Resolving the Complexity of Some Fundamental Problems in Computational Social Choice

A THESIS

SUBMITTED FOR THE DEGREE OF

Doctor of Philosophy

IN THE

Faculty of Engineering

BY

Palash Dey

Computer Science and Automation

Indian Institute of Science

Bangalore − 560 012 (INDIA)

August 2016

DEDICATED TO

My teachers

Acknowledgements

Abstract

In many real world situations, especially involving multiagent systems and artificial intelligence, participating agents often need to agree upon a common alternative even if they have differing preferences over the available alternatives. Voting is one of the tools of choice in these situations. Common and classic applications of voting in modern applications include collaborative filtering and recommender systems, metasearch engines, coordination and planning among multiple automated agents etc. Agents in these applications usually have computational power at their disposal. This makes the study of computational aspects of voting crucial. This thesis is devoted to a study of computational complexity of several fundamental algorithmic and complexity-theoretic problems arising in the context of voting theory.

The typical setting for our work is an "election"; an election consists of a set of voters or agents, a set of alternatives, and a voting rule. The vote of any agent can be thought of as a ranking (more precisely, a complete order) of the set of alternatives. A voting profile comprises a collection of votes of all the agents. Finally, a voting rule is a mapping that takes as input a voting profile and outputs an alternative, which is called the "winner" or "outcome" of the election. Our contributions in this thesis can be categorized into three parts and are described below.

Part I: Preference Elicitation. In the first part of the thesis, we study the problem of eliciting the preferences of a set of voters by asking a small number of comparison queries (such as who a voter prefers between two given alternatives) for various interesting domains of preferences.

We commence with considering the domain of single peaked preferences on trees in Chapter 3. This domain is a significant generalization of the classical well studied domain of single peaked preferences. The domain of single peaked preferences and its generalizations are

hugely popular among political and social scientists. We show tight dependencies between query complexity of preference elicitation and various parameters of the single peaked tree, for example, number of leaves, diameter, path width, maximum degree of a node etc.

We next consider preference elicitation for the domain of single crossing preference profiles in Chapter 4. This domain has also been studied extensively by political scientists, social choice theorists, and computer scientists. We establish that the query complexity of preference elicitation in this domain crucially depends on how the votes are accessed and on whether or not any single crossing ordering is a priori known.

Part II: Winner Determination. In the second part of the thesis, we undertake a study of the computational complexity of several important problems related to determining winner of an election.

We begin with a study of the following problem: Given an election, predict the winner of the election under some fixed voting rule by sampling as few votes as possible. We establish optimal or almost optimal bounds on the number of votes that one needs to sample for many commonly used voting rules when the margin of victory is at least εn (n is the number of voters and ε is a parameter). We next study efficient sampling based algorithms for estimating the margin of victory of a given election for many common voting rules. The margin of victory of an election is a useful measure that captures the robustness of an election outcome. The above two works are presented in Chapter 5.

In Chapter 6, we design an optimal algorithm for determining the plurality winner of an election when the votes are arriving one-by-one in a streaming fashion. This resolves an intriguing question on finding heavy hitters in a stream of items, that has remained open for more than 35 years in the data stream literature. We also provide near optimal algorithms for determining the winner of a stream of votes for other popular voting rules, for example, veto, Borda, maximin etc.

Voters' preferences are often partial orders instead of complete orders. This is known as the incomplete information setting in computational social choice theory. In an incomplete information setting, an extension of the winner determination problem which has been studied extensively is the problem of determining possible winners. We study the kernelization complexity (under the complexity-theoretic framework of parameterized complexity) of the possible winner problem in Chapter 7. We show that there do not exist kernels of size that is polynomial in the number of alternatives for this problem for commonly used voting rules

under a plausible complexity theoretic assumption. However, we also show that the problem of coalitional manipulation which is an important special case of the possible winner problem admits a kernel whose size is polynomially bounded in the number of alternatives for common voting rules.

Part III: Election Control.   In the final part of the thesis, we study the computational complexity of various interesting aspects of strategic behavior in voting.

First, we consider the impact of partial information in the context of strategic manipulation in Chapter 8. We show that lack of complete information makes the computational problem of manipulation intractable for many commonly used voting rules.

In Chapter 9, we initiate the study of the computational problem of detecting possible instances of election manipulation. We show that detecting manipulation may be computationally easy under certain scenarios even when manipulation is intractable.

The computational problem of bribery is an extensively studied problem in computational social choice theory. We study computational complexity of bribery when the briber is "frugal" in nature. We show for many common voting rules that the bribery problem remains intractable even when the briber's behavior is restricted to be frugal, thereby strengthening the intractability results from the literature. This forms the subject of Chapter 10.

Publications based on this Thesis

1. *Palash Dey*, Neeldhara Misra, and Yadati Narahari. *"Kernelization Complexity of Possible Winner and Coalitional Manipulation Problems in Voting"*. In *Theoretical Computer Science*, volume 616, pages 111-125, February 2016.
   Remark: A preliminary version of this work was presented in the 14th *International Conference on Autonomous Systems and Multiagent Systems (AAMAS-15), 2015*.

2. *Palash Dey*, Neeldhara Misra, and Yadati Narahari. *"Frugal Bribery in Voting"*. Accepted in *Theoretical Computer Science*, March 2017.
   Remark: A preliminary version of this work was presented in the 30th *AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.

3. Arnab Bhattacharyya, *Palash Dey*, and David P. Woodruff. *"An Optimal Algorithm for ℓ_1-Heavy Hitters in Insertion Streams and Related Problems"*. Proceedings of the 35th *ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-16)*, pages 385-400, San Francisco, USA, 26 June - 1 July, 2016.

4. *Palash Dey* and Neeldhara Misra. *"Elicitation for Preferences Single Peaked on Trees"*. Proceedings of the 25th *International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 215-221, New York, USA, 9-15 July, 2016.

5. *Palash Dey* and Neeldhara Misra. *"Preference Elicitation For Single Crossing Domain"*. Proceedings of the 25th *International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 222-228, New York, USA, 9-15 July, 2016.

6. *Palash Dey*, Neeldhara Misra, and Yadati Narahari. *"Complexity of Manipulation with Partial Information in Voting"*. Proceedings of the 25th *International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 229-235, New York, USA, 9-15 July, 2016.

7. *Palash Dey*, Neeldhara Misra, and Yadati Narahari. *"Frugal Bribery in Voting"*. Proceedings of the 30$^{\text{th}}$ *AAAI Conference on Artificial Intelligence (AAAI-16)*, vol. 4, pages 2466-2672, Phoenix, Arizona, USA, 2016.

8. *Palash Dey* and Yadati Narahari. *"Estimating the Margin of Victory of Elections using Sampling"*. Proceedings of the 24$^{\text{th}}$ *International Joint Conference on Artificial Intelligence (IJCAI-15)*, pages 1120-1126, Buenos Aires, Argentina, 2015.

9. *Palash Dey* and Arnab Bhattacharyya. *"Sample Complexity for Winner Prediction in Elections"*. Proceedings of the 14$^{\text{th}}$ *International Conference on Autonomous Systems and Multiagent Systems (AAMAS-15)*, pages 1421-1430, Istanbul, Turkey, 2015.

10. *Palash Dey*, Neeldhara Misra, and Yadati Narahari. *"Detecting Possible Manipulators in Elections"*. Proceedings of the 14$^{\text{th}}$ *International Conference on Autonomous Systems and Multiagent Systems (AAMAS-15)*, pages 1441-1450, Istanbul, Turkey, 2015.

11. *Palash Dey*, Neeldhara Misra, and Yadati Narahari *"Kernelization Complexity of Possible Winner and Coalitional Manipulation Problems in Voting"*. Proceedings of the 14$^{\text{th}}$ *International Conference on Autonomous Systems and Multiagent Systems (AAMAS-15)*, pages 87-96, Istanbul, Turkey, 2015.

Contents

II   Winner Determination   **67**

5   Winner Prediction and Margin of Victory Estimation   **69**

List of Figures

List of Tables

Chapter 1

Introduction

> In this chapter, we provide an informal introduction to the area called social choice theory and motivate its computational aspects. We then provide an overview of the thesis along with further motivations for the problems we study in this thesis.

People always have different opinions on almost everything and hence they often have to come to a decision collectively on things that affect all of them. There are various ways for people with different opinions to come to a decision and the use of *voting* is the most natural and common method for this task. Indeed, the use of voting can be traced back to Socrates (470-399 BC) who was sentenced to death by taking a majority vote. We also see the mention of voting rules in Plato's (424-348 BC) unfinished book *"Laws"*. Then, in the Middle Ages, the Catalan philosopher, poet, and missionary Ramon Llull (1232-1316) mentioned Llull voting in his manuscripts Ars notandi, Ars eleccionis, and Alia ars eleccionis, which were lost until 2001. The German philosopher, theologian, jurist, and astronomer Nicholas of Cusa (1401-1464) suggested the method for electing the Holy Roman Emperor in 1433. The first systematic study of the theory of voting was carried out by two French mathematicians, political scientists, and philosophers Marie Jean Antoine Nicolas de Caritat, marquis de Condorcet (1743-1794) and Jean-Charles, chevalier de Borda (1733-1799). A more involved study of voting was pioneered by Kenneth Arrow (a co-recipient of the *Nobel Memorial Prize in Economics* in 1972). The celebrated *Arrow's impossibility theorem* in 1951 along with *Gibbard-Satterthwaite's impossibility theorem* in 1973 and 1977 laid the foundation stone of modern voting theory.

1.1 Important Aspects of Voting

In a typical setting for voting, we have a set of alternatives or *candidates*, a set of agents called *voters* each of whom has an ordinal preference called *votes* over the candidates, and a *voting rule* which chooses one or a set of candidates as *winner(s)*. We call a set of voters along with their preferences over a set of candidates and a voting rule an *election*. In Table 1.1 we exhibit an election with a set of voters {Satya, Shweta, Swapnil, Divya, Ganesh} and a set of five ice cream flavors {Chocolate, Butterscotch, Pesta, Vanilla, Kulfi} as a set of candidates. Suppose we use the plurality voting rule which chooses the candidate that is placed at the first positions of the votes most often as winner. Then the winner of the plurality election in Table 1.1 is Chocolate. Plurality voting rule is among the simplest voting rules and only observes the first position of every vote to determine the winner. Mainly due to its simplicity, the plurality voting rule frequently finds its applications in voting scenarios where human beings are involved, for example, political elections. Indeed many countries including India, Singapore, United Kingdom, Canada use plurality voting at various levels of elections [plu].

Voters	Preferences
Satya	Chocolate $\succ$ Kulfi $\succ$ Butterscotch $\succ$ Vanilla $\succ$ Pesta
Shweta	Butterscotch $\succ$ Kulfi $\succ$ Chocolate $\succ$ Vanilla $\succ$ Pesta
Swapnil	Pesta $\succ$ Butterscotch $\succ$ Kulfi $\succ$ Vanilla $\succ$ Chocolate
Divya	Chocolate $\succ$ Vanilla $\succ$ Kulfi $\succ$ Pesta $\succ$ Butterscotch
Ganesh	Kulfi $\succ$ Butterscotch $\succ$ Chocolate $\succ$ Vanilla $\succ$ Pesta

Table 1.1: Example of an election. Chocolate is the winner if the plurality voting rule is used. Kulfi is the winner if the Borda voting rule is used. Butterscotch is the Condorcet winner.

However, there are more sophisticated voting rules which takes into account the entire preference ordering of the voters to choose a winner. One such voting rule which finds its widespread use specially in sports and elections in many educational institutions is the Borda voting rule [bor]. The Borda voting rule, first proposed by Jean-Charles, chevalier de Borda in 1770, gives a candidate a score equal to the number of other candidates it is preferred over for every vote. In Table 1.1 for example, the candidate Kulfi receives a score of 3 from Satya, 3 from Shweta, 2 from Swapnil, 2 from Divya, and 4 from Ganesh. Hence the Borda score of Kulfi is 14. It can be verified that the Borda score of every other candidate is less than 14

which makes Kulfi the Borda winner of the election. We can already observe how two widely used and intuitively appealing voting rules select two different winners for the same set of votes.

A closer inspection of Table 1.1 reveals something peculiar about what the plurality and Borda voting rules choose as winners. The plurality voting rule chooses Chocolate as the winner whereas a majority (three out of five) of the voters prefer Butterscotch over Chocolate. The same phenomenon is true for the Borda winner (Kulfi) too — a majority of the voters prefer Butterscotch over Kulfi. More importantly Butterscotch is preferred over every other alternative (ice cream flavors) by a majority of the voters. Such an alternative, if it exists, which is preferred over every other alternative by a majority of the voters, is called the *Condorcet winner* of the election, named after the French mathematician and philosopher Marie Jean Antoine Nicolas de Caritat, marquis de Condorcet. We provide a more detailed discussion to all these aspects of voting including others in Chapter 2.

Now let us again consider the example in Table 1.1. Suppose Shweta misreports her preference as "Butterscotch ≻ Pesta ≻ Chocolate ≻ Vanilla ≻ Kulfi." Observe that the misreported vote of Shweta when tallied with the other votes makes Butterscotch the Borda winner with a Borda score of 12. Moreover, misreporting her preference results in an outcome (that is Butterscotch) which Shweta prefers over the honest one (that is Kulfi). This example demonstrate a fundamental notion in voting theory called *manipulation* — there can be instances where an individual voter may have an outcome by misreporting her preference which she prefers over the outcome of honest voting (according to her preference). Obviously manipulation is undesirable since it leads to suboptimal societal outcome. Hence one would like to use voting rules that do not create such possibilities of manipulation. Such voting rules are called *strategy-proof voting rules*. When we have only two candidates, it is known that the majority voting rule, which selects the majority candidate (breaking ties arbitrarily) as the winner, is a strategy-proof voting rule. There exist strategy-proof voting rules even if we have more than two candidates. For example, consider the voting rule $f_\mathfrak{v}$ that always outputs the candidate which is preferred most by some fixed voter say $\mathfrak{v}$. Obviously $f_\mathfrak{v}$ is strategy-proof since any voter other than $\mathfrak{v}$ cannot manipulate simply because they can never change the outcome of the election and the winner is already the most preferred candidate of $\mathfrak{v}$. However, voting rules like $f_\mathfrak{v}$ are undesirable intuitively because it is not performing the basic job of voting which is aggregating preferences of all the voters. The voting rule $f_\mathfrak{v}$ is called a *dictatorship* since it always selects the candidate preferred most by some fixed voter $\mathfrak{v}$; such

a voter v is called a *dictator*. Hence an important question to ask is the following: do there exist voting rules other than dictatorship that are strategy-proof with more than two candidates? Unfortunately classical results in social choice theory by Gibbard and Satterthwaite prove that there does not exist any such *onto* voting rule [Gib73, Sat75] if we have more than two candidates. A voting rule is onto if every candidate is selected for some set of votes. We formally introduce all these fundamental aspects of voting including others in Chapter 2.

1.2 Applications and Challenges of Voting Theory Relevant to Computer Science

With rapid improvement of computational power, the theory of voting found its applications not only in decision making among human beings but also in aggregating opinions of computational agents. Indeed, voting is commonly used whenever any system with multiple autonomous agents wish to take a common decision. Common and classic applications of voting in multiagent systems in particular and artificial intelligence in general include collaborative filtering and recommender systems [PHG00], planning among multiple automated agents [ER93], metasearch engines [DKNS01], spam detection [CSS99], computational biology [JSA08], winner determination in sports competitions [BBN14], similarity search and classification of high dimensional data [FKS03]. The extensive use of voting by computational agents makes the study of computational aspects of voting extremely important. In many applications of voting in artificial intelligence, one often has a huge number of voters and candidates. For example, voting has been applied in the design of metasearch engines [DKNS01] where the number of alternatives the agents are voting for is in the order of trillions if not more — the alternatives are the set of web pages in this case. In a commercial recommender system, for example Amazon.com, we often have a few hundred millions of users and items. In these applications, we need computationally efficient algorithms for quickly finding a winner of an election with so many candidates and voters.

Moreover, the very fact that the agents now have computational power at their disposal allows them to easily strategize their behavior instead of acting truthfully. For example, the problem of manipulation is much more severe in the presence of computational agents since the voters can easily use their computational power to find a manipulative vote. The field known as *computational social choice theory (COMSOC)* studies various computational aspects

in the context of voting. We refer the interested readers to a recent book on computational social choice theory for a more elaborate exposition [MBC+16].

Other than practical applications, the theory of voting turns out to be crucial in proving various fundamental theoretical results in computer science. Indeed, a notion known as *noise stability* of majority voting has been beautifully utilized for proving a lower bound on the quality of any polynomial time computable approximate solution of the maximum cut problem in a undirected graph [KKMO07, MOO05]. We refer interested reader to [O'D14] for a more detailed discussion.

1.3 Thesis Overview

In this thesis, we resolve computational questions of several fundamental problems arising in the context of voting theory. We provide a schematic diagram of the contribution of the thesis in Figure 1.1. We begin our study with the first step in any election system – eliciting the preferences of the voters. We consider the problem of preference elicitation in the first part of the thesis and develop (often) optimal algorithms for learning the preferences of a set of voters. In the second part of the thesis, we develop (often) optimal sublinear time algorithms for finding the winner of an election. The problem of determining winner of an election is arguably the most fundamental problem that comes to our mind once the preferences of all the voters have been elicited. In the third part of the thesis, we exhibit complexity-theoretic results for various strategic aspects of voting.

1.3.1 Part I: Preference Elicitation

The first step in any election system is to receive preferences of the voters as input. However, it may often be impractical to simply ask the voters for their preferences due to the presence of a huge number of alternatives. For example, asking users of Amazon.com to provide a complete ranking of all the items does not make sense. In such scenarios, one convenient way to know these agents' preferences is to ask the agents to compare (a manageable number of) pairs of items. However, due to the existence of a strong query complexity lower bound (from sorting), we can hope to reduce the burden of eliciting preference from the agents only by assuming additional structure on the preferences. Indeed, in several application scenarios commonly considered, it is rare that preferences are entirely arbitrary, demonstrating no
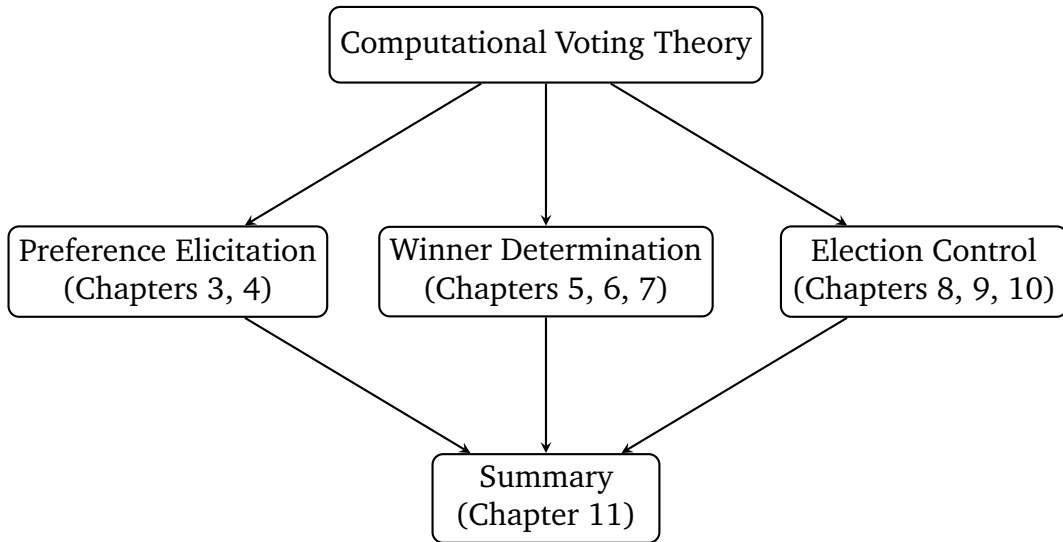
Figure 1.1: Structural overview of this dissertation

patterns whatsoever. For example, the notion of *single peaked preferences* forms the basis of several studies in the analytical political sciences. Intuitively, preferences of a set of agents is single peaked if the alternatives and the agents can be arranged in a linear order and the preferences "respect" this order. We defer the formal definition of single peaked profiles to Chapter 3.

We show in Chapter 3 and 4 that the query complexity for preference elicitation can be reduced substantially by assuming practically appealing restrictions on the domain of the preferences.

Chapter 3: Preference Elicitation for Single Peaked Preferences on Trees

Suppose we have n agents and m alternatives. Then it is known that if the preferences are single peaked, then elicitation can be done using $\Theta(mn)$ comparisons [Con09]. The notion of single peaked preferences has been generalized to single peaked preferences in a tree — intuitively, the preferences should be single peaked on every path of the tree. We show that if the single peaked tree has ℓ leaves, then elicitation can be done using $\Theta(mn \log \ell)$ comparisons [DM16a]. We also show that other natural parameters of the single peaked tree, for example, its diameter, path width, minimum degree, maximum degree do not decide the query complexity for preference elicitation.

Chapter 4: Preference Elicitation for Single Crossing Preferences

Another well studied domain is the domain of single crossing profiles. Intuitively, a set of preferences is single crossing if the preferences can be arranged in a linear order such that all the preferences who prefer an alternative x over another alternative y are consecutive, for every x and y. We consider in [DM16b] two distinct scenarios: when an ordering of the voters with respect to which the profile is single crossing is *known* a priori versus when it is *unknown*. We also consider two different access models: when the votes can be accessed at random, as opposed to when they arise in any arbitrary sequence. In the sequential access model, we distinguish two cases when the ordering is known: the first is that the sequence in which the votes appear is also a single-crossing order, versus when it is not. The main contribution of our work is to provide polynomial time algorithms with low query complexity for preference elicitation in all the above six cases. Further, we show that the query complexities of our algorithms are optimal up to constant factor for all but one of the above six cases.

1.3.2 Part II: Winner Determination

Once we have elicited the preferences of the voters, the next important task in an election is to determine the winner of this election. We show interesting complexity theoretic results on determining the winner of an election under a variety of application scenarios.

Chapter 5: Winner Prediction

We begin by studying the following problem: Given an election, predict the winner of the election under some fixed voting rule by sampling as few preferences as possible. Our results show that the winner of an election can be determined (with high probability) by observing only a few preferences that are picked uniformly at random from the set of preferences. We show this result for many common voting rules when the margin of victory is at least εn [DB15], where n is the number of voters and ε is a parameter. The margin of victory of an election is the smallest number of preferences that must be modified to change the winner of an election. We also establish optimality (in terms of the number of preferences our algorithms sample) of most of our algorithms by proving tight lower bounds on the sample complexity of this problem.

Another important aspect of an election is the robustness of the election outcome. A popular measure of robustness of an election outcome is its margin of victory. We develop

efficient sampling based algorithms for estimating the margin of victory of a given election for many common voting rules [DN15b]. We also show optimality of our algorithms for most of the cases under appealing practical scenarios.

Chapter 6: Winner Determination in Streaming

An important, realistic model of data in big data research is the streaming model. In the setting that we consider, the algorithms are allowed only one pass over the data. We give the first optimal bounds for returning the ℓ_1-heavy hitters in an insertion only data stream, together with their approximate frequencies, thus settling a long line of work on this problem. For a stream of m items in $\{1, 2, \ldots, n\}$ and parameters $0 < \varepsilon < \phi \leqslant 1$, let f_i denote the frequency of item i, i.e., the number of times item i occurs in the stream in [BDW16]. With arbitrarily large constant probability, our algorithm returns all items i for which $f_i \geqslant \phi m$, returns no items j for which $f_j \leqslant (\phi - \varepsilon)m$, and returns approximations $\tilde{f}_i$ with $|\tilde{f}_i - f_i| \leqslant \varepsilon m$ for each item i that it returns. Our algorithm uses $O(\varepsilon^{-1} \log \phi^{-1} + \phi^{-1} \log n + \log \log m)$ bits of space, processes each stream update in $O(1)$ worst-case time, and can report its output in time linear in the output size. We also prove a lower bound, which implies that our algorithm is optimal up to a constant factor in its space complexity. A modification of our algorithm can be used to estimate the maximum frequency up to an additive εm error in the above amount of space, resolving an open question on algorithms for data streams for the case of ℓ_1-heavy hitters. We also introduce several variants of the heavy hitters and maximum frequency problems, inspired by rank aggregation and voting schemes, and show how our techniques can be applied in such settings. Unlike the traditional heavy hitters problem, some of these variants look at comparisons between items rather than numerical values to determine the frequency of an item.

Chapter 7: Kernelization Complexity for Determining Possible Winners

Voters' preferences are often partial orders instead of complete orders. This is known as the incomplete information setting in computational social choice theory. In an incomplete information setting, an extension of the winner determination problem which has been studied extensively is the problem of determining possible winners. An alternative x is a possible winner in a set of incomplete preferences if there exists a completion of these incomplete preferences where the alternative x wins. Previous work has provided, for many common

voting rules, fixed parameter tractable algorithms for the *Possible winner* problem, with number of candidates as the parameter. However, the corresponding kernelization question is still open, and in fact, has been mentioned as a key research challenge [BCF$^+$14a]. In this work, we settle this open question for many common voting rules.

We show in [DMN15b, DMN16c] that the *Possible winner* problem for maximin, Copeland, Bucklin, ranked pairs, and a class of scoring rules that includes the Borda voting rule do not admit a polynomial kernel with the number of candidates as the parameter. We show however that the *Coalitional manipulation* problem which is an important special case of the *Possible winner* problem does admit a polynomial kernel for maximin, Copeland, ranked pairs, and a class of scoring rules that includes the Borda voting rule, when the number of manipulators is polynomial in the number of candidates. A significant conclusion of this work is that the *Possible winner* problem is computationally harder than the *Coalitional manipulation* problem since the *Coalitional manipulation* problem admits a polynomial kernel whereas the *Possible winner* problem does not admit a polynomial kernel.

