PP}.

Proof. Given a tournament T and threshold t, the NP^{PP} machine nondeterministically guesses a set of bribes and evaluate with a single call to a #P oracle (equivalently, a PP oracle). We accept if the probability of e^* winning is $> t$. Note that we have actually shown this is in $NP^{\#P[1]}$. ❑

Both variants of the possible winner problem for round robin tournaments have polynomial time algorithms. While at first glance this may seem odd, it is because answering the question of possible winner does not require enumerating all possible outcomes of the tournament. Instead, to answer the possible winner problem we only need to find one configuration of the matches in which e^* wins.

The case of possible winner with access to bribes is somewhat more complicated for round robin tournaments. We begin with this problem because, as we show, the possible winner problem without access to bribes can leverage the same algorithmic construction as the possible winner with access to bribes problem. Recall the MINIMUM COST FEASIBLE FLOW problem introduced in Section 2.1.2 [1].

Name: MINIMUM COST FEASIBLE FLOW

Given: A graph $G(V,E)$ with vertices V, edges E, source node s, and sink node t. Each $e \in E$ has flow, capacity, and cost. An edge labeled "$[x,y],c$" has capacity y and requires flow of at least x with cost per unit c. Cost is accumulated on a per unit flow basis (i.e., if $x = 2$ and $c = 3$ then the total cost of the edge is 6).

Question: Does there exist a flow from s to t such that the flow into each node is the same as the flow out, all minimum edge flows are satisfied, no maximum edge flows are violated, and cost is minimal?

There is a polynomial time algorithm for this problem and its solution (when the constraints are integers) is integral [1].

Using minimum cost feasible flows, Russell and Walsh provided an algorithm to compute minimal constructive manipulations (but not bribery) for deterministic round robin tournaments when the number of games that e^* can win is fixed [113]. The algorithm presented by Russell and Walsh did not allow for bribery (we do not bound the number of manipulable games) and assumed that the number of games that e^* could win is fixed. Likewise, Faliszewski et al. showed manipulation results for Copeland under deterministic information elections using feasible flows [55] and a similar construction was used by Faliszewski for the nonuniform bribery case (bribery where each voter may have different prices for changing their votes) [51]. While our construction is similar to theirs (and to an earlier constructions from Gusfield and Martel [68] and Ford and Fulkerson [64]) there are significant differences between the constructions. Our results are for a more flexible model, which is able to encapsulate bribery and manipulation, and works for even or odd numbers of entrants in the tournament.

Theorem 3.2.17 *Pos-Win-$ for* RR-PTBP *is in P.*

Proof. We give a polynomial time reduction of an instance of Pos-Win-$ to a minimal cost feasible flow problem. Our reduction constructs a set of minimum cost feasible flow instances; the minimum cost bribery corresponds to the cost of the flow in one of the polynomially many graphs we construct. There is a polynomial time algorithm for finding the solution to a minimum cost feasible flow problem [1]. We build this sequence of graphs so that, in each graph, if there is a way to bribe the entrants such that e^* can be made a possible winner, the cost of the bribery scheme will be equal to the cost of the minimal cost flow in that graph. We build a graph for each possible winning score e^* could have. There will be at most $n/2$ graphs built in this proof and therefore we can answer Pos-Win-$ for RR-PTBP in time polynomial in the size of the input.

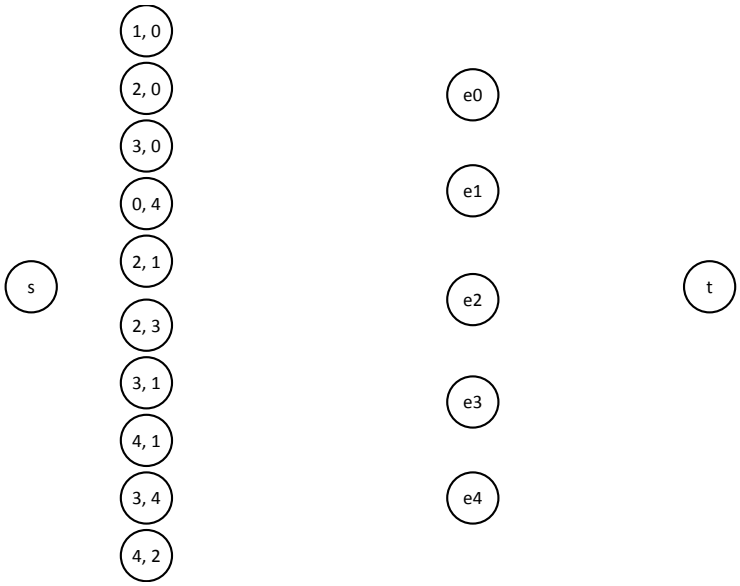
We first construct the *minimum cost matrix*, $min\$_{i,j}$, as in the proof of Theorem 3.2.5. We also construct a domination graph D, which is a complete directed graph that has all entrants as nodes and each directed edge goes from a winning entrant to a losing entrant. We construct D by placing a directed edge from all entrants that are more likely to win to those less likely to win. While constructing D, we set the deterministic outcome of all games with 0 bribery cost involving e^* such that e^* is a winner (maximizing its possible wins). Once we set all the nondetermined games in favor of e^* we let $t = outDeg(e^*)$. In this instance t represents the total number of games that e^* can possibly win without making any bribes. In order for e^* to be a possible winner, we need to bribe between 0 and $n-t-1$ entrants to lose to e^* (as well as possibly influencing the outcomes of other games).

For each i from 0 to $n-t$ we construct a feasible flow graph that tells us whether e^* can be made a possible winner by bribing exactly i games involving e^*. We iterate over these graphs until we find a situation where e^* is a possible winner. Intuitively, the graph selects the cheapest bribes in order to make e^* a winner. Each unit of flow in the graph is thought of as a point for a winning entrant. The costs of edges correspond to minimum bribe costs in order to "fix" deterministic games. We arrange the flow network such that, if there is a

minimum cost feasible flow, then e^* can be made a winner for cost equal to the cost of the flow.

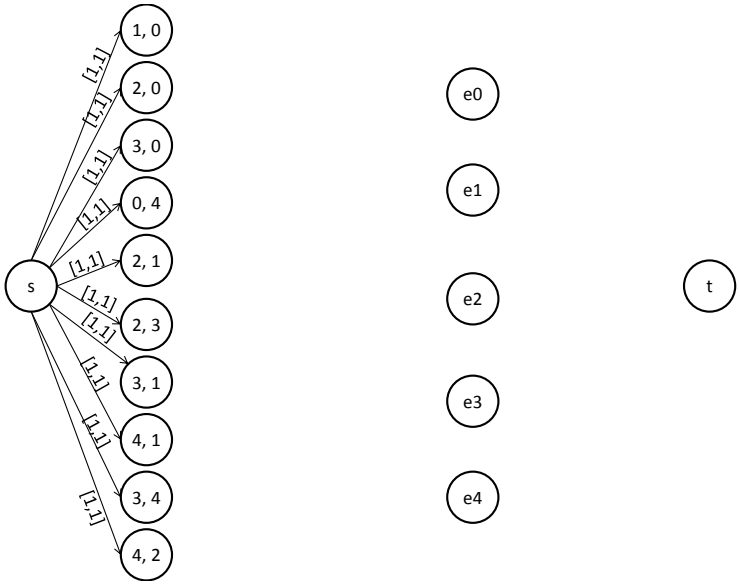
We detail the construction of the graph in three groups of edges between two interior stages. Figure 3.2.3 illustrates an example of a completely constructed graph and we proceed through a description of the construction illustrating a scenario with five entrants. The nodes other than the source and sink are classified as either left nodes or right nodes. The left half is the set of nodes that connect from the source. The right half is a set of nodes that are connected to the sink. The left and right nodes are connected by the interior edges of the graph.

Figure 3.3: Step 1 of the construction of a minimal cost winner determination graph. We have a source, sink, game nodes for each possible game, and collector nodes for each participating entrant.



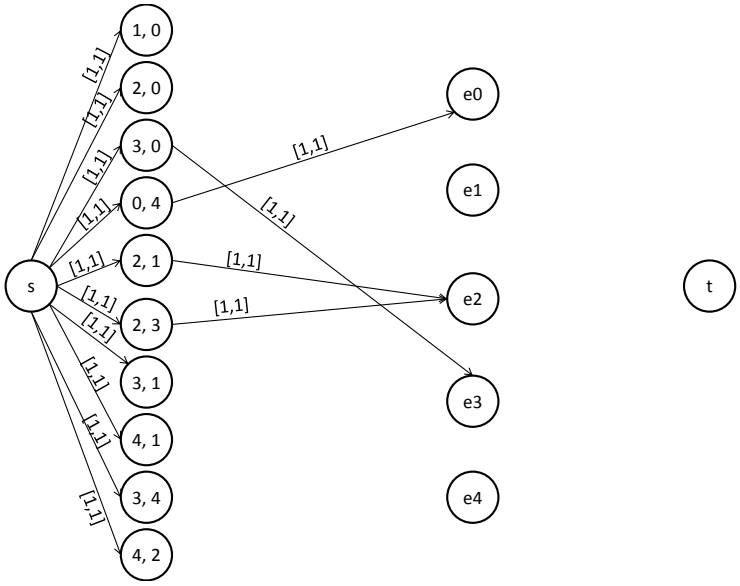
In order to build the graph we begin with a bipartite graph (not including the source and sink). The nodes on the left hand side of the graph represent all the games in the tournament. The nodes on the right hand side of the graph represent all the entrants to the tournament. Each unit of flow in this graph can be thought of, intuitively, as a "win" flowing to a particular entrant. We begin by building a source node s, a sink node t, a node for each game that will be played, and a collection node for each entrant in the tournament. The game nodes in Figure 3.2.3 are labeled with the numbers of the entrants participating in a particular game while the collector nodes are labeled with the individual entrants.

Figure 3.4: Step 2 of the construction of a minimal cost winner determination graph. We build edges from the source to all game nodes with 1 unit of flow.



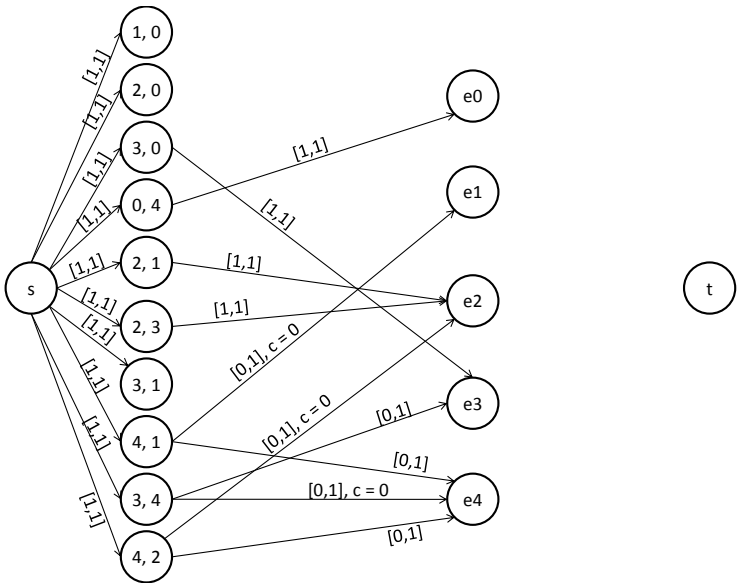
To build the edges we first set one edge from the source to each of the nodes on the left hand side with capacity and flow of 1 as illustrated in Figure 3.2.3. These units of flow will eventually count how many games a particular entrant will win. Since each game is potentially worth one point we have exactly one unit of flow into each game node.

Figure 3.5: Step 3 of the construction of a minimal cost winner determination graph. We build edges from all game nodes to their sure-to-win entrants.



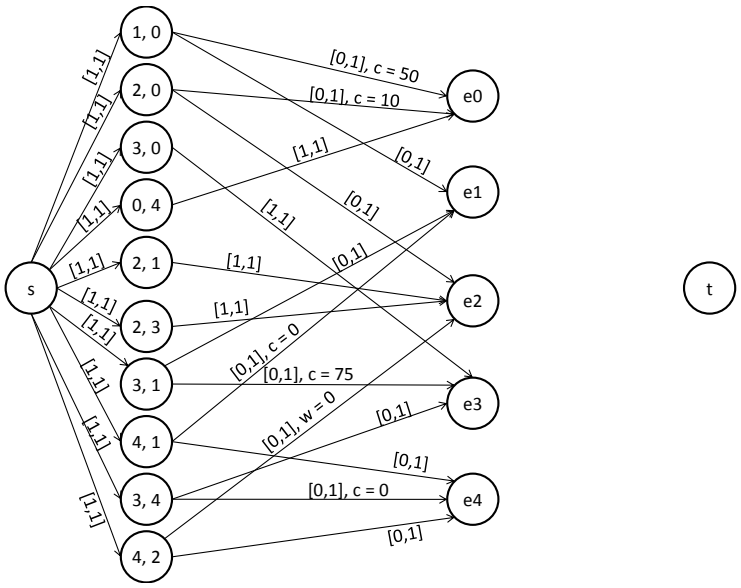
The internal edges (edges from matches to entrants) are constructed depending on the prices and probabilities of the individual games. If one entrant is sure to beat another entrant and we cannot afford any bribes with respect to these games, we build an edge $[1,1], c = 0$ from the game node to the sure-to-win entrant. Figure 3.2.3 illustrates these edges in our example graph.

Figure 3.6: Step 4 of the construction of a minimal cost winner determination graph. We build two edges in cases where either entrant is a possible winner.



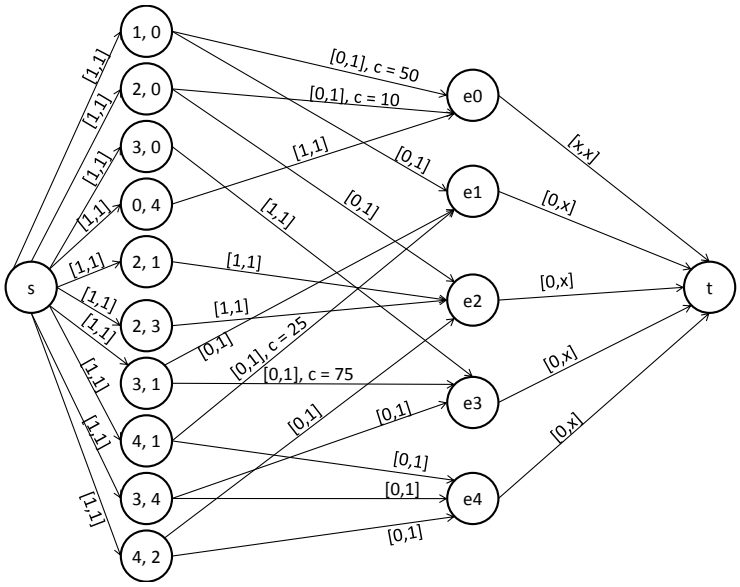
If both entrants have a nonzero probability of winning we build an edge $[0,1], c = 0$ to both entrant nodes. Since we are only attempting to find a configuration that would allow e^* to win, this construction allows us to check both possible game outcomes. This step is illustrated in Figure 3.2.3.

Figure 3.7: Step 5 of the construction of a minimal cost winner determination graph. We encode the cost of minimum bribes to change deterministic game outcomes.



The final case of internal edges is if one entrant is sure to beat another and we can possibly afford the bribe. In this case we build an edge $[0,1], c = 0$ from the game node to the sure-to-win entrant and an edge $[0,1], c = min\$_{i,j}$ from the game node to the node of the entrant who can be made a possible winner if the bribe is made. This construction allows our flow network to change the outcome of a deterministic game with cost equal to the amount of the minimum bribe. This step is illustrated in Figure 3.2.3.

Figure 3.8: A complete example of a minimal cost winner determination flow network.



We then build the final set of edges from entrant nodes to sink with capacity x, which we change between iterations of the graph. Recall that t is the total number of wins that e^* can achieve without resorting to bribery. We set $x = t + i$ and we iterate i from 0 to $n - t$ and build a graph as described above for each i. Here, x represents the most games that e^* can win making i bribes to entrants involved in games with e^*. We find the minimum cost feasible flow for each combination of bribes that e^* can make to achieve a given score. A complete graph construction is showing Figure 3.2.3.

We check each graph in turn for $i = 0, \ldots, n - t$ and if we attain a feasible flow with cost $\leq B$ we accept, otherwise, reject.

This method is complete and will find the minimum cost bribery to give e^* a chance to win. Assume that there is a cheaper flow then the one found by our network. This would mean that there existed a less expensive way to grant e^* a number of wins greater than or equal to any other entrant. Since each entrants win at most n games in any particular graph, and we check all n possible scores for e^* this cannot happen. Therefore, this construction will find the minimal cost bribery construction.

We build, at most, n networks with $\mathcal{O}(n^2)$ nodes and edges. Since the minimum cost

Table 3.4: Complexity results for deterministic tournaments. The cup and round robin results are from Russell and Walsh [113].

	Challenge	Cup	Round-Robin
Evaluation	P	P	P
CCM	P	P	P

feasible flow problem can be solved in polynomial time, the overall running time of our algorithm is polynomial. Therefore pos-win-$ for RR-PTBP is in P. ❑

We can straightforwardly extend the proof of Theorem 3.2.17 to the case where we do not have access to our budget. In the Pos-Win problem we cannot actually bribe anyone, we just have to choose winners of the non-deterministic games in a way that allows e^* a way to win. The most direct way to show this is to apply the algorithm presented in Theorem 3.2.17 with $B = 0$. We can optimize this construction somewhat but it is sufficient to state the following corollary.

Corollary 3.2.18 *Pos-Win for* RR-PTBP *is in P.*

3.2.4 Observations

Table 3.3 provides an overview of our complexity results on tournaments. Russell and Walsh [113] provided a thorough complexity analysis of similar problems in deterministic cases, all of which generate polynomial time solvable problems. Table 3.4 reveals that, in general, polynomial time evaluation procedures lead to polynomial time manipulation procedures (for deterministic systems).[6]

If we can compute a constructive manipulation in the deterministic case then we can determine a possible winner in the stochastic case. The problem of a possible winner with uncertain information is the same as constructive bribery in the deterministic setting with $B = \infty$. We don't yet have hardness proofs for some of the problems here. We believe that the problems are, in fact, complete for the classes we mention. We observe that in

[6]This is also observed in [33].

some cases the introduction of uncertainty has no effect on membership in these complexity classes while in others it represents a significant change in reasoning structure and complexity.

3.3 Summary

In this chapter we have studied the complexity of bribery and manipulation problems in elections with multiple referenda and sports tournaments, in which the actors seeking to affect the outcomes have access to only probabilities of how agents will vote or teams will perform. This chapter represents a significant departure from other work in the field of computational social choice which, generally, only considers deterministic information versions of the same problems. Many of the problems studied become computationally harder under the uncertain information setting. This increase in complexity is not uniform across problem types and, in many cases, the change in complexity seems to be closely tied to modeling choices of the particular setting.

In this chapter voters expressed their preferences over the issues and each issue was treated as an independent item. In the next chapter we investigate the situation where agents have structured preferences over the decisions to be made. This is an important step in understanding how the type and amount of information outside actors have access to affects the security of voting systems.

Chapter 4 Bribery and Manipulation in Combinatorial Domains

This chapter details work on the computational complexity of finding optimal bribery schemes in voting domains where the candidate set is the Cartesian product of a set of variables and their domains, and agents' preferences are represented as CP-nets. This work includes model building; extending and defining aggregation methods and bribery methods for these models; and complexity results of reasoning in these domains. We find that this domain structure, which may lead to an exponential number of candidates in the size of the input, causes many existing computational results for bribery to break down. We provide new algorithms and complexity results which show that, in most cases, bribery in combinatorial domains is easy. A summary of our results is shown in Table 4.1 and Table 4.2. This also holds for some cases of k-approval, where bribery is difficult in traditional domains. Portions of this work have been previously published as both an extended abstract [88] and as an invited special session paper [89].

4.1 Voting in Combinatorial Domains

It is often natural to express group decision problems as the combination of a sequence of decisions. This method is used in many settings, from the United States Congress (specifically, votes for amendments to a bill) to a group of friends deciding what appetizer, main course, and wine should be served for a group meal [18, 82]. In all these cases, agents express preferences and vote on parts of the overall decision to be taken. Moreover, agents may have dependent preferences: the choice of wine may depend on the choice of main course for the meal. We consider a scenario where agents use the CP-net formalism, which allows agents to compactly represent their preferences over many issues that may have conditional dependencies [15, 90].

As we have seen, bribery and manipulation directly compromise the security and ro-

bustness of an election system. These questions are ways that agents (outside or inside an election) can affect the election's outcome [7,52]. In particular, the bribery problem regards scenarios in which an outside agent with a limited budget attempts to affect the outcome of an election by paying some of the agents to change their preferences so that a particular candidate p wins the election. This problem was introduced to computational social choice by Faliszewski et al. [52] and has become a metric by which the security of election rules is judged [39, 51, 85]: if it is computationally difficult to find an optimal bribery in elections decided by a certain voting rule, we can say that the rule is resistant to bribery.

Manipulation occurs when one or more voters attempt to vote strategically in order to affect the result of the election [7,33]. Recall the **Constructive Coalitional Manipulation (CCM)** problem [33]: given a set of agents X with fixed votes, a set Y of agents with unfixed votes, and a candidate p, is there is an assignment of votes to the agents of Y such that running the election on the complete set of votes (profiles) $X \cup Y$ results in p winning the election. Certain forms of the manipulation problem can therefore be seen as bribery problems where the agents which can modify their vote do so for a unit cost, those that cannot modify their vote have a bribery price of ∞, and the budget is equal to the size of the manipulator set. While every reasonable voting rule is manipulable [3], a rule may be said to be resistant to manipulation if it is computationally difficult to decide how to manipulate [7]. The complexity of manipulation was first studied by Bartholdi et al. [7] with further study by Conitzer et al. [33] and is another metric for election security. These problems have been studied extensively in computational social choice, however, the literature is sparse when elections have combinatorially structured domains.

The setting we study in this chapter is novel for several reasons. We allow sets of candidates which are the Cartesian product of the domains of a set of possible decisions. This raises several interesting questions as to what it means to compute a winner and what bribery means in these combinatorial domains. The use of structured, combinatorial preferences leads to instances where some voters, depending on their preference dependencies,

cannot be bribed to vote for any arbitrary candidate. This restriction, along with the potentially exponential number of candidates, breaks some of the existing algorithms for bribery and manipulation.

When voting is structured as the combination of several decisions, there are two natural methods to employ when selecting a winner. We investigate multiple individual rules within two overarching winner determination strategies.

Sequential Methods: A sequential method aggregates each agents' preference issue-by-issue. This is akin to how amendments to a bill are passed in the US Congress [18]. The final decision is the sum total of all the sequential decisions. In this chapter we consider a sequential majority (SM) rule.

One-step Methods: A one-step method aggregates the agents' votes over the set of all combinations of value assignments to all issues. This would be the method where a group of friends votes on a complete meal for some shared group meal [82]. With a one-step method we can incorporate several individual voting rules. In this section we consider one-step plurality (OP), one-step veto (OV), and one-step k-approval (OK).

Unlike previous studies on bribery, CP-nets do not necessarily lend themselves to the convention that the cost of bribery is either a fixed constant for all voters, a fixed constant on a per-voter basis, or related to the number of swaps in the candidate ordering [43, 52]. In fact, we do not always want to work on the outcome ordering, since it is exponentially large, but on the CP-net. Since a single flip in a CP-net preference statement (cp-statement) can lead to a large change in the outcome ordering, we consider four different bribery cost schemes.

C_{EQUAL}**:** Any amount of change in a CP-net is a unit cost.

C_{FLIP}**:** The cost of bribery is the number of variable flips in the CP-net.

C_{LEVEL}: The cost is the number of variable flips weighted by variable position within the CP-net.

C_{ANY}: The cost is the number of variable flips weighted by a specific cost per variable.

In the rest of this section we provide an overview of structured preference models, specifically CP-nets, for compactly expressing preference in combinatorial domains and their use in voting procedures. In Section 4.2 we precisely define our domain and formulate our three bribery actions and four cost schemes that are used in combinatorial bribery. Section 4.3 gives the bribery and manipulation problems within our model. Section 4.4 gives our results about the complexity of computing optimal bribery and manipulation schemes in our models. Our results section is broken up according to voting rule and aggregation method in order to provide a cohesive narrative. Section 4.4.4 expands our model to encompass domains where variables are non-binary and provides results about the composition of multiple voting rules. We end in Section 4.5 which details the changes in complexity from traditional bribery domains to those when agents' preferences are represented as CP-nets.

4.1.1 Structured Preferences

Combinatorially structured election domains are used in many places including the United States House of Representatives [18]. This election structure considers several possibly independent variables of some whole described by the assignment of each of the variables to some value. The canonical example in the ComSoc community is choosing the appetizer, main course, desert, and wine for a meal. Each course has several alternatives and the combination of all possible choices for all possible courses creates the set of alternatives (or candidates). This sequential framework creates a possibly exponential number of complete configurations of meals in the description of the problem instance.

Sequential elections were first introduced and studied in computational social choice by Lang [79]. Lang provided a comprehensive axiomatic description of several voting rules

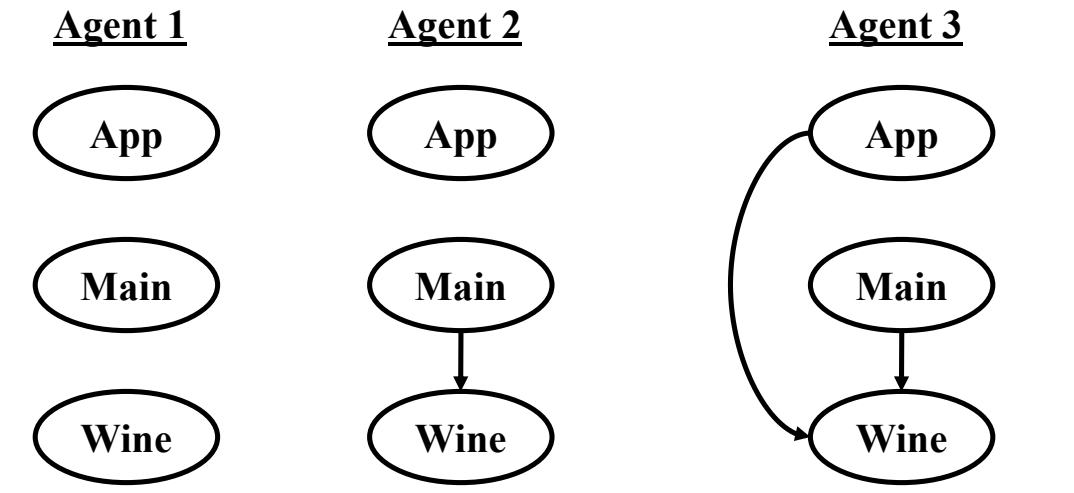
when extended to combinatorial domains when agents' preferences are expressed as CP-nets [79]. In later work, Lang and Xia considered the properties of several election rules and how these properties change when rules are used as part of a sequential election domain [82]. Additional work by Xia et al. investigates the properties of sequential election rules when cyclic preferences are allowed [137] and axiomatic characterizations of strategy-proof sequential voting rules [136].

While bribery has not been specifically investigated in any domain with sequential elections, Conitzer et al. investigated the control problem in sequential voting domains [29] when agents' preferences are recorded with a network structure similar to CP-nets. While Xia et al. have investigated the possibility of strategic voting by a single voter in sequential voting when agents' preferences are recorded as CB-nets [138]. Dalla Pozza et al. investigate sequential voting domains when agents' preferences are represented as soft constraints [37].

In our model of elections under uncertainty in Section 3.1, observe that a complete agenda with multiple referenda is similar to the combinatorial domains we study in this chapter. However, in the uncertain election model, all issues are independent whereas in the combinatorial domains in this chapter allow for issues to have conditional dependencies upon one another. In some cases one model can be described with the other (deterministic instances of the uncertain information model are equivalent to the combinatorial model with all independent variables). This subcase, however, does not take advantage of the fidelity of either model. We think of them as models designed to investigate different types and amounts of information that may be available to an outside agent.

A major challenge for computational social choice is describing agent preferences in domains where there is a large number of candidates. Artificial Intelligence provides several formalisms to do this, including CP-nets [15], the one we consider in this work. CP-nets are a graphical model for compactly representing conditional and qualitative preference relations. CP-nets are sets of *ceteris paribus* preference statements (cp-statements).

Figure 4.1: An example of a CP-net with three agents expressing O-legal profiles over three binary variables.



App	Soup > Salad
Main	Meat > Fish
Wine	Red > White

App	Salad > Soup
Main	Fish > Meat
Wine	Fish: White > Red Meat: Red > White

App	Salad > Soup
Main	Fish > Meat
Wine	Salad, Fish: White > Red Salad, Meat: Red > White Soup, Meat: Red > White Soup, Fish: Red > White

For instance, the cp-statement "I prefer red wine to white wine if meat is served." asserts that, given two meals that differ only in the kind of wine served and both containing meat, the meal with red wine is preferable to the meal with white wine.

Formally, a CP-net has a set of issues or variables $m = \{x_1, \ldots, x_n\}$, and each issue has a finite domain $D(x_1), \ldots, D(x_n)$. Each issue x_i has a set of parent issues $Pa(x_i)$ whose assignment can affect the ordering over the values of x_i. This set of dependencies defines a graph where each x_i has $Pa(x_i)$ as its immediate predecessors. In general, CP-nets allow for cyclic dependencies within the implied graph. However, in this work we only allow acyclic dependency graphs. Given the implied graph structure an agent specifies a total order (strict linear order) over the values of each x_i for each complete assignment of the variables in $Pa(x_i)$. These orders are referred to as cp-statements. A CP-net is **compact** if the maximum number of parents of a feature is bounded from above by a constant. This formalism allows for a superpolynomial number of outcome assignments in the description

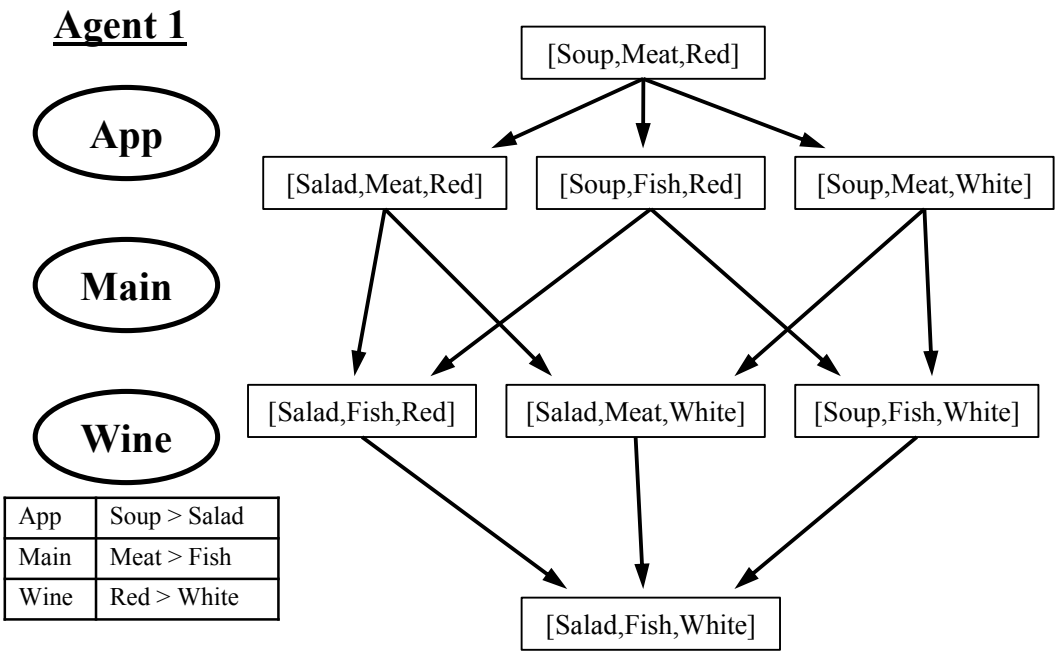
size of the CP-net. If we allowed for an unbounded number of parents for each variable, since we have to enumerate all linear orders for all combinations of parent variables, our tables would be exponential in size.

Figure 4.1.1 shows an example of a CP-net and corresponding cp-statements. In this graph we have three issues $m = \{App, Main, Wine\}$. The domains for the variables are $D(App) = \{Soup, Salad\}$, $D(Main) = \{Fish, Meat\}$, and $D(Wine) = \{Red, White\}$. We have three different agents and each agent has its own set of dependency relations within the preference graph. Agent 1 has no preferential dependencies and therefore none of his variables have parent variables ($Pa(App) = \emptyset$ for Agent 1). Agent 2's preferences on wine are dependent on his assignment for main course. The encoding of Agent 2's preferences make the statement that, "I prefer red wine to white if meat is served." So Agent 2, given two meals that differ only in the kind of wine, the meal with red wine is more preferred. Agent 3 has a more complex set of dependencies in his graph. Recall that, for each variable, a complete assignment must be provided for all assignments of the parent variables. This is why Agent 3's preference table is so much more complex. Since the *Wine* issue has two parents, Agent 3 must specify a wine preference for all possible combinations of App and $Main$.

In general we consider issues with only binary domains in this chapter. For a given issue A, we denote A's binary domain as a and $\overline{a}$ with an individual cp-statement of $a > \overline{a}$. Given a set of issues A, B where the assignment of B is dependent on the assignment of A, we write our cp-statements as: $a > \overline{a}$, $a : \overline{b} > b$, and $\overline{a} : b > \overline{b}$. These statements imply the dependency graph and explicitly state that $A = a$ is unconditionally preferred to $A = \overline{a}$ and that $B = \overline{b}$ is preferred when $A = a$. A complete assignment to all issues is called an **outcome** or a **candidate**. When examining binary domains, CP-nets allow us to compactly express preference over an outcome space of cardinality exactly 2^m, where m is the number of issues.

In general, finding the optimal (most preferred) outcome of a CP-net is NP-hard [15].

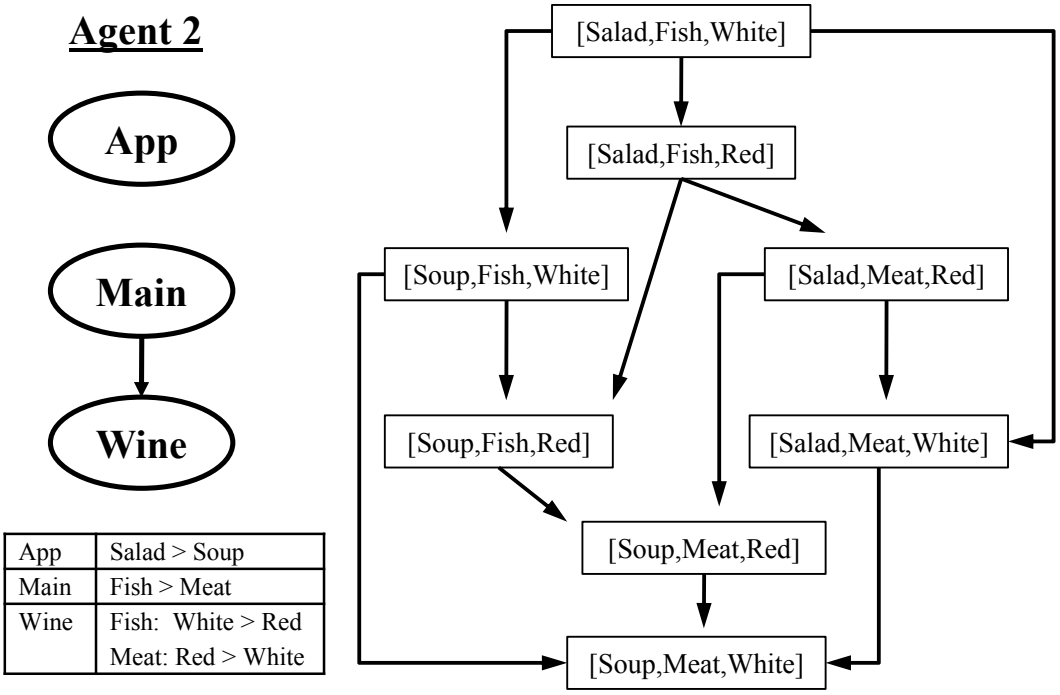
Figure 4.2: An example of a CP-net with the corresponding graph representing the partial order between all possible outcomes.



However, in acyclic CP-nets, there is only one optimal outcome and this outcome can be found in linear time by sweeping through the CP-net in topological order and assigning the most preferred values according to the cp-statements. Consider Agent 3 in the example illustrated in Figure 4.1.1. Agent 3 would choose $App = Salad$ and $Main = Fish$. After these two are set we investigate the cp-statement for the last variable and select the appropriate value $Wine = White$ to find his top outcome.

Given an outcome for an agent, a **worsening flip** is a change in the value of an issue to a less preferred value according to the relevant cp-statement for that issue. In general, one outcome α is better than another outcome β ($\alpha \succ \beta$) if and only if there is a chain of worsening flips from α to β. This definition induces a preorder (reflexive and transitive relation) over the outcomes, which is a partial order (reflexive, transitive, and anti-symmetric relation) if the dependency graph of the CP-net is acyclic. Figure 4.1.1 illustrates the partial order over all the outcomes defined by the CP-net of Agent 1. The top outcome according

Figure 4.3: An example of a CP-net with the corresponding graph representing the partial order between all possible outcomes. Ties are broken with independent variables being considered more important than dependent variables.



to the CP-net is $\{Soup, Meat, Red\}$. The arrows from this outcome represents all the possible worsening flips from the top outcome. Traversing any of the edges from one outcome to another down the induced graph of assignments means we are moving from a better to a worse outcome. Note that, since there are no dependencies and we have not defined any relationship between the independent variables, the nodes of the graph at each level are incomparable (occupy the same position in the partial order).

In an acyclic CP-net it is also computationally easy, given an outcome, to find the next best outcome in a linearization of the induced outcome ordering [17]. In a CP-net, all outcomes may not be ordered since the CP-net only implies a partial order and so some assignments may be incomparable. In some cases, we may require a total order over the outcome ordering and in these cases we enforce some kind of linearization as is standard [20, 34]. If finding the next best outcome according to the linearization scheme is computationally

easy then it is also easy to find the top k outcomes when k is polynomially bounded.

A standard way to define a linearization is to provide a total order over the variables and then to linearize incomparable elements using a lexicographical approach based on this total order and on the conditional preferences. We illustrate this in Figure 4.1.1 where we show Agent 2's CP-net and the induced graph according to worsening flips. The graph also shows the partial order when we consider independent variables more important than dependent variables. We have broken the tie between $\{Salad, Fish, Red\}$ and $\{Soup, Fish, White\}$ using this rule. Each of these assignments contains one variable with a less preferred assignment. The reason that $\{Salad, Fish, Red\}$ precedes $\{Soup, Fish, White\}$ in the partial order is because $\{Salad, Fish, Red\}$ has a less preferred assignment in a dependent variable (less important). If we were to linearize the order in Figure 4.1.1 into a strict linear order using a lexicographic linearization method by variable name we get: $\{Salad, Fish, White\} > \{Salad, Fish, Red\} > \{Salad, Meat, Red\} > \{Soup, Fish, White\} > \{Salad, Meat, White\} > \{Soup, Fish, Red\} > \{Soup, Meat, Red\} > \{Soup, Meat, White\}$.

Given a set of issues, domains, and some preferred outcome p, there is always a CP-net that has p as its optimal outcome [15]. However, given issues, domains, and an outcome ordering $p_1, \ldots, p_n$, there may be no CP-net whose induced outcome ordering coincides with the given ordering. In fact, CP-nets cannot model all outcome orderings. For example, two outcomes differing for the value of one issue must necessarily be ordered in the preorder induced by a CP-net. So, if we are given an ordering where two outcomes that differ on a single variable that is incomparable, there is no CP-net that can induce this ordering. Although CP-nets have a limited expressiveness, they also present inherent advantages because of their qualitative nature, especially in the context of preference elicitation.

4.1.2 Winner Determination and Voting with CP-nets

Voting theory provides a myriad voting rules to aggregate agents' preferences. We survey some of these methods in Section 2.2.1. Recall that each rule takes, as input, a partial or complete preference ordering of the agents and gives, as output, a winner (an outcome that is best according to the rule). If there are only two candidates, the best rule, according to many criteria, is majority voting [91]. When there are more than two candidates, there are many voting rules one could use (Plurality, Borda, STV, approval, etc.), each with its advantages and drawbacks.

In this chapter we consider the following voting rules, modified slightly to work when agents' express their preferences as CP-nets. Each candidate when voting in combinatorial domains is a complete assignment to all variables within the domain.

Plurality: The candidate ranked first receives a point. The candidate(s) with the most points win the election. When there are two candidates, plurality coincides with majority.

Veto: Each voter chooses a candidate to veto (disapprove) with all other candidates receiving a point. The candidate(s) with the most points wins the election.

k-approval: Each voter approves of k out of m candidates ($k \leq m$) and disapproves of the remaining candidates. All approved candidates receive one point. The candidate(s) with the most points win the election.

The study of voting theory, as discussed in Section 2.2.1, provides an axiomatic characterization of voting rules in terms of desirable properties such as: the absence of a dictator, unanimity, anonymity, neutrality, monotonicity, independence of irrelevant alternatives, and resistance to manipulation and bribery. In this chapter we focus on combinatorial voting domains and their resistance to bribery.

The model we consider in this chapter has a n agents with preferences over a common set of candidates. The candidate set has a combinatorial structure: there is a common set of

m binary issues and the set of candidates is the Cartesian product of the variable domains. Each candidate (or outcome) is an assignment of values to all issues, thus we have 2^m candidates.

Each agent expresses its preferences over the candidates via a compact acyclic CP-net. Moreover, when we use a sequential approach to voting we require additional restrictions on the CP-nets. The CP-nets for sequential methods must be compatible with one another: there is a total ordering O over the issues such that, in each CP-net, each issue must be independent of all issues following it in the ordering O and no dependency graph may contain cycles.

A **profile** (P, O) is a collection $P_1, \ldots, P_n$ of n CP-nets over m common issues and a total ordering O over the issues that satisfies the above property. This is called an O-legal profile and was introduced by Lang [79]. Notice that the CP-nets appearing in such profiles do not necessarily have the same dependency graphs. This is illustrated in Figure 4.1.1, our running example. Each of the three agents has the same ordering over the issues $O = \{App, Main, Wine\}$ but each agent has a different dependency graph. Moreover, each of the agents' CP-nets combine to form an O-legal profile: the CP-nets are acyclic, follow an ordering, and are compact.

It is worthwhile to clarify our notion of an agent: an agent is only a CP-net handler. He has the ability to specify a CP-net over a given set of binary issues and to compute the optimal outcome, the top k outcomes, and the worst outcome of his CP-net. He does not handle the explicit partial order over all the outcomes of the CP-net, but just the CP-net, which is a compact representation of the ordering.

To determine the winning outcome, we consider two different approaches: sequential and one-step rules. Within these, we define four distinct voting rules when selecting a winning outcome. Notice that the O-legality condition is not required for the one-step approaches; in these settings, the CP-nets in a profile can have very different dependency structures. Tie-breaking, as discussed in Section 2.2.1 is a complex issue and we discuss it

for our domain when we formally define our problem in Section 4.3.

Sequential Majority (SM)

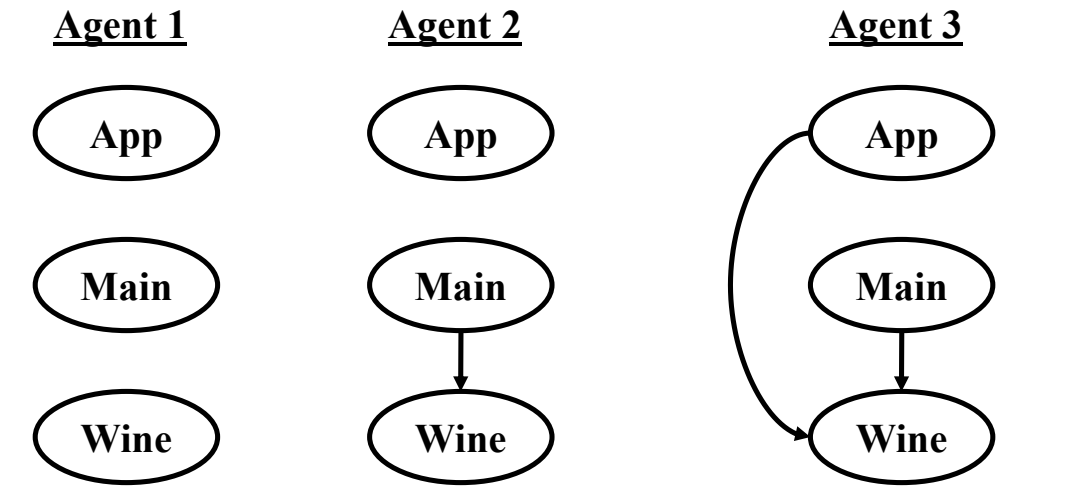
Sequential Majority describes Lang's sequential procedure [79] instantiated on binary issues. For SM we consider only O-legal profiles. For each issue in the ordering O, we select a winning assignment using majority (since issues are binary). The value chosen for an issue is returned to all agents who then fix the issue to that value in their respective CP-nets. When voting has completed on all issues, the winning outcome is defined by the issue instantiations chosen by majority at each of the voting steps.