1.3.3 Part III: Election Control

When agents in a multiagent system have conflicting goals, they can behave strategically to make the system produce an outcome that they favor. For example, in the context of voting, there can be instances where an agent, by misreporting her preference, can cause the election to result in an alternative which she prefers more than the alternative that would have been the outcome in case she had reported her preference truthfully. This phenomenon is called *manipulation*. Certainly, we would like to design systems that are robust to such manipulation. Unfortunately, classical results in social choice theory establish that any reasonable voting rule will inevitably create such opportunities for manipulation. In this context, computer scientists provide some hope by showing that the computational problem of manipulation can often be intractable. Indeed, if a computationally bounded agent has to solve an intractable problem to manipulate the election system, then the agent would, in all likelihood, fail to manipulate.

Chapter 8: Manipulation with Partial Information

The computational problem of manipulation has classically been studied in a complete information setting – the manipulators know the preference of every other voter. We extend

this line of work to the more practical, incomplete information setting where agents have only a partial knowledge about the preferences of other voters [DMN16a]. In our framework, the manipulators know a partial order for each voter that is consistent with the true preference of that voter. We say that an extension of a partial order is *viable* if there exists a manipulative vote for that extension. We propose the following notions of manipulation when manipulators have incomplete information about the votes of other voters.

1. WEAK MANIPULATION: the manipulators seek to vote in a way that makes their preferred candidate win in *at least one extension* of the partial votes of the non-manipulators.

2. OPPORTUNISTIC MANIPULATION: the manipulators seek to vote in a way that makes their preferred candidate win *in every viable extension* of the partial votes of the non-manipulators.

3. STRONG MANIPULATION: the manipulators seek to vote in a way that makes their preferred candidate win *in every extension* of the partial votes of the non-manipulators.

We consider several scenarios for which the traditional manipulation problems are easy (for instance, the Borda voting rule with a single manipulator). For many of them, the corresponding manipulative questions that we propose turn out to be computationally intractable. Our hardness results often hold even when very little information is missing, or in other words, even when the instances are very close to the complete information setting. Our overall conclusion is that computational hardness continues to be a valid obstruction or deterrent to manipulation, in the context of the more realistic setting of incomplete information.

Chapter 9: Manipulation Detection

Even if the computational problem of manipulation is almost always intractable, there have been many instances of manipulation in real elections. In this work, we initiate the study of the computational problem of detecting possible instances of manipulation in an election [DMN15a]. We formulate two pertinent computational problems in the context of manipulation detection - Coalitional Possible Manipulators (CPM) and Coalitional Possible Manipulators given Winner (CPMW), where a suspect group of voters is provided as input and we have to determine whether they can form a potential coalition of manipulators. In the absence of any suspect group, we formulate two more computational problems namely Coalitional Possible Manipulators Search (CPMS) and Coalitional Possible Manipulators Search

given Winner (CPMSW). We provide polynomial time algorithms for these problems, for several popular voting rules. For a few other voting rules, we show that these problems are NP-complete. We observe that detecting possible instances of manipulation may be easy even when the actual manipulation problem is computationally intractable, as seen for example, in the case of the Borda voting rule.

Chapter 10: Frugal Bribery

Another fundamental problem in the context of social choice theory is bribery. Formally, the computational problem of bribery is as follows: given (i) a set of votes, (ii) a cost model for changing the votes, (iii) a budget, and (iv) a candidate x, is it possible for an external agent to bribe the voters to change their votes (subject to the budget constraint) so that x wins the election? We introduce and study two important special cases of the classical $BRIBERY problem, namely, FRUGAL-BRIBERY and FRUGAL-$BRIBERY where the briber is frugal in nature in [DMN16b]. By this, we mean that the briber is only able to influence voters who benefit from the suggestion of the briber. More formally, a voter is *vulnerable* if the outcome of the election improves according to her own preference when she accepts the suggestion of the briber. In the FRUGAL-BRIBERY problem, the goal of the briber is to make a certain candidate win the election by changing *only* the vulnerable votes. In the FRUGAL-$BRIBERY problem, the vulnerable votes have prices and the goal is to make a certain candidate win the election by changing only the vulnerable votes, subject to a budget constraint. We further formulate two natural variants of the FRUGAL-$BRIBERY problem namely UNIFORM-FRUGAL-$BRIBERY and NONUNIFORM-FRUGAL-$BRIBERY where the prices of the vulnerable votes are, respectively, all same or different.

We observe that, even if we have only a small number of candidates, the problems are intractable for all voting rules studied here for weighted elections, with the sole exception of the FRUGAL-BRIBERY problem for the plurality voting rule. In contrast, we have polynomial time algorithms for the FRUGAL-BRIBERY problem for plurality, veto, k-approval, k-veto, and plurality with runoff voting rules for unweighted elections. However, the FRUGAL-$BRIBERY problem is intractable for all the voting rules studied here barring the plurality and the veto voting rules for unweighted elections. These intractability results demonstrate that bribery is a hard computational problem, in the sense that several special cases of this problem continue to be computationally intractable. This strengthens the view that bribery, although a possible attack on an election in principle, may be infeasible in practice.

We believe that the thesis work has attempted to break fresh ground by resolving the computational complexity of many long-standing canonical problems in computational social choice. We finally conclude in Chapter 11 with future directions of research.

Chapter 2

Background

This chapter provides an overview of selected topics required to understand the technical content in this thesis. We cover these topics, up to the requirement of following this thesis, in the following order: Voting and Elections, Computational Complexity, Probability Theory, Information Theory, Relevant Terminology for Trees, Universal Family of Hash Functions.

We present the basic preliminaries in this chapter. We denote, for any natural number $\ell \in \mathbb{N}$, the set $\{1, 2, \ldots, \ell\}$ by $[\ell]$. We denote the set of permutations of $[\ell]$ by $\mathfrak{S}_\ell$. For a set $\mathfrak{X}$, we denote the set of subsets of $\mathfrak{X}$ of size k by $\mathfrak{P}_k(\mathfrak{X})$ and the power set of $\mathfrak{X}$ by $2^{\mathfrak{X}}$.

2.1 Voting and Elections

In this section, we introduce basic terminologies of voting.

2.1.1 Basic Setting

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ be a set of candidates or alternatives and $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ a set of voters. If not mentioned otherwise, we denote the set of candidates by $\mathcal{C}$, the set of voters by $\mathcal{V}$, the number of candidates by m, and the number of voters by n. Every voter v_i has a preference or vote $\succ_i$ which is a complete order over $\mathcal{C}$. A complete order over any set $\mathfrak{X}$ is a relation on $\mathfrak{X}$ which is reflexive, transitive, anti-symmetric, and total. A relation $\mathcal{R}$ on $\mathfrak{X}$ is a subset of $\mathfrak{X} \times \mathfrak{X}$. A relation $\mathcal{R}$ is called reflexive if $(x, x) \in \mathcal{R}$ for every $x \in \mathfrak{X}$, transitive if $(x, y) \in \mathcal{R}$ and $(y, z) \in \mathcal{R}$ implies $(x, z) \in \mathcal{R}$ for every $x, y, z \in \mathfrak{X}$, anti-symmetric

is $(x, y) \in \mathcal{R}$ and $(y, x) \in \mathcal{R}$ implies $x = y$ for every $x, y \in \mathcal{X}$, and total is either $(x, y) \in \mathcal{R}$ or $(y, x) \in \mathcal{R}$ for every $x, y \in \mathcal{X}$. We denote the set of complete orders over $\mathcal{C}$ by $\mathcal{L}(\mathcal{C})$. We call a tuple of n preferences $(\succ_1, \succ_2, \cdots, \succ_n) \in \mathcal{L}(\mathcal{C})^n$ an n-voter preference profile. Given a preference $\succ = c_1 \succ c_2 \succ \cdots \succ c_m \in \mathcal{L}(\mathcal{C})$, we call the order $c_m \succ c_{m-1} \succ \cdots \succ c_2 \succ c_1$ the reverse order of $\succ$ and denote it by $\overleftarrow{\succ}$. We say that a candidate $x \in \mathcal{C}$ is at the ith position of a preference $\succ$ if there exists exactly $i - 1$ other candidates in $\mathcal{C}$ who are preferred over x in $\succ$, that is $|\{y \in \mathcal{C} \setminus \{x\} : y \succ x\}| = i - 1$. It is often convenient to view a preference as a subset of $\mathcal{C} \times \mathcal{C}$ — a preference $\succ$ corresponds to the subset $\mathcal{A} = \{(x, y) \in \mathcal{C} \times \mathcal{C} : x \succ y\}$. For a preference $\succ$ and a subset $\mathcal{A} \subseteq \mathcal{C}$ of candidates, we define $\succ (\mathcal{A})$ be the preference $\succ$ restricted to $\mathcal{A}$, that is $\succ (\mathcal{A}) = \succ \cap (\mathcal{A} \times \mathcal{A})$.

A voting correspondence is a function $\tilde{r} : \cup_{n \in \mathcal{N}} \mathcal{L}(\mathcal{C})^n \longrightarrow 2^{\mathcal{C}} \setminus \{\emptyset\}$ which selects, from a preference profile, a nonempty set of candidates as the winners. A tie breaking rule $\tau : 2^{\mathcal{C}} \setminus \{\emptyset\} \longrightarrow \mathcal{C}$ with $\tau(\mathcal{A}) \in \mathcal{A}$ for every $\mathcal{A} \in 2^{\mathcal{C}} \setminus \{\emptyset\}$ selects one candidate from a nonempty set of (tied) candidates. A voting rule $r : \cup_{n \in \mathcal{N}} \mathcal{L}(\mathcal{C})^n \longrightarrow \mathcal{C}$ selects one candidate as the winner from a preference profile. An important class of tie breaking rules is the class of *lexicographic* tie breaking rules where the ties are broken in a fixed complete order of candidates. More formally, given a complete order $\succ \in \mathcal{L}(\mathcal{C})$, we can define a tie breaking rule $\tau_{\succ}$ as follows: for every nonempty subset $A \subseteq \mathcal{C}$ of $\mathcal{C}$, $\tau_{\succ}(A) = x$ such that $x \in A$ and $x \succ y$ for every $y \in A \setminus \{x\}$. A natural way for defining a voting rule is to compose a voting correspondence $\tilde{r}$ with a tie breaking rule τ; that is $r = \tau \circ \tilde{r}$ is a voting rule. We call a preference profile over a set of candidates along with a voting rule an election. The winner (or winners) of an election is often called the outcome of the election.

2.1.2 Axioms of Voting Rules

Let us now look at important axiomatic properties of voting rules that one may wish to satisfy.

▷ *Onto:* A voting rule r is called onto if every candidate wins for at least one voting profile. More formally, a voting rule r is called onto if for every $x \in \mathcal{C}$ there exists a $(\succ_1, \succ_2, \ldots, \succ_n) \in \mathcal{L}(\mathcal{C})^n$ such that $r(\succ_1, \succ_2, \ldots, \succ_n) = x$.

▷ *Anonymity:* A voting rule is anonymous if the "names" of the voters does not affect the outcome. More formally, a voting rule r is said to satisfy anonymity if for every $(\succ_1, \succ_2, \ldots, \succ_n) \in \mathcal{L}(\mathcal{C})^n$, we have $r(\succ_1, \succ_2, \ldots, \succ_n) = r(\succ_{\sigma(1)}, \succ_{\sigma(2)}, \ldots, \succ_{\sigma(n)})$ for every permutation $\sigma \in \mathfrak{S}_n$.

▷ *Neutrality:* A voting rule is called neutral if the "names" of the candidates are immaterial for determining election outcome. That is, a voting rule r is called neutral if for every $(\succ_1, \succ_2, \ldots, \succ_n) \in \mathcal{L}(\mathcal{C})^n$, we have $r(\succ_1, \succ_2, \ldots, \succ_n) = \sigma(r(\sigma(\succ_1), \sigma(\succ_2), \ldots, \sigma(\succ_n)))$ for every permutation $\sigma \in \mathfrak{S}_m$. Given a preference $\succ = c_1 \succ c_2 \succ \cdots \succ c_m$ and a permutation $\sigma \in \mathfrak{S}_m$, we denote the preference $c_{\sigma(1)} \succ c_{\sigma(2)} \succ \cdots \succ c_{\sigma(m)}$ by $\sigma(\succ)$.

▷ *Homogeneity:* A voting rule is called homogeneous if the outcome of the election solely depends on the fraction of times (of the number of voters n) every complete order $\succ \in \mathcal{L}(\mathcal{C})$ appears in the preference profile.

▷ *Dictatorship:* A voting rule r is called a dictatorship if there exists an integer $i \in [n]$ such that $r(\succ_1, \succ_2, \ldots, \succ_n) = \succ_i(1)$ for every $(\succ_1, \succ_2, \ldots, \succ_n) \in \mathcal{L}(\mathcal{C})^n$, where $\succ_i(1)$ the candidate placed at the first position of the preference $\succ_i$.

▷ *Manipulability:* A voting rule r is called manipulable if there exists a preference profile $(\succ_1, \succ_2, \ldots, \succ_n) \in \mathcal{L}(\mathcal{C})^n$ and a preference $\succ \in \mathcal{L}(\mathcal{C})$ such that the following holds.

$$r(\succ_1, \succ_2, \ldots, \succ_{i-1}, \succ, \succ_{i+1}, \ldots, \succ_n) \succ_i r(\succ_1, \succ_2, \ldots, \succ_n)$$

That it, voter v_i prefers the election outcome if she reports $\succ$ to be her preference than the election outcome if she reports her true preference $\succ_i$.

▷ *Condorcet consistency:* A candidate is called the Condorcet winner of an election if it defeats every other candidate in pairwise election. More formally, given an election $\mathcal{E}$, let us define $N_{\mathcal{E}}(x, y) = |\{i : x \succ_i y\}|$ for any two candidates $x, y \in \mathcal{C}$. A candidate c is called the Condorcet winner of an election if $N_{\mathcal{E}}(c, x) > n/2$ for every candidate $x \in \mathcal{C} \setminus \{c\}$ other than c. A voting rule is called Condorcet consistent is it selects the Condorcet winner as the outcome of the election whenever such a candidate exists.

▷ *Weak Condorcet consistency:* A candidate is called a weak Condorcet winner of an election if it does not lose to any other candidate in pairwise election. More formally, given an election $\mathcal{E}$, let us define $N_{\mathcal{E}}(x, y) = |\{i : x \succ_i y\}|$ for any two candidates $x, y \in \mathcal{C}$. A candidate c is called the Condorcet winner of an election if $N_{\mathcal{E}}(c, x) \geqslant n/2$ for every candidate $x \in \mathcal{C} \setminus \{c\}$ other than c. A voting rule is called weak Condorcet consistent is it selects a weak Condorcet winner as the outcome of the election whenever one such candidate exists.

Given an election $\mathcal{E} = (\succ, \mathcal{C})$ and two candidates $x, y \in \mathcal{C}$, we say that the candidate x defeats the candidate y in *pairwise election* if $N_{\mathcal{E}}(x, y) > N_{\mathcal{E}}(y, x)$.

2.1.3 Majority Graph

Given an election $\mathcal{E} = (\succ = (\succ_1, \succ_2, \ldots, \succ_n), \mathcal{C})$, we can construct a weighted directed graph $\mathcal{G}_{\mathcal{E}} = (U = \mathcal{C}, E)$ as follows. The vertex set of the graph $\mathcal{G}_{\mathcal{E}}$ is the set of candidates $\mathcal{C}$. For any two candidates $x, y \in \mathcal{C}$ with $x \neq y$, let us define the margin $\mathcal{D}_{\mathcal{E}}(x, y)$ of x from y to be $N_{\mathcal{E}}(x, y) - N_{\mathcal{E}}(y, x)$. We have an edge from x to y in $\mathcal{G}_{\mathcal{E}}$ if $\mathcal{D}_{\mathcal{E}}(x, y) > 0$. Moreover, in that case, the weight $w(x, y)$ of the edge from x to y is $\mathcal{D}_{\mathcal{E}}(x, y)$. Observe that, a candidate c is the Condorcet winner of an election $\mathcal{E}$ if and only if there is an edge from c to every other vertices in the weighted majority graph $\mathcal{G}_{\mathcal{E}}$.

2.1.4 Condorcet Paradox

Given an election $\mathcal{E} = (\succ, \mathcal{C})$, there may exist candidates $c_1, c_2, \ldots, c_k \in \mathcal{C}$ for some integer $k \geqslant 3$ such that the candidate c_i defeats $c_{i+1 \bmod k}$ in pairwise election for every $i \in [k]$. This phenomenon was discovered by Marie Jean Antoine Nicolas de Caritat, marquis de Condorcet and is called Condorcet paradox or Condorcet cycles. Figure 2.1 shows an example of a Condorcet paradox where a defeats b, b defeats c, and c defeats a.

$$
\boxed{
\begin{array}{l}
a \succ b \succ c \\
b \succ c \succ a \\
c \succ a \succ b
\end{array}
}
$$

Figure 2.1: Condorcet paradox.

2.1.5 Incomplete Information Setting

A more general setting is an *election* where the votes are only *partial orders* over candidates. A *partial order* is a relation that is *reflexive, antisymmetric*, and *transitive*. A partial vote can be extended to possibly more than one linear votes depending on how we fix the order for the unspecified pairs of candidates. For example, in an election with the set of candidates $\mathcal{C} = \{a, b, c\}$, a valid partial vote can be $a \succ b$. This partial vote can be extended to three

linear votes namely, $a \succ b \succ c$, $a \succ c \succ b$, $c \succ a \succ b$. However, the voting rules always take as input a set of votes that are complete orders over the candidates.

2.1.6 Voting Rules

Examples of common voting correspondences are as follows. We can use any tie breaking rule, for example, any lexicographic tie breaking rule, with these voting correspondences to get a voting rule.

▷ **Positional scoring rules:** A collection of m-dimensional vectors $\overrightarrow{s_m} = (\alpha_1, \alpha_2, \ldots, \alpha_m) \in \mathbb{R}^m$ with $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_m$ and $\alpha_1 > \alpha_m$ for every $m \in \mathbb{N}$ naturally defines a voting rule — a candidate gets score α_i from a vote if it is placed at the ith position, and the score of a candidate is the sum of the scores it receives from all the votes. The winners are the candidates with maximum score. Scoring rules remain unchanged if we multiply every α_i by any constant $\lambda > 0$ and/or add any constant μ. Hence, we assume without loss of generality that for any score vector $\overrightarrow{s_m}$, there exists a j such that $\alpha_j - \alpha_{j+1} = 1$ and $\alpha_k = 0$ for all $k > j$. We call such a $\overrightarrow{s_m}$ a normalized score vector. A scoring rule is called *strict* if $\alpha_1 > \alpha_2 > \cdots > \alpha_m$ for every natural number m. If α_i is 1 for $i \in [k]$ and 0 otherwise, then we get the k-approval voting rule. For the k-veto voting rule, α_i is 0 for $i \in [m-k]$ and -1 otherwise. 1-approval is called the plurality voting rule and 1-veto is called the veto voting rule. For the Borda voting rule, we have $\alpha_i = m - i$ for every $i \in [m]$.

▷ **Bucklin and simplified Bucklin:** Let ℓ be the minimum integer such that at least one candidate gets majority within top ℓ positions of the votes. The winners under the simplified Bucklin voting rule are the candidates having more than $n/2$ votes within top ℓ positions. The winners under the Bucklin voting rule are the candidates appearing within top ℓ positions of the votes highest number of times. However, for brevity, other than Chapter 8, we use Bucklin to mean simplified Bucklin only.

▷ **Fallback and simplified Fallback:** For these voting rules, each voter v ranks a subset $\mathcal{X}_v \subset \mathcal{C}$ of candidates and disapproves the rest of the candidates [BS09]. Now for the Fallback and simplified Fallback voting rules, we apply the Bucklin and simplified Bucklin voting rules respectively to define winners. If there is no integer ℓ for which at least one candidate gets more than $n/2$ votes, both the Fallback and simplified Fallback

voting rules output the candidates with most approvals as winners. We assume, for simplicity, that the number of candidates each partial vote approves is known.

▷ **Maximin:** The maximin score of a candidate x in an election E is $\min_{y \neq x} D_E(x, y)$. The winners are the candidates with maximum maximin score.

▷ **Copeland$^\alpha$:** Given $\alpha \in [0, 1]$, the Copeland$^\alpha$ score of a candidate x is $|\{y \neq x : D_E(x, y) > 0\}| + \alpha|\{y \neq x : D_E(x, y) = 0\}|$. The winners are the candidates with maximum Copeland$^\alpha$ score. If not mentioned otherwise, we will assume α to be zero.

▷ **Ranked pairs:** Given an election E, we pick a pair $(c_i, c_j) \in \mathcal{C} \times \mathcal{C}$ such that $D_E(c_i, c_j)$ is maximum. We fix the ordering between c_i and c_j to be $c_i \succ c_j$ unless it contradicts previously fixed orders. We continue this process until all pairwise elections are considered. At this point, we have a complete order $\succ$ over the candidates. Now the top candidate of $\succ$ is chosen as the winner.

▷ **Plurality with runoff:** The top two candidates according to plurality score are selected first. The pairwise winner of these two candidates is selected as the winner of the election. This rule is often called the *runoff* voting rule.

▷ **Single transferable vote:** In single transferable vote (STV), a candidate with least plurality score is dropped out of the election and its votes are transferred to the next preferred candidate. If two or more candidates receive least plurality score, then some predetermined tie breaking rule is used. The candidate that remains after $(m - 1)$ rounds is the winner. The single transferable vote is also called the instant runoff vote.

We use the parallel-universes tie breaking [CRX09, BF12] to define the winning candidate for the ranked pairs voting rule. In this setting, a candidate c is a winner if and only if there exists a way to break ties in all of the steps such that c is the winner.

2.2 Computational Complexity

2.2.1 Parameterized Complexity

Preprocessing, as a strategy for coping with hard problems, is universally applied in practice. The main goal here is *instance compression* - the objective is to output a smaller instance

while maintaining equivalence. In the classical setting, NP-hard problems are unlikely to have efficient compression algorithms (since repeated application would lead to an efficient solution for the entire problem, which is unexpected). However, the breakthrough notion of *kernelization* in parameterized complexity provides a mathematical framework for analyzing the quality of preprocessing strategies. In parameterized complexity, each problem instance comes with a parameter k, and the central notion is *fixed parameter tractability* (FPT) which means, for a given instance (x, k), solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size $|x|$. We use the notation $O^*(f(k))$ to denote $O(f(k)\text{poly}(|x|))$. A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$, where Γ is a finite alphabet. An instance of a parameterized problem is a tuple (x, k), where k is the parameter. We refer the reader to the books [GN07, DF99, FG06] for a detailed introduction to this paradigm, and below we state only the definitions that are relevant to our work.

A *kernelization* algorithm is a set of preprocessing rules that runs in polynomial time and reduces the instance size with a guarantee on the output instance size. This notion is formalized below.

Definition 2.1. [Kernelization] *[Nie02, FG06] A kernelization algorithm for a parameterized problem $\Pi \subseteq \Gamma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Gamma^* \times \mathbb{N}$, outputs, in time polynomial in $|x| + k$, a pair $(x', k') \in \Gamma^* \times \mathbb{N}$ such that (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$ and (b) $|x'|, k' \leqslant g(k)$, where g is some computable function. The output instance x' is called the kernel, and the function g is referred to as the size of the kernel. If $g(k) = k^{O(1)}$, then we say that Π admits a polynomial kernel.*

For many parameterized problems, it is well established that the existence of a polynomial kernel would imply the collapse of the polynomial hierarchy to the third level (or more precisely, $\text{CoNP} \subseteq \text{NP/Poly}$). Therefore, it is considered unlikely that these problems would admit polynomial-sized kernels. For showing kernel lower bounds, we simply establish reductions from these problems.

Definition 2.2. [Polynomial Parameter Transformation] [BTY09] *Let Γ_1 and Γ_2 be parameterized problems. We say that Γ_1 is polynomial time and parameter reducible to Γ_2, written $\Gamma_1 \leqslant_{\text{Ptp}} \Gamma_2$, if there exists a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$, and a polynomial $p : \mathbb{N} \to \mathbb{N}$, and for all $x \in \Sigma^*$ and $k \in \mathbb{N}$, if $f((x, k)) = (x', k')$, then $(x, k) \in \Gamma_1$ if and only if $(x', k') \in \Gamma_2$, and $k' \leqslant p(k)$. We call f a polynomial parameter transformation (or a PPT) from Γ_1 to Γ_2.*

This notion of a reduction is useful in showing kernel lower bounds because of the following theorem.

Theorem 2.1. [BTY09, Theorem 3] *Let* P *and* Q *be parameterized problems whose derived classical problems are* P^c, Q^c, *respectively. Let* P^c *be* NP-*complete, and* $Q^c \in$ NP. *Suppose there exists a PPT from* P *to* Q. *Then, if* Q *has a polynomial kernel, then* P *also has a polynomial kernel.*

2.2.2 Approximation Factor of an Algorithm

For a minimization problem $\mathcal{P}$, we say an algorithm $\mathcal{A}$ archives an approximation factor of α if $\mathcal{A}(\mathcal{I}) \leqslant \alpha \mathrm{OPT}(\mathcal{I})$ for every problem instance $\mathcal{I}$ of $\mathcal{P}$. In the above, $\mathrm{OPT}(\mathcal{I})$ denotes the value of the optimal solution of the problem instance $\mathcal{I}$.

2.2.3 Communication Complexity

Communication complexity of a function measures the number of bits that need to be exchanged between two players to compute a function whose input is split among those two players [Yao79]. In a more restrictive *one-way communication model*, Alice, the first player, sends only one message to Bob, the second player, and Bob outputs the result. A protocol is a method that the players follow to compute certain functions of their input. Also the protocols can be randomized; in that case, the protocol needs to output correctly with probability at least $1-\delta$, for $\delta \in (0, 1)$ (the probability is taken over the random coin tosses of the protocol). The randomized one-way communication complexity of a function f with error probability δ is denoted by $\mathcal{R}_\delta^{\text{1-way}}(f)$. [KN97] is a standard reference for communication complexity.

2.3 Probability Theory

2.3.1 Statistical Distance Measures

Given a finite set X, a distribution μ on X is defined as a function $\mu : X \longrightarrow [0, 1]$, such that $\sum_{x \in X} \mu(x) = 1$. The finite set X is called the base set of the distribution μ. We use the following distance measures among distributions in our work.

Definition 2.3. *The* KL divergence *[KL51] and the* Jensen-Shannon divergence *[Lin91] between two distributions μ_1 and μ_2 on X are defined as follows.*

$$D_{KL}(\mu_1 \| \mu_2) = \sum_{x \in X} \mu_1(x) \ln \frac{\mu_1(x)}{\mu_2(x)}$$

$$JS(\mu_1, \mu_2) = \frac{1}{2} \left(D_{KL} \left(\mu_1 \| \frac{\mu_1 + \mu_2}{2} \right) + D_{KL} \left(\mu_2 \| \frac{\mu_1 + \mu_2}{2} \right) \right)$$

The Jensen-Shannon divergence has subsequently been generalized to measure the mutual distance among more than two distributions as follows.

Definition 2.4. *Given n distributions $\mu_1, \ldots, \mu_n$ over the same base set, the* generalized Jensen-Shannon divergence[1] *among them is:*

$$JS(\mu_1, \ldots, \mu_n) = \frac{1}{n} \sum_{i=1}^{n} D_{KL} \left(\mu_i \| \frac{1}{n} \sum_{j=1}^{n} \mu_j \right)$$

2.3.2 Concentration Inequalities of Probability

Suppose we have n events namely $A_1, A_2, \ldots, A_n$ in a probability space. Then the probability of occurring any of these n events can be bounded by what is known as union bound.

Theorem 2.2. *(Union Bound)*
The probability of happening any of a given n events $A_1, A_2, \ldots, A_n$ can be upper bounded as follows.

$$\Pr[\cup_{i=1}^{n} A_i] \leqslant \sum_{i=1}^{n} \Pr[A_i]$$

Given a probability space $(\Omega, \mathcal{F}, \mu)$, a random variable $X : \Omega \longrightarrow \mathbb{R}$ is a function from the sample space Ω to the set of real numbers $\mathbb{R}$ such that $\{\omega \in \Omega : X(\omega) \leqslant c\} \in \mathcal{F}$ for every $c \in \mathbb{R}$. We refer to [Dur10] for elementary notions of probability theory.

Given a positive random variable X with finite mean $\mathbb{E}[X]$, the classical Markov inequality bounds the probability by which the random variable X deviates from its mean.

Theorem 2.3. *(Markov Inequality)*
For a positive random variable X with mean $\mathbb{E}[X]$, the probability that the random variable X

---

[1]The generalized Jensen-Shannon divergence is often formulated with weights on each of the n distributions. The definition here puts equal weight on each distribution and is sufficient for our purposes.

takes value more than $c\mathbb{E}[X]$ can be upper bounded as follows.

$$\Pr[X \geqslant c\mathbb{E}[X]] \leqslant \frac{1}{c}$$

If the variance of a random variable X is known, the Chebyshev inequality provides a sharper concentration bound for a random variable.

Theorem 2.4. *(Chebyshev Inequality)*
For a random variable with mean $\mathbb{E}[X]$ and variance σ^2, the probability that the random variable X takes value more than $c\sigma$ away from its mean $\mathbb{E}[X]$ is upper bounded as follows.

$$\Pr[|X - \mathbb{E}[X]| \geqslant c\sigma] \leqslant \frac{1}{c^2}$$

For a sum of independent bounded random variables X, the Chernoff bound gives a much tighter concentration around the mean of X.

Theorem 2.5. *(Chernoff Bound)*
Let $X_1, \ldots, X_\ell$ be a sequence of ℓ independent random variables in $[0, 1]$ (not necessarily identical). Let $S = \sum_i X_i$ and let $\mu = \mathbb{E}[S]$. Then, for any $0 \leqslant \delta \leqslant 1$:

$$\Pr[|S - \mu| \geqslant \delta\mu] < 2\exp(-\delta^2\mu/3)$$

2.4 Information Theory

For a discrete random variable X with possible values $\{x_1, x_2, \ldots, x_n\}$, the Shannon entropy of X is defined as $H(X) = -\sum_{i=1}^{n} \Pr(X = x_i) \log_2 \Pr(X = x_i)$. Let $H_b(p) = -p\log_2 p - (1 - p)\log_2(1 - p)$ denote the binary entropy function when $p \in (0, 1)$. For two random variables X and Y with possible values $\{x_1, x_2, \ldots, x_n\}$ and $\{y_1, y_2, \ldots, y_m\}$, respectively, the conditional entropy of X given Y is defined as $H(X \mid Y) = \sum_{i,j} \Pr(X = x_i, Y = y_j) \log_2 \frac{\Pr(Y=y_j)}{\Pr(X=x_i, Y=y_j)}$. Let $I(X; Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$ denote the mutual information between two random variables X, Y. Let $I(X; Y \mid Z)$ denote the mutual information between two random variables X, Y conditioned on Z, i.e., $I(X; Y \mid Z) = H(X \mid Z) - H(X \mid Y, Z)$. The following summarizes several basic properties of entropy and mutual information.

Proposition 2.1. *Let X, Y, Z, W be random variables.*

1. *If X takes value in* $\{1, 2, \ldots, m\}$, *then* $H(X) \in [0, \log m]$.

2. $H(X) \geqslant H(X \mid Y)$ *and* $I(X; Y) = H(X) - H(X \mid Y) \geqslant 0$.

3. *If X and Z are independent, then we have* $I(X; Y \mid Z) \geqslant I(X; Y)$. *Similarly, if X, Z are independent given W, then* $I(X; Y \mid Z, W) \geqslant I(X; Y \mid W)$.

4. *(Chain rule of mutual information)* $I(X, Y; Z) = I(X; Z) + I(Y; Z \mid X)$. *More generally, for any random variables* $X_1, X_2, \ldots, X_n, Y$, $I(X_1, \ldots, X_n; Y) = \sum_{i=1}^{n} I(X_i; Y \mid X_1, \ldots, X_{i-1})$. *Thus,* $I(X, Y; Z \mid W) \geqslant I(X; Z \mid W)$.

5. *(Fano's inequality) Let X be a random variable chosen from domain* $\mathcal{X}$ *according to distribution* μ_X, *and Y be a random variable chosen from domain* $\mathcal{Y}$ *according to distribution* μ_Y. *For any reconstruction function* $g : \mathcal{Y} \to \mathcal{X}$ *with error* δ_g,

$$H_b(\delta_g) + \delta_g \log(|\mathcal{X}| - 1) \geqslant H(X \mid Y).$$

We refer readers to [CT12] for a nice introduction to information theory.

2.5   Relevant Terminology for Trees

A tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a set of vertices $\mathcal{V}$ along with a set of edges $\mathcal{E} \subset \mathfrak{P}_2(\mathcal{V})$ such that for every two vertices $x, y \in \mathcal{V}$, there exists exactly one path between x and y in $\mathcal{T}$. The following definitions pertaining to the structural aspects of trees will be useful.

▷ The *pathwidth* of $\mathcal{T}$ is the minimum width of a *path decomposition* of $\mathcal{T}$ [Hei93].

▷ A set of disjoint paths $\mathcal{Q} = \{Q_1 = (\mathcal{X}_1, \mathcal{E}_1), \ldots, Q_k = (\mathcal{X}_k, \mathcal{E}_k)\}$ is said to cover a tree $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ if $\mathcal{X} = \cup_{i \in [k]} \mathcal{X}_i, \mathcal{E}_i \subseteq \mathcal{E}, \mathcal{X}_i \cap \mathcal{X}_j = \emptyset, \mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ for every $i, j \in [k]$ with $i \neq j$. The *path cover number* of $\mathcal{T}$ is the cardinality of the smallest set $\mathcal{Q}$ of *disjoint* paths that cover $\mathcal{T}$.

▷ The *distance of a tree* $\mathcal{T}$ *from a path* is the smallest number of nodes whose removal makes the tree a path.

▷ The *diameter* of a tree $\mathcal{T}$ is the number of edges in the longest path in $\mathcal{T}$.
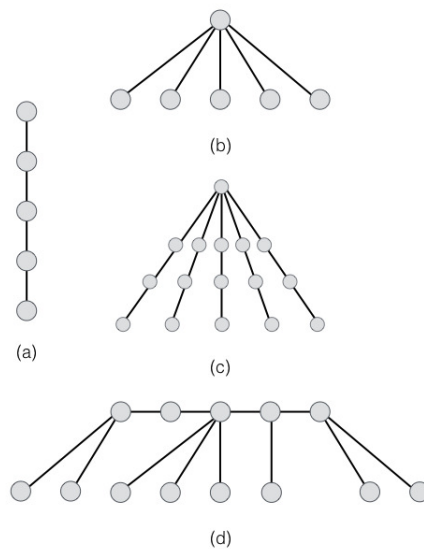
Figure 2.2: Depicting classes of trees: (a) a path, (b) a star, (c) a (balanced) subdivision of a star, (d) a caterpillar.

We also list some definitions of *subclasses* of trees (which are special types of trees, see also Figure 2.2).

▷ A tree is a *star* if there is a *center vertex* and every other vertex is a neighbor of this vertex.

▷ A tree is a *subdivision of a star* if it can be constructed by replacing each edge of a star by a path.

▷ A subdivision of a star is called *balanced* if there exists an integer ℓ such that the distance of every leaf node from the center is ℓ.

▷ A tree is a *caterpillar* if there is a *central path* and every other vertex is at a distance of one it.

▷ A tree is a *complete binary tree* rooted at r if every nonleaf node has exactly two children and there exists an integer h, called the height of the tree, such that every leaf node is at a distance of either h or $h - 1$ from the root node r.

2.6   Universal Family of Hash Functions

In this section, we discuss universal family of hash functions. Intuitively, a family of hash functions is universal if a hash function picked uniformly at random from the family behaves like a completely random function. The formal definition of universal family of hash functions is as follows. We use this notion crucially in Chapter 6.

Definition 2.5. *(Universal family of hash functions)*
A family of functions $\mathcal{H} = \{h | h : A \to B\}$ *is called a* universal family of hash functions *if for all* $a \neq b \in A$, $\Pr\{h(a) = h(b)\} = 1/|B|$, *where* h *is picked uniformly at random from* $\mathcal{H}$.

$$\Pr_{h \text{ picked uniformly at random from } \mathcal{H}}\{h(a) = h(b)\} = 1/|B|$$

We know that there exists a universal family of hash functions $\mathcal{H}$ from $[k]$ to $[\ell]$ for every positive integer ℓ and every prime k [Cor09]. Moreover, $|\mathcal{H}|$, the size of $\mathcal{H}$, is $O(k^2)$.

Part I

Preference Elicitation

This part of the thesis consists of the following two chapters.

▷ In Chapter 3 – Preference Elicitation for Single Peaked Profiles on Trees – we present efficient algorithms for preference elicitation when the preferences belong to the domain of single peaked preferences on a tree. We show interesting connections between query complexity for preference elicitation and various common parameters of trees.

▷ In Chapter 4 – Preference Elicitation for Single Crossing Profiles – we show optimal algorithms (except for one scenario) for eliciting single crossing preference profiles. Our results show that the query complexity for preference elicitation for single crossing preference profiles crucially depends on the way the votes are allowed to access.

Chapter 3

Preference Elicitation for Single Peaked Profiles on Trees

In multiagent systems, we often have a set of agents each of which has a preference ordering over a set of items. One would often like to know these preference orderings for various tasks like data analysis, preference aggregation, voting etc. However, we often have a large number of items which makes it impractical to directly ask the agents for their complete preference ordering. In such scenarios, one convenient way to know these agents' preferences is to ask the agents to compare (hopefully a small number of) pairs of items.

Prior work on preference elicitation focuses on unrestricted domain and the domain of single peaked preferences. They show that the preferences in single peaked domain can be elicited with much less number of queries compared to unrestricted domain. We extend this line of research and study preference elicitation for single peaked preferences on trees which is a strict superset of the domain of single peaked preferences. We show that the query complexity for preference elicitation crucially depends on the number of leaves, the path cover number, and the minimum number of nodes that should be removed to turn the single peaked tree into a path. We also observe that the other natural parameters of the single peaked tree like maximum and minimum degrees of a node, diameter, pathwidth do not play any direct role in determining the query complexity of preference elicitation. We

then investigate the query complexity for finding a weak Condorcet winner (we know that at least one weak Condorcet winner always exists for profiles which are single peaked on a tree) for preferences single peaked on a tree and show that this task has much less query complexity than preference elicitation. Here again we observe that the number of leaves in the underlying single peaked tree and the path cover number of the tree influence the query complexity of the problem.

3.1 Introduction

In multiagent systems, we often have scenarios where a set of agents have to arrive at a consensus although they possess different opinions over the alternatives available. Typically, the agents have preferences over a set of items, and the problem of aggregating these preferences in a suitable manner is one of the most well-studied problems in social choice theory [BCE+15]. There are many ways of expressing preferences over a set of alternatives. One of the most comprehensive ways is to specify a complete ranking over the set of alternatives. However, one of the downsides of this model is the fact that it can be expensive to solicit the preferences when a large number of alternatives and agents are involved.

Since asking agents to provide their complete rankings is impractical, a popular notion is one of *elicitation*, where we ask agents simple *comparison queries*, such as if they prefer an alternative x over another alternative y. This naturally gives rise to the problem of *preference elicitation*, where we hope to recover the complete ranking (or possibly the most relevant part of the ranking) based on a small number of comparison queries.

In the context of a fixed voting rule, we may also want to query the voters up to the point of determining the winner (or the aggregate ranking, as the case may be). Yet another refinement in this setting is when we have prior information about how agents are likely to vote, and we may want to determine which voters to query first, to be able to quickly rule out a large number of alternatives, as explored by [CS02].

When our goal is to elicit preferences that have no prior structure, one can demonstrate scenarios where it is imperative to ask each agent (almost) as many queries as would be required to determine an arbitrary ranking. However, in recent times, there has been considerable interest in voting profiles that are endowed with additional structure. The motivation for this is two-fold. The first is that in several application scenarios commonly considered, it is rare that votes are ad-hoc, demonstrating no patterns whatsoever. For example, the notion

of *single-peaked preferences*, which we will soon discuss at length, forms the basis of several studies in the analytical political sciences [HM97]. In his work on eliciting preferences that demonstrate the "single-peaked" structure, Conitzer argues that the notion of single-peakedness is also a reasonable restriction in applications outside of the domain of political elections [Con09].

The second motivation for studying restricted preferences is somewhat more technical, but is just as compelling. To understand why structured preferences have received considerable attention from social choice theorists, we must first take a brief detour into some of the foundational ideas that have shaped the landscape of voting theory as we understand them today. As it turns out, the axiomatic approach of social choice involves defining certain "properties" that formally capture the quality of a voting rule. For example, we would not want a voting rule to be, informally speaking, a *dictatorship*, which would essentially mean that it discards all but one voter's input. Unfortunately, a series of cornerstone results establish that it is impossible to devise voting rules which respect some of the simplest desirable properties. Indeed, the classic work of Arrow [Arr50] and Gibbard-Satterthwaite [Gib73, Sat75] show that there is no straight forward way to simultaneously deal with properties like voting paradoxes, strategy-proofness, nondictatorship, unanimity etc. We refer to [Mou91] for a more elaborate discussion on this. Making the matter worse, many classical voting rules, for example the Kemeny voting rule [Kem59, BNM⁺58, BITT89, HSV05], the Dodgson voting rule [Dod76, HHR97], the Young voting rule [You77] etc., turn out to be computationally intractable.

This brings us to the second reason for why structured preferences are an important consideration. The notion of single-peakedness that we mentioned earlier is an excellent illustration (we refer the reader to Section 3.2 for the formal definition). Introduced in [Bla48], it not only captures the essence of structure in political elections, but also turns out to be extremely conducive to many natural theoretical considerations. To begin with, one can devise voting rules that are "nice" with respect to several properties, when preferences are single-peaked. Further, they are structurally elegant from the point of view of winner determination, since they always admit a *weak Condorcet winner* — a candidate which is not defeated by any other candidate in pairwise election — thus working around the Condorcet paradox which is otherwise a prominent concern in the general scenario. In a landmark contribution, Brandt et al. [BBHH15] show that several computational problems in social choice theory that are

intractable in the general setting become polynomial time solvable when we consider single-peaked preferences.

A natural question at this point is if the problem of elicitation becomes any easier — that is, if we can get away with fewer queries — by taking advantage of the structure provided by single-peakedness. It turns out that the answer to this is in the affirmative, as shown in a detailed study by Conitzer [Con09]. The definition of single-peakedness involves an ordering over the candidates (called the *harmonious ordering* by some authors). The work of Conitzer [Con09] shows that $\mathcal{O}(mn)$ queries suffice, assuming either that the harmonious ordering is given, or one of the votes is known, where m and n are the number of candidates and voters respectively.

We now return to the theme of structural restrictions on preferences. As it turns out, the single peaked preference domain has subsequently been generalized to single peakedness on trees (roughly speaking, these are profiles that are single peaked on every path on a given tree) [Dem82, Tri89]. This is a class that continues to exhibit many desirable properties of single peaked domain. For example, there always exists a weak Condorcet winner and further, many voting rules that are intractable in an unrestricted domain are polynomial time computable if the underlying single peaked tree is "nice" [YCE13, PE16]. We note the class of profiles that are single peaked on trees are substantially more general than the class of single peaked preferences. Note that the latter is a special case since a path is, in particular, a tree. Our work here addresses the issue of elicitation on profiles that are single-peaked on trees, and can be seen as a significant generalization of the results in [Con09]. We now detail the specifics of our contributions.

3.1.1   Our Contribution

We study the query complexity for preference elicitation when the preference profile is single peaked on a tree. We provide *tight* connections between various parameters of the underlying single peaked tree and the query complexity for preference elicitation. Our broad goal is to provide a suitable generalization of preference elicitation for single peaked profiles to profiles that are single peaked on trees. Therefore, we consider various ways of quantifying the "closeness" of a tree to a path, and reflect on how these measures might factor into the query complexity of an algorithm that is actively exploiting the underlying tree structure.

We summarize our results for preference elicitation in Table 3.1, where the readers will note that most of the parameters (except diameter) chosen are small constants (typically

Parameter	Upper Bound	Lower Bound
Path width (w)	$\mathcal{O}(mn \log m)$ [Observation 3.1]	$\Omega(mn \log m)$ even for $w = 1, \log m$ [Corollary 3.5]
Maximum degree (Δ)	$\mathcal{O}(mn \log m)$ Observation 3.1]	$\Omega(mn \log m)$ even for $\Delta = 3, m - 1$ [Corollary 3.3]
Path cover number (k)	$\mathcal{O}(mn \log k)$ [Theorem 3.2]	$\Omega(mn \log k)$ [Corollary 3.2]
Number of leaves (ℓ)	$\mathcal{O}(mn \log \ell)$ [Corollary 3.1]	$\Omega(mn \log \ell)$ [Theorem 3.4]
Distance from path (d)	$\mathcal{O}(mn + nd \log d)$ [Theorem 3.3]	$\Omega(mn + nd \log d)$ [Theorem 3.6]
Diameter (ω)	$\mathcal{O}(mn \log m)$ [Observation 3.1]	$\Omega(mn \log m)$ even for $\omega = 2, m/2$ [Corollary 3.4]

Table 3.1: Summary of query complexity bounds for eliciting preferences which are single peaked on trees.

zero, one or two) when the tree under consideration is a path. Observe that in some cases — such as the number of leaves, or the path cover number — the dependence on the parameter is transparent (and we recover the results of [Con09] as a special case), while in other cases, it is clear that the perspective provides no additional mileage (the non-trivial results here are the matching lower bounds).

In terms of technique, our strategy is to "scoop out the paths from the tree" and use the algorithm form [Con09] to efficiently elicit the preference on the parts of the trees that are paths. We then efficiently merge this information across the board, and that aspect of the algorithm varies depending on the parameter under consideration. The lower bounds typically come from trees that provide large "degrees of freedom" in reordering candidates, typically these are trees that don't have too many long paths (such as stars). The arguments are often subtle but intuitive.

We then study the query complexity for finding a weak Condorcet winner of a preference profile which is single peaked on a tree. Here, we are able to show that a weak Condorcet winner can be found with far fewer queries than the corresponding elicitation problem. In particular, we establish that a weak Condorcet winner can be found using $\mathcal{O}(mn)$ many queries for profiles that are single peaked on trees [Theorem 3.7], and we also show that this bound is the best that we can hope for [Theorem 3.11]. We also consider the problem for the special

case of single peaked profiles. While Conitzer in [Con09] showed that $\Omega(mn)$ queries are necessary to determine the *aggregate ranking*, we show that only $\mathcal{O}(n \log m)$ queries suffice if we are just interested in (one of the) weak Condorcet winners. Moreover, we show this bound is tight under the condition that the algorithm does not interleave queries to different voters [Theorem 3.12] (our algorithm indeed satisfies this condition).

Finally, for expressing the query complexity for determining a weak Condorcet winner in terms of a measure of closeness to a path, we show an algorithm with query complexity $\mathcal{O}(nk \log m)$ where k is the path cover number of $\mathcal{T}$ [Theorem 3.10] or the number of leaves in $\mathcal{T}$ [Corollary 3.6]. We now elaborate further on our specific contributions for preference elicitation.

▷ We design novel algorithms for preference elicitation for profiles which are single peaked on a tree with ℓ leaves with query complexity $\mathcal{O}(mn \log \ell)$ [Corollary 3.1]. Moreover, we prove that there exists a tree $\mathcal{T}$ with ℓ leaves such that any preference elicitation algorithm for profiles which are single peaked on tree $\mathcal{T}$ has query complexity $\Omega(mn \log \ell)$ [Theorem 3.4]. We show similar results for the parameter path cover number of the underlying tree [Theorem 3.2 and Corollary 3.2].

▷ We provide a preference elicitation algorithm with query complexity $\mathcal{O}(mn + nd \log d)$ for single peaked profiles on trees which can be made into a path by deleting at most d nodes [Theorem 3.3]. We show that our query complexity upper bound is tight up to constant factors [Theorem 3.6]. These results show that the query complexity for preference elicitation tightly depends on the number of leaves, the path cover number, and the distance from path of the underlying tree.

▷ We then show that there exists a tree $\mathcal{T}$ with pathwidth one or $\log m$ [Corollary 3.5] or maximum degree is 3 or $m - 1$ [Corollary 3.3] or diameter is 2 or $^m/_2$ [Corollary 3.4] such that any preference elicitation algorithm for single peaked profiles on the tree $\mathcal{T}$ has query complexity $\Omega(mn \log m)$. These results show that the query complexity for preference elicitation does not directly depend on the parameters above.

We next investigate the query complexity for finding a weak Condorcet winner for profiles which are single peaked on trees and we have the following results.

▷ We show that a weak Condorcet winner can be found using $\mathcal{O}(mn)$ queries for profiles that are single peaked on trees [Theorem 3.7] which is better than the query complexity

for preference elicitation. Moreover, we prove that this bound is tight in the sense that any algorithm for finding a weak Condorcet winner for profiles that are single peaked on stars has query complexity $\Omega(mn)$ [Theorem 3.11].

▷ On the other hand, we can find a weak Condorcet winner using only $\mathcal{O}(n \log m)$ queries for single peaked profiles [Theorem 3.8]. Moreover, we show that this bound is tight under the condition that the algorithm does not interleave queries to different voters [Theorem 3.12] (our algorithm indeed satisfies this condition). For any arbitrary underlying single peaked tree $\mathcal{T}$, we provide an algorithm for finding a weak Condorcet winner with query complexity $\mathcal{O}(nk \log m)$ where k is the path cover number of $\mathcal{T}$ [Theorem 3.10] or the number of leaves in $\mathcal{T}$ [Corollary 3.6].

To summarize, we remark that our results non-trivially generalize earlier works on query complexity for preference elicitation in [Con09]. We believe revisiting the preference elicitation problem in the context of profiles that are single peaked on trees is timely, and that this work also provides fresh algorithmic and structural insights on the domain of preferences that are single peaked on trees.

3.1.2 Related Work

We have already mentioned the work in [Con09] addressing the question of eliciting preferences in single-peaked profiles, which is the closest predecessor to our work. Before this, Conitzer and Sandholm addressed the computational hardness for querying minimally for winner determination [CS02]. They also prove that one would need to make $\Omega(mn \log m)$ queries even to decide the winner for many commonly used voting rules [CS05] which matches with the trivial $\mathcal{O}(mn \log m)$ upper bound for preference elicitation in unrestricted domain (due to sorting lower bound). Ding and Lin study preference elicitation under partial information setting and show interesting properties of what they call a deciding set of queries [DL13a]. Lu and Boutilier provide empirical study of preference elicitation under probabilistic preference model [LB11b] and devise several novel heuristics which often work well in practice [LB11a].

3.2 Domain of Single Peaked Profiles on Trees

A preference $\succ \in \mathcal{L}(\mathcal{C})$ over a set of candidates $\mathcal{C}$ is called *single peaked* with respect to an order $\succ' \in \mathcal{L}(\mathcal{C})$ if, for every candidates $x, y \in \mathcal{C}$, we have $x \succ y$ whenever we have either $c \succ' x \succ' y$ or $y \succ' x \succ' c$, where $c \in \mathcal{C}$ is the candidate at the first position of $\succ$. A profile $\mathcal{P} = (\succ_i)_{i \in [n]}$ is called single peaked with respect to an order $\succ' \in \mathcal{L}(\mathcal{C})$ if $\succ_i$ is single peaked with respect to $\succ'$ for every $i \in [n]$. Notice that if a profile $\mathcal{P}$ is single peaked with respect to an order $\succ' \in \mathcal{L}(\mathcal{C})$, then $\mathcal{P}$ is also single peaked with respect to the order $\overleftarrow{\succ}'$. Example 3.1 exhibits an example of a single peaked preference profile.