Example 4.1.1 *Consider our running example of three agents illustrated in Figure 4.1.1. Since we have three issues we will hold three separate sub-elections, all decided by majority. Since* $O = \{App, Main, Wine\}$ *for this example, we first take a majority vote for* App *and Salad is preferred by a vote of two to one. We then return this value to all agents and have them fix this value in their CP-net. Moving to the next issue we have that* $Fish$ *is preferred to* $Meat$ *by a two to one majority. The partial assignment for all agents is now* $\{Salad, Fish\}$*. This has caused two of Agent 1's preferences to be reversed but we do not need to update his CP-net as it contains no dependencies. For* $Wine$ *we see that Agent 2's preferences are* $White > Red$ *given our assignment to* $Fish$*. Agent 3's preference is for* $White$ *as well and now we have a majority for* $White$ *of two to one. The winning outcome for this example with the SM procedure is* $\{Salad, Fish, White\}$.

One-step Plurality (OP)

In OP we ask each agent to provide their optimal outcome according to their CP-net. We then use the plurality rule to evaluate the winning outcome. Note that for this method (and all the other one-step methods) we do not require the O-legality constraint, merely that agents be able to compute their top outcome.

Figure 4.4: An example of a "reversed" CP-net with three agents expressing O-legal profiles over three binary variables with all cp-statements reversed.



App	Salad > Soup
Main	Fish > Meat
Wine	White > Red

App	Soup > Salad
Main	Meat > Fish
Wine	Fish: Red > White Meat: White > Red

App	Soup > Salad
Main	Meat > Fish
Wine	Salad, Fish: Red > White Salad, Meat: White > Red Soup, Meat: White > Red Soup, Fish: White > Red

Example 4.1.2 *Consider our running example of three agents illustrated in Figure 4.1.1. By investigation we see that Agent 1 will vote* $\{Soup, Meat, Red\}$*, Agent 2 will vote* $\{Salad, Fish, White\}$*, and Agent 3 will vote* $\{Salad, Fish, White\}$*. Collecting these votes under the plurality rule the winning outcome with two votes is* $\{Salad, Fish, White\}$*.*

One-step Veto (OV)

In OV we ask each agent to provide their worst outcome according to their CP-net. Each agent can determine their worst outcome by flipping all of their individual cp-statements, the reversed cp-statements for our running example are illustrated in Figure 4.1.2. For this method there may be many candidates in the winner set because the number of alternatives may be exponential in the number of voters. However, since we are using this as a social choice procedure, we only need to select one alternative from the winning set, which we can do in polynomial time.

Example 4.1.3 *Consider our running example of three agents illustrated in Figure 4.1.1. First each agent must reverse all the cp-statements in their CP-nets. This will leave the agents with the CP-nets illustrated in Figure 4.1.2. We see that Agent 1 will veto* $\{Salad, Fish, White\}$ *while Agent 2 and Agent 3 will both veto* {Soup, Meat, White}*. In this case we have veto's for two of the* $2^3 = 8$ *possible outcomes. Therefore, the winning set is made up of six alternatives (ordered by the variable ordering and then lexicographically):* {Salad, Fish, Red}, {Salad, Meat, White}, {Salad, Meat, Red}, {Soup, Fish, Red}, {Soup, Fish, White}, {Soup, Meat, Red}*. Here we would select* {Salad, Fish, Red} *if we employ a lexicographic tie breaking rule.*

One-step k-approval (OK)

We ask each agent to provide their top k outcomes according to their CP-net, and a linearization of the outcome ordering (unless otherwise noted we always assume that our linearization is according to the variable ordering, then lexicographically). We require the linearization here because it is possible that k is not a round power of two. It is also possible, though we do not discuss it in this work, for each agent to individually employ their own linearization procedure. In this case we could have that k includes some, but not all, elements of an incomparable set in the partial order.

Example 4.1.4 *Consider the situation where* $k = 2$ *in our running example illustrated in Figure 4.1.1. We see that Agent 1 will approve of* $\{Soup, Meat, Red\}$ *and* {Soup, Meat, White}*. Agent 2 and Agent 3 will approve of* $\{Salad, Fish, White\}$ *and* $\{Salad, Fish, Red\}$*. Therefore, the winning set consists of the two options chosen by both Agent 2 and 3.*

4.2 Bribery and Manipulation

The bribery problem we consider in this chapter has three parameters: the way a winner is chosen from the given profile, the bribery actions, and the cost scheme for the bribers'

requests. In this section we extend the traditional notions of bribery in order to account for combinatorial domains and our structured preference representation. Bribery is usually considered in domains with only one issue to be decided [52] or, as in Section 3.1, multiple independent issues. It is a non-trivial exercise to define bribery in the domain considered in this chapter.

4.2.1 Bribery Actions

Bribery actions are the changes agents perform in their preferences in response to a briber's request. In most bribery frameworks, agents express their preferences over the candidates via a total order, and the briber asks agents to make changes within this total order. In our setting, agents have a CP-net instead of an explicit outcome ordering. We define the bribery actions as changes in the total order of some cp-statements. If the briber could ask for any change in the ordering induced by the CP-net, some of these changes could be computationally hard or impossible to accomplish via changes to the CP-net. Therefore, we define the bribery actions as changes made directly to the cp-statements within the CP-net of an agent. Since a cp-statement gives a total order for the domain of an issue, the briber asks for a new total order over that domain. In this section we consider binary issues, so changing a cp-statement means flipping the positions of the two values of an issue.

In a CP-net, a cp-statement is associated with a certain issue, and issues are of two kinds: independent and dependent. Independent issues have indegree zero in the dependency graph while dependent issues have indegree ≥ 1. Independent variables can affect larger changes in the final preference order and, since they are independent, agents have "pure preferences" over them, independent of any other assignment. Therefore, we consider independent variables to be more important than dependent variables.

We distinguish bribery actions by specifying in which cp-statements the briber is allowed to request a change.

IV: The briber can only request a change in the cp-statements of independent variables.

DV: The briber can only request a change in the cp-statement of dependent variables.

IV+DV: The briber can ask for any change in any cp-statement.

Notice that, with any of these bribery actions, no dependency can be added in the CP-nets. In fact, the outside actor can only remove dependencies through the variable assignment swaps. Thus, the new CP-nets are still compatible with one another and with the ordering O over the issues. This is only important for SM. Additionally, if only one or the other type of bribe is available then there are outcomes that may not be able to be elevated to the most preferred outcome, as illustrated in the following example.

Example 4.2.1 *Consider our running example from Figure 4.1.1. Suppose* $p = \{Soup, Meat, White\}$ *and we need to bribe some of the agents to have this as their top outcome using the SM rule.*

When considering only IV bribes, all of Agent 1's preferences can be changed since they are all independent. For Agent 1, we only need to bribe him to swap his assignment such that $Wine = White$*; p now has one vote. For Agent 2, both* App *and* $Main$ *are independent variables and we can bribe the agent on these two variables only. When we bribe these variables (swapping them to make the variable assignments match p) we see that Agent 2's most preferred assignment is* $Wine = Red$*, given* $Main = Meat$*. Since we are only allowing IV bribes we cannot make Agent 2's preferences match p. For Agent 3 we have the same problem as Agent 2. Though we can set Agent 3's assignment to* $App = Soup$ *and* $Main = Meat$*, this causes* $Wine = Red$ *in the CP-net. We cannot make p the winner of the election.*

Consider the case of only DV bribes. In this case we cannot change any of Agent 1's preferences because all his preferences are independent. Likewise, we cannot bribe Agent 2 or Agent 3's assignment of App *because no agent has* App *as a dependent variable. In this case we could change the assignment of* $Wine$ *but because we cannot change the assignment of any other variable, we cannot make p a winner.*

In the case of IV+DV bribes we can make p a winner. We make the same bribes to the agents as we did in the IV example. This time, however, we can also bribe Agent 2 and Agent 3 to have $Wine = White$. Using IV+DV we can make all agents have p as their top preference and a winner in the election.

4.2.2 Cost Schemes

In traditional bribery domains, the cost of the bribery actions is either fixed for any amount of change for each agent [52], variable per agent for any amount of change [51], or it depends on the number of swaps to be performed on the current total order [43]. However, we feel that none of these methods captures an effective cost scheme when agents express their preferences as CP-nets and voting is done over a combinatorial domain. We do not want to work on the outcome ordering, since it can be exponentially large; we would rather modify the CP-net directly. However, a single flip in a CP-net preference statement can lead to a large change in the outcome ordering and we need to take this into account. In fact, a formalism closer to the one defined in Chapter 3, where agents have cost associated to the change in probability of a vote, is not sufficient in the CP-net context either. Therefore, in this chapter we consider four different cost schemes. Some of these cost schemes closely resemble the costs in more traditional bribery domains while others are more closely tied to the use of the CP-net formalism and applicable to combinatorial domains.

The four cost schemes we investigate in this chapter are listed here from least expressive to most expressive.

C_{EQUAL}: A unit cost allows any number of flips in the cp-statements of a CP-net. This method is the same as the one proposed by Faliszewski et al. in the original definition of bribery [52].

C_{FLIP}: The cost of changing a CP-net is the total number of individual cp-statements that must be flipped in order to change one profile P_i to a target profile P_t. This method

captures a notion of how much change the briber is asking for, and makes the cost of bribery proportional to the total amount of change. This method is similar to the swap bribery method introduced by Elkind et al. [43] applied to CP-nets instead of strict linear orders.

$C_{\textbf{LEVEL}}$**:** We expect an issue that is closer to being independent is more important, as in some other structured preference formalisms [14], than a issue buried deep in a dependency graph. Therefore, we expect that the cost of changing a more influential issue should be higher than lower level issues. We link the cost of a flip to this importance: the cost of changing a CP-net is the total number of flips performed in the cp-statements, each weighted according to the level of the relevant issue (in the original CP-net configuration). We define the *level* of a issue recursively as:

$$level(x) = \begin{cases} 1 & \text{if } x \text{ is an independent issue;} \\ i+1 & \text{if all parents of } x \text{ are in levels } \{1,\ldots,i\} \\ & \text{and there is a parent in level } i. \end{cases} \tag{4.1}$$

We can then precisely define the cost of flipping an issue as $\sum_x flip(x) \times (k+1-level(x))$, where x ranges over the issues, k is the number of levels in the CP-net, and $flip(x)$ is the number of flips performed in cp-statements associated with x.

$C_{\textbf{ANY}}$**:** The total cost is the number of flips, each weighted by a specific cost. Formally, we define a vector for each agent that, for every variable $v_i \in M$, maps $v_i \to \mathbb{Z}_0^+$ so that modifying any variable in the instance has its own associated price. This is similar to the nonuniform bribery model introduced by Faliszewski [51]

Each of the above cost schemes can be generalized to account for agents with different bribing costs by associating a certain cost to each agent. This allows each agent to have its own cost function. We assume, where necessary, that each agent has their own individual bribing cost and we are provided a vector $\vec{Q} \in \mathbb{Z}^+$ of size n where each $Q[i]$ denotes the

individual bribing cost of agent i. We multiply this bribing cost by the cost returned from the given cost scheme. This creates a situation similar to plurality-$bribery introduced by Faliszewski et al. [52].

In many variations of the bribery and manipulation problems it is standard to attach weights to each voter [33, 52]. Weights usually represent a group of voters who all think alike, or voters with differing levels of importance such as shareholder votes for publicly traded companies. We will sometimes assume a weight vector is provided $\vec{W} \in \mathbb{Z}^+$ where $W[i]$ represents the weight associated to each voter i. When we use weighted voters we require that p be elected according to the given voting rule.

It is important to note that the distance of an outcome from the optimal one is not necessarily linked to the number of flips needed in the cp-statements to make that outcome optimal. Consider the CP-net and associated ordering presented for Agent 2 in Figure 4.1.1. In the outcome ordering the optimal ordering is $\{Salad, Fish, White\}$ and $\{Salad, Meat, Red\}$ dominates $\{Salad, Meat, White\}$ in the CP-net and therefore $\{Salad, Meat, White\}$ is farther away from the optimal outcome in the partial order. To make $\{Salad, Meat, White\}$ the optimal outcome we only need to flip one cp-statement ($Fish \rightarrow Meat$). However, to make $\{Salad, Meat, Red\}$ the optimal outcome in the CP-net, we require two flips ($Fish \rightarrow Meat$ and $White \rightarrow Red$) for a total cost of 2 for C_{FLIP} (versus 1 for the other outcome). For C_{LEVEL} we have a similar situation: changing the optimal outcome to $\{Salad, Meat, White\}$ has cost 1 while switching to $\{Salad, Meat, Red\}$ has cost $2 \times 1 + 1 = 3$ (one independent and one dependent flip). Therefore, the cost schemes C_{LEVEL} and C_{FLIP} (and the more general C_{ANY}) do not respect the distance-by-flips notion.

4.3 The Combinatorial Bribery Problem

We are now ready to formally state the bribery problem we study in this chapter. We can state this problem for each choice of winner determination rule $D \in \{SM, OP, OK, OV\}$, bribery action $A \in \{IV, DV, IV + DV\}$, and cost scheme $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}, C_{\text{ANY}}\}$.

Name: (D,A,C)-Bribery

Given: A profile (P,O), where P is a collection of n compact, acyclic CP-nets with m binary issues and O is a total ordering of the issues (when necessary), a budget $B \in \mathbb{Z}^+$, a preferred outcome p, and (when necessary) a bribing cost vector $\vec{Q} \in \mathbb{Z}^+$, and weights $\vec{W} \in \mathbb{Z}^+$.

Question: Is there a way for an outside actor to make p win in profile (P,O) with winner determination rule $D \in \{SM, OP, OK, OV\}$ using bribery actions according to $A \in \{IV, DV, IV+DV\}$, paying according to the cost scheme $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}, C_{\text{ANY}}\}$ and bribing cost vector $\vec{Q}$, weight $\geq W$ and without exceeding budget B?

The CCM problem is defined analogously for our domain with the condition that all entrants in the manipulator set have a unit cost to bribe and all other voters have infinite cost. This means that the CCM problem is actually a sub-problem of the bribery problem in this case.

Name: (D)-Manipulation

Given: A preferred outcome p and profile (P,O), where P is divided into two disjoint sets X and Y where $||X|| + ||Y|| = n$. The voters in X have preferences expressed as compact, acyclic CP-nets with m binary issues that follow the ordering O over the issues (when necessary) and the voters in Y have no preferences.

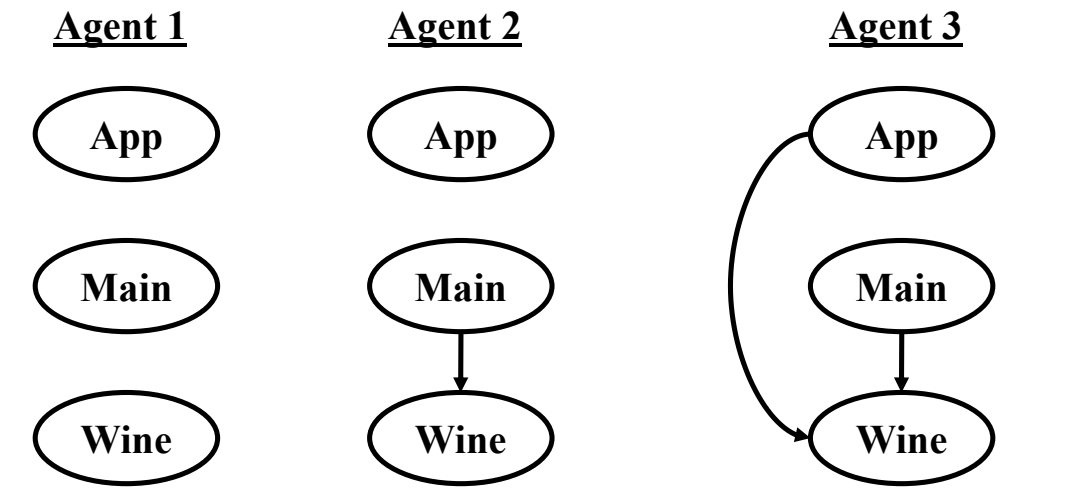
Question: Are there CP-nets for the agents in Y that are consistent with the ordering O (when necessary) such that p is made a winner in profile (P,O) with winner determination rule $D \in SM, OP, OK, OV$?

We assume, as in other works on bribery such as those by Elkind et al. [43] and Faliszewski et al. [52], a non-unique winner model: we are only asked to elevate p into the

winning set. Some of our reductions can be extended to a unique winner model but we focus on the general case. While we obtain mostly easiness results under this winner model (as did Bartholdi et al. [7]), a complete investigation of tie-breaking and its effect on the complexity of manipulation is outside the scope of this chapter. In fact, the careful study of tie-breaking mechanisms is extremely complex as shown by Obraztsova et al. [98] among others [58] and requires its own course of study.

Example 4.3.1 *Recall our running example with the CP-nets depicted below.*

Figure 4.5: CP-net with three agents expressing O-legal profiles over three binary variables for Example 4.3.1.



App	Soup > Salad
Main	Meat > Fish
Wine	Red > White

App	Salad > Soup
Main	Fish > Meat
Wine	Fish: White > Red Meat: Red > White

App	Salad > Soup
Main	Fish > Meat
Wine	Salad, Fish: White > Red Salad, Meat: Red > White Soup, Meat: Red > White Soup, Fish: Red > White

Given these profiles and $O = \{App, Main, Wine\}$ we can consider several examples of our bribery problem parameterized by different attributes.

(SM-IV-C_{EQUAL})-Bribery: *Consider a case where $B = 10$, $\vec{Q} = [5,5,5]$, and $p = \{Soup, Meat, Red\}$. We need to select some bribes, only in the IV variables, in order to make p a winner of the election decided by SM. By investigation we see that Agent 1 has exactly p as its top outcome so we do not need to bribe him at all. Additionally, we see that if we give bribes to either Agent 2 or Agent 3 on the App and Main, they will change $Wine \rightarrow Red$. Since we are using C_{EQUAL} and $Q[2] = Q[3] = 5$ we can make both assignment changes in either agents' cp-statements for a cost of 5. This makes each individual element of p a majority winner at its respective level. Therefore this is a "yes" instance of our problem.*

(OV-DV-C_{FLIP})-Bribery: *Consider a case where $B = 5$, $\vec{Q} = [1,1,2]$, and $p = \{Salad, Fish, White\}$. Notice that we are using OV and we have three agents and $2^3 = 8$ possible outcomes. In this instance there will, necessarily, be some outcomes that do not have any vetoes. Therefore, we must make sure that p has no vetoes since the winner set will contain only outcomes that have no vetoes. In this case we are limited to only DV bribes. This means we cannot bribe Agent 1 as he has no dependent variables. Unfortunately, Agent 1 is the only one vetoing p and we can't change his mind. Therefore, this is a "no" instance of our problem.*

(OP-$\{IV + DV\}$-C_{LEVEL})-Bribery: *Consider a case where $B = 3$, $\vec{Q} = [2,1,2]$, and $p = \{Salad, Meat, White\}$. Currently, no agent has this as their top preference and p is not winning. Notice that, in order to set Agent 1 to have p as his top preference requires two bribes to independent variables. Since all variables for Agent 1 are at level one out of the two levels of the CP-nets, the cost of this bribe is $Q[1] \times 2 = 4$. However, bribing either Agent 2 or Agent 3 requires a bribe at level one and a bribe at level two and therefore the cost of the bribe for Agent 2 is $Q[2] \times (1 \times 1 + 1 \times 2) = 3$ while Agent 3's bribing cost is $Q[3] \times (1 \times 1 + 1 \times 2) = 6$. We chose to bribe Agent 2 to now have p as his top candidate. This means that we have one vote each for p, $\{Soup, Meat, Red\}$, and $\{Salad, Fish, White\}$. We have elevated p into the winning set and spent less than our budget, so this is a "yes" instance.*

4.4 Results

In the rest of the chapter we study the computational complexity of this problem, considering all possible combinations of winner determination rules in $D \in \{SM, OP, OV, OK\}$, bribery actions in $A \in \{IV, DV, IV + DV\}$, and cost schemes in $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}, C_{\text{ANY}}\}$. Our results are summarized in Table 4.1 and Table 4.2. Note that in some cases we are still missing results and we will address these gaps while we unfold our results.

We begin by looking at the complexity of winner determination in our domain. We then consider the complexity of finding optimal bribery actions in our domain. After these preliminaries, we consider each voting rule in turn and conclude our results section with some extensions to non-binary domains.

4.4.1 Winner Determination and Changing a Vote

The computational complexity of the bribery problem is interesting only when it is easy to determine the winner of an election, since otherwise bribery would be trivially hard. Winner determination is easy to compute for all approaches we consider.

Theorem 4.4.1 *Winner determination is in P for SM, OP, OV, and OK (when k is bounded from above by a constant).*

Proof. SM works issue-by-issue, and at each issue, it applies the majority voting rule. As there are a polynomial number of issues, the overall procedure is in P.

For OP, we need to compute the optimal outcome for each acyclic CP-net, which is polynomial. Then plurality is applied, in time polynomial in the number of voters.

The winner according to One-step Veto (OV) can be obtained by reversing each total order in all cp-statements of the CP-nets. Thus, the optimal outcome of the new CP-net is the worst outcome of the original one. Notice that in an acyclic CP-net we have exactly one worst outcome. This reversal step takes time polynomial in the size of the CP-nets. After the reversal step, we just choose an outcome which appears the smallest number of times (possibly zero) as the optimal outcome of the reversed CP-nets. In some cases enumerating the entire winning set could take exponential time. However, since we assume p wins all ties we can determine if p is in the winning set by looking at the polynomial number of candidates who receive vetoes. If p is not in the winner set, then we randomly choose some other outcome receiving no vetoes as the winner of the election. This again takes polynomial time with respect to the size of the input.

OK needs to get the top k outcomes from each CP-net. If k is bounded, this is easy since the CP-nets are acyclic: each agent can just start from the optimal outcome and find the best k outcomes by applying the *next* operation $k-1$ times [17]. Therefore, we can produce the top k lists and apply the voting rule in polynomial time. ❑

In a bribery scenario, the briber has a desired candidate p. To make p the overall winner, the briber may need to ask some agents to change their current vote to a vote for p or to vote for some other candidate, paying for this change according to the cost scheme. In non-combinatorial bribery scenarios, agents specify their preferences explicitly and, therefore, the cost of bribery is easy to compute. In the four cost schemes that we have proposed, however, this computation is not so straightforward.