Example 3.1. *(Example of single peaked preference profile) Consider a set $\mathcal{C}$ of m candidates, corresponding m distinct points on the Real line, a set $\mathcal{V}$ of n voters, also corresponding n points on the Real line, and the preference of every voter are based on their distance to the candidates – given any two candidates, every voter prefers the candidate nearer to her (breaking the tie arbitrarily). Then the set of n preferences is single peaked with respect to the ordering of the candidates according to the ascending ordering of their positions on the Real line.*

Given a path $\mathcal{Q} = (x_1, x_2, \ldots, x_\ell)$ from a vertex x_1 to another vertex x_ℓ in a tree $\mathcal{T}$, we define the order induced by the path $\mathcal{Q}$ to be $x_1 \succ x_2 \succ \cdots \succ x_\ell$. Given a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ with the set of nodes as the set of candidates $\mathcal{C}$, a profile $\mathcal{P}$ is called single peaked on the tree $\mathcal{T}$ if $\mathcal{P}$ is single peaked on the order induced by every path of the tree $\mathcal{T}$; that is for every two candidates $x, y \in \mathcal{C}$, the profile $\mathcal{P}(\mathcal{X})$ is single peaked with respect to the order $\succ$ on the set of candidates $\mathcal{X}$ induced by the unique path from x to y in $\mathcal{T}$. We call the tree $\mathcal{T}$ the *underlying single peaked tree* of the preference profile $\mathcal{P}$. It is known (c.f. [Dem82]) that there always exists a weakly Condorcet winner for a profile $\mathcal{P}$ which is single peaked on a tree $\mathcal{T}$.

3.3 Problem Definitions and Known Results

Suppose we have a profile $\mathcal{P}$ with n voters and m candidates. For any pair of distinct candidates x and y, and a voter $\ell \in [n]$, we introduce the boolean-valued function $\text{QUERY}(x \succ_\ell y)$ as follows. The output of this function is TRUE if the voter ℓ prefers the candidate x over the candidate y and FALSE otherwise. We now formally state the two problems that we consider in this work.

Definition 3.1. PREFERENCE ELICITATION

Given a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ and an oracle access to the function QUERY *($\cdot$) for a profile $\mathcal{P}$ which is single peaked on $\mathcal{T}$, find $\mathcal{P}$.*

Suppose we have a set of candidates $\mathcal{C} = \{c_1, \ldots, c_m\}$. We say that an algorithm $\mathcal{A}$ makes q queries if there are exactly q distinct tuples $(\ell, c_i, c_j) \in [n] \times \mathcal{C} \times \mathcal{C}$ with $i < j$ such that $\mathcal{A}$ calls QUERY $(c_i \succ_\ell c_j)$ or QUERY $(c_j \succ_\ell c_i)$. We call the maximum number of queries made by an algorithm $\mathcal{A}$ for any input its *query complexity*.

We state some known results that we will appeal to later. The first observation employs a sorting algorithm like merge sort to elicit every vote with $\mathcal{O}(m \log m)$ queries, while the second follows from the linear-time merge subroutine of merge sort ([Cor09]).

Observation 3.1. *There is a* PREFERENCE ELICITATION *algorithm with query complexity* $\mathcal{O}(mn \log m)$.

Observation 3.2. *Suppose $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{C}$ form a partition of $\mathcal{C}$ and $\succ$ is a ranking of the candidates in $\mathcal{C}$. Then there is a linear time algorithm that finds $\succ$ given $\succ (\mathcal{C}_1)$ and $\succ (\mathcal{C}_2)$ with query complexity $\mathcal{O}(|\mathcal{C}|)$.*

Theorem 3.1. *[Con09] There is a* PREFERENCE ELICITATION *algorithm with query complexity* $\mathcal{O}(mn)$ *for single peaked profiles.*

We now state the WEAK CONDORCET WINNER problem, which asks for eliciting only up to the point of determining a weak Condercet winner (recall that at least one such winner is guaranteed to exist on profiles that are single-peaked on trees).

Definition 3.2. WEAK CONDORCET WINNER

Given a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ and an oracle access to the function QUERY *($\cdot$) for a profile $\mathcal{P}$ which is single peaked on $\mathcal{T}$, find a weak Condorcet winner of $\mathcal{P}$.*

3.4 Results for Preference Elicitation

In this section, we present our results for PREFERENCE ELICITATION for profiles that are single peaked on trees. Recall that we would like to generalize Theorem 3.1 in a way to profiles that are single peaked on trees. Since the usual single peaked profiles can be viewed as profiles single peaked with respect to a path, we propose the following measures of how much a tree resembles a path.

▷ *Leaves.* Recall any tree has at least two leaves, and paths are the trees that have exactly two leaves. We consider the class of trees that have ℓ leaves, and show an algorithm with query complexity of $\mathcal{O}(mn\log\ell)$.

▷ *Path Cover.* Consider the notion of a *path cover number* of a tree, which is the smallest number of disjoint paths that the tree can be partitioned into. Clearly, the path cover number of a path is one; and for trees that can be covered with k paths, we show an algorithm with query complexity $\mathcal{O}(mn\log k)$.

▷ *Distance from Paths.* Let d be the size of the smallest set of vertices whose removal makes the tree a path. Again, if the tree is a path, then the said set is simply the empty set. For trees that are at a distance d from being a path (in the sense of vertex deletion), we provide an algorithm with query complexity $\mathcal{O}(mn\log d)$.

▷ *Pathwidth and Maximum Degree.* Finally, we note that paths are also trees that have pathwidth one, and maximum degree two. These perspectives turn out to be less useful: in particular, there are trees where these parameters are constant, for which we show that elicitation is as hard as it would be on an arbitrary profile, and therefore the easy algorithm from Observation 3.1 is actually the best that we can hope for.

For the first three perspectives that we employ, that seemingly capture an appropriate aspect of paths and carry it forward to trees, the query complexities that we obtain are tight — we have matching lower bounds in all cases. Also, while considering structural parameters, it is natural to wonder if there is a class of trees that are incomparable with paths but effective for elicitation. Our attempt in this direction is to consider trees of bounded diameter. However, again, we find that this is not useful, as we have examples to show that there exist trees of diameter two that are as hard to elicit as general profiles.

We remark at this point that all these parameters are polynomially computable for trees, making the algorithmic results viable. For example, the distance from path can be found by finding the longest path in the single peaked tree which can be computed in polynomial amount of time [Cor09]. Also, for the parameters of pathwidth, maximum degree and diameter, we show lower bounds on trees where these parameters are large (such as trees with pathwidth $O(\log m)$, maximum degree $m-1$, and diameter $m/2$), which — roughly speaking — also rules out the possibility of getting a good inverse dependence. As a concrete example, motivated by the $\mathcal{O}(mn)$ algorithm for paths, which have diameter m, one might wonder if

there is an algorithm with query complexity $\mathcal{O}(\frac{mn\log m}{\log\omega})$ for single peaked profiles on a tree with diameter ω. This possibility, in particular, is ruled out. We are now ready to discuss the results in Table 3.1.

We begin with showing a structural result about trees: any tree with ℓ leaves can be partitioned into ℓ paths. The idea is to fix some non-leaf node as root and iteratively find a path from low depth nodes (depth of a node is its distance from root) to some leaf node which is disjoint from all the paths chosen so far. We formalize this idea below.

Lemma 3.1. *Let $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ be a tree with ℓ leaves. Then there is a polynomial time algorithm which partitions $\mathcal{T}$ into ℓ disjoint paths $\mathcal{Q}_i = (\mathcal{X}_i, \mathcal{E}_i), i \in [\ell]$; that is we have $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset, \mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ for every $i, j \in [\ell]$ with $i \neq j$, $\mathcal{X} = \cup_{i \in [\ell]} \mathcal{X}_i$, $\mathcal{E} = \cup_{i \in [\ell]} \mathcal{E}_i$, and $\mathcal{Q}_i$ is a path in $\mathcal{T}$ for every $i \in [\ell]$.*

Proof. We first make the tree $\mathcal{T}$ rooted at any arbitrary nonleaf node r. We now partition the tree $\mathcal{T}$ into paths iteratively as follows. Initially every node of the tree $\mathcal{T}$ is *unmarked* and the set of paths $\mathcal{Q}$ we have is empty. Let $Q_1 = (\mathcal{X}_1, \mathcal{E}_1)$ be the path in $\mathcal{T}$ from the root node to any leaf node. We put Q_1 into $\mathcal{Q}$ and *mark* all the nodes in Q_1. More generally, in the ith iteration we pick an unmarked node u that is closest to the root r, breaking ties arbitrarily, and add any path $\mathcal{Q}_i$ in $\mathcal{T}$ from u to any leaf node in the subtree $\mathcal{T}_u$ rooted at u, and *mark* all the nodes in $\mathcal{Q}_i = (\mathcal{X}_i, \mathcal{E}_i)$. Since u is unmarked, we claim that every node in $\mathcal{T}_u$ is unmarked. Indeed, otherwise suppose there is a node w in $\mathcal{T}_u$ which is already marked. Then there exists two paths from r to w one including u and another avoiding u since u is currently unmarked and w is already marked. This contradicts the fact that $\mathcal{T}$ is a tree. Hence every node in $\mathcal{T}_u$ is unmarked. We continue until all the leaf nodes are marked and return $\mathcal{Q}$. Since $\mathcal{T}$ has ℓ leaves and in every iteration at least one leaf node is marked (since every $\mathcal{Q}_i$ contains a leaf node), the algorithm runs for at most ℓ iterations. Notice that, since the algorithm always picks a path consisting of unmarked vertices only, the set of paths in $\mathcal{Q}$ are pairwise disjoint. We claim that $\mathcal{Q}$ forms a partition of $\mathcal{T}$. Indeed, otherwise there must be a node x in $\mathcal{T}$ that remains unmarked at the end. From the claim above, we have all the nodes in the subtree $\mathcal{T}_x$ rooted at x unmarked which includes at least one leaf node of $\mathcal{T}$. This contradicts the fact that the algorithm terminates when all the leaf nodes are marked. $\qquad\square$

Using Lemma 3.1, and the fact that any path can account for at most two leaves, we have that the path cover number of a tree is the same as the number of leaves up to a factor of two.

Lemma 3.2. *Suppose the path cover number of a tree $\mathcal{T}$ with ℓ leaves is k. Then we have $\ell/2 \leqslant k \leqslant \ell$.*

Proof. The inequality $\ell/2 \leqslant k$ follows from the fact that any path in $\mathcal{T}$ can involve at most two leaves in $\mathcal{T}$ and there exists k paths covering all the leaf nodes. The inequality $k \leqslant \ell$ follows from the fact from Lemma 3.1 that any tree $\mathcal{T}$ with ℓ leaves can be partitioned into ℓ paths. □

3.4.1 Algorithmic Results for PREFERENCE ELICITATION

We now present our main algorithmic results. We begin with generalizing the result of Theorem 3.1 to any single peaked profiles on trees whose path cover number is at most k. The idea is to partition the tree into k disjoint paths, use the algorithm from Theorem 3.1 on each paths to obtain an order of the candidates on each path of the partition, and finally merge these suborders intelligently. We now formalize this idea as follows.

Theorem 3.2. *There is a PREFERENCE ELICITATION algorithm with query complexity $\mathcal{O}(mn \log k)$ for profiles that are single peaked on trees with path cover number at most k.*

Proof. Since the path cover number is at most k, we can partition the tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ into t disjoint paths $\mathcal{P}_i = (\mathcal{C}_i, \mathcal{E}_i), i \in [t]$, where t is at most k. We now show that we can elicit any preference $\succ$ which is single peaked on the tree $\mathcal{T}$ by making $\mathcal{O}(m \log t)$ queries which in turn proves the statement. We first find the preference ordering restricted to $\mathcal{C}_i$ using Theorem 3.1 by making $\mathcal{O}(|\mathcal{C}_i|)$ queries for every $i \in [t]$. This step needs $\sum_{i \in [t]} \mathcal{O}(|\mathcal{C}_i|) = \mathcal{O}(m)$ queries since $\mathcal{C}_i, i \in [t]$ forms a *partition* of $\mathcal{C}$. We next merge the t orders $\succ (\mathcal{C}_i), i \in [t]$, to obtain the complete preference $\succ$ by using a standard divide and conquer approach for t-way merging as follows which makes $O(m \log t)$ queries [HUA83].

Initially we have t orders to merge. We arbitrarily pair the t orders into $\lceil t/2 \rceil$ pairs with at most one of them being singleton (when t is odd). By renaming, suppose the pairings are as follows: $(\succ (\mathcal{C}_{2i-1}), \succ (\mathcal{C}_{2i})), i \in [\lfloor t/2 \rfloor]$. Let us define $\mathcal{C}'_i = \mathcal{C}_{2i-1} \cup \mathcal{C}_{2i}$ for every $i \in [\lfloor t/2 \rfloor]$ and $\mathcal{C}'_{\lceil t/2 \rceil} = \mathcal{C}_t$ if t is an odd integer. We merge $\succ (\mathcal{C}_{2i-1})$ and $\succ (\mathcal{C}_{2i})$ to get $\succ (\mathcal{C}'_i)$ for every $i \in [\lfloor t/2 \rfloor]$ using Observation 3.2. The number queries the algorithm makes in this iteration is $\sum_{i \in [\lfloor t/2 \rfloor]} \mathcal{O}(|\mathcal{C}_{2i-1}| + |\mathcal{C}_{2i}|) = \mathcal{O}(m)$ since $(\mathcal{C}_i)_{i \in [t]}$ forms a partition of $\mathcal{C}$. At the end of the first iteration, we have $\lceil t/2 \rceil$ orders $\succ (\mathcal{C}'_i), i \in [\lceil t/2 \rceil]$ to merge to get $\succ$. The algorithm repeats the step above $\mathcal{O}(\log t)$ times to obtain $\succ$ and the query complexity of each iteration is $\mathcal{O}(m)$. Thus the query complexity of the algorithms is $\mathcal{O}(m + m \log t) = \mathcal{O}(m \log k)$. □

Using the fact from Lemma 3.2 that a tree with ℓ leaves can be partitioned into at most ℓ paths, we can use the algorithm from Theorem 3.2 to obtain the following bound on query complexity for preference elicitation in terms of leaves.

Corollary 3.1. *There is a* PREFERENCE ELICITATION *algorithm with query complexity* $\mathcal{O}(mn\log\ell)$ *for profiles that are single peaked on trees with at most ℓ leaves.*

Finally, if we are given a subset of candidates whose removal makes the single peaked tree a path, then we have an elicitation algorithm that makes $\mathcal{O}(mn + nd\log d)$ queries. As before, we first determine the ordering among the candidates on the path (after removing those d candidates) with $\mathcal{O}(m - d)$ queries. We then determine the ordering among the rest in $\mathcal{O}(d\log d)$ queries using Observation 3.1 and merge using Observation 3.2 these two orderings. This leads us to the following result.

Theorem 3.3. *There is a* PREFERENCE ELICITATION *algorithm with query complexity* $\mathcal{O}(mn + nd\log d)$ *for profiles that are single peaked on trees with distance d from path.*

Proof. Let $\mathcal{X}$ be the smallest set of nodes of a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ such that $\mathcal{T} \setminus \mathcal{X}$, the subgraph of $\mathcal{T}$ after removal of the nodes in $\mathcal{X}$, is a path. We have $|\mathcal{X}| \leqslant d$. For any preference $\succ$, we make $\mathcal{O}(d\log d)$ queries to find $\succ(\mathcal{X})$ using Observation 3.1, make $\mathcal{O}(|\mathcal{C} \setminus \mathcal{X}|) = \mathcal{O}(m - d)$ queries to find $\succ(\mathcal{C} \setminus \mathcal{X})$ using Theorem 3.1, and finally make $\mathcal{O}(m)$ queries to find $\succ$ by merging $\succ(\mathcal{X})$ and $\succ(\mathcal{C} \setminus \mathcal{X})$ using Observation 3.2. This gives an overall query complexity of $\mathcal{O}(mn + nd\log d)$. $\qquad\square$

3.4.2 Lower Bounds for PREFERENCE ELICITATION

We now turn to query complexity lower bounds for preference elicitation. Our first result is based on a counting argument, showing that the query complexity for preference elicitation in terms of the number of leaves, given by Corollary 3.1, is tight up to constant factors. Indeed, let us consider a subdivision of a star $\mathcal{T}$ with ℓ leaves and let t denote the distance from the center to the leaves, so that we have a total of $t\ell + 1$ candidates. One can show that if the candidates are written out in level order, the candidates that are distance i from the star center can be ordered arbitrarily within this ordering. This tells us that the number of possible preferences $\succ$ that are single peaked on the tree $\mathcal{T}$ is at least $(\ell!)^t$. We obtain the lower bound by using a decision tree argument, wherein we are able to show that it is always possible for an oracle to answer the comparison queries asked by the algorithm in such a way

that the total number of possibilities for the preference of the current voter decreases by at most a factor of two. Since the decision tree of the algorithm must entertain at least $(\ell!)^t$ leaves to account for all possibilities, we obtain the claimed lower bound.

Theorem 3.4. *Let $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ be a balanced subdivision of a star on m nodes with ℓ leaves. Then any deterministic* PREFERENCE ELICITATION *algorithm for single peaked profiles on $\mathcal{T}$ has query complexity $\Omega(mn \log \ell)$.*

Proof. Suppose the number of candidates m be $(t\ell + 1)$ for some integer t. Let c be the center of $\mathcal{T}$. We denote the shortest path distance between any two nodes $x, y \in \mathcal{C}$ in $\mathcal{T}$ by $d(x, y)$. We consider the partition $(\mathcal{C}_0, \dots, \mathcal{C}_t)$ of the set of candidates $\mathcal{C}$ where $\mathcal{C}_i = \{x \in \mathcal{C} : d(x, c) = i\}$. We claim that the preference $\succ = \pi_0 \succ \pi_1 \succ \cdots \succ \pi_t$ of the set of candidates $\mathcal{C}$ is single peaked on the tree $\mathcal{T}$ where π_i is any arbitrary order of the candidates in $\mathcal{C}_i$ for every $0 \leqslant i \leqslant t$. Indeed; consider any path $\mathcal{Q} = (\mathcal{X}, \mathcal{E}')$ in the tree $\mathcal{T}$. Let y be the candidate closest to c among the candidates in $\mathcal{X}$; that is $y = \text{argmin}_{x \in \mathcal{X}} d(x, c)$. Then clearly $\succ (\mathcal{X})$ is single peaked with respect to the path $\mathcal{Q}$ having peak at y. We have $|\mathcal{C}_i| = \ell$ for every $i \in [t]$ and thus the number of possible preferences $\succ$ that are single peaked on the tree $\mathcal{T}$ is at least $(\ell!)^t$.

Let $\mathcal{A}$ be any PREFERENCE ELICITATION algorithm for single peaked profiles on the tree $\mathcal{T}$. We now describe our oracle to answer the queries that the algorithm $\mathcal{A}$ makes. For every voter v, the oracle maintains the set $\mathcal{R}_v$ of possible preferences of the voter v which is consistent with the answers to all the queries that the algorithm $\mathcal{A}$ have already made for the voter v. At the beginning, we have $|\mathcal{R}_v| \geqslant (\ell!)^t$ for every voter v as argued above. Whenever the oracle receives a query on v for any two candidates x and y, it computes the numbers n_1 and n_2 of orders in $\mathcal{R}_v$ which prefers x over y and y over x respectively; the oracle can compute the integers n_1 and n_2 since the oracle has infinite computational power. The oracle answers that the voter v prefers the candidate x over y if and only if $n_1 \geqslant n_2$ and updates the set $\mathcal{R}_v$ accordingly. Hence, whenever the oracle is queried for a voter v, the size of the set $\mathcal{R}_v$ decreases by a factor of at most two. On the other hand, we must have, from the correctness of the algorithm, $\mathcal{R}_v$ to be a singleton set when the algorithm terminates for every voter v – otherwise there exists a voter v (whose corresponding $\mathcal{R}_v$ is not singleton) for which there exist two possible preferences which are single peaked on the tree $\mathcal{T}$ and are consistent with all the answers the oracle has given and thus the algorithm $\mathcal{A}$ fails to output the preference of the voter v correctly. Hence every voter must be queried at least $\Omega(\log((\ell!)^t)) = \Omega(t\ell \log \ell) = \Omega(m \log \ell)$ times. $\qquad \square$

Since the path cover number of a subdivided star on ℓ leaves is at least $\ell/2$, we also obtain the following.

Corollary 3.2. *There exists a tree $\mathcal{T}$ with path cover number k such that any deterministic PREFERENCE ELICITATION algorithm for single peaked profiles on $\mathcal{T}$ has query complexity $\Omega(mn \log k)$.*

Mimicking the level order argument above on a generic tree with ℓ leaves, and using the connection between path cover and leaves, we obtain lower bounds that are functions of (n, ℓ) and (n, k), as given below. This will be useful for our subsequent results.

Theorem 3.5. *Let $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ be any arbitrary tree with ℓ leaves and path cover number k. Then any deterministic PREFERENCE ELICITATION algorithm for single peaked profiles on $\mathcal{T}$ has query complexity $\Omega(n\ell \log \ell)$ and $\Omega(nk \log k)$.*

Proof. Let $\mathcal{X}$ be the set of leaves in $\mathcal{T}$. We choose any arbitrary nonleaf node r as the root of $\mathcal{T}$. We denote the shortest path distance between two candidates $x, y \in \mathcal{C}$ in the tree $\mathcal{T}$ by $d(x, y)$. Let t be the maximum distance of a node from r in $\mathcal{T}$; that is $t = \max_{y \in \mathcal{C} \setminus \mathcal{X}} d(r, x)$. We partition the candidates in $\mathcal{C} \setminus \mathcal{X}$ as $(\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_t)$ where $\mathcal{C}_i = \{y \in \mathcal{C} \setminus \mathcal{X} : d(r, y) = i\}$ for $0 \leqslant i \leqslant t$. We claim that the preference $\succ = \pi_0 \succ \pi_1 \succ \cdots \succ \pi_t \succ \pi$ of the set of candidates $\mathcal{C}$ is single peaked on the tree $\mathcal{T}$ where π_i is any arbitrary order of the candidates in $\mathcal{C}_i$ for every $0 \leqslant i \leqslant t$ and π is an arbitrary order of the candidates in $\mathcal{C} \setminus \mathcal{X}$. Indeed, otherwise consider any path $\mathcal{Q} = (\mathcal{Y}, \mathcal{E}')$ in the tree $\mathcal{T}$. Let y be the candidate closest to r among the candidates in $\mathcal{Y}$; that is $y = \operatorname{argmin}_{x \in \mathcal{Y}} d(x, r)$. Then clearly $\succ (\mathcal{Y})$ is single peaked with respect to the path $\mathcal{Q}$ having peak at y. We have the number of possible preferences $\succ$ that are single peaked on the tree $\mathcal{T}$ is at least $|\mathcal{X}|! = \ell!$. Again using the oracle same as in the proof of Theorem 3.4, we deduce that any PREFERENCE ELICITATION algorithm $\mathcal{A}$ for profiles that are single peaked on the tree $\mathcal{T}$ needs to make $\Omega(n\ell \log \ell)$ queries. The bound with respect to the path cover number now follows from Lemma 3.2. □

The following results can be obtained simply by applying Theorem 3.5 on particular graphs. For instance, we use the fact that stars have $(m - 1)$ leaves and have pathwidth one to obtain the first part of Corollary 3.5, while appealing to complete binary trees that have $O(m)$ leaves and pathwidth $O(\log m)$ for the second part. These examples also work in the context of maximum degree, while for diameter we use stars and caterpillars with a central path of length $m/2$ in Corollary 3.4.

Corollary 3.3. *There exist two trees $\mathcal{T}$ and $\mathcal{T}'$ with maximum degree $\Delta = 3$ and $m-1$ respectively such that any deterministic* PREFERENCE ELICITATION *algorithm for single peaked profiles on $\mathcal{T}$ and $\mathcal{T}'$ respectively has query complexity $\Omega(mn \log m)$.*

Proof. Using Theorem 3.5, we know that any PREFERENCE ELICITATION algorithm for profiles which are single peaked on a complete binary tree has query complexity $\Omega(mn \log m)$ since a complete binary tree has $\Omega(m)$ leaves. The result now follows from the fact that the maximum degree Δ of a node is three for any binary tree. The case of $\Delta = m - 1$ follows immediately from Theorem 3.5 applied on stars. $\qquad\square$

Corollary 3.4. *There exists two trees $\mathcal{T}$ and $\mathcal{T}'$ with diameters $\omega = 2$ and $\omega = m/2$ respectively such that any deterministic* PREFERENCE ELICITATION *algorithm for profiles which are single peaked on $\mathcal{T}$ and $\mathcal{T}'$ respectively has query complexity $\Omega(mn \log m)$.*

Proof. The $\omega = 2$ and $\omega = m/2$ cases follow from Theorem 3.5 applied on star and caterpillar graphs with a central path of length $m/2$ respectively. $\qquad\square$

We next consider the parameter pathwidth of the underlying single peaked tree. We immediately get the following result for PREFERENCE ELICITATION on trees with pathwidths one or $\log m$ from Theorem 3.5 and the fact that the pathwidths of a star and a complete binary tree are one and $\log m$ respectively.

Corollary 3.5. *There exist two trees $\mathcal{T}$ and $\mathcal{T}'$ with pathwidths one and $\log m$ respectively such that any deterministic* PREFERENCE ELICITATION *algorithm for single peaked profiles on $\mathcal{T}$ and $\mathcal{T}'$ respectively has query complexity $\Omega(mn \log m)$.*

Our final result, which again follows from Theorem 3.5 applied of caterpillar graphs with a central path of length $m - d$, shows that the bound in Theorem 3.3 is tight.

Theorem 3.6. *For any integers m and d with $1 \leqslant d \leqslant m/4$, there exists a tree $\mathcal{T}$ with distance d from path such that any deterministic* PREFERENCE ELICITATION *algorithm for profiles which are single peaked on $\mathcal{T}$ has query complexity $\Omega(mn + nd \log d)$.*

Proof. Consider the caterpillar graph where the length of the central path $\mathcal{Q}$ is $m - d$; there exists such a caterpillar graph since $d \leqslant m/4$. Consider the order $\succ = \pi \succ \sigma$ of the set of candidates $\mathcal{C}$ where π is an order of the candidates in $\mathcal{Q}$ which is single peaked on $\mathcal{Q}$ and σ is any order of the candidates in $\mathcal{C} \setminus \mathcal{Q}$. Clearly, $\succ$ is single peaked on the tree $\mathcal{T}$. Any elicitation algorithm $\mathcal{A}$ needs to make $\Omega(m - d)$ queries involving only the candidates in $\mathcal{Q}$ to elicit π

due to [Con09] and $\Omega(d \log d)$ queries to elicit σ due to sorting lower bound for every voter. This proves the statement. $\qquad\square$

3.5 Results for Weak Condorcet Winner

In this section, we present our results for the query complexity for finding a weak Condorcet winner in profiles that are single peaked on trees.

3.5.1 Algorithmic Results for Weak Condorcet Winner

We now show that we can find a weak Condorcet winner of profiles that are single peaked on trees using fewer queries than the number of queries needed to find the profile itself. We note that if a Condorcet winner is guaranteed to exist for a profile, then it can be found using $\mathcal{O}(mn)$ queries — we pit an arbitrary pair of candidates x, y and use $\mathcal{O}(n)$ queries to determine if x defeats y. We push the winning candidate forward and repeat the procedure, clearly requiring at most m rounds. Now, if a profile is single peaked with respect to a tree, and there are an odd number of voters, then we have a Condorcet winner and the procedure that we just described would work. Otherwise, we simply find a Condorcet winner among the first $(n-1)$ voters. It can be easily shown that such a winner is one of the weak Condorcet winners for the overall profile, and we therefore have the following results. We begin with the following general observation.

Observation 3.3. *Let $\mathcal{P}$ be a profile where a Condorcet winner is guaranteed to exist. Then we can find the Condorcet winner of $\mathcal{P}$ by making $\mathcal{O}(mn)$ queries.*

Proof. For any two candidates $x, y \in \mathcal{C}$ we find whether x defeats y or not by simply asking all the voters to compare x and y; this takes $\mathcal{O}(n)$ queries. The algorithms maintains a set $\mathcal{S}$ of candidates which are *potential* Condorcet winners. We initialize $\mathcal{S}$ to $\mathcal{C}$. In each iteration we pick any two candidates $x, y \in \mathcal{S}$ from $\mathcal{S}$, remove x from $\mathcal{S}$ if x does not defeat y and vice versa using $\mathcal{O}(n)$ query complexity until $\mathcal{S}$ is singleton. After at most $m-1$ iterations, the set $\mathcal{S}$ will be singleton and contain only Condorcet winner since we find a candidate which is not a Condorcet winner in every iteration and thus the size of the set $\mathcal{S}$ decreases by at least one in every iteration. This gives a query complexity bound of $\mathcal{O}(mn)$. $\qquad\square$

Using Observation 3.3 we now develop a WEAK CONDORCET WINNER algorithm with query complexity $\mathcal{O}(mn)$ for profiles that are single peaked on trees.

Theorem 3.7. *There is a* WEAK CONDORCET WINNER *algorithm with query complexity* $\mathcal{O}(mn)$ *for single peaked profiles on trees.*

Proof. Let $\mathcal{P} = (\succ_i)_{i \in [n]}$ be a profile which is single peaked on a tree $\mathcal{T}$. If n is an odd integer, then we know that there exists a Condorcet winner in $\mathcal{P}$ since no two candidates can tie and there always exists at least one weak Condorcet winner in every single peaked profile on trees. Hence, if n is an odd integer, then we use Observation 3.3 to find a weak Condorcet winner which is the Condorcet winner too. Hence let us now assume that n is an even integer. Notice that $\mathcal{P}_{-1} = (\succ_2, \ldots, \succ_n)$ is also single peaked on $\mathcal{T}$ and has an odd number of voters and thus has a Condorcet winner. We use Observation 3.3 to find the Condorcet winner c of $\mathcal{P}_{-1}$ and output c as a weak Condorcet winner of $\mathcal{P}$. We claim that c is a weak Condorcet winner of $\mathcal{P}$. Indeed otherwise there exists a candidate x other than c who defeats c in $\mathcal{P}$. Since n is an even integer, x must defeat c by a margin of at least two (since all pairwise margins are even integers) in $\mathcal{P}$. But then x also defeats c by a margin of at least one in $\mathcal{P}_{-1}$. This contradicts the fact that c is the Condorcet winner of $\mathcal{P}_{-1}$. $\qquad\square$

For the special case of single peaked profiles, we can do even better. Here we take advantage of the fact that a "median candidate" [MCWG95] is guaranteed to be a weak Condorcet winner. We make $\mathcal{O}(\log m)$ queries per vote to find the candidates placed at the first position of all the votes using the algorithm in [Con09] and find a median candidate to have an algorithm for finding a weak Condorcet winner. If a profile is single peaked (on a path), then there is a WEAK CONDORCET WINNER algorithm with query complexity $\mathcal{O}(n \log m)$ as shown below. Let us define the frequency $f(x)$ of a candidate $x \in \mathcal{C}$ to be the number of votes where x is placed at the first position. Then we know that a median candidate according to the single peaked ordering of the candidates along with their frequencies as defined above is a weak Condorcet winner for single peaked profiles [MCWG95].

Theorem 3.8. *There is a* WEAK CONDORCET WINNER *algorithm with query complexity* $\mathcal{O}(n \log m)$ *for single peaked profiles (on a path).*

Proof. Let $\mathcal{P}$ be a profile that is single peaked with respect to an ordering $\succ \in \mathcal{L}(\mathcal{C})$ of candidates. Then we find, for every voter v, the candidate the voter v places at the first position using $\mathcal{O}(\log m)$ queries using the algorithm in [Con09] and return a median candidate. $\qquad\square$

The next result uses Theorem 3.8 on paths in a path cover of the single peaked tree eliminating the case of even number of voters by the idea of setting aside one voter that was used in Theorem 3.7.

Theorem 3.9. *Let $\mathcal{T}$ be a tree with path cover number at most k. Then there is an algorithm for* WEAK CONDORCET WINNER *for profiles which are single peaked on $\mathcal{T}$ with query complexity $\mathcal{O}(nk \log m)$.*

Recalling that the number of leaves bounds the path cover number, we have the following consequence.

Corollary 3.6. *Let $\mathcal{T}$ be a tree with ℓ leaves. Then there is an algorithm for* WEAK CONDORCET WINNER *for profiles which are single peaked on $\mathcal{T}$ with query complexity $\mathcal{O}(n\ell \log m)$.*

From Theorem 3.7 and 3.8 we have the following result for any arbitrary tree.

Theorem 3.10. *Let $\mathcal{T}$ be a tree with path cover number at most k. Then there is an algorithm for* WEAK CONDORCET WINNER *for profiles which are single peaked on $\mathcal{T}$ with query complexity $\mathcal{O}(nk \log m)$.*

Proof. Let $\mathcal{P}$ be the input profile and $\mathcal{Q}_i = (\mathcal{X}_i, \mathcal{E}_i)$ $i \in [t]$ be $t(\leqslant k)$ disjoint paths that cover the tree $\mathcal{T}$. Here again, if the number of voters is even, then we remove any arbitrary voter and the algorithm outputs the Condorcet winner of the rest of the votes. The correctness of this step follows from the proof of Theorem 3.7. Hence we assume, without loss of generality, that we have an odd number of voters. The algorithm proceeds in two stages. In the first stage, we find the Condorcet winner w_i of the profile $\mathcal{P}(\mathcal{X}_i)$ for every $i \in [t]$ using Theorem 3.8. The query complexity of this stage is $\mathcal{O}(n \sum_{i \in [t]} \log |\mathcal{X}_i|) = \mathcal{O}(nt \log(m/t))$. In the second stage, we find the Condorcet winner w of the profile $\mathcal{P}(\{w_i : i \in [t]\})$ using Theorem 3.7 and output w. The query complexity of the second stage is $\mathcal{O}(nt \log t)$. Hence the overall query complexity of the algorithm is $\mathcal{O}(nt \log(m/t)) + \mathcal{O}(nt \log t) = \mathcal{O}(nk \log m)$. $\qquad\square$

3.5.2 Lower Bounds for Weak Condorcet Winner

We now state the lower bounds pertaining to WEAK CONDORCET WINNER. First, we show that any algorithm for WEAK CONDORCET WINNER for single peaked profiles on stars has query complexity $\Omega(mn)$, showing that the bound of Theorem 3.7 is tight.

Theorem 3.11. *Any deterministic* WEAK CONDORCET WINNER *algorithm for single peaked profiles on stars has query complexity $\Omega(mn)$.*

Proof. Let $\mathcal{T}$ be a star with center vertex c. We now design an oracle that will "force" any WEAK CONDORCET WINNER algorithm $\mathcal{A}$ for single peaked profiles on $\mathcal{T}$ to make $\Omega(mn)$ queries. For every voter v, the oracle maintains a set of *"marked"* candidates which can not be placed at the first position of the preference of v. Suppose the oracle receives a query to compare two candidates x and y for a voter ℓ. If the order between x and y for the voter ℓ follows from the answers the oracle has already provided to all the queries for the voter ℓ, then the oracle answers accordingly. Otherwise it answers $x \succ_\ell y$ if y is unmarked and marks y; otherwise the oracle answers $y \succ_\ell x$ and marks x. Notice that the oracle marks at most one unmarked candidate every time it is queried. We now claim that there must be at least $n/10$ votes which have been queried at least $m/4$ times. If not, then there exists $n - n/10 = 9n/10$ votes each of which has at least $m - m/4 = 3m/4$ candidates unmarked. In such a scenario, there exists a constant N_0 such that for every $m, n > N_0$, we have at least two candidates x and y who are unmarked in at least $(\lfloor n/2 \rfloor + 1)$ votes each. Now if the algorithm outputs x, then we put y at the first position in at least $(\lfloor n/2 \rfloor + 1)$ votes and at the second position in the rest of the votes and this makes y the (unique) Condorcet winner. If the algorithm does not output x, then we put x at the first position in at least $(\lfloor n/2 \rfloor + 1)$ votes and at the second position in the rest of the votes and this makes x the (unique) Condorcet winner. Hence the algorithm fails to output correctly in both the cases contradicting the correctness of the algorithm. Also the resulting profile is single peaked on $\mathcal{T}$ with center at y in the first case and at x in the second case. Therefore the algorithm $\mathcal{A}$ must have query complexity $\Omega(mn)$. $\qquad\square$

Our next result uses an adversary argument, and shows that the query complexity for WEAK CONDORCET WINNER for single peaked profiles in Theorem 3.8 is essentially optimal, provided that the queries to different voters are not interleaved, as is the case with our algorithm.

Theorem 3.12. *Any deterministic* WEAK CONDORCET WINNER *algorithm for single peaked profiles which does not interleave the queries to different voters has query complexity* $\Omega(n \log m)$.

Proof. Let a profile $\mathcal{P}$ be single peaked with respect to the ordering of the candidates $\succ = c_1 \succ c_2 \succ \cdots \succ c_m$. The oracle maintains two indices ℓ and r for every voter such that any candidate from $\{c_\ell, c_{\ell+1}, \ldots, c_r\}$ can be placed at the first position of the preference of the voter v and still be consistent with all the answers provided by the oracle for v till now and single peaked with respect to $\succ$. The algorithm initializes ℓ to one and r to m for every voter. The oracle answers any query in such a way that maximizes the new value of $r - \ell$. More

specifically, suppose the oracle receives a query to compare candidates c_i and c_j with $i < j$ for a voter ν. If the ordering between c_i and c_j follows, by applying transitivity, from the answers to the queries that have already been made so far for this voter, then the oracle answers accordingly. Otherwise the oracle answers as follows. If $i < \ell$, then the oracle answers that c_j is preferred over c_i; else if $j > r$, then the oracle answers that c_i is preferred over c_j. Otherwise (that is when $\ell \leqslant i < j \leqslant r$), if $j - \ell > r - i$, then the oracle answers that c_i is preferred over c_j and changes r to j; if $j - \ell \leqslant r - i$, then the oracle answers that c_j is preferred over c_i and changes ℓ to i. Hence whenever the oracle answers a query for a voter ν, the value of $r - \ell$ for that voter ν decreases by a factor of at most two. Suppose the election instance has an odd number of voters. Let $\mathcal{V}$ be the set of voters. Now we claim that the first $\lfloor n/5 \rfloor$ voters must be queried $(\log m - 1)$ times each. Suppose not, then consider the first voter ν' that is queried less than $(\log m - 1)$ times. Then there exist at least two candidates c_t and c_{t+1} each of which can be placed at the first position of the vote ν'. The oracle fixes the candidates at the first positions of all the votes that have not been queried till ν' is queried (and there are at least $\lceil 4n/5 \rceil$ such votes) in such a way that $\lfloor n/2 \rfloor$ voters in $\mathcal{V} \setminus \{\nu'\}$ places some candidate in the left of c_t at the first positions and $\lfloor n/2 \rfloor$ voters in $\mathcal{V} \setminus \{\nu'\}$ places some candidate in the right of c_{t+1}. If the algorithm outputs c_t as the Condorcet winner, then the oracle makes c_{t+1} the (unique) Condorcet winner by placing c_{t+1} at the top position of ν', because c_{t+1} is the unique median in this case. If the algorithm does not output c_t as the Condorcet winner, then the oracle makes c_t the (unique) Condorcet winner by placing c_t at the top position of ν', because c_t is the unique median in this case. Hence the algorithm fails to output correctly in both the cases thereby contradicting the correctness of the algorithm. $\qquad \square$

3.6 Conclusion

In this work, we present algorithms for eliciting preferences of a set of voters when the preferences are single peaked on a tree thereby significantly extending the work of [Con09]. Moreover, we show non-trivial lower bounds on the number of comparison queries any preference elicitation algorithm would need to ask in a domain of single peaked profiles on a tree. From this, we conclude that our algorithms asks optimal number of comparison queries up to constant factors. Our main finding in this work is the interesting dependencies between the number of comparison queries any preference elicitation algorithm would ask and various parameters of the single peaked tree. For example, our results show that the query complexity

for preference elicitation is a monotonically increasing function on the number of leaf nodes in the single peaked tree. On the other hand, the query complexity for preference elicitation does not directly depend on other tree parameters like, maximum degree, minimum degree, path width, diameter etc.

We then move on to study query complexity for finding a weak Condorcet winner of a set of votes which is single peaked on a tree. Here, our results show that a weak Condorcet winner can be found with much less number of queries compared to preference elicitation.

In the next chapter, we explore the preference elicitation problem again, but for another well-studied domain known as single crossing profiles.

Chapter 4

Preference Elicitation for Single Crossing Profiles

In this chapter, we consider the domain of single crossing preference profiles and study the query complexity of preference elicitation under various situations. We consider two distinct scenarios: when an ordering of the voters with respect to which the profile is single crossing is *known* a priori versus when it is *unknown*. We also consider two different access models: when the votes can be accessed at random, as opposed to when they are coming in a predefined sequence. In the sequential access model, we distinguish two cases when the ordering is known: the first is that the sequence in which the votes appear is also a single-crossing order, versus when it is not. The main contribution of our work is to provide polynomial time algorithms with low query complexity for preference elicitation in all the above six cases. Further, we show that the query complexities of our algorithms are optimal up to constant factors for all but one of the above six cases.

4.1 Introduction

Eliciting the preferences of a set of agents is a nontrivial task since we often have a large number of candidates (ranking restaurants for example) and it will be infeasible for the

agents to rank all of them. Hence it becomes important to elicit the preferences of the agents by asking them (hopefully a small number of) comparison queries only - ask an agent i to compare two candidates x and y.

Unfortunately, it turns out that one would need to ask every voter $\Omega(m \log m)$ queries to know her preference (due to sorting lower bound). However, if the preferences are not completely arbitrary, and admit additional structure, then possibly we can do better. Indeed, an affirmation of this thought comes from the work of [Con09, DM16a], who showed that we can elicit preferences using only $O(m)$ queries per voter if the preferences are single peaked. The domain of single peaked preferences has been found to be very useful in modeling preferences in political elections. We, in this work, study the problem of eliciting preferences of a set of agents for another popular domain namely the domain of single crossing profiles. A profile is called *single crossing* if the voters can be arranged in a complete order $\succ$ such that for every two candidates x and y, all the voters who prefer x over y appear consecutively in $\succ$ [Mir71, Rob77]. Single crossing profiles have been extensively used to model income of a set of agents [Rob77, MR81]. The domain is single crossing profiles are also popular among computational social scientists since many computationally hard voting rules become tractable if the underlying preference profile is single crossing [MBC$^+$16].

4.1.1 Related Work

Conitzer and Sandholm show that determining whether we have enough information at any point of the elicitation process for finding a winner under some common voting rules is computationally intractable [CS02]. They also prove in their classic paper [CS05] that one would need to make $\Omega(mn \log m)$ queries even to decide the winner for many commonly used voting rules which matches with the trivial $O(mn \log m)$ upper bound (based on sorting) for preference elicitation in unrestricted domain.

A natural question at this point is if these restricted domains allow for better elicitation algorithms as well. The answer to this is in the affirmative, and one can indeed elicit the preferences of the voters using only $O(mn)$ many queries for the domain of single peaked preference profiles [Con09]. Our work belongs to this kind of research– we investigate the number of queries one has to ask for preference elicitation in single crossing domains. When some partial information is available about the preferences, Ding and Lin prove interesting properties of what they call a deciding set of queries [DL13a]. Lu and Boutilier empirically show that several heuristics often work well [LB11b, LB11a].

4.1.2 Single Crossing Domain

A preference profile $\mathcal{P} = (\succ_1, \ldots, \succ_n)$ of n agents or voters over a set $\mathcal{C}$ of candidates is called a single crossing profile if there exists a permutation $\sigma \in \mathbb{S}_n$ of $[n]$ such that, for every two distinct candidates $x, y \in \mathcal{C}$, whenever we have $x \succ_{\sigma(i)} y$ and $x \succ_{\sigma(j)} y$ for two integers i and j with $1 \leqslant i < j \leqslant n$, we have $x \succ_{\sigma(k)} y$ for every $i \leqslant k \leqslant j$. Example 4.1 exhibits an example of a single crossing preference profile.

Example 4.1. *(Example of single crossing preference profile) Consider a set $\mathcal{C}$ of m candidates, corresponding m distinct points on the Real line, a set $\mathcal{V}$ of n voters, also corresponding n points on the Real line, and the preference of every voter are based on their distance to the candidates – given any two candidates, every voter prefers the candidate nearer to her (breaking the tie arbitrarily). Then the set of n preferences is single crossing with respect to the ordering of the voters according to the ascending ordering of their positions on the Real line.*

The following observation is immediate from the definition of single crossing profiles.

Observation 4.1. *Suppose a profile $\mathcal{P}$ is single crossing with respect to an ordering $\sigma \in \mathbb{S}_n$ of votes. Then $\mathcal{P}$ is single crossing with respect to the ordering $\overleftarrow{\sigma}$ too.*

4.1.3 Single Crossing Width

A preference $\mathcal{P} = (\succ_1, \ldots, \succ_n)$ of n agents or voters over a set $\mathcal{C}$ of candidates is called a single crossing profile with width w if the set of candidates $\mathcal{C}$ can be partitioned into $(\mathcal{C}_i)_{i \in [k]}$ such that $|\mathcal{C}_i| \leqslant w$ for every i and for every two candidates $x \in \mathcal{C}_\ell$ and $y \in \mathcal{C}_t$ with $\ell \neq t$ from two different subsets of the partition, whenever we have $x \succ_{\sigma(i)} y$ and $x \succ_{\sigma(j)} y$ for two integers i and j with $1 \leqslant i < j \leqslant n$, we have $x \succ_{\sigma(k)} y$ for every $i \leqslant k \leqslant j$.

4.1.4 Our Contribution

In this work we present novel algorithms for preference elicitation for the domain of single crossing profiles in various settings. We consider two distinct situations: when an ordering of the voters with respect to which the profile is single crossing is *known* versus when it is *unknown*. We also consider different access models: when the votes can be accessed at random, as opposed to when they are coming in a pre-defined sequence. In the sequential access model, we distinguish two cases when the ordering is known: the first is that sequence

53

in which the votes appear is also a single-crossing order, versus when it is not. We also prove lower bounds on the query complexity of preference elicitation for the domain of single crossing profiles; these bounds match the upper bounds up to constant factors (for a large number of voters) for all the six scenarios above except the case when we know a single crossing ordering of the voters and we have a random access to the voters; in this case, the upper and lower bounds match up to a factor of $\mathcal{O}(m)$. We summarize our results in Table 6.1.

Ordering	Access model	Query Complexity	
		Upper Bound	Lower Bound
Known	Random	$\mathcal{O}(m^2 \log n)$ [Lemma 4.1]	$\Omega(m \log m + m \log n)$ [Theorem 4.1]
	Sequential single crossing order	$\mathcal{O}(mn + m^2)$ [Theorem 4.2]	$\Omega(m \log m + mn)$ [Theorem 4.3]
	Sequential any order	$\mathcal{O}(mn + m^2 \log n)$ [Theorem 4.4]	
Unknown	Sequential any order	$\mathcal{O}(mn + m^3 \log m)$ [Theorem 4.5]	
	Random	$\mathcal{O}(mn + m^3 \log m)$ [Corollary 4.1]	$\Omega(m \log m + mn)$ [Theorem 4.6]

Table 4.1: Summary of Results for preference elicitation for single crossing profiles.

We then extend our results to domains which are single crossing of width w in Section 4.3.3. We also prove that a weak Condorcet winner and the Condorcet winner (if it exists) of a single crossing preference profile can be found out with much less number of queries in Section 4.4.

4.2 Problem Formulation

The query and cost model is the same as Definition 3.1 in Chapter 3. We recall it below for ease of access.

Suppose we have a profile $\mathcal{P}$ with n voters and m candidates. Let us define a function QUERY($x \succ_\ell y$) for a voter ℓ and two different candidates x and y to be TRUE if the voter ℓ prefers the candidate x over the candidate y and FALSE otherwise. We now formally define the problem.

Definition 4.1. PREFERENCE ELICITATION

Given an oracle access to QUERY *(·) for a single crossing profile* $\mathcal{P}$*, find* $\mathcal{P}$*.*

For two distinct candidates $x, y \in \mathcal{C}$ and a voter ℓ, we say a PREFERENCE ELICITATION algorithm $\mathcal{A}$ *compares* candidates x and y for voter ℓ, if $\mathcal{A}$ makes a call to either QUERY$(x \succ_\ell y)$ or QUERY$(y \succ_\ell x)$. We define the number of queries made by the algorithm $\mathcal{A}$, called the *query complexity* of $\mathcal{A}$, to be the number of distinct tuples $(\ell, x, y) \in \mathcal{V} \times \mathcal{C} \times \mathcal{C}$ with $x \neq y$ such that the algorithm $\mathcal{A}$ compares the candidates x and y for voter ℓ. Notice that, even if the algorithm $\mathcal{A}$ makes multiple calls to QUERY $(·)$ with same tuple (ℓ, x, y), we count it only once in the query complexity of $\mathcal{A}$. This is without loss of generality since we can always implement a wrapper around the oracle which memorizes all the calls made to the oracle so far and whenever it receives a duplicate call, it replies from its memory without "actually" making a call to the oracle. We say two query complexities $q(m, n)$ and $q'(m, n)$ are tight up to a factor of ℓ *for a large number of voters* if $1/\ell \leqslant \lim_{n \to \infty} q(m,n)/q'(m,n) \leqslant \ell$.

Note that by using a standard sorting routine like merge sort, we can fully elicit an unknown preference using $\mathcal{O}(m \log m)$ queries. We state this explicitly below, as it will be useful in our subsequent discussions.

Observation 4.2. *There is a* PREFERENCE ELICITATION *algorithm with query complexity* $\mathcal{O}(m \log m)$ *for eliciting one vote from a single crossing preference profile.*

4.2.1 Model of Input

We study two models of input for PREFERENCE ELICITATION for single crossing profiles.

▷ **Random access to voters:** In this model, we have a set of voters and we are allowed to ask any voter to compare any two candidates at any point of time. Moreover, we are also allowed to interleave the queries to different voters. Random access to voters is the model of input for elections within an organization where every voter belongs to the organization and can be queried any time.

▷ **Sequential access to voters:** In this model, voters are arriving in a sequential manner one after another to the system. Once a voter ℓ arrives, we can query voter ℓ as many times as we like and then we "release" the voter ℓ from the system to access the next voter in the queue. Once voter ℓ is released, it can never be queried again. Sequential

access to voters is indeed the model of input in many practical elections scenarios such as political elections, restaurant ranking etc.

4.3 Results for Preference Elicitation

In this section, we present our technical results. We first consider the (simpler) situation when one single crossing order is known, and then turn to the case when no single crossing order is a priori known. In both cases, we explore all the relevant access models.

4.3.1 Known Single Crossing Order

We begin with a simple PREFERENCE ELICITATION algorithm when we are given a random access to the voters and one single crossing ordering is known.

Lemma 4.1. *Suppose a profile $\mathcal{P}$ is single crossing with respect to a known permutation of the voters. Given a random access to voters, there is a* PREFERENCE ELICITATION *algorithm with query complexity* $\mathcal{O}(m^2 \log n)$.

Proof. By renaming, we assume that the profile is single crossing with respect to the identity permutation of the votes. Now, for every $\binom{m}{2}$ pair of candidates $\{x, y\} \subset \mathcal{C}$, we perform a binary search over the votes to find the index $i(\{x, y\})$ where the ordering of x and y changes. We now know how any voter j orders any two candidates x and y from $i(\{x, y\})$ and thus we have found $\mathcal{P}$. $\qquad\square$

Interestingly, the simple algorithm in Lemma 4.1 turns out to be optimal up to a multiplicative factor of $\mathcal{O}(m)$ as we prove next. The idea is to "pair up" the candidates and design an oracle which "hides" the vote where the ordering of the two candidates in any pair (x, y) changes unless it receives at least $(\log m - 1)$ queries involving only these two candidates x and y. We formalize this idea below. Observe that, when a single crossing ordering of the voters in known, we can assume without loss of generality, by renaming the voters, that the preference profile is single crossing with respect to the identity permutation of the voters.