Theorem 4.4.2 *Given a CP-net and an outcome p, determining if the CP-net can be changed to make p its optimal outcome, and, if so, determining the minimum cost to perform such a change, can be computed in polynomial time if we use the cost schemes $C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}$, or C_{ANY}.*

Proof. Assume we can use all bribery actions, $IV + DV$. Therefore, any CP-net can be changed to vote for p (that is, to make p its optimal outcome).

For C_{EQUAL}, if p is already the top outcome of the CP-net, the cost is 0, otherwise it is 1. To find the p sweep through the CP-net and assign each variable to have the same assignment as in p.

For C_{FLIP}, we need to compute the minimum number of flips to be performed on the cp-statements within the CP-net in order to make p the top outcome. Starting from the independent issues, we flip the cp-statements which do not have the corresponding value of p as their preferred value. We then proceed to each dependent variable, after all its parents are processed, and do the same in its relevant cp-statement. At the end, p is the top outcome and the number of flips performed is the minimum cost to make p win.

Table 4.1: Bribery complexity results for Sequential Majority and Weighted Sequential Majority.

	Sequential Maj.	Theorem	Weighted Sequential Maj.	Theorem
C_{EQUAL}	NP-complete	4.4.4	NP-complete	4.4.5 and 4.4.6
C_{FLIP}	P	4.4.3	NP-complete	4.4.5 and 4.4.6
C_{LEVEL}	P	4.4.3	NP-complete	4.4.5 and 4.4.6
C_{ANY}	P	4.4.3	NP-complete	4.4.5 and 4.4.6

For C_{LEVEL} or C_{ANY}, finding the flips is the same as above but the cost takes into account the levels or the cost of each flip.

Notice that this algorithm gives us the minimum set of flips, but the same result could be achieved also by performing other useless flips. However, what we are interested in is computing the minimum cost to change the CP-net.

If we only had access to IV or DV, as soon as a necessary flip cannot be performed, we know that p cannot be the optimal outcome. ❑

4.4.2 Sequential Rules

In this section we consider the computational complexity of bribery when we determine the winner via sequential majority. A summary of our results can be found in Table 4.1.

Theorem 4.4.3 *(SM,IV,C)-Bribery, (SM,DV,C)-Bribery, and (SM,IV+DV,C)-Bribery are in P when* $C \in \{C_{\text{FLIP}}, C_{\text{LEVEL}}, C_{\text{ANY}}\}$.

Proof. We show a polynomial time algorithm for solving the bribery problem in an instance of $(SM, IV + DV, C_{\text{FLIP}})$-Bribery. The argument is independent of the selection of C and the same algorithm can be used for DV and IV bribery actions. We can determine whether or not the outside agent can achieve its agenda in polynomial time. We provide an algorithm to determine the cost of the optimal sequence of bribes

Given a profile (P,O), we consider the issues in the order O. For the first issue, we compute the minimum number of flips to be performed in the CP-nets in order to achieve

a majority for the same value that appears in p for that issue. Since we are using SM, we can select the agents to bribe by starting from the cheapest ones (according to $\vec{Q}$), until we achieve a majority for the considered issue. In fact, using SM, the resulting preferred value for this issue will be set in all agents' CP-nets (not just those where this value was most preferred). This allows us to effectively separate the levels so we can consider bribery for each variable independently of all other variables.

For each of the other issues in O we perform the same computation. We can ignore any conditional dependencies; as long as we achieve our agenda at this level, our bribery actions on a per agent basis will have no effect on the cost of changing an agents' preference for the next issue in O. This is because we must pass every issue at every level and passing an issue sets its value in *all* agents' CP-nets. Therefore, all effects farther down in the variable ordering are the same for all agents and we can ignore them.

When all issues have been considered, the number of performed flips required to get the correct majority for each issue is the cost of bribing so that p wins. The total cost is within the budget $\leq B$ if and only if the bribery problem has a solution.

If we use the cost scheme C_{ANY} the choice of the agents to bribe for each issue still only requires us to greedily select the cheapest agent at each level.

For C_{LEVEL} we must consider the level of the variables when computing the cost for each agent. However, since the topological order of any agents' preference dependencies must respect O we can traverse the variables in the order described in O. This will ensure that we consider the variables in cost order for each agent and that we do not change a variable before processing all variables it depends upon. Therefore, we can greedily select the cheapest agents to bribe at every level with C_{LEVEL}.

A very similar algorithm can be used for IV or DV by giving a cost of $B+1$ to an agents' bribery prices if p cannot be made its best outcome given the restrictions. ❑

With C_{EQUAL}, the cost for changing a CP-net is always 1, no matter how many flips are needed. Therefore, the decision of which agents' CP-nets to change for a particular issue

is strictly related to this same decision for another issue: if we change the same agents' CP-net for multiple issues, the cost will not increase. So, to compute the minimum bribery cost, the computation we do for one issue depends on the computation for a different issue since we are trying to minimize the total number of agents whose CP-nets we need to modify. This makes the bribery problem computationally hard.

Recall the OPTIMAL LOBBYING (OL) problem introduced by Christian et al. [24].

Name: OPTIMAL LOBBYING (OL)

Given : An $n \times m$ $0/1$ matrix E and a $0/1$ vector $\vec{x}$ of length m where each column of E represents an issue and each row of E represents a voter. We say E is a binary approval matrix with 1 corresponding to a "yes" vote and $\vec{x}$ is the target group decision.

Parameter: $k \in \mathbb{Z}^+$: the number of agents to be influenced.

Question: Is there a choice of k rows of the matrix E such that these rows can be edited so that the majority of votes in each column matches the target vector $\vec{x}$?

This problem is shown to be $W[2]$-complete via a reduction from k-DOMINATING SET [24]. We give a polynomial reduction from OL to our bribery problem, thus showing that our bribery problem is NP-complete [40]. In fact, since OL is $W[2]$-complete for parameter k and our reduction is parameter preserving, all the bribery problems in the following proof are $W[2]$-hard with parameter B.

Theorem 4.4.4 *$(SM, IV, C_{\text{EQUAL}})$-Bribery, $(SM, DV, C_{\text{EQUAL}})$-Bribery, and $(SM, IV + DV, C_{\text{EQUAL}})$-Bribery are NP-complete.*

Proof. Membership in NP is an immediate consequence of Theorem 4.4.1 and a guess and check algorithm.

To show completeness, we provide a polynomial reduction from OL. We start with $(SM, IV, C_{\text{EQUAL}})$-Bribery. Given an instance $(E, \vec{x}, k)$ of OL, we construct an instance of

$(SM, IV, C_{\text{EQUAL}})$-Bribery containing CP-nets with only independent variables. The set of issues, m, is equal to the number of columns in E. For each row of E, we create a voter with the preferences over the m variables as described in the row of E. Finally, we set the price of bribery for each voter to be 1, the budget $B = k$, the weights of the agents all equal, and the preferred outcome $p = \vec{x}$.

Thus, p wins the election if and only if there is a selection of k rows of E such that $\vec{x}$ becomes the winning agenda of the OL instance. Therefore $(SM, IV, C_{\text{EQUAL}})$-Bribery is NP-complete.

The same reduction works for IV+DV as well. For DV, we choose an instance of the bribery problem as above, except we add one more issue and each voter therefore has an additional independent variable on which all others depend. The preference of all voters on the new variable is $1 > 0$. For the other variables, the corresponding row of E will provide the preference associated to the value 1 of the new variable, while for the value 0 we give the opposite preference. The last difference is that now p is $1\vec{x}$. With this mapping, p wins the election if and only if the given OL instance has a positive answer. ❑

With weighted agents, the winner of the election will be computed by a weighted majority for each issue. We prove that bribery with all cost schemes is computationally difficult. We denote by **weighted-X** the bribery problem X with weighted agents.

Faliszewski et al. [52] show NP-completeness of the plurality-weighted-$bribery, which we state below, with a reduction from PARTITION to their single binary issue bribery problem.

Name: plurality-weighted-$bribery.

Given: A set C of candidates. A collection V of voters specified via their preference lists (most preferred candidate), weights $h_1, \ldots, h_m \in \mathbb{Z}^+$, and prices $\$_1, \ldots, \$_m \in \mathbb{Z}^+$. A distinguished candidate $p \in C$ and $k \in \mathbb{Z}^+$ (the budget).

Question: Is there a set $B \subseteq \{1,\ldots,m\}$ such that $\sum_{i \in B} \$_i \leq k$ and there is a way to bribe the voters from B in such a way that p becomes a winner?

In fact, Faliszewski et al. showed this problem to be NP-complete for just two candidates. The following theorem can be proven via polynomial reductions from plurality-weighted-$bribery over two candidates to our problems.

Theorem 4.4.5 *Weighted-(SM,A,C)-Bribery for* $A \in \{IV, IV+DV\}$ *and* $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}, C_{\text{ANY}}\}$ *is NP-complete.*

Proof. Membership in NP is an immediate consequence of Theorem 4.4.1 and a guess and check algorithm.

We reduce the plurality-weighted-$bribery problem over two candidates to an instance of weighted-(SM,IV,C_{EQUAL})-Bribery. We construct our instance with the same number of voters ($P = V$); the same costs ($Q[i] = \$_i$); weights ($w_i = h_i$); and the same budget ($B = k$) as the original problem. In our instance we create only one issue. This issue is an independent variable in our construction and the two values for the issue correspond exactly to the two candidates. We assign the votes and p the same as in the instance of plurality-weighted-$bribery.

The instance of our problem that we construct is exactly the given instance of plurality-weighted-$bribery: our problem is a yes instance if and only if there is a solution to the original problem. Therefore, weighted-(SM,IV,C_{EQUAL})-Bribery is NP-complete.

We can extend this proof for all variants of weighted-(SM,A,C)-Bribery where $A \in \{IV, IV+DV\}$ and $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}, C_{\text{ANY}}\}$. Since we are only using one independent variable in the construction there is no difference between IV and (IV+DV). Likewise, since there is only one variable and therefore only one variable level, the cost scheme does not matter. ❑

We can also extend the result of Theorem 4.4.5 to the case of DV bribery.

Corollary 4.4.6 *Weighted-(SM,DV,C)-Bribery where* $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}, C_{\text{ANY}}\}$ *is NP-complete.*

Proof. Membership in NP is an immediate consequence of Theorem 4.4.1 and a guess and check algorithm.

We apply the same mapping as shown in Theorem 4.4.5 to an instance of weighted-(SM,DV,C_{EQUAL})-Bribery except for the definition of the variables in our instance. Instead of only having one independent issue, we introduce an independent and dependent issue. The preference of all voters on the independent variable is $1 > 0$. We now map the preferences of the instance of plurality-weighted-$bribery to the dependent variable when the independent variable is set to 1. We set p to be $1x$ where x is the definition of p from the instance of plurality-weighted-$bribery. Since we are using only DV there will be no way to change the preference of any voter over the independent variable. Therefore, all bribery will take place on the dependent variable. This single dependent variable has the same candidate mapping as in the proof of Theorem 4.4.5. Therefore, there will be a solution to our constructed problem if and only if there is a solution to the given instance of plurality-weighted-$bribery and our problem is NP-complete.

We can extend this construction to any version of Weighted-(SM,DV,C)-Bribery where $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}, C_{\text{LEVEL}}, C_{\text{ANY}}\}$. We can set the costs of bribery on the dependent variables in each of these cost schemes to work with the above reduction.

For C_{FLIP} we set the entries of Q to be the same as in the construction for C_{EQUAL}. Since there is only one available bribery action for each voter, this will make the prices equivalent. We can set the costs in the same way for C_{ANY}. For C_{LEVEL}, each of the variables will be at level 2 (since they are all dependent on the independent issue). Therefore, we can set the entries of Q in the same way as for the other cost schemes. ❑

However, if we use the cost scheme C_{FLIP} and we assume that the bribing costs are the same for all voters, the bribery problem becomes easy.

Theorem 4.4.7 *Weighted-(SM,IV,C_{FLIP})-Bribery, weighted-(SM,DV,C_{FLIP})-Bribery, and weighted-(SM,IV+DV,C_{FLIP})-Bribery are in P when the bribing costs in $\vec{Q}$ are all the same.*

Proof. We can use the same argument as in the proof of Theorem 4.4.3. Since voters are weighted, we only need to achieve a weighted majority for each individual issue.

Since all the bribing costs at each level are the same with the C_{FLIP} pricing function, the cost to bribe any voter at any particular level is equal for all voters. We can therefore bribe the voters from heaviest to lightest (in terms of the individual voters' weight). This will be an optimal sequence of bribes at each individual level and lead to the optimal bribery scheme overall.

This is true no matter which bribery actions are allowed, since using IV or DV just restricts the issues over which it is possible to perform a flip in the CP-nets.

❑

4.4.3 One-Step Rules

In this section we will investigate the three one-step rules that we have introduced in this chapter: OP, OV, and OK. We will consider each of these rules in turn and we will see that these rules pose more significant challenges then the sequential rules. A summary of our results can be found in Table 4.2.

One-step Plurality

Recall the MINIMUM COST FEASIBLE FLOW problem introduced in Section 2.1.2 [1]. Faliszewski [51] shows that plurality bribery in single issue elections with nonuniform cost functions is in P through the use of flow networks. We showed a similar algorithm in Theorem 3.2.17 for finding possible winners with bribery in round robin sports tournaments. Both of these algorithm require the enumeration of all possible elements of the candidate

Table 4.2: Bribery and complexity results for one-step rules. OP(A) stands for voting rule OP with bribery actions A, and similarly for OV and OK, OK* stands for OK when k is a power of 2. In some cases we have not been able to provide lower bounds. In these cases we note the upper bounds with $\in$.

	C_{EQUAL}	C_{FLIP}
OP (IV)	P (Thm. 4.4.8)	P (Thm. 4.4.8)
OP (DV, IV+DV)	P (Thm. 4.4.8)	P (Thm. 4.4.8)
OV (IV)	P (Thm. 4.4.12)	P (Thm. 4.4.12)
OV (DV, IV+DV)	P (Thm. 4.4.12)	P (Thm. 4.4.12)
OK*(IV)	P (Thm. 4.4.15)	P (Thm. 4.4.15)
OK*(DV, IV+DV)	P (Thm. 4.4.15)	P (Thm. 4.4.15)
	C_{LEVEL}	C_{ANY}
OP (IV)	P (Thm. 4.4.9)	$\in$ NP (Thm. 4.4.10)
OP (DV, IV+DV)	$\in$ NP (Thm. 4.4.10)	$\in$ NP (Thm. 4.4.10)
OV (IV)	P (Coro. 4.4.13)	$\in$ NP (Thm. 4.4.10)
OV (DV, IV+DV)	$\in$ NP (Thm. 4.4.10)	$\in$ NP (Thm. 4.4.10)
OK*(IV)	P (Coro. 4.4.16)	$\in$ NP (Thm. 4.4.10)
OK*(DV, IV+DV)	$\in$ NP (Thm. 4.4.10)	$\in$ NP (Thm. 4.4.10)

set as part of the construction of the flow network. In our model, the number of candidates can be exponential in the size of the input, so we cannot use that construction directly.

Theorem 4.4.8 *(OP,A,C)-Bribery, where* $A \in \{IV, DV, IV + DV\}$ *and* $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}\}$*, is in P.*

Proof. If the number of candidates, which is 2^m, is polynomial in the number of voters (n), we can directly use the construction in [51]. Therefore, we assume we have a number of candidates that is superpolynomial in the number of voters. We show that a similar technique using flow networks as in [51] can be used in the superpolynomial case while still assuring a polynomial algorithm.

We can build our flow network without listing all candidates. Let candidate c_i be **safe** if $score(c_i) \leq score(p) - 1$. The key insight is that, at most n of the candidates will receive votes. For each voter v_i, we only need to consider the next n safe candidates enumerated in cost order. This is because, in the worst case, we will have to redistribute the votes of all n

voters to candidates other than those currently receiving votes. We can safely assume that we will redistribute the votes as cheaply as possible, so we need only consider, for each voter, the $\leq n$ unsafe candidates (those receiving votes) and then at most n more.

We then construct a series of networks, one for each $r \in \{1, \ldots, n\}$. We want to know whether, for any such r, p can be made a winner with exactly r votes without exceeding B. If so, then the answer to the bribery question is yes. Otherwise, we reject. We show that, for each r, the network can be constructed in time polynomial in the number of voters. Since there are only n values of r of interest, this shows that the whole (Turing) reduction can be performed in P.

To solve the decision problem for a given r, we transform this problem to a minimum-cost flow problem [1] (and discussed in Section 2.1.2). In our graph, units of flow represent votes and the minimum cost of the graph flow is the minimum cost of bribery necessary to elect p with exactly r votes.

We describe the construction similarly to Theorem 3.2.17 in Chapter 3. The network has a source s, a sink t, and two internal "layers" of nodes. We will describe the construction, moving through these three internal layers in turn.

The first layer has one node for each voter $v_1, \ldots, v_n$. There are also n edges (s, v_i), with capacity 1 and cost 0 giving one vote to every voter.

The second layer models the profile modified by the bribery, where each voter can maintain or change his current vote. The important point is that the other non-voted candidates that we do not include in the second layer would never be used in the new profile obtained through bribery since either they cost too much or they already receive too many votes. Providing n safe candidates for each voter is enough, since there are n voters and in the worst case each of them votes for a different candidate. In the worst case there are $n-1$ unsafe candidates before we encounter the first safe candidate. Once we reach the first safe candidate we enumerate $n-1$ more candidates. Therefore, we only need to enumerate a polynomial number of candidates.

The second layer of nodes represents the subset of the candidates that v_i can make his top candidate. For each voter v_i, we define a subset C_i of candidates consisting of p, the candidate that v_i currently has as his top outcome, all candidates that currently receive a vote (at most n), and the n safe, cheapest candidates for this voter, according to the cost scheme, for a total of at most $2n+1$ candidates. The second layer of nodes in the network is the union, S, of all the C_is. The set S contains p and all $\leq n$ top-choice candidates, plus at most n candidates for each of the n voters. Thus, $|S| \leq (n^2+n+1)$.

For each voter node v_i in the first layer, we build an edge to every node in that voter's set of considered candidates in S. The edge from v_i to c_j has capacity $+\infty$ and cost equal to the cost to bribe v_i to vote for c_j.

Finding the candidates to include for each voter and the cost of bribery for the particular voter/candidate pair takes polynomial time no matter the cost scheme. Finding the voted candidates is easy since finding the optimal outcome in acyclic CP-nets takes polynomial time. In order to find the n cheapest safe candidates for C_{EQUAL} we start from the optimal outcome for each voters' CP-net and move down in a linearization of the outcome ordering. This method takes polynomial time, as shown by Brafman et al. [17]. For C_{FLIP}, we first consider the candidates which can be obtained by performing one flip in the CP-net and then computing the new optimal outcome. Then we proceed by increasing the number of flips, until we collect n non-voted candidates.

Given a voter, computing the cost for such a voter to vote for one of the collected candidates is easy for the non-voted candidates, since they are obtained by using the cost schemes. For C_{EQUAL} the cost is always 1. For the voted candidates, including p, it is easy to compute the cost if we use C_{FLIP} as shown in Theorem 4.4.2.

To enforce the constraint that no candidate besides p can receive more than r votes we build an edge from each candidate to the sink with capacity r. The cost is 0 for the edge from p to the sink, while for all other candidates it is a large integer M to force as much flow through the node p as possible.

If we had included nodes for all the candidates in the second layer, we would have used a network equivalent to the one used in the proof of Theorem 3.1 in [51], which shows that there is a minimum cost flow of value n if and only if there is a way to solve the bribery problem. However, since we have a number of candidates which is superpolynomial in the size of the input, we would not have a polynomial algorithm. By including only the cheapest n alternative candidates for each voter, along with p and all the voted candidates, the result still holds. Assume there is a minimal flow in the larger network which goes through one of the nodes which we omit. This would mean that a voter has been forced to vote for another, more expensive, non-voted candidate since all its cheapest candidates already had r votes each. But, this is not possible, since we have only a total of $n-1$ votes that can be given by the other voters, and we provide n non-voted candidates.

We build, at worst, n networks with $O(n^2)$ nodes and edges. Since min cost feasible flow problem can be solved in $\mathcal{O}(|V||E|\log(|V|))$, the overall running time of this method is polynomial in the size of the CP-nets and n. ❑

Leveraging the above construction, we get the following theorem.

Theorem 4.4.9 *(OP,IV,C_{LEVEL})-Bribery is in P.*

Proof. The result of Theorem 4.4.8 also holds when we use the cost schemes C_{LEVEL} or C_{ANY} with the IV bribery criteria.

When not all variables are independent and we use IV, it may be impossible to find n alternative candidates for a voter, since that voter may be able to vote for fewer than n candidates given the restrictions on the bribery actions. In this case we provide all possible alternative candidates for the voter (since we may have to shuffle his vote).

For any set of voters we only need to be able, in polynomial time, to enumerate the next n possible votes in cost order. If we can achieve this enumeration then we are guaranteed, by use of the pigeon hole principal, that all voters can be moved as in Theorem 4.4.8. For the C_{LEVEL} criteria we can enumerate the next safe candidate in polynomial time. Notice

that, on a per agent basis, all independent variables will have the same bribing cost. Specifically, for some agent i, the cost to make any bribery action on any single independent variables will be $k \times Q[i]$, where k is the number of levels in agent i's CP-net. This means that, when only IV is used, C_{LEVEL} is equivalent to C_{FLIP}.

Now that we can enumerate the next candidate in cost order on a per agent basis, we directly use the construction in Theorem 4.4.8. Therefore, (OP,IV,C_{LEVEL})-Bribery is in P. ❑

There are combinations of cost schemes and bribery actions for which we do not have a result for any of our one-step rules. Specifically, we have no results for C_{LEVEL} with bribery actions involving DV and C_{ANY} with any bribery actions. However, all these problems are in NP by virtue of Theorem 4.4.1 and guess and check algorithms. Given an instance of the problem and a set of bribes, we can check whether p wins in polynomial time.

Theorem 4.4.10 *(D,A,C_{LEVEL})-Bribery with $A \in \{DV, IV+DV\}$ and (D,A,C_{ANY})-Bribery with $A \in \{IV, DV, IV+DV\}$ are in NP when $D \in \{OP, OV, OK\}$.*

The following theorem leads us to the conjecture that all the problem variants listed in Theorem 4.4.10 are hard when using C_{ANY}. Recall the SUBSET SUM PROBLEM statement from Garey and Johnson [65].