Theorem 4.1. *Suppose a profile $\mathcal{P}$ is single crossing with respect to the identity permutation of votes. Given random access to voters, any deterministic* PREFERENCE ELICITATION *algorithm has query complexity* $\Omega(m \log m + m \log n)$.

Proof. The $\Omega(m \log m)$ bound follows from the query complexity lower bound of sorting and the fact that any profile consisting of only one preference $\succ \in \mathcal{L}(\mathcal{C})$ is single crossing. Let $\mathcal{C} = \{c_1, \ldots, c_m\}$ be the set of m candidates where m is an even integer. Consider the ordering $Q = c_1 \succ c_2 \succ \cdots \succ c_m \in \mathcal{L}(\mathcal{C})$ and the following pairing of the candidates: $\{c_1, c_2\}, \{c_3, c_4\}, \ldots, \{c_{m-1}, c_m\}$. Our oracle answers QUERY $(\cdot)$ as follows. The oracle fixes the preferences of the voters one and n to be Q and $\overleftarrow{Q}$ respectively. For every odd integer $i \in [m]$, the oracle maintains θ_i (respectively β_i) which corresponds to the largest (respectively smallest) index of the voter for whom (c_i, c_{i+1}) has already been queried and the oracle answered that the voter prefers c_i over c_{i+1} (c_{i+1} over c_i respectively). The oracle initially sets $\theta_i = 1$ and $\beta_i = n$ for every odd integer $i \in [m]$. Suppose oracle receives a query to compare candidates c_i and c_j for $i, j \in [m]$ with $i < j$ for a voter ℓ. If i is an even integer or $j - i \geqslant 2$ (that is, c_i and c_j belong to different pairs), then the oracle answers that the voter ℓ prefers c_i over c_j. Otherwise we have $j = i+1$ and i is an odd integer. The oracle answers the query to be $c_i \succ c_{i+1}$ and updates θ_i to ℓ keeping β_i fixed if $|\ell - \theta_i| \leqslant |\ell - \beta_i|$ and otherwise answers $c_{i+1} \succ c_i$ and updates β_i to ℓ keeping θ_i fixed (that is, the oracle answers according to the vote which is closer to the voter ℓ between θ_i and β_i and updates θ_i or β_i accordingly). If the pair (c_i, c_{i+1}) is queried less than $(\log n - 2)$ times, then we have $\beta_i - \theta_i \geqslant 2$ at the end of the algorithm since every query for the pair (c_i, c_{i+1}) decreases $\beta_i - \theta_i$ by at most a factor of two and we started with $\beta_i - \theta_i = n - 1$. Consider a voter κ with $\theta_i < \kappa < \beta_i$. If the elicitation algorithm outputs that the voter κ prefers c_i over c_{i+1} (respectively c_{i+1} over c_i), then the oracle sets all the voters κ' with $\theta_i < \kappa' < \beta_i$ to prefer c_{i+1} over c_i (respectively c_i over c_{i+1}). Clearly, the algorithm does not elicit the preference of the voter κ correctly. Also, the profile is single crossing with respect to the identity permutation of the voters and consistent with the answers of all the queries made by the algorithm. Hence, for every odd integer $i \in [m]$, the algorithm must make at least $(\log n - 1)$ queries for the pair (c_i, c_{i+1}) thereby making $\Omega(m \log n)$ queries in total. $\qquad\square$

We now present our PREFERENCE ELICITATION algorithm when we have a sequential access to the voters according to a single crossing order. We elicit the preference of the first voter using Observation 4.2. From second vote onwards, we simply use the idea of *insertion sort* relative to the previously elicited vote [Cor09]. Since we are using insertion sort, any particular voter may be queried $\mathcal{O}(m^2)$ times. However, we are able to bound the query complexity of our algorithm due to two fundamental reasons: (i) consecutive preferences will often be almost similar in a single crossing ordering, (ii) our algorithm takes only $\mathcal{O}(m)$

queries to elicit the preference of the current voter if its preference is indeed the same as the preference of the voter preceding it. In other words, every time we have to "pay" for shifting a candidate further back in the current vote, the relative ordering of that candidate with all the candidates that it jumped over is now fixed, because for these pairs, the one permitted crossing is now used up. We begin with presenting an important subroutine called Elicit($\cdot$) which finds the preference of a voter ℓ given another preference $\mathcal{R}$ by performing an insertion sort using $\mathcal{R}$ as the order of insertion.

Algorithm 1 Elicit($\mathcal{C}$, $\mathcal{R}$, ℓ)

Input: A set of candidates $\mathcal{C} = \{c_i : i \in [m]\}$, an ordering $\mathcal{R} = c_1 \succ \cdots \succ c_m$ of $\mathcal{C}$, a voter ℓ
Output: Preference ordering $\succ_\ell$ of voter ℓ on $\mathcal{C}$
 1: $\mathcal{Q} \leftarrow c_1$ $\triangleright$ $\mathcal{Q}$ will be the preference of the voter ℓ
 2: **for** $i \leftarrow 2$ to m **do** $\triangleright$ c_i is inserted in the ith iteration
 3:      Scan $\mathcal{Q}$ linearly *from index* $i - 1$ *to* 1 to find the index j where c_i should be inserted according to the preference of voter ℓ and insert c_i in $\mathcal{Q}$ at j
 4: **end for**
 5: **return** $\mathcal{Q}$

For the sake of the analysis of our algorithm, let us introduce a few terminologies. Given two preferences $\succ_1$ and $\succ_2$, we call a pair of candidates $(x, y) \in \mathcal{C} \times \mathcal{C}, x \neq y$, *good* if both $\succ_1$ and $\succ_2$ order them in a same way; a pair of candidates is called *bad* if it is not good. We divide the number of queries made by our algorithm into two parts: goodCost($\cdot$) and badCost($\cdot$) which are the number of queries made between good and respectively bad pair of candidates. In what follows, we show that goodCost($\cdot$) for Elicit($\cdot$) is small and the total badCost($\cdot$) across all the runs of Elicit($\cdot$) is small.

Lemma 4.2. *The goodCost(Elicit($\mathcal{C}$, $\mathcal{R}$, ℓ)) of Elicit($\mathcal{C}$, $\mathcal{R}$, ℓ) is $\mathcal{O}(m)$ (good is with respect to the preferences $\mathcal{R}$ and $\succ_\ell$).*

Proof. Follows immediately from the observation that in any iteration of the for loop at line 2 in Algorithm 1, only one good pair of candidates are compared. □

We now use Algorithm 1 iteratively to find the profile. We present the pseudocode in Algorithm 2 which works for the more general setting where a single crossing ordering is known but the voters are arriving in any arbitrary order π. We next compute the query complexity of Algorithm 2 when voters are arriving in a single crossing order.

58

Algorithm 2 PreferenceElicit(π)

Input: $\pi \in \mathbb{S}_n$
Output: Profile of all the voters
 1: $\Omega[\pi(1)] \leftarrow$ Elicit $\succ_{\pi(1)}$ using Observation 4.2 $\triangleright$ Ω stores the profile
 2: $\mathfrak{X} \leftarrow \{\pi(1)\}$ $\triangleright$ Set of voters' whose preferences have already been elicited
 3: **for** $i \leftarrow 2$ to n **do** $\triangleright$ Elicit the preference of voter $\pi(i)$ in iteration i
 4:      $k \leftarrow \min_{j \in \mathfrak{X}} |j - i|$ $\triangleright$ Find the closest known preference
 5:      $\mathcal{R} \leftarrow \Omega[k], \mathfrak{X} \leftarrow \mathfrak{X} \cup \{\pi(i)\}$
 6:      $\Omega[\pi(i)] \leftarrow$ Elicit$(\mathcal{C}, \mathcal{R}, \pi(i))$
 7: **end for**
 8: **return** Ω

Theorem 4.2. *Assume that the voters are arriving sequentially according to an order with respect to which a profile $\mathcal{P}$ is single crossing. Then there is a* PREFERENCE ELICITATION *algorithm with query complexity $\mathcal{O}(mn + m^2)$.*

Proof. By renaming, let us assume, without loss of generality, that the voters are arriving according to the identity permutation id_n of the voters and the profile $\mathcal{P}$ is single crossing with respect to id_n. Let the profile $\mathcal{P}$ be $(P_1, P_2, \ldots, P_n) \in \mathcal{L}(\mathcal{C})^n$. For two candidates $x, y \in \mathcal{C}$ and a voter $i \in \{2, \ldots, n\}$, let us define a variable $b(x, y, i)$ to be one if x and y are compared for the voter i by Elicit$(\mathcal{C}, P_{i-1}, i)$ and (x, y) is a bad pair of candidates with respect to the preferences of voter i and $i - 1$; otherwise $b(x, y, i)$ is defined to be zero. Then we have the following.

$$
\begin{aligned}
& \text{CostPreferenceElicit}(id_n) \\
=~& \mathcal{O}(m \log m) + \sum_{i=2}^{n} \Big(\text{goodCost}(\text{QUERY}(\mathcal{C}, P_{i-1}, i)) + \text{badCost}(\text{QUERY}(\mathcal{C}, P_{i-1}, i)) \Big) \\
\leqslant~& \mathcal{O}(m \log m + mn) + \sum_{i=2}^{n} \text{badCost}(\text{QUERY}(\mathcal{C}, P_{i-1}, i)) \\
=~& \mathcal{O}(m \log m + mn) + \sum_{(x,y) \in \mathcal{C} \times \mathcal{C}} \left(\sum_{i=2}^{n} b(x, y, i) \right) \\
\leqslant~& \mathcal{O}(m \log m + mn) + \sum_{(x,y) \in \mathcal{C} \times \mathcal{C}} 1 \\
=~& \mathcal{O}(mn + m^2)
\end{aligned}
$$

The first inequality follows from Lemma 4.2, the second equality follows from the definition of $b(x, y, i)$, and the second inequality follows from the fact that $\sum_{i=2}^{n} b(x, y, i) \leqslant 1$ for every pair of candidates $(x, y) \in \mathcal{C}$ since the profile $\mathcal{P}$ is single crossing. $\square$

We show next that, when the voters are arriving in a single crossing order, the query complexity upper bound in Theorem 4.3 is tight for a large number of voters up to constant factors. The idea is to pair up the candidates in a certain way and argue that the algorithm must compare the candidates in every pair for every voter thereby proving a $\Omega(mn)$ lower bound on query complexity.

Theorem 4.3. *Assume that the voters are arriving sequentially according to an order with respect to which a profile $\mathcal{P}$ is single crossing. Then any deterministic* PREFERENCE ELICITATION *algorithm has query complexity $\Omega(m \log m + mn)$.*

Proof. The $\Omega(m \log m)$ bound follows from the fact that any profile consisting of only one preference $P \in \mathcal{L}(\mathcal{C})$ is single crossing. By renaming, let us assume without loss of generality that the profile $\mathcal{P}$ is single crossing with respect to the identity permutation of the voters. Suppose we have an even number of candidates and $\mathcal{C} = \{c_1, \ldots, c_m\}$. Consider the order $\mathcal{Q} = c_1 \succ c_2 \succ \cdots \succ c_m$ and the pairing of the candidates $\{c_1, c_2\}, \{c_3, c_4\}, \ldots, \{c_{m-1}, c_m\}$. The oracle answers all the query requests consistently according to the order Q till the first voter κ for which there exists at least one odd integer $i \in [m]$ such that the pair (c_i, c_{i+1}) is not queried. If there does not exist any such κ, then the algorithm makes at least $mn/2$ queries thereby proving the statement. Otherwise, let κ be the first vote such that the algorithm does not compare c_i and c_{i+1} for some odd integer $i \in [m]$. The oracle answers the queries for the rest of the voters $\{\kappa + 1, \ldots, n\}$ according to the order $Q' = c_1 \succ c_2 \succ \cdots \succ c_{i-1} \succ c_{i+1} \succ c_i \succ c_{i+2} \succ \cdots \succ c_m$. If the algorithm orders $c_i \succ_\kappa c_{i+1}$ in the preference of the voter κ, then the oracle sets the preference of the voter κ to be $\mathcal{Q}'$. On the other hand, if the algorithm orders $c_{i+1} \succ_\kappa c_i$ in the preference of voter κ, then the oracle sets the preference of voter κ to be $\mathcal{Q}$. Clearly, the elicitation algorithm fails to correctly elicit the preference of the voter κ. However, the profiles for both the cases are single crossing with respect to the identity permutation of the voters and are consistent with the answers given to all the queries made by the algorithm. Hence, the algorithm must make at least $mn/2$ queries. $\square$

We next move on to the case when we know a single crossing order of the voters; however, the voters arrive in an arbitrary order $\pi \in \mathbb{S}_n$. The idea is to call the function Elicit($\mathcal{C}, \mathcal{R}, i$) where the current voter is the voter i and $\mathcal{R}$ is the preference of the voter which is closest to

i according to a single crossing ordering and whose preference has already been elicited by the algorithm.

Theorem 4.4. *Assume that a profile $\mathcal{P}$ is known to be single crossing with respect to a known ordering of voters $\sigma \in \mathbb{S}_n$. However, the voters are arriving sequentially according to an arbitrary order $\pi \in \mathbb{S}_n$ which may be different from σ. Then there is a* PREFERENCE ELICITATION *algorithm with query complexity $\mathcal{O}(mn + m^2 \log n)$.*

Proof. By renaming, let us assume, without loss of generality, that the profile $\mathcal{P}$ is single peaked with respect to the identity permutation of the voters. Let the profile $\mathcal{P}$ be $(P_1, P_2, \ldots, P_n) \in \mathcal{L}(\mathcal{C})^n$. Let $f : [n] \longrightarrow [n]$ be the function such that $f(i)$ is the k corresponding to the i at line 4 in Algorithm 2. For candidates $x, y \in \mathcal{C}$ and voter ℓ, we define $b(x, y, \ell)$ analogously as in the proof of Theorem 4.2. We claim that $B(x, y) = \sum_{i=2}^{n} b(x, y, i) \leqslant \log n$. To see this, we consider any arbitrary pair $(x, y) \in \mathcal{C} \times \mathcal{C}$. Let the set of indices of the voters that have arrived immediately after the first time (x, y) contributes to $B(x, y)$ be $\{i_1, i_2, \ldots, i_t\}$. Without loss of generality, let us assume $i_1 < i_2 < \cdots < i_t$. Again, without loss of generality, let us assume that voters $i_1, i_2, \ldots, i_j$ prefer x over y and voters $i_{j+1}, \ldots, i_t$ prefer y over x. Let us define Δ to be the difference between smallest index of the voter who prefers y over x and the largest index of the voter who prefers x over y. Hence, we currently have $\Delta = i_{j+1} - i_j$. A crucial observation is that if a new voter ℓ contributes to $B(x, y)$ then we must necessarily have $i_j < \ell < i_{j+1}$. Another crucial observation is that whenever a new voter contributes to $B(x, y)$, the value of Δ gets reduced at least by a factor of two by the choice of k at line 4 in Algorithm 2. Hence, the pair (x, y) can contribute at most $(1 + \log \Delta) = \mathcal{O}(\log n)$ to $B(x, y)$ since we have $\Delta \leqslant n$ to begin with. Then we have the following.

61

$$\text{CostPreferenceElicit}(\pi)$$

$$= \; \mathcal{O}(m \log m) + \sum_{i=2}^{n} \text{goodCost}(\text{Query}(\mathcal{C}, P_{f(i)}, i)) + \text{badCost}(\text{Query}(\mathcal{C}, P_{f(i)}, i))$$

$$\leqslant \; \mathcal{O}(m \log m + mn) + \sum_{i=2}^{n} \text{badCost}(\text{Query}(\mathcal{C}, P_{f(i)}, i))$$

$$= \; \mathcal{O}(m \log m + mn) + \sum_{(x,y) \in \mathcal{C} \times \mathcal{C}} \sum_{i=2}^{n} b(x, y, i)$$

$$\leqslant \; \mathcal{O}(m \log m + mn) + \sum_{(x,y) \in \mathcal{C} \times \mathcal{C}} \log n$$

$$= \; \mathcal{O}(mn + m^2 \log n)$$

The first inequality follows from Lemma 4.2, the second equality follows from the definition of $b(x, y, i)$, and the second inequality follows from the fact that $\sum_{i=2}^{n} b(x, y, i) \leqslant \log n$. $\qquad \square$

4.3.2 Unknown Single Crossing Order

We now turn our attention to PREFERENCE ELICITATION for single crossing profiles when no single crossing ordering is known. Before we present our PREFERENCE ELICITATION algorithm for this setting, let us first prove a few structural results about single crossing profiles which we will use crucially later. We begin with showing an upper bound on the number of distinct preferences in any single crossing profile.

Lemma 4.3. *Let $\mathcal{P}$ be a profile on a set $\mathcal{C}$ of candidates which is single crossing. Then the number of distinct preferences in $\mathcal{P}$ is at most $\binom{m}{2} + 1$.*

Proof. By renaming, let us assume, without loss of generality, that the profile $\mathcal{P}$ is single crossing with respect to the identity permutation of the voters. We now observe that whenever the ith vote is different from the $(i+1)^{\text{th}}$ vote for some $i \in [n-1]$, there must exist a pair of candidates $(x, y) \in \mathcal{C} \times \mathcal{C}$ whom the ith vote and the $(i+1)^{\text{th}}$ vote order differently. Now the statement follows from the fact that, for every pair of candidates $(a, b) \in \mathcal{C} \times \mathcal{C}$, there can exist at most one $i \in [n-1]$ such that the ith vote and the $(i+1)^{\text{th}}$ vote order a and b differently. $\qquad \square$

We show next that in every single crossing preference profile $\mathcal{P}$ where all the preferences are *distinct*, there exists a pair of candidates $(x, y) \in \mathcal{C} \times \mathcal{C}$ such that nearly half of the voters in $\mathcal{P}$ prefer x over y and the other voters prefer y over x.

Lemma 4.4. *Let $\mathcal{P}$ be a preference profile of n voters such that all the preferences are distinct. Then there exists a pair of candidates $(x, y) \in \mathcal{C}$ such that x is preferred over y in at least $\lfloor n/2 \rfloor$ preferences and y is preferred over x in at least $\lfloor n/2 \rfloor$ preferences in $\mathcal{P}$.*

Proof. Without loss of generality, by renaming, let us assume that the profile $\mathcal{P}$ is single crossing with respect to the identity permutation of the voters. Since all the preferences in $\mathcal{P}$ are distinct, there exists a pair of candidates $(x, y) \in \mathcal{C} \times \mathcal{C}$ such that the voter $\lfloor n/2 \rfloor$ and the voter $\lfloor n/2 \rfloor + 1$ order x and y differently. Let us assume, without loss of generality, that the voter $\lfloor n/2 \rfloor$ prefers x over y. Now, since the profile $\mathcal{P}$ is single crossing, every voter in $[\lfloor n/2 \rfloor]$ prefer x over y and every voter in $\{\lfloor n/2 \rfloor + 1, \ldots, n\}$ prefer y over x. $\qquad\square$

Using Lemma 4.3 and 4.4 we now design a PREFERENCE ELICITATION algorithm when no single crossing ordering of the voters is known. The overview of the algorithm is as follows. At any point of time in the elicitation process, we have the set $\mathcal{Q}$ of all the distinct preferences that we have already elicited completely and we have to elicit the preference of a voter ℓ. We first search the set of votes $\mathcal{Q}$ for a preference which is *possibly* same as the preference $\succ_\ell$ of the voter ℓ. It turns out that we can find a possible match $\succ \in \mathcal{Q}$ using $\mathcal{O}(\log |\mathcal{Q}|)$ queries due to Lemma 4.4 which is $\mathcal{O}(\log m)$ due to Lemma 4.3. We then check whether the preference of the voter ℓ is indeed the same as $\succ$ or not using $\mathcal{O}(m)$ queries. If $\succ$ is the same as $\succ_\ell$, then we have elicited $\succ_\ell$ using $\mathcal{O}(m)$ queries. Otherwise, we elicit $\succ_\ell$ using $\mathcal{O}(m \log m)$ queries using Observation 4.2. Fortunately, Lemma 4.3 tells us that we would use the algorithm in Observation 4.2 at most $\mathcal{O}(m^2)$ times. We present the pseudocode of our PREFERENCE ELICITATION algorithm in this setting in Algorithm 4. It uses Algorithm 3 as a subroutine which returns TRUE if the preference of any input voter is same as any given preference.

Theorem 4.5. *Assume that a profile $\mathcal{P}$ is known to be single crossing. However, no ordering of the voters with respect to which $\mathcal{P}$ is single crossing is known a priori. The voters are arriving sequentially according to an arbitrary order $\pi \in \mathbb{S}_n$. Then there is a PREFERENCE ELICITATION algorithm with query complexity $\mathcal{O}(mn + m^3 \log m)$.*

Proof. We present the pesudocode in Algorithm 4. We maintain two arrays in the algorithm. The array $\mathcal{R}$ is of length n and the jth entry stores the preference of voter j. The other array

---

Algorithm 3 Same($\mathcal{R}, \ell$)

Input: $\mathcal{R} = c_1 \succ c_2 \succ \cdots \succ c_m \in \mathcal{L}(\mathcal{C}), \ell \in [n]$
Output: TRUE if the preference of the ℓth voter is $\mathcal{R}$; FALSE otherwise
 1: **for** $i \leftarrow 1$ to $m-1$ **do**
 2:     **if** QUERY $(c_i \succ_\ell c_{i+1})$ = FALSE **then**
 3:         **return** FALSE   ▷ We have found a mismatch.
 4:     **end if**
 5: **end for**
 6: **return** TRUE

$\mathcal{Q}$ stores all the votes seen so far after removing duplicate votes; more specifically, if some specific preference $\succ$ has been seen ℓ many times for any $\ell > 0$, $\mathcal{Q}$ stores only one copy of $\succ$. Upon arrival of voter i, we first check whether there is a preference in $\mathcal{Q}$ which is "potentially" same as the preference of voter i. At the beginning of the search, our search space $\mathcal{Q}' = \mathcal{Q}$ for a potential match in $\mathcal{Q}$ is of size $|\mathcal{Q}|$. We next iteratively keep halving the search space as follows. We find a pair of candidates $(x, y) \in \mathcal{C} \times \mathcal{C}$ such that at least $\lfloor |\mathcal{Q}'|/2 \rfloor$ preferences in $\mathcal{Q}'$ prefer x over y and at least $\lfloor |\mathcal{Q}'|/2 \rfloor$ preferences prefer y over x. The existence of such a pair of candidates is guaranteed by Lemma 4.4 and can be found in $\mathcal{O}(m^2)$ time by simply going over all possible pairs of candidates. By querying how voter i orders x and y, we reduce the search space $\mathcal{Q}'$ for a potential match in $\mathcal{Q}$ to a set of size at most $\lfloor |\mathcal{Q}'|/2 \rfloor + 1$. Hence, in $\mathcal{O}(\log m)$ queries, the search space reduces to only one preference since we have $|\mathcal{Q}| \leqslant m^2$ by Lemma 4.3. Once we find a potential match w in $\mathcal{Q}$ (line 12 in Algorithm 4), we check whether the preference of voter i is the same as w or not using $\mathcal{O}(m)$ queries. If the preference of voter i is indeed same as w, then we output w as the preference of voter i. Otherwise, we use Observation 4.2 to elicit the preference of voter i using $\mathcal{O}(m \log m)$ queries and put the preference of voter i in $\mathcal{Q}$. Since the number of times we need to use the algorithm in Observation 4.2 is at most the number of distinct votes in $\mathcal{P}$ which is known to be at most m^2 by Lemma 4.3, we get the statement. $\qquad\square$

Theorem 4.5 immediately gives us the following corollary in the random access to voters model when no single crossing ordering is known.

Corollary 4.1. *Assume that a profile $\mathcal{P}$ is known to be single crossing. However, no ordering of the voters with respect to which $\mathcal{P}$ is single crossing is known. Given a random access to voters, there is a* PREFERENCE ELICITATION *algorithm with query complexity $\mathcal{O}(mn + m^3 \log m)$.*

Algorithm 4 PreferenceElicitUnknownSingleCrossingOrdering(π)

Input: $\pi \in \mathbb{S}_n$
Output: Profile of all the voters
1: $\mathcal{R}, \mathcal{Q} \leftarrow \emptyset$ ▷ $\mathcal{Q}$ stores all the votes seen so far without duplicate. $\mathcal{R}$ stores the profile.
2: **for** $i \leftarrow 1$ to n **do** ▷ Elicit preference of the ith voter in i^{th} iteration of this for loop.
3:     $\mathcal{Q}' \leftarrow \mathcal{Q}$
4:     **while** $|\mathcal{Q}'| > 1$ **do** ▷ Search $\mathcal{Q}$ to find a vote potentially same as the preference of $\pi(i)$
5:        Let $x, y \in \mathcal{C}$ be two candidates such that at least $\lfloor |\mathcal{Q}'|/2 \rfloor$ votes in $\mathcal{Q}'$ prefer x over y and at least $\lfloor |\mathcal{Q}'|/2 \rfloor$ votes in $\mathcal{Q}'$ prefer y over x.
6:           **if** QUERY $(x \succ_{\pi(i)} y) = $ TRUE **then**
7:              $\mathcal{Q}' \leftarrow \{v \in \mathcal{Q}' : v \text{ prefers } x \text{ over } y\}$
8:           **else**
9:              $\mathcal{Q}' \leftarrow \{v \in \mathcal{Q}' : v \text{ prefers } y \text{ oer } x\}$
10:          **end if**
11:       **end while**
12:       Let w be the only vote in $\mathcal{Q}'$ ▷ w is potentially same as the preference of $\pi(i)$
13:       **if** Same$(w, \pi(i)) = $ TRUE **then** ▷ Check whether the vote $\pi(i)$ is potentially same as w
14:          $\mathcal{R}[\pi(i)] \leftarrow w$
15:       **else**
16:          $\mathcal{R}[\pi(i)] \leftarrow$ Elicit using Observation 4.2
17:          $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathcal{R}[\pi(i)]\}$
18:       **end if**
19:    **end for**
20:    **return** $\mathcal{R}$

Proof. Algorithm 4 works for this setting also and exact same bound on the query complexity holds. □

We now show that the query complexity upper bound of Corollary 4.1 is tight up to constant factors for large number of voters.

Theorem 4.6. *Given a random access to voters, any deterministic* PREFERENCE ELICITATION *algorithm which do not know any ordering of the voters with respect to which the input profile is single crossing has query complexity* $\Omega(m \log m + mn)$.

Proof. The $\Omega(m \log m)$ bound follows from sorting lower bound and the fact that any profile consisting of only one preference $P \in \mathcal{L}(\mathcal{C})$ is single crossing. Suppose we have an even number of candidates and $\mathcal{C} = \{c_1, \ldots, c_m\}$. Consider the ordering $\mathcal{Q} = c_1 \succ c_2 \succ \cdots \succ c_m$ and the pairing of the candidates $\{c_1, c_2\}, \{c_3, c_4\}, \ldots, \{c_{m-1}, c_m\}$. The oracle answers all the query requests consistently according to the ordering Q. We claim that any PREFERENCE

ELICITATION algorithm $\mathcal{A}$ must compare c_i and c_{i+1} for every voter and for every odd integer $i \in [m]$. Indeed, otherwise, there exist a voter κ and an odd integer $i \in [m]$ such that the algorithm $\mathcal{A}$ does not compare c_i and c_{i+1}. Suppose the algorithm outputs a profile $\mathcal{P}'$. If the voter κ prefers c_i over c_{i+1} in $\mathcal{P}'$, then the oracle fixes the preference $\succ_\kappa$ to be $c_1 \succ c_2 \succ \cdots \succ c_{i-1} \succ c_{i+1} \succ c_i \succ c_{i+2} \succ \cdots \succ c_m$; otherwise the oracle fixes $\succ_\kappa$ to be $\mathcal{Q}$. The algorithm fails to correctly output the preference of the voter κ in both the cases. Also the final profile with the oracle is single crossing with respect to any ordering of the voters that places the voter κ at the end. Hence, $\mathcal{A}$ must compare c_i and c_{i+1} for every voter and for every odd integer $i \in [m]$ and thus has query complexity $\Omega(mn)$. □

4.3.3 Single Crossing Width

We now consider preference elicitation for profiles which are nearly single crossing. We begin with profiles with bounded single crossing width.

Proposition 4.1. *Suppose a profile $\mathcal{P}$ is single crossing with width w. Given a* PREFERENCE ELICITATION *algorithm $\mathcal{A}$ with query complexity $q(m, n)$ for random (or sequential) access to the voters when a single crossing order is known (or unknown), there exists a* PREFERENCE ELICITATION *algorithm $\mathcal{A}'$ for the single crossing profiles with width w which has query complexity $\mathcal{O}(q(m/w, n) + mn \log w)$ under same setting as $\mathcal{A}$.*

Proof. Let the partition of the set of candidates $\mathcal{C}$ with respect to which the profile $\mathcal{P}$ is single crossing be $\overline{\mathcal{C}}_i, i \in [\lceil m/w \rceil]$. Hence, $\mathcal{C} = \cup_{i \in [\lceil m/w \rceil]} \overline{\mathcal{C}}_i$ and $\overline{\mathcal{C}}_i \cap \overline{\mathcal{C}}_j = \emptyset$ for every $i, j \in [\lceil m/w \rceil]$ with $i \neq j$. Let $\mathcal{C}'$ be a subset of candidates containing exactly one candidate from $\overline{\mathcal{C}}_i$ for each $i \in [\lceil m/w \rceil]$. We first find $\mathcal{P}(\mathcal{C}')$ using $\mathcal{A}$. The query complexity of this step is $\mathcal{O}(q(m/w, n))$. Next we find $\mathcal{P}(\overline{\mathcal{C}}_i)$ using Observation 4.2 for every $i \in [m/w]$ thereby finding $\mathcal{P}$. The overall query complexity is $\mathcal{O}(q(m/w, n) + (m/w)nw \log w) = \mathcal{O}(q(m/w, n) + mn \log w)$. □

From Proposition 4.1, Lemma 4.1, Theorem 4.2, 4.4 and 4.5, and Corollary 4.1 we get the following.

Corollary 4.2. *Let a profile $\mathcal{P}$ be single crossing with width w. Then there exists a* PREFERENCE ELICITATION *algorithm with query complexity $\mathcal{O}((m^2/w) \log(n/w) + mn \log w)$ for known single crossing order and random access to votes, $\mathcal{O}(m^2/w^2 + mn \log w)$ for sequential access to votes according to a single crossing order, $\mathcal{O}((m^2/w^2) \log(n/w) + mn \log w)$ for known single crossing*

order but arbitrary sequential access to votes, $\mathcal{O}(m^3/w^3 \log(m/w) + mn \log w)$ for unknown single crossing order and arbitrary sequential access to votes or random access to votes.

4.4   Results for Condorcet Winner

Lemma 4.5. *Let an n voter profile $\mathcal{P}$ is single crossing with respect to the ordering $v_1, v_2, \ldots, v_n$. Then the candidate which is placed at the top position of the vote $v_{\lceil n/2 \rceil}$ is a weak Condorcet winner.*

Proof. Let c be the candidate which is placed at the top position of the vote $v_{\lceil n/2 \rceil}$ and w be any other candidate. Now the result follows from the fact that either all the votes in $\{v_i : 1 \leqslant i \leqslant \lceil n/2 \rceil\}$ prefer c to w or all the votes in $\{v_i : \lceil n/2 \rceil \leqslant i \leqslant n\}$ prefer c to w. ☐

Lemma 4.5 immediately gives the following.

Corollary 4.3. *Given either sequential or random access to votes, there exists a* WEAK CONDORCET WINNER *algorithm with query complexity $\mathcal{O}(m)$ when a single crossing ordering is known.*

Proof. Since we know a single crossing ordering $v_1, v_2, \ldots, v_n$, we can find the candidate at the first position of the vote $v_{\lceil n/2 \rceil}$ (which is a weak Condorcet winner by Lemma 4.5) by making $\mathcal{O}(m)$ queries. ☐

We now move on to finding the Condorcet winner. We begin with the following observation regarding existence of the Condorcet winner of a single crossing preference profile.

Lemma 4.6. *Let an n voter profile $\mathcal{P}$ is single crossing with respect to the ordering $v_1, v_2, \ldots, v_n$. If n is odd, then the Condorcet winner is the candidate placed at the first position of the vote $v_{\lceil n/2 \rceil}$. If n is even, then there exists a Condorcet winner if and only there exists a candidate which is placed at the first position of both the votes $v_{n/2}$ and $v_{n/2+1}$.*

Proof. If n is odd, the result follows from Lemma 4.5 and the fact that a candidate c is a Condorcet winner if and only if c is a weak Condorcet winner.

Now suppose n is even. Let a candidate c is placed at the first position of the votes $v_{n/2}$ and $v_{n/2+1}$. Then for every candidate $x \in \mathcal{C} \setminus \{c\}$, either all the votes in $\{v_i : 1 \leqslant i \leqslant n/2 + 1\}$ or all the votes in $\{v_i : n/2 \leqslant i \leqslant n\}$ prefer c over x. Hence, c is a Condorcet winner. Now suppose a candidate w is a Condorcet winner. Then we prove that w must be placed at the

first position of both the votes $v_{n/2}$ and $v_{n/2+1}$. If not, then both the two candidates placed at the first positions of the votes $v_{n/2}$ and $v_{n/2+1}$ are weak Condorcet winners according to Lemma 4.5 applied to single crossing orders $v_1, v_2, \ldots, v_n$ and $v_n, v_{n-1}, \ldots, v_1$. $\qquad\square$

Lemma 4.6 immediately gives us the following.

Corollary 4.4. *Given either sequential or random access to votes, there exists a* Weak Condorcet Winner *algorithm with query complexity* $\mathcal{O}(m)$ *when a single crossing ordering is known.*

Proof. Since we know a single crossing ordering $v_1, v_2, \ldots, v_n$, we can find the candidates which are placed at the first positions of the votes $v_{n/2}$ and $v_{n/2+1}$ by making $\mathcal{O}(m)$ queries. Now the result follows from Lemma 4.6. $\qquad\square$

4.5 Conclusion

In this work, we have presented Preference Elicitation algorithms with low query complexity for single crossing profiles under various settings. Moreover, we have proved that the query complexity of our algorithms are tight for a large number of voters up to constant factors for all but one setting namely when the voters can be accessed randomly but we do not know any ordering with respect to which the voters are single crossing. We then move on to show that a weak Condorcet winner and the Condorcet winner (if one exists) can be found from a single crossing preference profile using much less number of queries.

With this, we conclude the first part of the thesis. In the next part of the thesis, we study the problem of finding a winner of an election under various real world scenarios.

Part II

Winner Determination

In the second part of the thesis, we present our work on determining the winner of an election under various circumstances. This part consists of the following chapters.

▷ In Chapter 5 – Winner Prediction and Margin of Victory Estimation – we present efficient algorithms based on sampling to predict the winner of an election and its robustness. We prove that both the problems of winner prediction and robustness estimation can be solved simultaneously by sampling only a few votes uniformly at random.

▷ In Chapter 6 – Streaming Algorithms for Winner Determination – we develop (often) optimal algorithms for determining the winner of an election when the votes are arriving in a streaming fashion. Our results show that an approximate winner can be determined fast with a small amount of space.

▷ In Chapter 7 – Kernelization for Possible Winner and Coalitional Manipulation – we present interesting results on kernelization for determining possible winners from a set of incomplete votes. Our results prove that the problem of determining possible winners with incomplete votes does not have any efficient preprocessing strategies under plausible complexity theoretic assumptions even when the number of candidates is relatively small. We also present efficient kernelization algorithms for the problem of manipulating an election.

Chapter 5

Winner Prediction and Margin of Victory Estimation

Predicting the winner of an election and estimating the margin of victory of that election are favorite problems both for news media pundits and computational social choice theorists. Since it is often infeasible to elicit the preferences of all the voters in a typical prediction scenario, a common algorithm used for predicting the winner and estimating the margin of victory is to run the election on a small sample of randomly chosen votes and predict accordingly. We analyze the performance of this algorithm for many commonly used voting rules.

More formally, for predicting the winner of an election, we introduce the (ε, δ)-WINNER DETERMINATION problem, where given an election $\mathcal{E}$ on n voters and m candidates in which the margin of victory is at least εn votes, the goal is to determine the winner with probability at least $1 - \delta$ where ε and δ are parameters with $0 < \varepsilon, \delta < 1$. The margin of victory of an election is the smallest number of votes that need to be modified in order to change the election winner. We show interesting lower and upper bounds on the number of samples needed to solve the (ε, δ)-WINNER DETERMINATION problem for many

common voting rules, including all scoring rules, approval, maximin, Copeland, Bucklin, plurality with runoff, and single transferable vote. Moreover, the lower and upper bounds match for many common voting rules up to constant factors.

For estimating the margin of victory of an election, we introduce the (c, ε, δ)–MARGIN OF VICTORY problem, where given an election $\mathcal{E}$ on n voters, the goal is to estimate the margin of victory $M(\mathcal{E})$ of $\mathcal{E}$ within an additive error of $cM(\mathcal{E}) + \varepsilon n$ with probability of error at most δ where ε, δ, and c are the parameters with $0 < \varepsilon, \delta < 1$ and $c > 0$. We exhibit interesting bounds on the sample complexity of the (c, ε, δ)–MARGIN OF VICTORY problem for many commonly used voting rules including all scoring rules, approval, Bucklin, maximin, and Copeland$^\alpha$. We observe that even for the voting rules for which computing the margin of victory is NP-hard, there may exist efficient sampling based algorithms for estimating the margin of victory, as observed in the cases of maximin and Copeland$^\alpha$ voting rules.

5.1 Introduction

In many situations, one wants to predict the winner without holding the election for the entire population of voters. The most immediate such example is an *election poll*. Here, the pollster wants to quickly gauge public opinion in order to predict the outcome of a full-scale election. For political elections, exit polls (polls conducted on voters after they have voted) are widely used by news media to predict the winner before official results are announced. In *surveys*, a full-scale election is never conducted, and the goal is to determine the winner, based on only a few sampled votes, for a hypothetical election on all the voters. For instance, it is not possible to force all the residents of a city to fill out an online survey to rank the local Chinese restaurants, and so only those voters who do participate have their preferences aggregated.

If the result of the poll or the survey has to reflect the true election outcome, it is obviously necessary that the number of sampled votes not be too small. Here, we investigate this fundamental question:

> What is the minimum number of votes that need to be sampled so that the winner of the election on the sampled votes is the same as the winner of the election on all the votes?

This question can be posed for any voting rule. The most immediate rule to study is the *plurality* voting rule, where each voter votes for a single candidate and the candidate with most votes wins. Although the plurality rule is the most common voting rule used in political elections, it is important to extend the analysis to other popular voting rules. For example, the *single transferable vote* is used in political elections in Australia, India and Ireland, and it was the subject of a nationwide referendum in the UK in 2011. The *Borda* voting rule is used in the Icelandic parliamentary elections. Outside politics, in private companies and competitions, a wide variety of voting rules are used. For example, the *approval* voting rule has been used by the Mathematical Association of America, the American Statistical Institute, and the Institute of Electrical and Electronics Engineers, and *Condorcet consistent* voting rules are used by many free software organizations.

A voting rule is called *anonymous* if the winner does not change after any renaming of the voters. All popular voting rules including the ones mentioned before are anonymous. For any anonymous voting rule, the question of finding the minimum number of vote samples required becomes trivial if a single voter in the election can change the winning candidate. In this case, all the votes need to be counted, because otherwise that single crucial vote may not be sampled. We get around this problem by assuming that in the elections we consider, the winning candidate wins by a considerable *margin of victory*. Formally, the margin of victory of an election is defined as the minimum number of votes that must be changed in order to change the election winner. Note that the margin of victory depends not only on the votes cast but also on the voting rule used in the election.

Other than predicting the winner of an election, one may also like to know how robust the election outcome is with respect to the changes in votes [SYE13, CPS14, RGMT06]. One way to capture robustness of an election outcome is the margin of victory of that election. An election outcome is considered to be robust if the margin of victory of that election is large.

In addition to formalizing the notion of robustness of an election outcome, the margin of victory of an election plays a crucial role in many practical applications. One such example is post election audits — methods to observe a certain number of votes (which is often selected randomly) after an election to detect an incorrect outcome. There can be a variety of reasons for an incorrect election outcome, for example, software or hardware bugs in voting machine [NL07], machine output errors, use of various clip-on devices that can tamper with the memory of the voting machine [WWH+10], human errors in counting votes. Post election audits have nowadays become common practice to detect problems in electronic

voting machines in many countries, for example, USA. As a matter of fact, at least thirty states in the USA have reported such problems by 2007 [NL07]. Most often, the auditing process involves manually observing some sampled votes. Researchers have subsequently proposed various *risk limiting auditing* methodologies that not only minimize the cost of manual checking, but also limit the risk of making a human error by sampling as few votes as possible [Sta08a, Sta08b, Sta09, SCS11]. The sample size in a risk limiting audit critically depends on the margin of victory of the election.

Another important application where the margin of victory plays an important role is polling. One of the most fundamental questions in polling is: how many votes should be sampled to be able to correctly predict the outcome of an election? It turns out that the *sample complexity* in polling too crucially depends on the margin of victory of the election from which the pollster is sampling [CEG95, DB15]. Hence, computing (or at least approximating sufficiently accurately) the margin of victory of an election is often a necessary task in many practical applications. However, in many applications including the ones discussed above, one cannot observe all the votes. For example, in a survey or polling, one cannot first observe all the votes to compute the margin of victory and then sample the required number of votes based on the margin of victory computed. Hence, one often needs a *"good enough"* estimate of the margin of victory by observing a small number votes. We precisely address this problem: estimate the margin of victory of an election by sampling as few votes as possible.

5.1.1   Our Contribution

In this work, we show nontrivial bounds for the sample complexity of predicting the winner of an election and estimating the margin of victory of an election for many commonly used voting rules.

5.1.1.1   Winner Prediction

Let n be the number of voters, m the number of candidates, and r a voting rule. We introduce and study the following problem in the context of winner prediction:

Definition 5.1. $((\varepsilon, \delta)$-Winner Determination$)$
Given a r-election $\mathcal{E}$ whose margin of victory is at least εn, determine the winner of the election with probability at least $1 - \delta$. (The probability is taken over the internal coin tosses of the algorithm.)

We remind the reader that there is no assumption about the distribution of votes in this problem. Our goal is to solve the (ε, δ)-WINNER DETERMINATION problem by a randomized algorithm that is allowed to query the votes of arbitrary voters. Each query reveals the full vote of the voter. The minimum number of votes queried by any algorithm that solves the (ε, δ)-WINNER DETERMINATION problem is called the *sample complexity* of this problem. The sample complexity can of course depend on ε, δ, n, m, and the voting rule in use.

A standard result in [CEG95] implies that solving the above problem for the majority rule on 2 candidates requires at least $(1/4\varepsilon^2) \ln(\frac{1}{8e\sqrt{\pi}\delta})$ samples (Theorem 5.1). Also, a straightforward argument (Theorem 5.3) using Chernoff bounds shows that for any homogeneous voting rule, the sample complexity is at most $(9m!^2/2\varepsilon^2) \ln(2m!/\delta)$. So, when m is a constant, the sample complexity is of the order $\Theta((1/\varepsilon^2) \ln(1/\delta))$ for any homogeneous voting rule that reduces to the majority voting rule on 2 candidates and this bound is tight up to constant factors. We note that all the commonly used voting rules including the ones we study here, are homogeneous and they reduce to the majority voting rule when we have only 2 candidates. Note that this bound is independent of n if ε and δ are independent of n.

Our main technical contribution is in understanding the dependence of the sample complexity on the number of candidates m. Note that the upper bound cited above has very bad dependence on m and is clearly unsatisfactory in situations when m is large (such as in online surveys about restaurants).

▷ We show that the sample complexity of the (ε, δ)-WINNER DETERMINATION problem is at most $(9/2\varepsilon^2) \ln(2k/\delta)$ for the k-approval voting rule (Theorem 5.6) and at most $(27/\varepsilon^2) \ln(4/\delta)$ for the plurality with runoff voting rule (Theorem 5.11). In particular, for the plurality rule, the sample complexity is independent of m as well as n.

▷ We show that the sample complexity of the (ε, δ)-WINNER DETERMINATION problem is at most $(9/2\varepsilon^2) \ln(2m/\delta)$ and $\Omega((1-\delta)(1/\varepsilon^2) \ln m)$ for the Borda (Theorem 5.2), approval (Theorem 5.4), maximin (Theorem 6.6), and Bucklin (Theorem 5.10) voting rules. Note that when δ is a constant, the upper and lower bounds match up to constant factors.

▷ We show a sample complexity upper bound of $(25/2\varepsilon^2) \ln^3(m/\delta)$ for the (ε, δ)-WINNER DETERMINATION problem for the Copeland$^\alpha$ voting rule (Theorem 5.9) and $(3m^2/\varepsilon^2)(m \ln 2 + \ln(2m/\delta))$ for the STV voting rule (Theorem 5.12).

We summarize these results in Table 5.1.

Voting rules	Sample complexity for (ε, δ)-WINNER DETERMINATION	
	Upper bounds	Lower bounds
k-approval	$(9/2\varepsilon^2)\ln(2k/\delta)$ [Theorem 5.6]	$\Omega\left((\ln(k+1)/\varepsilon^2)\,(1-\delta)\right)^{\S}$ [Theorem 5.2]
k-veto	$(9/2\varepsilon^2)\ln(2k/\delta)$ [Theorem 5.7]	$(1/4\varepsilon^2)\ln(1/8e\sqrt{\pi}\delta)^{*}$ [Corollary 5.1]
Scoring rules	$(9/2\varepsilon^2)\ln(2m/\delta)$ [Theorem 5.5]	$\Omega\left((\ln m/\varepsilon^2)\,(1-\delta)\right)^{\dagger}$ [Theorem 5.2]
Borda		
Approval	$(9/2\varepsilon^2)\ln(2m/\delta)$ [Theorem 5.4]	
Maximin	$(9/2\varepsilon^2)\ln(2m/\delta)$ [Theorem 6.6]	
Copeland	$(25/2\varepsilon^2)\ln^3(2m/\delta)$ [Theorem 5.9]	
Bucklin	$(9/2\varepsilon^2)\ln(2m/\delta)$ [Theorem 5.10]	
Plurality with runoff	$(27/\varepsilon^2)\ln(4/\delta)$ [Theorem 5.11]	$(1/4\varepsilon^2)\ln(1/8e\sqrt{\pi}\delta)^{*}$ [Corollary 5.1]
STV	$(3m^2/\varepsilon^2)(m\ln 2 + \ln(2m/\delta))$ [Theorem 5.12]	
Any homogeneous voting rule	$(9m!^2/2\varepsilon^2)\ln(2m!/\delta)$ [Theorem 5.3]	

Table 5.1: Sample complexity of the (ε, δ)-WINNER DETERMINATION problem for common voting rules. †–The lower bound of $\Omega((\ln m/\varepsilon^2)\,(1-\delta))$ also applies to any voting rule that is Condorcet consistent. *– The lower bound of $(1/4\varepsilon^2)\ln(1/8e\sqrt{\pi}\delta)$ holds for any voting rule that reduces to the plurality voting rule for elections with two candidates. §– The lower bound holds for $k \leqslant .999m$.

5.1.1.2 Estimating Margin of Victory

The margin of victory of an election is defined as follows.

Definition 5.2. *Margin of Victory (MOV)*
Given an election $\mathcal{E}$, the margin of victory of $\mathcal{E}$ is defined as the smallest number of votes that must be changed to change the winner of the election $\mathcal{E}$.

We abbreviate margin of victory as MOV. We denote the MOV of an election $\mathcal{E}$ by $M(\mathcal{E})$. We introduce and study the following computational problem for estimating the margin of victory of an election:

Definition 5.3. $((c, \varepsilon, \delta)$–MARGIN OF VICTORY(*MOV*))
Given a r-election $\mathcal{E}$, determine the margin of victory $M_r(\mathcal{E})$ of $\mathcal{E}$, within an additive error of at most $cM_r(\mathcal{E}) + \varepsilon n$ with probability at least $1 - \delta$. The probability is taken over the internal coin tosses of the algorithm.

Our goal here is to solve the (c, ε, δ)–MARGIN OF VICTORY problem by observing as few sample votes as possible. Our main technical contribution is to come up with efficient sampling based *polynomial time* randomized algorithms to solve the (c, ε, δ)–MARGIN OF VICTORY problem for common voting rules. Each sample reveals the entire preference order of the sampled vote. We summarize the results on the (c, ε, δ)–MARGIN OF VICTORY problem in Table 5.2.

Table 5.2 shows a practically appealing positive result — *the sample complexity of all the algorithms presented here is independent of the number of voters*. Our lower bounds on the sample complexity of the (c, ε, δ)–MARGIN OF VICTORY problem for all the voting rules studied here match with the upper bounds up to constant factors when we have a constant number of candidates. Moreover, the lower and upper bounds on the sample complexity for the k-approval voting rule match up to constant factors irrespective of number of candidates, when k is a constant.

> ▷ We show a sample complexity lower bound of $\left({(1-c)^2}/{36\varepsilon^2} \right) \ln \left({1}/{8e\sqrt{\pi}\delta} \right)$ for the (c, ε, δ)–MARGIN OF VICTORY problem for all the commonly used voting rules, where $c \in [0, 1)$ (Theorem 5.13 and Corollary 5.3).

> ▷ We show a sample complexity upper bound of $\left({12}/{\varepsilon^2} \right) \ln(2m/\delta)$ for the $(1/3, \varepsilon, \delta)$–MARGIN OF VICTORY problem for arbitrary scoring rules (Theorem 5.14). However, for a special

Voting rules	Sample complexity for (c, ε, δ)–MARGIN OF VICTORY	
	Upper bounds	Lower bounds
Scoring rules	$(1/3, \varepsilon, \delta)$–MOV, $(12/\varepsilon^2) \ln(2m/\delta)$ [Theorem 5.14]	
k-approval	$(0, \varepsilon, \delta)$–MOV, $(12/\varepsilon^2) \ln(2k/\delta)$, [Theorem 5.15]	(c, ε, δ)–MOV†, $((1-c)^2/36\varepsilon^2) \ln(1/8e\sqrt{\pi}\delta)$, [Theorem 5.13] [Corollary 5.3]
Approval	$(0, \varepsilon, \delta)$–MOV, $(12/\varepsilon^2) \ln(2m/\delta)$, [Theorem 5.16]	
Bucklin	$(1/3, \varepsilon, \delta)$–MOV, $(12/\varepsilon^2) \ln(2m/\delta)$, [Theorem 5.17]	
Maximin	$(1/3, \varepsilon, \delta)$–MOV, $(24/\varepsilon^2) \ln(2m/\delta)$, [Theorem 5.18]	
Copeland$^\alpha$	$(1 - O(1/\ln m), \varepsilon, \delta)$–MOV, $(96/\varepsilon^2) \ln(2m/\delta)$, [Theorem 5.19]	

Table 5.2: Sample complexity for the (c, ε, δ)–MARGIN OF VICTORY problem for common voting rules. †The result holds for any $c \in [0, 1)$.

class of scoring rules, namely, the k-approval voting rules, we prove a sample complexity upper bound of $(12/\varepsilon^2) \ln(2k/\delta)$ for the $(0, \varepsilon, \delta)$–MARGIN OF VICTORY problem (Theorem 5.15).

One key finding of our work is that, there may exist efficient sampling based polynomial time algorithms for estimating the margin of victory, even if computing the margin of victory is NP-hard for a voting rule [Xia12], as observed in the cases of maximin and Copeland$^\alpha$ voting rules.