Name: SUBSET SUM

Given: A set $H \in \mathbb{Z}_+$, $h_1, \ldots, h_k$, and a target $S \in \mathbb{Z}_+$.

Question: Does there exist a subset $I \subseteq \{1, \ldots, k\}$ such that $\sum_{i \in I} h_i = S$?

Using this problem we can show the following.

Theorem 4.4.11 *Computing the n cheapest next candidates with C_{ANY} is NP-hard.*

Proof. We map a given SUBSET SUM instance (H,S) in the problem of finding the next cheapest voter in an instance of (D,A,C_{ANY})-Bribery. Consider a domain with $m = k$ binary features. We set the cost m_i for some voter v_i on feature f_i to be h_i, his current vote to 0^m, and set the budget B to S. If we were to enumerate v_i's next n cheapest candidates, we will need to find a candidate such that the cost of all the flips will be exactly $B = S$. In order to perform this enumeration we need to solve an NP-complete sub-problem. Therefore computing the n cheapest next candidates with C_{ANY} is NP-hard. ❑

Theorem 4.4.11 does not clarify the results for C_{LEVEL} and the dependent variable bribery schemes. These problems create an instance similar to C_{ANY}, however, the price structure for C_{LEVEL} must be strictly decreasing as we go down levels. This does not give us the freedom to obviously embed a SUBSET SUM instance like we constructed in the proof of Theorem 4.4.11. We are continuing to study the complexity of problems involving C_{LEVEL} and dependent variable bribery.

One-step Veto

The OV provides a particular set of challenges as noted in Theorem 4.4.1 because we cannot necessarily enumerate the entire winning set. However, as we show, bribery with the OV rule is still computationally easy in most cases.

Theorem 4.4.12 *(OV,A,C)-Bribery is in P if* $A \in \{IV, DV, IV + DV\}$ *and* $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}\}$

Proof. The choice of IV, DV, or IV+DV, does not matter in this context. We can either change a voters' last place candidate or we can't, the method doesn't matter.

We note that to compute the worst candidate for every voter it is sufficient to reverse the orderings in all the cp-statements as discussed in Section 4.1.2. Let us call any CP-net on which this is performed a reversed CP-net. If p has no vetoes then we trivially accept (since he is, at worst, tied for a winning position). Otherwise we break into two cases:

when $n < 2^m$ we must bribe all vetoes for p to some other candidate. In this case there will necessarily be some candidates that receive no vetoes. Therefore, in order for p to win, he must have no vetoes. When $n \geq 2^m$ we consider the reversed CP-nets and we use an algorithm based on a flow network similar to the one in Theorem 4.4.8 to redistribute vetoes. Note that the number of candidates, in this case, is polynomial in the size of the input, and thus we can use all of the candidates in the flow network. We consider one network for each $r \in \{0, \ldots, n\}$. The network is the same as the one in Theorem 4.4.8 except that nodes not representing p have capacity ∞ into the sink. ❑

Additionally, we can state the following corollary similar to Section 4.4.3. The proof is a straightforward combination of Theorem 4.4.9 and the observation in Theorem 4.4.12.

Corollary 4.4.13 *(OV,IV,C_{LEVEL})-Bribery is in P.*

One-step k-Approval

In One-step k-approval (OK) we aggregate the n profiles by taking the top k outcomes from each agents' profile. In a CP-net, the top outcome is followed by a series of outcomes obtained through a sequence of worsening flips. Since CP-nets induce a partial order, we mean the top k according to some linearization. Here we assume that the linearization is according to variable order and then lexicographically. Other linearizations would produce the same results. In the traditional bribery domain, the bribery problem for k-approval, when $k \geq 3$, is NP-complete when all the bribery costs are equal [52]. However, when agents' preferences are expressed as CP-nets, bribery schemes for k-approval can be computed in polynomial time under certain conditions. First we need a lemma.

Lemma 4.4.14 *Given an acyclic CP-net X and a constant linearization scheme across all agents and $k = 2^j$, the top k outcomes of a X are all the outcomes differing from the top one on the value of exactly j issues. Moreover, given two CP-nets in the same profile and with the same top element, they have the same top k elements.*

Proof. Given a CP-net, consider any order of its issues which is compatible with its dependency graph. One can think of a CP-net as a binary string with elements labeled $x_1 > \cdots > x_i > x_{i+1} > \cdots > x_m$ regardless of the dependencies that are present (imagine the last variable in the ordering is the lowest bit of the binary string). For a given top outcome (assignment to variables $x_1, \ldots, x_m$) the next outcome will be the one with $\bar{x_m}$ and variables $x_1, \ldots, x_{m-1}$ maintaining their assignments. That is, the next outcome in terms of worsening flips will be the outcome that varies only on the last element. If $k = 2^j$, then the top k outcomes have the same values for issues $x_1, \ldots, x_{m-j}$ and differ on the values of issues $x_{m-j+1}, \ldots, x_m$. Since we can use the same topological order for any two CP-nets in a profile, if two CP-nets have the same top outcome then they have the same top k outcomes when k is a power of 2. ❑

A practical use of this result, apart from its use in the following proof, is for the elicitation problem for CP-nets: if the ordering over the issues is known, then we only need to elicit the top candidate in order to know a voters' top k candidates, when k is a power of 2.

Theorem 4.4.15 *When $k = 2^j$, (OK,A,C)-Bribery is in P if $A \in \{IV, DV, IV + DV\}$ and $C \in \{C_{\text{EQUAL}}, C_{\text{FLIP}}\}$,*

Proof. When k is a power of two Lemma 4.4.14 tells us that, if we fix the top outcome, the rest of the outcomes must follow in some unique order. Therefore, we treat the top k outcomes as one bundle (group of candidates that are all approved together). For each voter we will need to find n cheapest un-voted bundles. We can do this in polynomial time as all we require is the ability to apply *next* some multiple of k times. We then apply the algorithm in the proof of Theorem 4.4.8 in order to decide the cheapest bribery scheme to elevate the bundle that includes p into the winning set. ❑

Again, similar to our results for OV, we can state the following corollary.

Corollary 4.4.16 *When $k = 2^j$, (OK,IV,C_{LEVEL})-Bribery is in P.*

4.4.4 Manipulation and Non-binary Domains

We will now show that, in the setting we have considered, manipulation is easy for two of the voting procedures.

Theorem 4.4.17 *Weighted-CCM and unweighted-CCM are in P for SM, OP, OV, and OK (when $k = 2^j$).*

Proof. Note that in the CCM problems (weighted or unweighted) we are given two sets of voters, X and Y. We are asked to set the CP-nets of Y in such a way as to ensure a victory for a candidate p. All the voters in Y have completely unspecified CP-nets.

When voting with SM and OP the optimal thing to do is to set the CP-nets of voters in Y in such a way that p is their optimal outcome. This can be done in polynomial time. If p can win with all the votes from Y then we accept, otherwise we reject.

When voting with OV we can count how many vetoes p has in the set X. If p currently receives no vetoes then we set the votes of Y to veto any arbitrary candidate that is not p. If the total number of vetoes p receives in X is such that we can evenly distribute the votes of Y among the other candidates so that every candidate has as many or more vetoes than p then we can make p a winner. We can do this in polynomial time since we must check each voter's veto and the number of voters is polynomial. However, if the number of vetoes for p is greater than the number we can apply to all other candidates, then we cannot make p a winner.

For OK when $k = 2^j$ we can leverage the observation in Lemma 4.4.14. In this case, as in the case for SM and OP, we set all the voters in Y in such a way that p is the top candidate. Since we have no control over the next candidates this is the optimal action.

If p can win with all the votes from Y assigned in the above ways then we accept, otherwise we reject. ❑

Up to this point we have required that all variables in the domain be binary. While this is a convenient and highly expressive model, we wish to relax this assumption. This allows

the sequential composition of general voting rules (not just majority) as in other work on voting in sequential domains [82]. This allows for interesting extensions in our domain and may lead to some novel insights about bribery and manipulation in sequential domains when agent's preferences are represented as CP-nets.

In what follows, each issue has a domain with two or more values and the CP tables specify strict total orderings on such domains. For each variable $x \in M$, we associate a domain $D(x)$ of values to that variable. When using the DV bribery action, we fix some part of the linear order. The briber will modify the linear orders through a sequence of swaps, much like in the swap bribery domain described by Elkind et al. [43]. Note that editing dependencies could change the linear order on a particular level.

Now we can use different voting rules at every level. We associate to a profile (P, O) a vector R of voting rules $\langle r_1, \ldots, r_m \rangle$. At level i we use voting rule r_i to aggregate the preferences at that level and propagate the winning outcome at that level to the next level for all agents' CP-nets. This is similar to Della Pozza et al.'s [37] definition for sequential voting rules with multivariate domains with soft constraints. Winner determination in this domain is polynomial if evaluation of the underlying voting rules are polynomial procedures and we can use the same algorithm as in Theorem 4.4.1. With this model we can show some additional theorems about sequential voting rules.

Theorem 4.4.18 *If bribery or CCM is computationally hard for any rule in* $\langle r_1, \ldots, r_m \rangle$ *then bribery is hard for the sequential rule.*

Proof. This property can be seen from a direct example. Suppose that we had a voting scenario with two levels: the first to be aggregated by any easy rule shown in this section and the second level by any hard rule (such as the Borda rule [12, 38]). Since we are using a sequential rule, the first level will be computationally easy to manipulate since we can ignore effects on the variables of the second level. However, the second level will remain

computationally difficult to manipulate. Therefore, the inclusion of any hard rule in the sequential rules R makes bribery hard for the overall scenario. ❑

Theorem 4.4.19 *If bribery or CCM is easy for every rule in $\langle r_1, \ldots, r_m \rangle$ then bribery is easy for the sequential rule.*

Proof. This follows from the separability property shown in Theorem 4.4.3. Since the levels of the ordering O can be treated independently, the problem separates into a sequence of independent bribery problems. If each one of these sub-problems are computationally easy, the entire sequence is computationally easy because we must win at every level. ❑

4.5 Observations

The results of our study are summarized in Table 4.1 and Table 4.2. We note that not all our results are in these tables, since some of the proofs of properties and interesting sub-cases do not neatly fit into tabular form. Additional results not appearing in our tables are our results for complexity of the CCM problem in sequential domains and our results about sequential rules with non-binary issues.

We show that SM bribery is easy except when we use C_{EQUAL}. However, when voters are weighted, bribery for SM is almost always difficult. OP and OV bribery is easy except for the two cases we have C_{LEVEL} with any variant of DV or any case with C_{ANY}. In these cases, we do not have a formal result although we believe the problem is difficult. Bribery is easy also for OK when k is a power of 2. This is particularly surprising at first sight, since bribery for k-approval in a non-combinatorial setting is difficult.

We have also shown several interesting properties of acyclic CP-nets that can be leveraged for computational and preference elicitation reasons. In fact, when agents express their preferences as acyclic compatible CP-nets, there are cases of the bribery problem that become computationally easier than the single issue domains, since such CP-nets have a restricted expressive power in terms of the orderings they can induce.

4.6 Summary

In this chapter we have studied the complexity of bribery and manipulation problems in elections with combinatorially structured domains where agents' express their preferences as CP-nets. To perform this study we introduced four cost schemes, three bribery schemes, and two broad classes of voting rules for combinatorial domains. This chapter represents novel work in that we are the first to extend the bribery and manipulation problems to combinatorially structured domains. In general we find that bribery in these settings is computationally easy for all the non-weighted sequential rules and many of the one-step rules. We are continuing to extend this work with studies of additional voting rules for the one-step instances, allowing domains with non-binary variables, and investigating the possibility of the bribery actor being able to add conditional preferences to the individual agents.

The complexity questions we have answered in this chapter are unsettling. We have seen that it may be computationally easy to affect elections in domains of these types. At first glance, this is odd since we have the possibility of an exponential number of candidates with respect to the size of the problem input. However, as we have seen, we can effectively work around this possible computational barrier. Our results may be due to an over-constrained model and our continuing work will address this issue.

In this chapter and the previous chapter we have focused on the theoretical implications of the security of election systems. We have established some baselines as to the computational complexity of reasoning in a variety of bribery domains. However, we are left with the underlying question of whether or not our results are applicable in actual elections (either involving people or multiagent systems). We dive into this question of practicality with the next chapter that looks at elections with real data and seeks to understand the empirical implications of our theoretical results.

Chapter 5 Empirical Analysis of Voting Rules and Election Paradoxes

This chapter details work on empirically testing the rates of occurrence of many of the paradoxes and issues that arise when using different voting rules. This work includes surveying existing datasets and empirical verification studies; the identification and mining of a novel dataset for theory testing; and extensive statistical analysis of the new dataset. The study of voting systems often takes place in the theoretical domain due to a lack of large samples of sincere, strictly ordered voting data. We derive several million elections (more than all the existing studies combined) from publicly available data: the Netflix Prize dataset [10]. The Netflix data is derived from millions of Netflix users who have an incentive to report sincere preferences, unlike random survey takers. We evaluate each of these elections under the Plurality, Borda, k-Approval, and Repeated Alternative Vote (RAV) voting rules. We examine the Condorcet Efficiency of each of the rules and the probability of occurrence of Condorcet's Paradox. We compare our votes to existing theories of domain restriction (e.g., single-peakedness) and statistical models used to generate election data for testing (e.g., Impartial Culture). We find a high consensus among the different voting rules; almost no instances of Condorcet's Paradox; almost no support for restricted preference profiles, and very little support for many of the statistical models currently used to generate election data for testing. Portions of this work have been previously published in refereed conference proceedings [86]. However, while the overall analysis is the same, this chapter details a greatly extended version of the work including several hundred million more elections than reported in the initial publication.

5.1 Motivation

As we have seen, voting rules and social choice methods have been used for centuries in order to make group decisions. Increasingly, in computer science, data collection and

reasoning systems are moving towards distributed and multi-agent design paradigms [96]. With this design shift comes the need to aggregate the (possibly disjoint) observations and preferences of individual agents into a total ordering in order to synthesize knowledge and data.

One of the most common methods of preference aggregation and group decision making in human systems is voting. Many societies, both throughout history and across the planet, use voting to arrive at group decisions on a range of topics from deciding what to have for dinner to declaring war. Unfortunately, results in the field of social choice prove that there is no perfect voting system and, in fact, voting systems can succumb to a host of problems. Arrow's Theorem demonstrates that any preference aggregation scheme for three or more alternatives will fail to meet a set of simple fairness conditions [2]. Each voting method violates one or more properties that most would consider important for a voting rule (such as non-dictatorship) [61]. Questions about voting and preference aggregation have circulated in the math and social choice communities for centuries [3, 27, 97].

Many scholars wish to empirically study how often and under what conditions individual voting rules fall victim to various voting irregularities [22, 61]. Due to a lack of large, accurate datasets, many computer scientists and political scientists are turning towards statistical distributions to generate election scenarios in order to verify and test voting rules and other decision procedures [111, 130]. These statistical models may or may not be grounded in reality and it is an open problem in both the political science and social choice fields as to what, exactly, election data looks like [124].

A fundamental problem in research into properties of voting rules is the lack of large data sets to run empirical experiments [108, 124]. There have been studies of several distinct datasets but these are limited in both number of elections analyzed [22] and size of individual elections within the datasets analyzed [61, 124]. While there is little agreement about the frequency with which different voting paradoxes occur or the consensus between voting methods, all the studies so far have found little evidence of **Condorcet's Voting**

Paradox [66] (a cyclical majority ordering) or **preference domain restrictions** such as **single peakedness** [13] (where one candidate out of a set of three is never ranked last). Additionally, most of the studies find a strong consensus between most voting rules except Plurality [22, 61, 108].

As the computational social choice community continues to grow there is increasing attention on empirical results (see, e.g., [130]). The empirical data will support and justify the theoretical concerns prevalent in the literature and that we have discussed in previous chapters [33,57]. Walsh explicitly called for the establishment of a repository of voting data in his COMSOC 2010 talk [131]. We begin to respond to this call through the identification, analysis, and posting of a new repository of voting data.

In this Chapter we detail the discovery and evaluation of an extremely large number of distinct 3 and 4 candidate elections derived from a novel dataset. We begin in Section 5.2 with a survey of the datasets that are commonly used in the literature. We then detail in Section 5.3 our new dataset, including summary statistics and a basic overview of the data. We then move into Section 5.4 which is broken into multiple subsections where we attempt to answer many of the questions about voting. Section 5.4.1 details an analysis that attempts to answer the questions "How often does Concert's Paradox occur?" and "How often does any voting cycle occur?" We continue with Section 5.4.2 which looks at the prevalence of single peaked preferences and other domain restricted election profiles [13, 117]. Section 5.4.3 investigates the consensus between multiple voting rules. We evaluate our millions of elections under the voting rules: Plurality, Copeland, Borda, Repeated Alternative Vote, and k-Approval. In Section 5.4.4 we evaluate our new dataset against many of the statistical models that are in use in the ComSoc and social choice communities to generate synthetic election data. We conclude in Section 5.5 with observations about our data in the context election systems and the current trends in computational social choice.

5.2 Survey of Existing Datasets

The literature on the empirical analysis of large voting datasets is somewhat sparse, and many studies use the same datasets [61, 124]. These problems can be attributed to the lack of large amounts of data from real elections [108]. Chamberlin et al. [22] provided empirical analysis of five elections of the American Psychological Association (APA). These elections range in size from 11,000 to 15,000 ballots (some of the largest elections studied). Within these elections there are no cyclical majority orderings and, of the six voting rules under study, only Plurality fails to coincide with the others on a regular basis. Similarly, Regenwetter et al. analyzed APA data from later years [109] and observed the same phenomena: a high degree of stability between elections rules. Felsenthal et al. [61] analyzed a dataset of 36 unique voting instances from unions and other professional organizations in Europe. Under a variety of voting rules Felsenthal et al. also found a high degree of consensus between voting rules (with the notable exception of Plurality).

All of the empirical studies surveyed [22, 61, 95, 108, 109, 124] came to a similar conclusion: there is scant evidence for occurrences of Condorcet's Paradox [97]. Many of these studies find no occurrence of majority cycles (and those that find cycles find them in rates of much less than 1% of elections). Additionally, each of these (with the exception of Niemi and his study of university elections, which he observes is a highly homogeneous population [95]) find almost no occurrences of either single-peaked preferences [13] or the more general value-restricted preferences [117].

Given this lack of data and the somewhat surprising results regarding voting irregularities, some authors have taken a more statistical approach. Over the years multiple statistical models have been proposed to generate election pseudo-data to analyze (e.g., [108, 124]). Gehrlein [66] provides an analysis of the probability of occurrence of Condorcet's Paradox in a variety of election cultures. Gehrlein exactly quantifies these probabilities and concludes that Condorcet's Paradox probably will only occur with very small electorates. Gehrlein states that some of the statistical cultures used to generate election pseudo-data,

specifically the Impartial Culture, may actually represent a worst-case scenario when analyzing voting rules for single-peaked preferences and the likelihood of observing Condorcet's Paradox [66]

Tideman and Plassmann have undertaken the task of verifying the statistical cultures used to generate pseudo-election data [124]. Using one of the largest datasets available, Tideman and Plassmann find little evidence supporting the models currently in use to generate election data. Additionally, Tideman and Plassmann propose several novel statistical models which better fit their empirical data.

5.3 The New Data

We have mined strict preference orders from the Netflix Prize Dataset [10]. The Netflix dataset offers a vast amount of preference data; compiled and publicly released by Netflix for its Netflix Prize [10]. There are 100,480,507 distinct ratings in the database. These ratings cover a total of 17,770 movies and 480,189 distinct users. Each user provides a numerical ranking between 1 and 5 (inclusive) of some subset of the movies. While all movies have at least one ranking it is not that case that all users have rated all movies. The dataset contains every movie rating received by Netflix, from its users, between when Netflix started tracking the data (early 2002) up to when the competition was announced (late 2005). This data has been perturbed to protect privacy and is conveniently coded for use by researchers.

The Netflix data is rare in preference studies: it is more sincere than most other preference data sets. Since users of the Netflix service will receive better recommendations from Netflix if they respond truthfully to the rating prompt, there is an incentive for each user to express sincere preference. This is in contrast to many other datasets which are compiled through surveys or other methods where the individuals questioned about their preferences have no stake in providing truthful responses.

We define an election as $E(m,n)$, where m is a set of candidates, $\{c_1,\ldots,c_m\}$, and n is a

set of votes. A vote is a strict preference ordering over all the candidates $c_1 > c_2 > \cdots > c_m$. For convenience and ease of exposition we will often speak in the terms of a three candidate election and label the candidates as A, B, C and preference profiles as $A > B > C$. All results and discussion can be extended to the case of more than three candidates. A voting rule takes, as input, a set of candidates and a set of votes and returns a set of winners which may be empty or contain one or more candidates. In our discussion, elections return a complete ordering over all the candidates in the election with no ties between candidates (after a tiebreaking rule has been applied). The candidates in our data set correspond to movies from the Netflix dataset and the votes correspond to strict preference orderings over these movies. We break ties according to the lowest numbered movie identifier in the Netflix set; these are random, sequential numbers assigned to every movie.

We construct vote instances from this dataset by looking at combinations of three movies. If we find a user with a strict preference ordering over the three moves, we tally that as a vote. For example, given movies A,B, and C: if a user rates movie $A = 1$, $B = 3$, and $C = 5$, then the user has a strict preference profile over the three movies we are considering and hence a vote. If we can find 350 or more votes for a particular movie triple then we regard that movie triple as an election and we record it. We use 350 as a cutoff for an election as it is the number of votes used by Tideman and Plassmann [124] in their study of voting data. While this is a somewhat arbitrary cutoff, Tideman and Plassmann claim it is a sufficient number to eliminate random noise in the elections [124]. We use the 350 number so that our results are directly comparable to the results reported by Tideman and Plassmann.

The dataset is too large to use completely ($\binom{17770}{3} \approx 1 \times 10^{12}$) and we have therefore subdivide. We have divided the movies into 10 independent (non-overlapping with respect to movies), randomly drawn samples of 1777 movies. This completely partitions the set of movies. For each sample we search all the $\binom{17770}{3} \approx 9.33 \times 10^8$ possible elections for those with more than 350 votes. For 3 candidate elections, this search generated 14,003,522

Figure 5.1: Empirical CDF of Set 1 for 3 candidate elections.

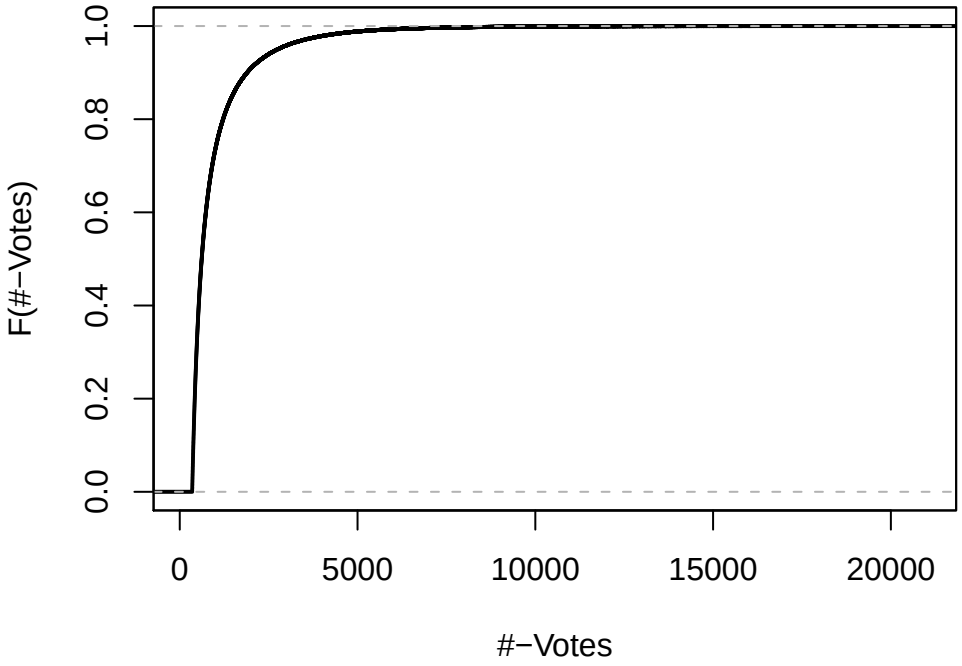
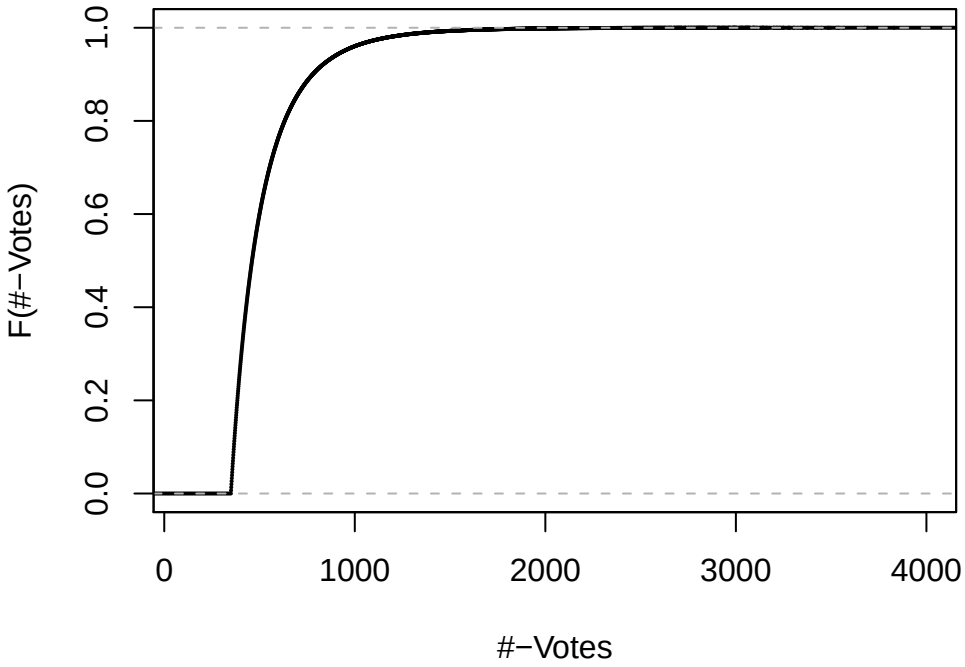


Figure 5.2: Empirical CDF of Set 1 for 4 candidate elections.



distinct movie triples in total over all the subdivisions. Not all users have rated all movies so the actual number of elections for each set is not consistent. The maximum election size found in the dataset is 24,670 votes; metrics of central tendency are presented in Tables 5.3 and 5.3. Figures 5.1 and 5.2 show the empirical cumulative distribution functions (ECFD) for Set 1 with 3 and 4 candidates respectively. The other set CDF's are similar.

Table 5.1: Summary statistics for 3 candidate elections.

	Set 1	Set 2	Set 3	Set 4	Set 5
Min.	350.0	350.0	350.0	350.0	350.0
1st Qu.	443.0	438.0	440.0	435.0	435.0
Median	610.0	592.0	597.0	583.0	581.0
Mean	964.8	880.6	893.3	843.3	829.9
3rd Qu.	1,011.0	958.0	960.0	921.0	915.0
Max.	18,270.0	19,480.0	19,040.0	17,930.0	12,630.0
Elements	1,453,012.0	1,640,584.0	1,737,858.0	1,495,316.0	1,388,892.0
	Set 6	Set 7	Set 8	Set 9	Set 10
Min.	350.0	350.0	350.0	350.0	350.0
1st Qu.	435.0	435.0	435.0	441.0	433.0
Median	584.0	585.0	580.0	600.0	573.0
Mean	853.2	868.4	841.3	862.7	779.2
3rd Qu.	923.0	935.0	911.0	963.0	876.0
Max.	20,250.0	24,670.0	21,260.0	17,750.0	13,230.0
Elements	1,344,775.0	931,403	1,251,478	1,500,040	1,260,164

Using the notion of item-item extension [70] we attempted to extend every triple found in the initial search. Item-item extension allows us to trim our search space by only searching for 4 movie combinations which contain a combination of 3 movies that was a valid voting instance. For each set we only searched for extensions within the same draw of 1777 movies, making sure to remove any duplicate extensions. The results of this search are summarized in Table 5.3. For 4 candidate elections, this search generated 11,362,358 distinct movie triples over all subdivisions. Our constructed datasets contains more than 5 orders of magnitude more distinct elections than all the previous studies *combined* and the largest single election contains slightly more votes than the largest previously studied election from data.

Table 5.2: Summary statistics for 4 candidate elections.

	Set 1	Set 2	Set 3	Set 4	Set 5
Min.	350.0	350.0	350.0	350.0	350.0
1st Qu.	397.0	390.0	392.0	388.0	386.0
Median	471.0	450.0	458.0	446.0	440.0
Mean	555.6	512.2	532.7	508.0	490.2
3rd Qu.	623.0	566.0	588.0	558.0	514.0
Max.	3,519.0	2,965.0	4,032.0	2,975.0	2,192.0
Elements	1,881,695.0	1,489,814.0	1,753,990	1,122,227.0	1,032,874
	Set 6	Set 7	Set 8	Set 9	Set 10
Min.	350.0	350.0	350.0	350.0	350.0
1st Qu.	389.0	390.0	388.0	383.0	380.0
Median	449.0	454.0	447.0	432.0	424.0
Mean	512.2	521.3	513.0	475.8	468.2
3rd Qu.	563.0	579.0	561.0	521.0	507.0
Max.	3,400.0	3,511.0	3,874.0	2,574.0	2,143.0
Elements	1,082,377.0	642,537	811,130	1,117,798	427,916

The data mining and experiments were performed on a pair of dedicated machines with dual-core Athlon 64x2 5000+ processors and 4 gigabytes of RAM. All the programs for searching the dataset and performing the experiments were written in C++. All of the statistical analysis was performed in R using RStudio.

The initial search of three movie combinations took approximately 36 hours (parallelized over the two cores) for each of the ten independently drawn sets. The four movie extension searches took approximately 250 hours per set. Computing the results of the various voting rules, checking for domain restrictions, and checking for cycles took approximately 20 hours per dataset. Calibrating and verifying the statistical distributions took approximately 20 hours per dataset. All the computations for this project are straightforward, the benefit of modern computational power allows our parallelized code to more quickly search the billions of possible movie combinations.

5.4 Analysis and Discussion

We have found a large correlation between each pair of voting rules under study with the exception of Plurality (when $m = 3,4$) and 2-Approval (when $m = 3$). A **Condorcet Winner** is a candidate who is preferred by a majority of the voters to each of the other candidates in an election [61]. The voting rules under study, with the exception of Copeland, are not **Condorcet Consistent**: they do not necessarily select a Condorcet Winner if one exists [97]. Therefore, we also analyze the voting rules in terms of their **Condorcet Efficiency**, the rate at which the rule selects a Condorcet Winner if one exists [93]. In Section 5.4.3 we see that the voting rules exhibit a high degree of Condorcet Efficiency in our dataset. The results in Section 5.4.1 show extremely small evidence for cases of single peaked preferences and very low rates of occurrence of preference cycles. Finally, the experiments in Section 5.4.4 indicate that several statistical models currently in use for testing new voting rules [111] do not reflect the reality of our dataset. All of these results are in keeping with the analysis of other, distinct, datasets [22,61,95,108,109,124] and provide support for their conclusions.

5.4.1 Preference Cycles

Condorcet's Paradox of Voting is the observation that rational group preferences can be aggregated, through a voting rule, into an irrational total preference [97]. It is an important theoretical and practical concern to evaluate how often the scenario arises in empirical data. In addition to analyzing instances of **total cycles** (Condorcet's Paradox) involving all candidates in an election, we check for two other types of cyclic preferences. We also search our results for both **partial cycles**, a cyclic ordering that does not include the top candidate (Condorcet Winner), and **partial top cycles**, a cycle that includes the top candidate but excludes one or more other candidates [61].

Tables 5.3 and 5.4 summarize the rates of occurrence of the different types of voting cycles found in our data sets. The cycle counts for $m = 3$ are all equivalent due to the fact that there is only one type of possible cycle when $m = 3$. There is an extremely low

instance of total cycles for all our data ($< 0.11\%$ of all elections). This corresponds to findings in the empirical literature that support the conclusion that Condorcet's Paradox has a low incidence of occurrence. Likewise, cycles of any type occur in rates $< 0.4\%$ and therefore seem of little practical importance in our dataset as well. Our results for cycles that do not include the winner mirror the results of Felsenthal et al. [61]: many cycles occur in the lower ranks of voters' preference orders in the election due to the voters' inability to distinguish between, or indifference towards, candidates the voter has a low ranking for or considers irrelevant.

Table 5.3: Number of elections demonstrating various types of voting cycles for 3 candidate elections.

	Set 1	Set 2	Set 3	Set 4	Set 5
Partial Cycle	667 (0.05%)	775 (0.05%)	995 (0.06%)	576 (0.04%)	638 (0.05%)
Partial Top	667 (0.05%)	775 (0.05%)	995 (0.06%)	576 (0.04%)	638 (0.05%)
Total	667 (0.05%)	775 (0.05%)	995 (0.06%)	576 (0.04%)	638 (0.05%)
	Set 6	Set 7	Set 8	Set 9	Set 10
Partial Cycle	560 (0.04%)	446 (0.05%)	512 (0.04%)	556 (0.04%)	896 (0.07%)
Partial Top	560 (0.04%)	446 (0.05%)	512 (0.04%)	556 (0.04%)	896 (0.07%)
Total	560 (0.04%)	446 (0.05%)	512 (0.04%)	556 (0.04%)	896 (0.07%)

Table 5.4: Number of elections demonstrating various types of voting cycles for 4 candidate elections.

	Set 1	Set 2	Set 3	Set 4	Set 5
Partial Cycle	4,088 (0.22%)	4,360 (0.29%)	3,879 (0.22%)	1,599 (0.14%)	1,316 (0.13%)
Partial Top	2,847 (0.15%)	3,042 (0.20%)	2,951 (0.17%)	1,165 (0.10%)	974 (0.09%)
Total	892 (0.05%)	1,110 (0.07%)	937 (0.05%)	427 (0.04%)	293 (0.03%)
	Set 6	Set 7	Set 8	Set 9	Set 10
Partial Cycle	1,597 (0.15%)	1,472 (0.23%)	1,407 (0.17%)	1,274 (0.11%)	1,646 (0.38%)
Partial Top	1,189 (0.11%)	1,222 (0.19%)	1,018 (0.13%)	870 (0.08%)	1,123 (0.26%)
Total	325 (0.03%)	438 (0.07%)	331 (0.04%)	198 (0.02%)	451 (0.11%)

5.4.2 Domain Restrictions

Black first introduced the notion of single-peaked preferences [13], a domain restriction that states that the candidates can be ordered along one axis of preference and there is a single peak to the graph of all votes by all voters if the candidates are ordered along this axis. Informally, the idea is that every member of the society has an (not necessarily identical) ideal point along a single axis and that, the farther an alternative is from the bliss point, the lower that candidate will be ranked. A conical example is that everyone has a preference for the volume of music in a room, the farther away (either louder or softer) the music is set, the less prefered that volume is.

This is expressed in an election as the scernio when some candidate, in a three candidate election, is never ranked last. The notion of restricted preference profiles was extended by Sen [117] to include the idea of candidates who are never ranked first (single-bottom) and candidates who are always ranked in the middle (single-mid). Domain restrictions can be expanded to the case where elections contain more than three candidates [3]. Preference restrictions have important theoretical applications and are widely studied in the area of election manipulation. Many election rules become trivially easy to affect through bribery or manipulation when electorates preferences are single-peaked [19].

Table 5.5 and Table 5.6 summarizes our results for the analysis of different restricted preference profiles. There is (nearly) a complete lack (10 total instances over all sets) of preference profile restrictions when $m = 4$ and near lack ($< 0.05\%$) when $m = 3$. It is important to remember that the underlying objects in this dataset are movies, and individuals, most likely, evaluate movies for many different reasons. Therefore, as the results of our analysis confirm, there are very few items that users rate with respect to a single dimension.

Table 5.5: Number of 3 candidate elections demonstrating preference profile restrictions.

	Set 1	Set 2	Set 3	Set 4	Set 5
Single Peaked	29 (0.002%)	92 (0.006%)	624 (0.036%)	54 (0.004%)	11 (0.001%)
Single Mid	0 (0.000%)	0 (0.000%)	0 (0.000%)	0 (0.000%)	0 (0.000%)
Single Bottom	44 (0.003%)	215 (0.013%)	412 (0.024%)	176 (0.012%)	24 (0.002%)
	Set 6	Set 7	Set 8	Set 9	Set 10
Single Peaked	162 (0.012%)	148 (0.016%)	122 (0.010%)	168 (0.011%)	43 (0.003%)
Single Mid	0 (0.000%)	0 (0.000%)	0 (0.000%)	0 (0.000%)	0 (0.000%)
Single Bottom	590 (0.044%)	147 (0.016%)	152 (0.012%)	434 (0.029%)	189 (0.015%)

Table 5.6: Number of 4 candidate elections demonstrating preference profile restrictions.

	Set 1	Set 2	Set 3	Set 4	Set 5
Single Peaked	0 (0.000%)	0 (0.000%)	1 (0.000%)	0 (0.000%)	0 (0.000%)
Single Mid	0 (0.000%)	0 (0.000%)	0 (0.000%)	0 (0.000%)	0 (0.000%)
Single Bottom	0 (0.000%)	0 (0.000%)	2 (0.000%)	0 (0.000%)	0 (0.000%)
	Set 6	Set 7	Set 8	Set 9	Set 10
Single Peaked	0 (0.000%)	3 (0.000%)	0 (0.000%)	1 (0.000%)	0 (0.000%)
Single Mid	0 (0.000%)	0 (0.000%)	0 (0.000%)	0 (0.000%)	0 (0.000%)
Single Bottom	1 (0.000%)	3 (0.000%)	0 (0.000%)	2 (0.000%)	0 (0.000%)

5.4.3 Voting Rules

The variety of voting rules and election models that have been implemented or "improved" over time is astounding. Arrow shows that any preference aggregation scheme for three or more alternatives cannot meet some simple fairness conditions [2]. This leads most scholars to question "which voting rule is the best?" We analyze our dataset under the voting rules Plurality, Borda, 2-Approval, and Repeated Alternative Vote (RAV). We briefly describe the voting rules under analysis. A more complete treatment of voting rules and their properties can be found in Nurmi [97], Arrow, Sen, and Suzumura [3], or Section 2.2.1.

Plurality: Plurality is the most widely used voting rule [97] (and, to many Americans, synonymous with the term "voting"). The Plurality score of a candidate is the sum of all the first place votes for that candidate. No other candidates in the vote are considered besides the first place vote. The winner is the candidate with the highest score.

k-Approval: Under k-Approval voting, when a voter casts a vote, the first k candidates each receive the same number of points. In a 2-Approval scheme, the first 2 candidates of every voter's preference order would receive the same number of points. The winner of a k-Approval election is the candidate with the highest total score.

Copeland: In a Copeland election each pairwise contest between candidates is considered. If candidate a defeats candidate b in a head-to-head comparison of first place votes then candidate a receives 1 point; a loss is -1 and a tie is worth 0 points. After all head-to-head comparisons are considered, the candidate with the highest total score is the winner of the election.

Borda: Borda's System of Marks involves assigning a numerical score to each position. In most implementations [97] the first place candidate receives $c-1$ points, with each candidate later in the ranking receiving one less points down to 0 points for the last ranked candidate. The winner is the candidate with the highest total score.

Repeated Alternative Vote: Repeated Alternative Vote (RAV) is an extension of the Alternative Vote (AV) into a rule which returns a complete order over all the candidates

[61]. For the selection of a single candidate there is no difference between RAV and AV. Scores are computed for each candidate as in Plurality. If no candidate has a strict majority of the votes the candidate receiving the fewest first place votes is dropped from all ballots and the votes are re-counted. If any candidate now has a strict majority, they are the winner. This process, for c candidates, is repeated up to $c-1$ times [61]. In RAV this procedure is repeated, removing the winning candidate from all votes in the election after they have won, until no candidates remain. The order in which the winning candidates were removed is the total ordering of all the candidates.

We follow the analysis outlined by Felsenthal et al. [61]. We establish the Copeland order as "ground truth" in each election; Copeland always selects the Condorcet Winner if one exists and many feel the ordering generated by the Copeland rule is the "most fair" when no Condorcet Winner exists [61, 97]. After determining the results of each election, for each voting rule, we compare the order produced by each rule to the Copeland order and compute the Spearman's Rank Order Correlation Coefficient (Spearman's ρ) to measure similarity [61].

Table 5.7 and Table 5.8 lists the mean and standard deviation for Spearman's Rho between the various voting rules and Copeland. All sets had a median value of 1.0. Our analysis supports other empirical studies in the field that find a high consensus between the various voting rules [22, 61, 109]. Plurality performs the worst as compared to Copeland across all the datasets. 2-Approval does fairly poorly when $m = 3$ but does surprisingly well when $m = 4$. We suspect this discrepancy is due to the fact that when $m = 3$, individual voters are able to select a full $2/3$ of the available candidates.

Table 5.7: Voting results (Spearman's ρ) for 3 candidate elections.

		Set 1	Set 2	Set 3	Set 4	Set 5
Plurality	Mean	0.9277	0.9275	0.9192	0.9375	0.9279
	SD	0.2013	0.1998	0.2172	0.1849	0.1983
2-Approval	Mean	0.9171	0.9157	0.9042	0.9191	0.9219
	SD	0.2132	0.2149	0.2299	0.2058	0.2048
Borda	Mean	0.9789	0.9792	0.9761	0.9801	0.9792
	SD	0.1024	0.1021	0.1090	0.0995	0.1019
RAV	Mean	0.9982	0.9982	0.9978	0.9987	0.9984
	SD	0.0367	0.0372	0.0414	0.0318	0.0353
		Set 6	Set 7	Set 8	Set 9	Set 10
Plurality	Mean	0.9308	0.9226	0.9280	0.9358	0.9130
	SD	0.1966	0.2110	0.1998	0.1873	0.2263
2-Approval	Mean	0.9222	0.9095	0.9152	0.9318	0.9065
	SD	0.2112	0.2252	0.2130	0.1920	0.2317
Borda	Mean	0.9803	0.9778	0.9782	0.9819	0.9756
	SD	0.0993	0.1208	0.1039	0.0953	0.1105
RAV	Mean	0.9985	0.9980	0.9984	0.9986	0.9975
	SD	0.0333	0.0373	0.0337	0.0331	0.0443

Table 5.8: Voting results (Spearman's ρ) for 4 candidate election.

		Set 1	Set 2	Set 3	Set 4	Set 5
Plurality	Mean	0.8757	0.8851	0.8551	0.9178	0.9037
	SD	0.2003	0.1853	0.2192	0.1487	0.1590
2-Approval	Mean	0.9504	0.9540	0.9511	0.9562	0.9554
	SD	0.1028	0.1000	0.1032	0.0950	0.0943
Borda	Mean	0.9747	0.9762	0.9739	0.9779	0.9792
	SD	0.0734	0.0717	0.0745	0.0679	0.0652
RAV	Mean	0.9962	0.9958	0.9956	0.9980	0.9982
	SD	0.0365	0.0395	0.0386	0.0258	0.0246
		Set 6	Set 7	Set 8	Set 9	Set 10
Plurality	Mean	0.8983	0.8835	0.8922	0.9047	0.7979
	SD	0.1765	0.1940	0.1802	0.1675	0.2738
2-Approval	Mean	0.9575	0.9554	0.9536	0.9677	0.9370
	SD	0.0975	0.0984	0.0993	0.0838	0.1168
Borda	Mean	0.9797	0.9765	0.9755	0.9847	0.9649
	SD	0.0656	0.0710	0.0722	0.0567	0.0865
RAV	Mean	0.9976	0.9962	0.9973	0.9982	0.9929
	SD	0.0278	0.0357	0.0293	0.0253	0.0496

There are many considerations one must make when selecting a voting rule for use within a given system. Merrill suggests that one of the most powerful metrics is Condorcet Efficiency [93]. Table 5.9 and Table 5.10 shows the proportion of Condorcet Winners selected by the various voting rules under study. We eliminated all elections that did not have a Condorcet Winner in this analysis. All voting rules select the Condorcet Winner a surprising majority of the time. The worst case is 2-Approval, when $m = 3$, as it results in the lowest Condorcet Efficiency in our dataset. The high rate of elections that have a Condorcet Winner ($> 80\%$) could be an artifact of how we select elections. By virtue of enforcing strict orders we are causing a selection bias in our set: we are only checking elections where many voters have a preference between any two items in the dataset.

Table 5.9: Condorcet Efficiency of the various voting rules for 3 candidate elections.

	Total Elections	Condorcet Winners	Plurality	2-Approval	Borda	RAV
Set 1	1,453,012	1,447,881	0.9661	0.8789	0.9785	0.9977
Set 2	1,640,584	1,634,829	0.9664	0.8775	0.9785	0.9975
Set 3	1,737,858	1,731,039	0.9583	0.8615	0.9744	0.9968
Set 4	1,495,316	1,490,265	0.9703	0.8752	0.9782	0.9982
Set 5	1,388,892	1,383,674	0.9676	0.8860	0.9795	0.9977
Set 6	1,344,775	1,340,321	0.9694	0.8882	0.9799	0.9979
Set 7	931,403	927,878	0.9612	0.8709	0.9771	0.9970
Set 8	1,251,478	1,247,199	0.9667	0.8729	0.9773	0.9977
Set 9	1,500,040	1,495,133	0.9725	0.9003	0.9825	0.9981
Set 10	1,260,164	1,254,712	0.9562	0.8711	0.9762	0.9964

Table 5.10: Condorcet Efficiency of the various voting rules for 4 candidate elections.

	Total Elections	Condorcet Winners	Plurality	2-Approval	Borda	RAV
Set 1	1,881,695	1,864,788	0.9450	0.9171	0.9610	0.9950
Set 2	1,489,814	1,474,697	0.9319	0.9100	0.9582	0.9926
Set 3	1,753,990	1,737,939	0.9382	0.9263	0.9629	0.9924
Set 4	1,122,227	1,113,378	0.9506	0.9010	0.9529	0.9961
Set 5	1,032,874	1,024,918	0.9583	0.9181	0.9632	0.9975
Set 6	1,082,377	1,074,434	0.9667	0.9380	0.9716	0.9970
Set 7	642,537	636,469	0.9294	0.9212	0.9601	0.9923
Set 8	811,130	804,496	0.9394	0.9058	0.9555	0.9946
Set 9	1,117,798	1,110,069	0.9708	0.9564	0.9816	0.9973
Set 10	427,916	422,489	0.9054	0.9205	0.9573	0.9891

Overall, we find a consensus between the various voting rules in our tests. This supports the findings of other empirical studies in the field [22, 61, 109]. Merrill finds much lower rates for Condorcet Efficiency than we do in our study [93]. However, Merrill uses statistical models to generate elections rather than empirical data to compute his numbers and this is likely the cause of the discrepancy [66].

5.4.4 Statistical Models of Elections

We evaluate our dataset to see how it matches up to different probability distributions found in the literature. We briefly detail several probability distributions (or "cultures") here that we test. Tideman and Plassmann provide a more complete discussion of the variety of statistical cultures in the literature [124]. There are other election generating cultures, such as weighted Independent Anonymous Culture, which generate preference profiles that are skewed towards single-peakedness or single-bottomness. As we have found no support in our analysis for restricted preference profiles we do not analyze these cultures (a further discussion and additional election generating statistical models can be found in [124]).

We follow the general outline in Tideman and Plassmann to guide us in this study [124]. For ease of discussion we divide the models into two groups: probability models (IC, DC, UC, UUP) and generative models (IAC, Urn, IAC-Fit). Probability models define a probability vector over each of the $m!$ possible strict preference rankings. We note these probabilities as $pr(ABC)$, which is the probability of observing a vote $A > B > C$ for each of the possible orderings. In order to compare how the statistical models describe the empirical data, we compute the mean Euclidean distance between the empirical probability distribution and the one predicted by the model.

Impartial Culture (IC): An even distribution over every vote exists. That is, for the $m!$ possible votes, each vote has probability $1/m!$ (a uniform distribution).

Dual Culture (DC): The dual culture assumes that the probability of opposite preference orders is equal. So, $pr(ABC) = pr(CBA)$, $pr(ACB) = pr(BCA)$ etc. This culture is

based on the idea that some groups are polarized over certain issues.

Uniform Culture (UC): The uniform culture assumes that the probability of distinct pairs of lexicographically neighboring orders (that share the same top candidate) are equal. For example, $pr(ABC) = pr(ACB)$ and $pr(BAC) = pr(BCA)$ but not $pr(ACB) = pr(CAB)$ (as, for three candidates, we pair them by the same winner). This culture corresponds to situations where voters have strong preferences over the top candidates but may be indifferent over candidates lower in the list.

Unequal Unique Probabilities (UUP): The unequal unique probabilities culture defines the voting probabilities as the maximum likelihood estimator over the entire dataset. We determine, for each of the data sets, the UUP distribution as described below.

For DC and UC each election generates its own statistical model according to the definition of the given culture. In order to calibrate the UUP we need to determine a multinomial probability distribution over the vote vectors. We follow a similar method described in Tideman and Plassmann [124]. To simplify discussion assume we have 3 candidates ($m = 3$) and therefore $m! = 6$ possible vote vectors. Call these $p_1 = ABC$, $p_2 = ACB$, $p_3 = BAC$, $p_4 = BCA$, $p_5 = CAB$, and $p_6 = CBA$.

We are calibrating UUP to some empirical (or observed) set of vote vectors. For each observation we re-label the voters so that, in the most common order, A is first, B is second, and C is third. Once this relabeling has occurred we want to find, for each election, the probability vector that maximizes the log likelihood of Equation 5.1.

$$f(N_1,\ldots,N_6;N,p_1,\ldots,p_6) = \frac{N!}{\prod_{r=1}^{6} N_r!}\prod_{r=1}^{6} p_r^{N_r} \tag{5.1}$$

Where N_r is the number of votes for vector r; N is the total number of votes in an election; and p_r is the probability (proportional share) of votes received by the particular preference ordering r. Note that $0 \leq p_r \leq 1$, $p_r = {}^{N_r}/_{N}$, and $\sum_{r=1}^{6} p_r = 1$. When we take the

log of this equation we end up with Equation 5.2.

$$g(N_1,\ldots,N_6;N,p_1,\ldots,p_6) = \left\{ \log N! - (\sum_{r=1}^{6} \log N_r!) \right\} + (\sum_{r=1}^{6} N_r \log p_r) \tag{5.2}$$

Since the term in the brackets is not dependent on the probability distribution we can drop it from our maximization problem. So, after rewriting we want to maximize Equation 5.3.

$$\max_{p_r} \left\{ \sum_{r=1}^{6} N_r \log(p_r) \right\} \tag{5.3}$$

Using Theorem 1.2.3 from Roman [112], page 22, we know that the vector which maximizes this equation is exactly the vector p_r from the empirical observation. So the maximum log-likelihood estimator for each election, according to the link function in Equation 5.1, is the empirical vector of vote shares.

In order to find the UUP distribution for an entire group of elections we have to find a probability vector that maximizes the log-likelihood of predicting all the reordered vote vectors. Let our set of relabeled elections be E. Again, we can drop the first term from Equation 5.2 as it has no effect since it is a constant scalar. Let $N_{r,i}$ be the number of votes for order $r \in R$ for election $i \in E$. Rewriting Equation 5.3 for all elections gives us an equation for our UUP distribution.

$$UUP = \max_{p_r} \left\{ \sum_{r=1}^{6} \left(\sum_{i \in E} N_{r,i} \right) \log(p_r) \right\} \tag{5.4}$$

With this equation we can again apply Theorem 1.2.3 from Roman [112]. Let

$$M = \sum_{r=0}^{6} \sum_{i \in E} N_{r,i}.$$

And let

$$S_r = \sum_{i \in E} N_{r,i}.$$

Then we can rewrite Equation 5.4 and are left with Equation 5.5.

$$\forall r \in \{1,\ldots,6\} : p_r = \frac{S_r}{M} \tag{5.5}$$

We can expand the above maximization problem for all 24 possible vectors when $m = 4$. To compute the error between the culture's distribution and the empirical observations, we re-label the culture distribution so that the preference order with the most votes in the empirical distribution matches the culture distribution and compute the error as the mean Euclidean distance between the discrete probability distributions.

Urn Model: The Polya Eggenberger urn model is a method designed to introduce some correlation between votes and does not assume a complete uniform random distribution [11]. We use a setup as described by Walsh [130]; we start with a jar containing one of each possible vote. We draw a vote at random and place it back into the jar with $a \in \mathbb{Z}_+$ additional votes of the same kind. We repeat this procedure until we have created a sufficient number of votes.

Impartial Anonymous Culture (IAC): Every distribution over orders has an equal likelihood. For each generated election we first randomly draw a distribution over all the $m!$ possible voting vectors and then use this model to generate votes in an election.

IAC-Fit: For this model we first determine the vote vector that maximizes the log-likelihood of Equation 5.1 without the reordering described for UUP. Using the probability vector obtained for $m = 3$ and $m = 4$ we randomly generate elections. This method generates a probability distribution or culture that represents our entire dataset.

For the generative models we must generate data in order to compare them to the culture distributions. To do this we average the total elections found for $m = 3$ and $m = 4$ and generate 1,400,352 and 1,132,636 elections, respectively. We then draw the individual election sizes randomly from the distribution represented in our dataset. After we generate these random elections we compare them to the probability distributions predicted by the various cultures.

Table 5.11: Mean Euclidean distance between the empirical data set and different statistical cultures (standard error in parentheses) for elections with 3 candidates.

	IC	DC	UC	UUP
Set 1	0.3064 (0.0137)	0.2742 (0.0113)	0.1652 (0.0087)	0.2817 (0.0307)
Set 2	0.3106 (0.0145)	0.2769 (0.0117)	0.1661 (0.0089)	0.2818 (0.0311)
Set 3	0.3005 (0.0157)	0.2675 (0.0130)	0.1639 (0.0091)	0.2860 (0.0307)
Set 4	0.3176 (0.0143)	0.2847 (0.0113)	0.1758 (0.0100)	0.2833 (0.0332)
Set 5	0.2974 (0.0125)	0.2677 (0.0104)	0.1610 (0.0082)	0.2774 (0.0300)
Set 6	0.3425 (0.0188)	0.3027 (0.0143)	0.1734 (0.0108)	0.3113 (0.0399)
Set 7	0.3043 (0.0154)	0.2704 (0.0125)	0.1660 (0.0095)	0.2665 (0.0289)
Set 8	0.3154 (0.0141)	0.2816 (0.0114)	0.1712 (0.0091)	0.2764 (0.0318)
Set 9	0.3248 (0.0171)	0.2906 (0.0130)	0.1686 (0.0100)	0.3005 (0.0377)
Set 10	0.2934 (0.0144)	0.2602 (0.0121)	0.1583 (0.0087)	0.2634 (0.0253)
Urn	0.6228 (0.0249)	0.4745 (0.0225)	0.4745 (0.0225)	0.4914 (0.1056)
IAC	0.2265 (0.0056)	0.1691 (0.0056)	0.1690 (0.0056)	0.2144 (0.0063)
IAC-Fit	0.0363 (0.0002)	0.0282 (0.0002)	0.0262 (0.0002)	0.0347 (0.0002)

Table 5.12: Mean Euclidean distance between the empirical data set and different statistical cultures (standard error in parentheses) for elections with 4 candidates.

	IC	DC	UC	UUP
Set 1	0.2394 (0.0046)	0.1967 (0.0031)	0.0991 (0.0020)	0.2533 (0.0120)
Set 2	0.2379 (0.0064)	0.1931 (0.0042)	0.0975 (0.0023)	0.2491 (0.0127)
Set 3	0.2633 (0.0079)	0.2129 (0.0051)	0.1153 (0.0032)	0.2902 (0.0159)
Set 4	0.2623 (0.0069)	0.2156 (0.0039)	0.1119 (0.0035)	0.2767 (0.0169)
Set 5	0.2458 (0.0044)	0.2040 (0.0028)	0.1059 (0.0027)	0.2633 (0.0138)
Set 6	0.3046 (0.0077)	0.2443 (0.0045)	0.1214 (0.0040)	0.3209 (0.0223)
Set 7	0.2583 (0.0088)	0.2094 (0.0053)	0.1060 (0.0038)	0.2710 (0.0161)
Set 8	0.2573 (0.0052)	0.2095 (0.0034)	0.1059 (0.0023)	0.2508 (0.0145)
Set 9	0.2981 (0.0090)	0.2414 (0.0049)	0.1202 (0.0045)	0.3258 (0.0241)
Set 10	0.2223 (0.0046)	0.1791 (0.0035)	0.1053 (0.0021)	0.2327 (0.0085)
Urn	0.6599 (0.0201)	0.4744 (0.0126)	0.4745 (0.0126)	0.6564 (0.1022)
IAC	0.1258 (0.0004)	0.0899 (0.0004)	0.0900 (0.0004)	0.1274 (0.0004)
IAC-Fit	0.0463 (0.0001)	0.0340 (0.0001)	0.0318 (0.0001)	0.0472 (0.0001)

Table 5.11 and Table 5.12 summarizes our results for the analysis of different statistical models used to generate elections. In general, none of the probability models captures our empirical data. Uniform Culture (UC) has the lowest error in predicting the distributions found in our empirical data. We conjecture that this is due to the process by which we select

movies and the fact that these are ratings on movies. Since we require strict orders and, generally, most people rate good movies better than bad movies, we obtain elections that look like UC scenarios. By this we mean that *The Godfather* is an objectively good movie while *Mega Shark vs. Crocosaurus* is pretty bad. While there are some people who may reverse these movies, most users will rate *The Godfather* higher. This gives the population something close to a UC when investigated in the way that we do here.

The data generated by our IAC-Fit model fits very closely to the various statistical models. This is most likely due to the fact that the distributions generated by the IAC-Fit procedure closely resemble an Impartial Culture (since our sample size is so large). We, like Tideman and Plassmann, find little support for the static cultures' ability to model real data [124]

5.5 Observations and Summary

In this chapter we have identified and thoroughly evaluated a novel dataset as a source of sincere election data. We find overwhelming support for many of the existing conclusions in the empirical literature. Namely, we find a high consensus among a variety of voting methods; low occurrences of Condorcet's Paradox and other voting cycles; low occurrences of preference domain restrictions such as single-peakedness; and a lack of support for existing statistical models which are used to generate election pseudo-data. Our study is significant as it adds more results to the current discussion of what an election is and how often voting irregularities occur. Voting is a common method by which agents make decisions both in computers and as a society. Understanding the statistical and mathematical properties of voting rules, as verified by empirical evidence across multiple domains, is an important step. We provide a new look at this question with a novel dataset that is several orders of magnitude larger than the sum of the data in previous studies.

This chapter represents an initial foray into empirically testing properties of elections. While we have not directly addressed the questions of manipulation and bribery with our

empirical study, we have laid the groundwork. This chapter provides us with perspective on the overall discussion of voting rules. By testing some of the theoretical properties of voting rules, and coming to the conclusion that some of the theoretical results are of little practical importance, we establish that more needs to be done to develop the empirical side of ComSoc. This empirical work is very much in the spirit of the overall ComSoc approach: we are using computational tools (data mining and access to extremely large sets of preference data) to address concerns in the social choice community. It is our hope that, with this dataset, we inspire others to look for novel datasets and empirically test some of their theoretical results.

Chapter 6 Conclusions and Future Directions

In this dissertation we have taken a comprehensive look at the effect that the type and amount of information can have on the bribery and manipulation problems. We have also surveyed a novel dataset that we are just starting to understand. The plethora of data present in the Netflix dataset will continue to support research in this area for several years.

In Chapter 3 we saw that relaxing the deterministic information assumption changes the reasoning complexity, with the problems generally becoming more difficult. However, this change in complexity is not uniform and we are left with several problem instances that are still computationally easy. This novel line of investigation has shown that, even in the face of uncertain information, many election rules are still susceptible to bribery and manipulation.

In Chapter 4 we saw the effect that structured information can have on the reasoning complexity of the bribery and manipulation problem. In the combinatorial domains with preferences modeled as CP-nets, many of the bribery problems are computationally tractable. In fact, for some cases of k-approval, the complexity of bribery actually became easier.

In Chapter 5 we identified and mined a novel set of data for use in empirically testing various voting rules and paradoxes. We saw that Condorcet's paradox is of little concern in our elections. Furthermore, we saw that in many cases voting rules return the same winner and are highly Condorcet efficient. We also saw that there is little to no support for existing statistical models of election data in the literature.

Though we present a significant body of research in this dissertation, we feel we have just begun to scratch the surface of some of these questions. The Netflix dataset, along with other publicly available datasets that we will mention, allow us unprecedented access to data with which we can empirically test and verify many of the assumptions about voting

rules in the ComSoc community. Additionally, despite some hard instances, we still have not hit truly computationally hard barriers for many of our models; many of them are either approximable to close factors or tractable in an FPT sense. These observations form the basis for much of the ongoing work.

For our theoretical work detailed in Chapter 3 and Chapter 4 we would like to complete our results. We were unable to provide lower bounds for some of the problems and we are continuing to work on these proofs. We would like to investigate heuristics and approximability results for the computationally hard bribery and manipulation problems. We are also interested in considering additional manipulation actions for our work with CP-nets: in the work presented in this dissertation, the manipulator can only delete dependencies, while we would like to investigate the complexity of the manipulation problem when the outside agent is also allowed to add dependencies within the CP-net. We would also like to verify some of our models using the vote histories from the US Congress from 1789-2000 from the University of California, Berkeley, *Vote World* project [133]. We can also obtain detailed records of the money received by candidates and lobbying performed by industries from the *Open Secrets* dataset [63]. Additionally, we would like to augment our models to use elements of inter-agent influence and look at the bribery and manipulation problem in the context of social-networks and other factors [60].

For the empirical work detailed in Chapter 5 we would like to empirically investigate voting rules and their potential as maximum likelihood estimators [31]. Additionally, we would like to expand our empirical evaluation of election cultures to include several new models of population distributions proposed by Tideman and Plassmann [124], and others. There has been extensive work in the computational social choice community on the probability that an election is computationally hard to manipulate based on the number of voters and candidates [129, 135]. Additional work has shown that voting rules are susceptible to small coalitions (relative to the total number of voters) for many voting rules [103] and protocol tweaks to decrease the probability that a small coalition will be able to manipu-

late an election [44]. We also plan to leverage the dataset to investigate manipulation in scenarios where voters do not express strict or complete orders [81, 139]. Additionally, we will be in a unique position to evaluate the effect of partial orders on election outcomes by classifying the sets of possible winners [80]. This plethora of data puts us in the position to empirically test many of the theoretical assumptions and/or predictions in the computational social choice literature.

Bibliography

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] K. J. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, 1963.

[3] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*, volume 1. North-Holland, 2002.

[4] Y. Bachrach, N. Betzler, and P. Faliszewski. Probabilistic possible winner determination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[5] 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(2):157–165, 1989.

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

[7] J.J. Bartholdi, C.A. Tovey, and M.A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[8] M. Baye, D. Kovenock, and C. de Vries. Rigging the lobbying process: An application of the all-pay auction. *The American Economic Review*, 83(1):289–294, 1993.

[9] M. Baye, D. Kovenock, and C. de Vries. The all-pay auction with complete information. *Economic Theory*, 8(2):291–305, 1996.

[10] J. Bennett and S. Lanning. The Netflix Prize. In *Proceedings of the KDD Cup and Workshop*, 2007. `www.netflixprize.com`.

[11] S. Berg. Paradox of voting under an urn model: The effect of homogeneity. *Public Choice*, 47(2):377–387, 1985.

[12] N. Betzler, R. Niedermeier, and G. Woeginger. Unweighted coalitional manipulation under the borda rule is NP-hard. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 2011.

[13] D. Black. On the rationale of group decision-making. *The Journal of Political Economy*, 56(1), 1948.

[14] C. Boutilier, F. Bacchus, and R.I. Brafman. UCP-networks: A directed graphical representation of conditional utilities. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI 2001)*, 2001.

[15] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21(1):135–191, 2004.

[16] D. Bouyssou, T. Marchant, P. Perny, M. Pirlot, A. Tsoukás, and P. Vincke. Evaluation and decision models. Springer, 2006.

[17] R. I. Brafman, F. Rossi, D. Salvagnin, K. B. Venable, and T. Walsh. Finding the next solution in constraint- and preference-based knowledge representation formalisms. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2010)*, 2010.

[18] S.J. Brams, D.M. Kilgour, and W.S. Zwicker. The paradox of multiple elections. *Social Choice and Welfare*, 15(2):211–236, 1998.

[19] F. Brandt, M. Brill, E. Hemaspaandra, and L. A. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[20] F. Brandt, V. Conitzer, and U. Endriss. Computational social choice. In G. Weiss, editor, *Multiagent Systems*. MIT Press, 2012.

[21] Martino Da Canal. *Les Estoires de Venise*. 1275. Translation by L.K. Morreale. Padua: Unipress, 2009.

[22] J. R. Chamberlin, J. L. Cohen, and C. H. Coombs. Social choice observed: Five presidential elections of the American Psychological Association. *The Journal of Politics*, 46(2):479 – 502, 1984.

[23] 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)*, 2007.

[24] R. Christian, M. Fellows, F. Rosamond, and A. Slinko. On complexity of lobbying in multiple referenda. *Review of Economic Design*, 11(3):217–224, 2007.

[25] J. D. Clinton. The statistical analysis of roll call data. *American Political Science Review*, 98(2):355–370, 2004.

[26] J. D. Clinton. Representation in Congress: Constituents and roll calls in the 106th house. *Journal of Politics*, 2(68):397–409, 2006.

[27] M. Condorcet. *Essay sur l'application de l'analyse de la probabilit des decisions: Redues et pluralit des voix*. Paris, 1785.

[28] V. Conitzer. *Computational Aspects of Preference Aggregation*. PhD thesis, Duke University, 2006.

[29] V. Conitzer, J. Lang, and L. Xia. How hard is it to control sequential elections via the agenda? In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 2009.

[30] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the 18th AAAI Conference on Artificial Intelligence (AAAI 2002)*, 2002.

[31] V. Conitzer and T. Sandholm. Common voting rules as maximum likelihood estimators. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*, 2005.

[32] V. Conitzer and T. Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI 2006)*, 2006.

[33] 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.

[34] V. Conitzer and L. Xia. Approximating common voting rules by sequential voting in multi-issue domains. In *12th International Symposium on Artificial Intelligence and Mathematics (ISAIM-12) Special Session on Computational Social Choice*, 2012.

[35] T. Corman, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[36] C. Crombez. Information, lobbying and the legislative process in the European Union. *European Union Politics*, 3(1):7–32, 2002.

[37] G. Dalla Pozza, M.S. Pini, F. Rossi, and K.B. Venable. Multi-agent soft constraint aggregation via sequential voting. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 2011.

[38] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for borda manipulation. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI 2011)*, 2011.

[39] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 2011. Online version, soon to appear in print.

[40] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[41] J. Duggan and T. Schwartz. Strategic manipulability without resoluteness or shared beliefs: Gibbard-satterthwaite generalized. *Social Choice and Welfare*, 17(1):85–93, 2000.

[42] E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *Proceedings of the 6th International Workshop on Internet and Network Economics (WINE 2010)*. Springer-Verlag, 2010.

[43] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. *Algorithmic Game Theory*, pages 299–310, 2009.

[44] E. Elkind and H. Lipmaa. Small coalitions cannot manipulate voting. In *Proceedings of the 9th International Conference on Financial Cryptography and Data Security (FC 2005)*, 2005.

[45] G. Erdélyi, H. Fernau, J. Goldsmith, N. Mattei, D. Raible, and J. Rothe. The complexity of probabilistic lobbying. In *Proceedings of the 1st International Conference on Algorithmic Decision Theory (ADT 2009)*, 2009.

[46] G. Erdélyi, H. Fernau, J. Goldsmith, N. Mattei, D. Raible, and J. Rothe. The complexity of probabilistic lobbying. Technical Report arXiv:0906.4431, CoRR: Computer Science Research Repository, 2011.

[47] G. Erdélyi, L. Hemaspaandra, J. Rothe, and H. Spakowski. On approximating optimal weighted lobbying, and frequency of correctness versus average-case polynomial time. In *Proceedings of the 16th International Symposium on Fundamentals of Computation Theory*, 2007.

[48] Diana Evans. Before the roll call: Interest group lobbying and public policy outcomes in house committees. *Political Research Quarterly*, 49(2):287–304, 1996.

[49] P. Faliszewski. Nonuniform bribery. Technical Report 2007-922, University of Rochester, Department of Computer Science, 2007.

[50] P. Faliszewski. *Manipulation of Elections: Algorithms and Infeasibility Results*. PhD thesis, University of Rochester, 2008.

[51] P. Faliszewski. Nonuniform bribery. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, 2008.

[52] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. How hard is bribery in elections? *Journal of Artificial Intelligence Research*, 35:485–532, 2009.

[53] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Llull and Copeland voting broadly resist bribery and control. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI 2007)*, 2007.

[54] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Copeland voting fully resists constructive control. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, 2008.

[55] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research*, 35:275–341, 2009.

[56] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Using complexity to protect elections. *Communications of the ACM*, 53(11):74–82, 2010.

[57] P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. A richer understanding of the complexity of election systems. In S. Ravi and S. Shukla, editors, *Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz*, pages 375 – 406. Springer, 2009.

[58] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: Ties matter. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, 2008.

[59] P. Faliszewski and A.D. Procaccia. AI's war on manipulation: Are we winning. *AI Magazine*, 31(4):53–64, 2010.

[60] K. Faust and S. Wasserman. *Social Network Analysis: Methods and Applications*. Cambridge University Press, New York, 1994.

[61] D. S. Felsenthal, Z. Maoz, and Rapoport A. An empirical evaluation of six voting procedures: Do they really make any difference? *British Journal of Political Science*, 23:1 – 27, 1993.

[62] J. Flum and M. Grohe. *Parameterized Complexity Theory*. EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2006.

[63] The Center for Responsive Politics. Open secrets: Money in politics. `http://www.opensecrets.org/`. Last accessed, 4/17/2012.

[64] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[65] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[66] W. V. Gehrlein. Condorcet's paradox and the likelihood of its occurrence: Different perspectives on balanced preferences. *Theory and Decisions*, 52(2):171 – 199, 2002.

[67] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973.

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

[69] Richard L. Hall and Frank W. Wayman. Buying time: Moneyed interests and the mobilization of bias in congressional committees. *American Political Science Review*, 84(3):797–820, 1990.

[70] J. Han and M. Kamber, editors. *Data Mining*. Morgan Kaufmann, 2006.

[71] N. Hazon, Y. Aumann, S. Kraus, and M. Wooldridge. Evaluation of election outcomes under uncertainty. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, 2008.

[72] N. Hazon, P. Dunne, S. Kraus, and M. Wooldridge. How to rig elections and competitions. In *Proceedings of the 2nd International Workshop on Computational Social Choice (COMSOC 2008)*, 2008.

[73] A. Healy, N. Malhotra, and C. H. Mo. Irrelevant events affect voters' evaluations of government performance. *Proceedings of the National Academy of Sciences of the United States of America*, 107(29):12804–12809, 2010.

[74] L. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion*. Springer-Verlag, 1998.

[75] RM Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 43(4):85–103, 1972.

[76] W. Kern and D. Paulusma. The computational complexity of the elimination problem in generalized sports competitions. *Discrete Optimization*, 1(2):201–214, 2004.

[77] Dave Kiffer. Tie-breakers. SitNews, `http://www.sitnews.us/DaveKiffer/100705_kiffer.html`, October 2005.

[78] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005.

[79] J. Lang. Vote and aggregation in combinatorial domains with structured preferences. In *Proceedings of the 20nd International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007.

[80] J. Lang, M.S. Pini, F. Rossi, D. Salvagnin, K.B. Venable, and T. Walsh. Winner determination in voting trees with incomplete preferences and weighted votes. *Autonomous Agents and Multi-Agent Systems*, 25(1):130–157, 2011.

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

[82] J. Lang and L. Xia. Sequential composition of voting rules in multi-issue domains. *Mathematical Social Sciences*, 57(3):304–324, 2009.

[83] Steven Levitt. How do senators vote? disentangling the role of voter preferences, party affiliation, and senator ideology. *American Economic Review*, 86(3):425–441, 1996.

[84] R.D. Luce and H. Raiffa. *Games and decisions: Introduction and critical survey*. Dover, 1989.

[85] T. Magrino, R. Rivest, E. Shen, and D. Wagner. Computing the margin of victory in IRV elections. In *Proceedings of the 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE11)*, 2011.

[86] N. Mattei. Empirical evaluation of voting rules with strictly ordered preference data. In *Proceedings of the 2nd International Conference on Algorithmic Decision Theory (ADT 2011)*, 2011.

[87] N. Mattei, J. Goldsmith, and A. Klapper. On the complexity of bribery and manipulation in tournaments with uncertain information. In *Proceedings of the 25th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2012)*, 2012.

[88] N. Mattei, M. S. Pini, F. Rossi, and K. B. Venable. Bribery in voting over combinatorial domains is easy. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, 2012. Extended Abstract.

[89] N. Mattei, M. S. Pini, F. Rossi, and K. B. Venable. Bribery in voting over combinatorial domains is easy. In *12th International Symposium on Artificial Intelligence and Mathematics (ISAIM-12) Special Session on Computational Social Choice*, 2012.

[90] N. Maudet, M. S. Pini, F. Rossi, and K. B. Venable. Influencing and aggregating agents' preferences over combinatorial domains. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011) Workshop on Social Choice and Artificial Intelligence*, 2011.

[91] K. May. A set of independent necessary and sufficient conditions for simple majority decisions. *Econometrica*, 20(4):680–684, 1952.

[92] I. McLean and A. Urken. *Classics of Social Choice*. University of Michigan Press, 1995.

[93] S. Merrill, III. A comparison of efficiency of multicandidate electoral systems. *American Journal of Political Science*, 28(1):23 – 48, 1984.

[94] M. Mowbray and D. Gollmann. Electing the Doge of Venice: Analysis of a 13th century protocol. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF 2007)*, pages 295–310, 2007.

[95] R. G. Niemi. The occurrence of the paradox of voting in university elections. *Public Choice*, 8(1):91–100, 1970.

[96] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[97] H. Nurmi. Voting procedures: A summary analysis. *British Journal of Political Science*, 13(2):181 – 208, 1983.

[98] S. Obraztsova, E. Elkind, and N. Hazon. Ties matter: Complexity of voting manipulation revisited. In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, 2011.

[99] C. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, Inc., 1994.

[100] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover, 1998.

[101] M.S. Pini. *Reasoning with Preferences and Uncertainty*. PhD thesis, University of Padova, 2007.

[102] K. Poole and H. Rosenthal, editors. *Congress: A political-economic history of roll call voting*. Oxford University Press, 1997.

[103] G. Pritchard and A. Slinko. On the average minimum size of a manipulating coalition. *Social Choice and Welfare*, 27(2):263–277, 2006.

[104] A. D. Procaccia. *Computational Voting Theory: Of the Agents, By the Agents, For the Agents*. PhD thesis, The Hebrew University of Jerusalem, 2008.

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

[106] Ariel D. Procaccia. Computational social choice: The first four centuries. *ACM Crossroads*, 18(2):31–34, 2011.

[107] K.A. Raaflaub, J. Ober, and R. W. Wallace. *Origins of Democracy in Ancient Greece*. University of California Press, 2007.

[108] M. Regenwetter, B. Grogman, A. A. J. Marley, and I. M. Testlin. *Behavioral Social Choice: Probabilistic Models, Statistical Inference, and Applications*. Cambridge Univ. Press, 2006.

[109] M. Regenwetter, A. Kim, A. Kantor, and M. R. Ho. The unexpected empirical consensus among consensus methods. *Psychological Science*, 18(7):629 – 635, 2007.

[110] J. Reinganum. A formal theory of lobbying behaviour. *Optimal Control Applications and Methods*, 4(4):71–84, 1983.

[111] R. L. Rivest and E. Shen. An optimal single-winner preferential voting system based on game theory. In *Proceedings of the 3rd International Workshop on Computational Social Choice (COMSOC 2010)*, 2010.

[112] S. Roman. *Coding and Information Theory*. Springer, 1992.

[113] T. Russell and T. Walsh. Manipulating tournaments in cup and round robin competitions. In *Proceedings of the 1st International Conference on Algorithmic Decision Theory (ADT 2009)*, 2009.

[114] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, 2002.

[115] M. Satterthwaite. Strategy-proofness and Arrow's Conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–216, 1975.

[116] I. Schlotter, P. Faliszewski, and E. Elkind. Campaign management under approval-driven voting rules. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI 2011)*, 2011.

[117] A. K. Sen. A possibility theorem on majority decisions. *Econometrica*, 34(2):491–499, 1966.

[118] L.S. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48:787–792, 1954.

[119] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[120] M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, 2006.

[121] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7(5):16–27, 2000.

[122] A. D. Taylor. *Social Choice and the Mathematics of Manipulation*. Cambridge University Press, 2005.

[123] N. Tideman. *Collective Decisions and Voting: The Potential for Public Choice*. Ashgate Publishing, 2006.

[124] N. Tideman and F. Plassmann. Modeling the outcomes of vote-casting in actual elections. In D.S. Felsenthal and M. Machover, editors, *Electoral Systems: Paradoxes, Assumptions, and Procedures*. Springer, 2012.

[125] J. Uckelman. *More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains*. PhD thesis, University of Amsterdam, 2009.

[126] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[127] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[128] T. Vu, A. Altman, and Y. Shoham. On the complexity of schedule control problems for knockout tournaments. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, 2009.

[129] T. Walsh. Where are the really hard manipulation problems? The phase transition in manipulating the veto rule. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 2009.

[130] T. Walsh. An empirical study of the manipulability of single transferable voting. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 2010.

[131] T. Walsh. Where are the hard manipulation problems? In V. Conitzer and J. Rothe, editors, *Proceedings of the 3rd International Workshop on Computational Social Choice (COMSOC 2010)*, 2010.

[132] V.V. Williams. Fixing a tournament. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[133] Vote World. The international legislative roll-call voting website. `http://voteworld.berkeley.edu/index.html`. Last accessed, 4/17/2012.

[134] L. Xia. *Computational Voting Theory: Game-Theoretic and Combinatorial Aspects*. PhD thesis, Duke University, 2011.

[135] L. Xia and V. Conitzer. Generalized scoring rules and the frequency of coalitional manipulability. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC 2008)*, 2008.

[136] L. Xia and V. Conitzer. Strategy-proof voting rules over multi-issue domains with restricted preferences. In *Proceedings of the 6th International Workshop on Internet and Network Economics (WINE 2010)*, 2010.

[137] L. Xia, V. Conitzer, and J. Lang. Voting on multi-attribute domains with cyclic preferential dependencies. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, 2008.

[138] L. Xia, V. Conitzer, and J. Lang. Strategic sequential voting in multi-issue domains and multiple-election paradoxes. In *Proceedings of the 12th ACM Conference on Electronic Commerce (EC 2011)*, 2011.

[139] L. Xia, M. Zuckerman, A. Procaccia, V. Conitzer, and J. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 348–353, 2009.

Vita

Name: Nicholas Scott Mattei, _____
Date and Place of Birth: November 8th, 1983, Louisville, Kentucky, USA.

Education

- M.S., Computer Science,
  University of Kentucky, May 2010.
- B.S., Computer Engineering,
  Magna cum Laude, University of Kentucky, August 2006.
- B.S., Computer Science, Honors Program, Minor in Mathematics,
  Magna cum Laude, University of Kentucky, May 2006.

Employment

2010 – Current, Research Assistant, Department of Computer Science, University of Kentucky, Lexington, KY.

2012 – Current, Instructor, Bluegrass Community and Technical College, Lexington, KY.

2011, Visiting Researcher, University of Padova, Padova, Italy.

2009 – 2010, Teaching Assistant, Department of Computer Science, University of Kentucky, Lexington, KY.

2007–2011, Aerospace Technologist, NASA Ames Research Center, Moffett Field, CA.

2008, Research Assistant, Department of Computer Science, University of Kentucky, Lexington, KY.

2006 – 2007, Teaching Assistant, Department of Computer Science, University of Kentucky, Lexington, KY.

2005, Undergraduate Student Research Program Intern, NASA Ames Research Center, Moffett Field, CA.

2005 – 2006, UK/CISCO Sponsored Work-Study Program Intern, University of Kentucky Network Engineering and Operations Management Center, Lexington, KY.

2004, LERCIP Intern, NASA Glenn Research Center, Cleveland, OH.

Honors and Awards

- Co-Author with Dr. Judy Goldsmith (PI) on NSF-EAGER Grant: "Changing Minds, Changing Probabilities."
- 2012 University of Kentucky Myrle E. and Verle D. Nietzel Visiting Distinguished Faculty Program Award which includes funding to include Dr. Francesca Rossi on my dissertation committee.
- Selected to attend the 2011 Intl. Joint Conference on Artificial Intelligence Doctoral Consortium which included funding from the NSF for an extended research visit with Dr. Francesca Rossi in Padova, Italy, Summer 2011.
- 2011 University of Kentucky Provost's Award for Outstanding Teaching, Finalist.
- 2010 Northern Kentucky Alumni Association Fellowship.
- 2010 NASA Ames Honor Award for Outstanding Team (Nanosatellite Team).
- 2010 University of Kentucky Department of Computer Science ACM/UPE Outstanding Teaching Assistant Award Honorable Mention.
- 2008 University of Kentucky Department of Computer Science ACM/UPE Outstanding Teaching Assistant Award.
- Full Academic Scholarship to the University of Kentucky (2002–2006).
- University of Kentucky, College of Engineering Academic Scholarship (2002–2006).
- Member and Graduate Student Board Representative for the UK Chapter of the ACM (Previously held: President: 2007–2008; Treasurer: 2006–2007; Undergraduate Representative: 2005–2006; Graduate Representative: 2010–2011)
- Member of the Tau Beta Pi Engineering Honors Society
- Member of the Upsilon Pi Epsilon CS Honors Society (Secretary 2004–2008)
- Member of AAAI and the ACM.
- 2001 Kentucky Governors Scholar

Publications

Refereed Conferences and Workshops

1. Nicholas Mattei, Judy Goldsmith, and Andrew Klapper, "On the Complexity of Bribery and Manipulation in Tournaments with Uncertain Information," *Proc. 25th Intl. Florida Artificial Intelligence Research Society Conference (FLAIRS 2012)*, June 2012.

2. Nicholas Mattei, Maria Silvia Pini, Francesca Rossi, K. Brent Venable, "Bribery in Voting Over Combinatorial Domains Is Easy,"

   - *Proc. 11th Intl. Conference on Autonomous Agents and Multiagent Systems (AAMAS-12, short paper)*, June 2012.
   - *12th Intl. Symposium on Artificial Intelligence and Mathematics (ISAIM-12), Special Session on Computational Social Choice*, January 2012.

3. Nicholas Mattei, "Empirical Evaluation of Voting Rules with Strictly Ordered Preference Data," *Proc. 2nd Intl. Conference on Algorithmic Decision Theory (ADT-11)*, October, 2011.

4. Thomas Dodson, Nicholas Mattei, and Judy Goldsmith, "A Natural Language Argumentation Interface for Explanation Generation in Markov Decision Processes,"

   - *Proc. 2nd Intl. Conference on Algorithmic Decision Theory (ADT-11)*, October, 2011.
   - *Proc. 6th Workshop on Explanation Aware Computing (EXACT-11)*, July 2011.

5. Judy Goldsmith and Nicholas Mattei, "Science Fiction as an Introduction to AI Research," *2nd Symposium on Educational Advances in Artificial Intelligence (EAAI-11),* August 2011.

6. Nicholas Mattei, "Decision Making Under Uncertainty: Social Choice and Manipulation," *Proc. 22nd Intl. Joint Conference on Artificial Intelligence (IJCAI-11, extended abstract)*, July 2011.

7. Gábor Erdélyi, Henning Fernau, Judy Goldsmith, Nicholas Mattei, Daniel Raible and Jörg Rothe, "The Complexity of Probabilistic Lobbying," *Proc. 1st Intl. Conference on Algorithmic Decision Theory (ADT-09)*, October, 2009.

   - Also available as: Technical Report arXiv:0906.4431v3 [cs.CC], *ACM Computing Research Repository (CoRR)*, 27 pages, June 2009. Revised, November 2009.

8. Aaron Swank, Stevan Spremo, Nicholas Mattei, David Bui and Pete Klupar, "COTSAT: Small Spacecraft Cost Optimization for Government and Commercial Use" *AIAA SPACE 2009 Conference and Exposition*, August, 2009.

9. Jesse Bregman, Christopher Dallara, Shakib Ghassemieh, James Hanratty, Evan Jackson, Chris Kitts, Pete Klupar, Michael Lindsay, Ignacio Mas, Nicholas Mattei, David Mayer, Emmett Quigley, Mike Rasay, Aaron Swank, Adam Talcott, Jeroen Vandersteen, Zion Young, "Low Cost Rapid Response Spacecraft, (LCRSS): A Case Study in Small Satellite Cost Optimization for Government and Commercial Use," *AIAA SPACE 2008 Conference and Exposition*, August, 2008.

Refereed by Abstract:

1. Shakib Ghassemieh, Nicholas Mattei, Gregory Defouw, Michael J. McIntyre, Millan Diaz-Aguado, "Nanosatellite Payload Software: Managing Full Space Science Missions," *Small Satellite Systems and Services Symposium*, June 2010.

2. Michael C. Lindsay, Peter Klupar, Stevan Spremo, Aaron Swank, David Bui, Evan Jackson, Nicholas Mattei, David Mayer, Emmett Quigley, Zion Young, "COTSAT-1: Cost Optimization and Technology Enablement on a Small Spacecraft Platform,"

    - *NATO Research and Technology Organization Conference: RTO-MP-AVT-171 Multifunctional Structures and System Technologies for Small Spacecraft*, May, 2010.
    - *Small Satellite Systems and Services Symposium*, June 2010.

Patents:

1. Christopher Dallara, Stevan Spremo, Nicholas Mattei, "Low Burden Star Tracker," *NASA Ames Research Center ARC-16289-1,* Submitted to ARC patent office.

Invited Talks and Poster Presentations:

1. "Influence, Bribery, and Manipulation in Voting Systems: Current Results and Ongoing Work."

    - University of Padova, Italy (June 2011).
    - Capital Area Theory Seminar (CATS), University of Maryland, College Park (Feb 2012).
    - Mississippi State University (March 2012).

2. "Attitudes and Expectations of Undergraduate Students in STEM and non-STEM Disciplines." Joshua T. Guerin, Nicholas Mattei, and Thomas Dodson, Poster Presentation, 3rd Annual University of Kentucky STEM Education Symposium, Feb. 2012.

3. Nicholas Mattei, "Empirical Evaluation of Voting Rules with Strictly Ordered Preference Data," *1st Intl. Conference on Comparative Decision Making Studies,* May 2011.

4. Thomas Dodson, Nicholas Mattei, and Judy Goldsmith, "A Natural Language Argumentation Interface for Explanation Generation in Markov Decision Processes," *1st Intl. Conference on Comparative Decision Making Studies,* May 2011.

5. "Academic Advising: Automatically Generating Convincing Explanations of Recommendations from Complex Models"

- Eastern Kentucky University 25th Annual Symposium in the Mathematics, Statistical, and Computer Sciences (April 2011).
- University of Texas at Austin (August 2011).

6. "Decision Making Under Uncertainty: Social Choice and Manipulation"
   - Mid-West Theory Day, Indiana University (May 2010).
   - Comparative Decision Making Seminar, University of Kentucky (May 2010).
7. Nicholas Mattei, Judy Goldsmith, Gábor Erdélyi and Jörg Rothe, "The Complexity of Lobbying in an Uncertain World," *Poster, AAAI Workshop on Advances in Preference Handling,* 2008.
8. Nicholas Mattei, "Wireless Emergency Stop for the K-10 Lunar Rover," *NASA Ames Research Center USRP Poster Day,* 2005.
9. Nicholas Mattei, "Real-Time Linux Testing and Results," *NASA Glenn Research Center LERCIP Presentation Day,* 2004.

Working Papers

1. Gábor Erdélyi, Henning Fernau, Judy Goldsmith, Nicholas Mattei, Daniel Raible and Jörg Rothe, "The Complexity of Probabilistic Lobbying." Expanded version under review for journal.
2. Christopher R. Stieha, Nathaniel T. Wheelwright, and Nicholas Mattei "The Effects of Asymmetrical Information on Mating Decisions." In final preparation for journal submission.
3. Thomas Dodson, Nicholas Mattei, Joshua T. Guerin, and Judy Goldsmith, "A Natural Language Argumentation Interface for Explanation Generation in Markov Decision Processes." Expanded version under review for journal.
4. Nicholas Mattei, "Empirical Evaluation of Voting Rules with Strictly Ordered Preference Data." Finalizing expanded version for journal submission.
5. Nicholas Mattei, "A Longitudinal Study of Alternative Teaching Methods in Computer Science Education." Finalizing revisions for submission to conference.
6. Joshua T. Guerin, Nicholas Mattei, Robert Crawford, and Judy Goldsmith, "Constructing a Dynamic Bayes Net Model of Academic Advising With Collaborative Filtering." Under review for journal.
7. Nicholas Mattei, Thomas Dodson, Joshua T. Guerin, and Judy Goldsmith. "Attitudes and Expectations of Undergraduate Students in STEM and non-STEM Disciplines." In preparation.
8. Judy Goldsmith and Nicholas Mattei, "Science Fiction as an Introduction to AI Research." Finalizing follow-up study and preparing extended version for journal submission.