5.1.2 Related Work

The subject of voting is at the heart of (computational) social choice theory, and there is a vast amount of literature in this area. Elections take place not only in human societies but also in man made social networks [BBCV09, RSW$^+$07] and, generally, in many multiagent systems [ER91, PHG00]. The winner determination problem is the task of finding the winner in an election, given the voting rule in use and the set of all votes cast. It is known that there are natural voting rules, e.g., Kemeny's rule and Dodgson's method, for which the winner determination problem is NP-hard [BITT89, HSV05, HHR97].

The basic model of election has been generalized in several other ways to capture real world situations. One important consideration is that the votes may be incomplete

rankings of the candidates and not a complete ranking. There can also be uncertainty over which voters and/or candidates will eventually turn up. The uncertainty may additionally come up from the voting rule that will be used eventually to select the winner. In these incomplete information settings, several winner models have been proposed, for example, robust winner [BLOP14, LB11a, SYE13], multi winner [LB13], stable winner [FMT12], approximate winner [DLC14], probabilistic winner [BBF10], possible winner [MBC⁺16, DMN15b, DMN16c]. Hazon et al. [HAKW08] proposed useful methods to evaluate the outcome of an election under various uncertainties. We do not study the role of uncertainty in this work.

The general question of whether the outcome of an election can be determined by less than the full set of votes is the subject of *preference elicitation*, a central category of problems in AI. The (ε, δ)-WINNER DETERMINATION problem also falls in this area when the elections are restricted to those having margin of victory at least εn. For general elections, the preference elicitation problem was studied by Conitzer and Sandholm [CS02], who defined an elicitation policy as an adaptive sequence of questions posed to voters. They proved that finding an efficient elicitation policy is NP-hard for many common voting rules. Nevertheless, several elicitation policies have been developed in later work [Con09, LB11a, LB11b, DL13b, OFB13, DM16a, DM16b] that work well in practice and have formal guarantees under various assumptions on the vote distribution. Another related work is that of Dhamal and Narahari [DN13] who show that if the voters are members of a social network where neighbors in the network have similar candidate votes, then it is possible to elicit the votes of only a few voters to determine the outcome of the full election.

In contrast, in our work on winner prediction, we posit no assumption on the vote distribution other than that the votes create a substantial margin of victory for the winner. Under this assumption, we show that even for voting rules in which winner determination is NP-hard in the worst case, it is possible to sample a small number of votes to determine the winner. Our work falls inside the larger framework of *property testing* [Ron01], a class of problems studied in theoretical computer science, where the inputs are promised to either satisfy some property or have a "gap" from instances satisfying the property. In our case, the instances are elections which either have some candidate w as the winner or are "far" from having w being the winner (in the sense that many votes need to be changed).

There have been quite a few work on computing the margin of victory of an election. Most prominent among them is the work of Xia [Xia12]. Xia presents polynomial time algorithms

for computing the margin of victory of an election for various voting rules, for example the scoring rules, and proved intractability results for several other voting rules, for example the maximin and Copeland$^\alpha$ voting rules. Magrino et al. [MRSW11] present approximation algorithms to compute the margin of victory for the instant runoff voting (IRV) rule. Cary [Car11] provides algorithms to estimate the margin of victory of an IRV election. Endriss et al. [EL14] compute the complexity of exact variants of the margin of victory problem for Schulze, Cup, and Copeland voting rules. However, all the existing algorithms to either compute or estimate the margin of victory *need to observe all the votes,* which defeats the purpose in many applications including the ones we discussed. We, in this work, show that we can estimate the margin of victory of an election for many commonly used voting rules quite accurately by sampling a few votes only. Moreover, the accuracy of our estimation algorithm is good enough for many practical scenarios. For example, Table 5.2 shows that it is enough to select only 3600 many votes uniformly at random to estimate $\frac{\text{MOV}}{n}$ of a plurality election within an additive error of 0.1 with probability at least 0.99, where n is the number of votes. We note that in all the sampling based applications we discussed, the sample size is inversely proportional to $\frac{\text{MOV}}{n}$ [CEG95] and thus it is enough to estimate $\frac{\text{MOV}}{n}$ accurately (see Table 5.1).

The problem of finding the margin of victory in an election is the same as the optimization version of the destructive bribery problem introduced by Faliszewski et al. [FHH06, FHH09]. However, to the best of our knowledge, there is no prior work on estimating the cost of bribery by sampling votes.

5.2 Results for Winner Prediction

In this section, we present our results for the (ε, δ)-Winner Determination problem.

5.2.1 Results on Lower Bounds

We begin with presenting our lower bounds for the (ε, δ)-Winner Determination problem for various voting rules. Our lower bounds for the sample complexity of the (ε, δ)-Winner Determination problem are derived from information-theoretic lower bounds for distinguishing distributions.

We start with the following basic observation. Let X be a random variable taking value 1 with probability $\frac{1}{2} - \varepsilon$ and 0 with probability $\frac{1}{2} + \varepsilon$; Y be a random variable taking value 1

with probability $\frac{1}{2} + \varepsilon$ and 0 with probability $\frac{1}{2} - \varepsilon$. Then, it is known that every algorithm needs at least $\frac{1}{4\varepsilon^2} \ln \frac{1}{8e\sqrt{\pi}\delta}$ many samples to distinguish between X and Y with probability of making an error being at most δ [CEG95, BYKS01]. We immediately have the following:

Theorem 5.1. *The sample complexity of the* (ε, δ)-WINNER DETERMINATION *problem for the plurality voting rule is at least* $\frac{1}{4\varepsilon^2} \ln \frac{1}{8e\sqrt{\pi}\delta}$ *even when the number of candidates is* 2.

Proof. Consider an election with two candidates a and b. Consider two vote distributions X and Y. In X, exactly $\frac{1}{2} + \varepsilon$ fraction of voters prefer a to b and thus a is the plurality winner of the election. In Y, exactly $\frac{1}{2} + \varepsilon$ fraction of voters prefer b to a and thus b is the plurality winner of the election. Also, the margin of victory of both the elections corresponding to the vote distributions X and Y is εn, since each vote change can change the plurality score of any candidate by at most one. Observe that any (ε, δ)-WINNER DETERMINATION algorithm for plurality will give us a distinguisher between the distributions X and Y with probability of error at most δ. and hence will need $\frac{1}{4\varepsilon^2} \ln \frac{1}{8e\sqrt{\pi}\delta}$ samples. □

Theorem 5.1 immediately gives us the following sample complexity lower bounds for the (ε, δ)-WINNER DETERMINATION problem for other voting rules.

Corollary 5.1. *Every* (ε, δ)-WINNER DETERMINATION *algorithm needs at least* $\frac{1}{4\varepsilon^2} \ln \frac{1}{8e\sqrt{\pi}\delta}$ *samples for any voting rule which reduces to the plurality rule for two candidates. In particular, the lower bound holds for approval, scoring rules, maximin, Copeland, Bucklin, plurality with runoff, and STV voting rules.*

Proof. All the voting rules mentioned in the statement except the approval voting rule is same as the plurality voting rule for elections with two candidates. Hence, the result follows immediately from Theorem 5.1 for the above voting rules except the approval voting rule. The result for the approval voting rule follows from the fact that any arbitrary plurality election is also a valid approval election where every voter approves exactly one candidate. □

We derive stronger lower bounds in terms of m by explicitly viewing the (ε, δ)-WINNER DETERMINATION problem as a *statistical classification* problem. In this problem, we are given a black box that contains a distribution μ which is guaranteed to be one of ℓ known distributions $\mu_1, \ldots, \mu_\ell$. A *classifier* is a randomized oracle which has to determine the identity of μ, where each oracle call produces a sample from μ. At the end of its execution, the classifier announces a guess for the identity of μ, which has to be correct with probability at least $1 - \delta$. Using information-theoretic methods, Bar-Yossef [BY03] showed the following:

Lemma 5.1. *The worst case sample complexity q of a classifier C for ℓ probability distributions $\mu_1, \ldots, \mu_\ell$ which does not make error with probability more than δ satisfies following.*

$$q \geqslant \Omega\left(\frac{\ln \ell}{JS(\mu_1, \ldots, \mu_\ell)} \cdot (1 - \delta)\right)$$

The connection with our problem is the following. A set V of n votes on a candidate set $\mathcal{C}$ generates a probability distribution μ_V on $\mathcal{L}(\mathcal{C})$, where $\mu_V(\succ)$ is proportional to the number of voters who vote $\succ$. Querying a random vote from V is then equivalent to sampling from the distribution μ_V. The margin of victory is proportional to the minimum statistical distance between μ_V and μ_W, over all the voting profiles W having a different winner than the winner of V.

Now suppose we have m voting profiles $V_1, \ldots, V_m$ having different winners such that each V_i has margin of victory at least εn. Any (ε, δ)-WINNER DETERMINATION algorithm must also be a statistical classifier for $\mu_{V_1}, \ldots, \mu_{V_m}$ in the above sense. It then remains to construct such voting profiles for various voting rules which we do in the proof of the following theorem:

Theorem 5.2. *Every (ε, δ)-WINNER DETERMINATION algorithm needs $\Omega\left(\frac{\ln m}{\varepsilon^2} \cdot (1 - \delta)\right)$ samples for approval, Borda, Bucklin, and any Condorcet consistent voting rules, and $\Omega\left(\frac{\ln k}{\varepsilon^2} \cdot (1 - \delta)\right)$ samples for the k-approval voting rule for $k \leqslant cm$ for any constant $c \in (0, 1)$.*

Proof. For each voting rule mentioned in the theorem, we will show d ($d = k + 1$ for the k-approval voting rule and $d = m$ for the rest of the voting rules) distributions $\mu_1, \ldots, \mu_d$ on the votes with the following properties. Let V_i be an election where each vote $v \in \mathcal{L}(\mathcal{C})$ occurs exactly $\mu_i(v) \cdot n$ many times. Let $\mu = \frac{1}{d} \sum_{i=1}^{d} \mu_i$.

1. For every $i \neq j$, the winner in V_i is different from the winner in V_j.

2. For every i, the margin of victory of V_i is $\Omega(\varepsilon n)$.

3. $D_{KL}(\mu_i \| \mu) = O(\varepsilon^2)$

The result then follows from Lemma 5.1. The distributions for different voting rules are as follows. Let the candidate set be $\mathcal{C} = \{c_1, \ldots, c_m\}$.

    ▷ k-**approval voting rule for** $k \leqslant cm$ **for any constant** $c \in (0, 1)$**:** Fix any arbitrary $M := k + 1$ many candidates $c_1, \ldots, c_M$. For $i \in [M]$, we define a distribution μ_i on all

k sized subsets of $\mathcal{C}$ (for the k-approval voting rule, each vote is a k-sized subset of $\mathcal{C}$) as follows. Each k sized subset corresponds to top k candidates in a vote.

$$
\mu_i(x) = \begin{cases} \left(\varepsilon/\binom{M-1}{k-1}\right) + \left(1-\varepsilon/\binom{M}{k}\right) & \text{if } c_i \in x \text{ and } x \subseteq \{c_1,\ldots,c_M\} \\ (1-\varepsilon)/\binom{M}{k} & c_i \notin x \text{ and } x \subseteq \{c_1,\ldots,c_M\} \\ 0 & \text{else} \end{cases}
$$

The score of c_i in V_i is $n\left(\varepsilon + (1-\varepsilon)\binom{M-1}{k-1}/\binom{M}{k}\right)$, the score of any other candidate $c_j \in \{c_1,\ldots,c_M\} \setminus \{c_i\}$ is $n(1-\varepsilon)\binom{M-1}{k-1}/\binom{M}{k}$, and the score of the rest of the candidates is zero. Hence, the margin of victory is $\Omega(\varepsilon n)$, since each vote change can reduce the score of c_i by at most one and increase the score of any other candidate by at most one and $k \leqslant cm$ for constant $c \in (0,1)$. This proves the result for the k-approval voting rule. Now we show that $D_{KL}(\mu_i\|\mu)$ to be $O(\varepsilon^2)$.

$$
\begin{aligned}
D_{KL}(\mu_i\|\mu) &= \left(\varepsilon + (1-\varepsilon)\frac{k}{M}\right)\ln\left(1-\varepsilon+\varepsilon\frac{M}{k}\right) + (1-\varepsilon)\left(1-\frac{k}{M}\right)\ln(1-\varepsilon) \\
&\leqslant \left(\varepsilon + (1-\varepsilon)\frac{k}{M}\right)\left(\varepsilon\frac{M}{k}-\varepsilon\right) - (1-\varepsilon)\left(1-\frac{k}{M}\right)\varepsilon \\
&= \varepsilon^2\left(\frac{M}{k}-1\right) \\
&\leqslant 2\varepsilon^2
\end{aligned}
$$

▷ **Approval voting rule:** The result follows from the fact that every $\frac{m}{2}$-approval election is also a valid approval election and Lemma 5.2.

▷ **Borda and any Condorcet consistent voting rule:** The score vector for the Borda voting rule which we use in this proof is $(m, m-1, \ldots, 1)$. For $i \in [m]$, we define a distribution μ_i on all possible linear orders over $\mathcal{C}$ as follows.

$$
\mu_i(x) = \begin{cases} 2\varepsilon/m! + (1-\varepsilon)/m! & \text{if } c_i \text{ is within top } \frac{m}{2} \text{ positions in } x. \\ (1-\varepsilon)/m! & \text{else} \end{cases}
$$

The score of c_i in V_i is at least $n(3\varepsilon m/5 + (1-\varepsilon)m/2) = n(m/2 + \varepsilon m/10)$ whereas the score of any other candidate $c_j \neq c_i$ is $mn/2$. Hence, the margin of victory is at least $\varepsilon n/8$, since each vote change can reduce the score of c_i by at most m and increase the score of any

other candidate by at most m. Also, in the weighted majority graph for the election V_i, $w(c_i, c_j) \geqslant \varepsilon n/10$. Hence, the margin of victory is at least $\varepsilon n/4$, since each vote change can change the weight of any edge in the weighted majority graph by at most two. Now we show that $D_{KL}(\mu_i \| \mu)$ to be $O(\varepsilon^2)$.

$$
\begin{aligned}
D_{KL}(\mu_i \| \mu) &= \frac{1 + \varepsilon}{2} \ln (1 + \varepsilon) + \frac{1 - \varepsilon}{2} \ln (1 - \varepsilon) \\
&\leqslant \frac{1 + \varepsilon}{2} \varepsilon - \frac{1 - \varepsilon}{2} \varepsilon \\
&= \varepsilon^2
\end{aligned}
$$

▷ **Bucklin:** For $i \in [m]$, we define a distribution μ_i on all $m/4$ sized subsets of $\mathcal{C}$ as follows. Each $m/4$ sized subset corresponds to the top $\frac{m}{4}$ candidates in a vote.

$$
\mu_i(x) = \begin{cases} \left(\varepsilon / \binom{m-1}{\frac{m}{4}-1} \right) + \left((1-\varepsilon) / \binom{m}{\frac{m}{4}} \right) & \text{if } c_i \in x \\ (1-\varepsilon) / \binom{m}{\frac{m}{4}} & \text{else} \end{cases}
$$

The candidate c_i occurs within the top $m(\frac{1}{2} - \frac{\varepsilon}{10})$ positions at least $n(\frac{1}{2} - \frac{\varepsilon}{10} + \varepsilon) = n(\frac{1}{2} + \frac{9\varepsilon}{10})$ times. On the other hand any candidate $c_j \neq c_i$ occurs within the top $m(\frac{1}{2} - \frac{\varepsilon}{10})$ positions at most $n(\frac{1}{2} - \frac{\varepsilon}{10})$ times. Hence the margin of victory is at least $\frac{\varepsilon n}{30} = \Omega(\varepsilon n)$. Now we show that $D_{KL}(\mu_i \| \mu)$ to be $O(\varepsilon^2)$.

$$
\begin{aligned}
D_{KL}(\mu_i \| \mu) &= \left(\varepsilon + \frac{1 - \varepsilon}{4} \right) \ln (1 + 3\varepsilon) + \frac{3}{4} (1 - \varepsilon) \ln (1 - \varepsilon) \\
&= \frac{1}{4} (3\varepsilon (1 + 3\varepsilon) - 3\varepsilon (1 - \varepsilon)) \\
&= 3\varepsilon^2
\end{aligned}
$$

□

5.2.2 Results on Upper Bounds

In this section, we present the upper bounds on the sample complexity of the (ε, δ)-WINNER DETERMINATION problem for various voting rules. The general framework for proving the upper bounds is as follows. For each voting rule, we first prove a useful structural property

about the election when the margin of victory is known to be at least εn. Then, we sample a few votes uniformly at random to estimate either the score of the candidates for score based voting rules or weights of the edges in the weighted majority graph for the voting rules which are defined using weighted majority graph (maximin and Copeland for example). Finally, appealing to the structural property that has been established, we argue that, the winner of the election on the sampled votes will be the same as the winner of the election, if we are able to estimate either the scores of the candidates or the weights of the edges in the weighted majority graph to a certain level of accuracy.

Before getting into specific voting rules, we prove a straightforward bound on the sample complexity for the (ε, δ)-winner determination problem for *any homogeneous* voting rule.

Theorem 5.3. *There is a (ε, δ)-WINNER DETERMINATION algorithm for every homogeneous voting rule with sample complexity $(9m!^2/2\varepsilon^2) \ln(2m!/\delta)$.*

Proof. We sample ℓ votes uniformly at random from the set of votes with replacement. For $x \in \mathcal{L}(\mathcal{C})$, let X_i^x be an indicator random variable that is 1 exactly when x is the i'th sample, and let $g(x)$ be the total number of voters whose vote is x. Define $\hat{g}(x) = \frac{n}{\ell} \sum_{i=1}^{\ell} X_i^x$. Using the Chernoff bound (Theorem 2.5), we have the following:

$$\Pr\left[|\hat{g}(x) - g(x)| \geqslant \frac{\varepsilon n}{3m!}\right] \leqslant 2 \cdot \exp\left(-\frac{2\varepsilon^2 \ell}{9m!^2}\right)$$

By using the union bound, we have the following,

$$\Pr\left[\exists x \in \mathcal{L}(\mathcal{C}), |\hat{g}(x) - g(x)| > \frac{\varepsilon n}{3m!}\right] \leqslant 2m! \cdot \exp\left(-\frac{2\varepsilon^2 \ell}{9m!^2}\right)$$

Since the margin of victory is εn and the voting rule is anonymous, the winner of the ℓ sample votes will be same as the winner of the election if $|\hat{g}(x) - g(x)| \leqslant \varepsilon n/3m!$ for every linear order $x \in \mathcal{L}(\mathcal{C})$. Hence, it is enough to take $\ell = (9m!^2/2\varepsilon^2) \ln(2m!/\delta)$. $\qquad\square$

5.2.2.1 Approval Voting Rule

We derive the upper bound on the sample complexity for the (ε, δ)-WINNER DETERMINATION problem for the approval voting rule.

Lemma 5.2. *If MOV $\geqslant \varepsilon n$ and w be the winner of an approval election, then, $s(w) - s(x) \geqslant \varepsilon n$, for every candidate $x \neq w$, where $s(y)$ is the number of approvals that a candidate y receives.*

Proof. Suppose there is a candidate $x \neq w$ such that $s(w) - s(x) < \varepsilon n$. Then there must exist $\varepsilon n - 1$ votes which does not approve the candidate x. We modify these votes to make it approve x. This makes w not the unique winner in the modified election. This contradicts the fact that the MOV is at least εn. $\qquad\square$

Theorem 5.4. *There is a (ε, δ)-WINNER DETERMINATION algorithm for the approval voting rule with sample complexity at most $9\ln(2m/\delta)/2\varepsilon^2$.*

Proof. Suppose w is the winner. We sample ℓ votes uniformly at random from the set of votes with replacement. For a candidate x, let X_i^x be a random variable indicating whether the i'th vote sampled approved x. Define $\hat{s}(x) = \frac{n}{\ell} \sum_{i=1}^{\ell} X_i^x$. Then, by an argument analogous to the proof of Theorem 5.3, $\Pr[\exists x \in \mathcal{C}, |\hat{s}(x) - s(x)| \geqslant \varepsilon n/3] \leqslant 2m \cdot \exp\left(-2\varepsilon^2\ell/9\right)$. Thus since MOV $\geqslant \varepsilon n$ and by Lemma 5.2, if we take $\ell = 9\ln(2m/\delta)/2\varepsilon^2$, $\hat{s}(w)$ is greater than $\hat{s}(x)$ for all $x \neq w$. $\qquad\square$

5.2.2.2 Scoring Rules

Now we move on to the scoring rules. Again, we first establish a structural consequence of having large MOV.

Lemma 5.3. *Let $\alpha = (\alpha_1, \ldots, \alpha_m)$ be any normalized score vector (hence, $\alpha_m = 0$). If w and z are the candidates that receive highest and second highest score respectively in a α–scoring rule election instance $\mathcal{E} = (V, C)$ and $M_\alpha(\mathcal{E})$ is the margin of victory of $\mathcal{E}$, then,*

$$\alpha_1(M_\alpha(\mathcal{E}) - 1) \leqslant s(w) - s(z) \leqslant 2\alpha_1 M_\alpha(\mathcal{E})$$

Proof. We claim that there must be at least $M_\alpha(\mathcal{E}) - 1$ many votes $v \in V$ where w is preferred over z. Indeed, otherwise, we swap w and z in all the votes where w is preferred over z. This makes z win the election. However, we have changed at most $M_\alpha(\mathcal{E}) - 1$ votes only. This contradicts the definition of margin of victory (see Definition 5.2). Let $v \in V$ be a vote where w is preferred over z. Let α_i and $\alpha_j(\leqslant \alpha_i)$ be the scores received by the candidates w and z respectively from the vote v. We replace the vote v by $v' = z \succ \cdots \succ c$. This vote change reduces the value of $s(w) - s(z)$ by $\alpha_1 + \alpha_i - \alpha_j$ which is at least α_1. Hence, $\alpha_1(M_\alpha(\mathcal{E}) - 1) \leqslant s(w) - s(z)$. Each vote change reduces the value of $s(w) - s(z)$ by at most $2\alpha_1$ since $\alpha_m = 0$. Hence, $s(w) - s(z) \leqslant 2\alpha_1 M_\alpha(\mathcal{E})$. $\qquad\square$

Using Lemma 5.3, we prove the following sample complexity upper bound for the (ε, δ)-WINNER DETERMINATION problem for the scoring rules.

Theorem 5.5. *Suppose $\alpha = (\alpha_1, \ldots, \alpha_m)$ be a normalized score vector. There is a (ε, δ)-WINNER DETERMINATION algorithm for the α-scoring rule with sample complexity at most $9\ln(2m/\delta)/2\varepsilon^2$.*

Proof. We sample ℓ votes uniformly at random from the set of votes with replacement. For a candidate x, define $X_i = \alpha_i/\alpha_1$ if x gets a score of α_i from the ith sample vote, and let $\hat{s}(x) = \frac{n\alpha_1}{\ell} \sum_{i=1}^{\ell} X_i$. Now using Chernoff bound (Theorem 2.5), we have:

$$\Pr\left[|\hat{s}(x) - s(x)| \geqslant \alpha_1 \varepsilon n/3\right] \leqslant 2\exp\left(-\frac{2\varepsilon^2 \ell}{9}\right)$$

The rest of the proof follows from an argument analogous to the proof of Theorem 5.4 using Lemma 5.3. $\qquad\square$

From Theorem 5.5, we have a (ε, δ)-winner determination algorithm for the k-approval voting rule which needs $9\ln(2m/\delta)/2\varepsilon^2$ many samples for any k. We now improve this bound to $9\ln(2k/\delta)/2\varepsilon^2$ for the k-approval voting rule. Before embarking on the proof of the above fact, we prove the following lemma which we will use crucially in Theorem 5.6.

Lemma 5.4. *Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a function defined by $f(x) = e^{-\lambda/x}$. Then,*

$$f(x) + f(y) \leqslant f(x + y), \text{ for } x, y > 0, \frac{\lambda}{x+y} > 2, x < y$$

Proof. For the function $f(x)$, we have following.

$$
\begin{aligned}
f(x) &= e^{-\lambda/x} \\
\Rightarrow f'(x) &= \frac{\lambda}{x^2} e^{-\lambda/x} \\
\Rightarrow f''(x) &= \frac{\lambda^2}{x^4} e^{-\lambda/x} - \frac{2\lambda}{x^3} e^{-\lambda/x}
\end{aligned}
$$

Hence, for $x, y > 0, \frac{\lambda}{x+y} > 2, x < y$ we have $f''(x), f''(y), f''(x+y) > 0$. This implies following for $x < y$ and an infinitesimal positive δ.

$$
\begin{aligned}
f'(x) &\leqslant f'(y) \\
\Rightarrow \frac{f(x-\delta)-f(x)}{\delta} &\geqslant \frac{f(y)-f(y-\delta)}{\delta} \\
\Rightarrow f(x)+f(y) &\leqslant f(x-\delta)+f(y+\delta) \\
\Rightarrow f(x)+f(y) &\leqslant f(x+y)
\end{aligned}
$$

$\square$

We now present our (ε, δ)-WINNER DETERMINATION algorithm for the k-approval voting rule.

Theorem 5.6. *There is a* (ε, δ)-WINNER DETERMINATION *algorithm for the* k-*approval voting rule with sample complexity at most* $9\ln(2k/\delta)/2\varepsilon^2$.

Proof. We sample ℓ votes uniformly at random from the set of votes with replacement. For a candidate x, let X_i^x be a random variable indicating whether x is among the top k candidates for the ith vote sample. Define $\hat{s}(x) = \frac{n}{\ell}\sum_{i=1}^{\ell} X_i^x$, and let $s(x)$ be the actual score of x. Then by the multiplicative Chernoff bound (Theorem 2.5), we have:

$$
\Pr\left[|\hat{s}(x)-s(x)| > \varepsilon n/3\right] \leqslant 2\exp\left(-\frac{2\varepsilon^2\ell n}{9s(x)}\right)
$$

By union bound, we have the following,

$$
\begin{aligned}
&\Pr[\exists x \in \mathcal{C}, |\hat{s}(x)-s(x)| > \varepsilon n/3] \\
&\leqslant \sum_{x \in \mathcal{C}} 2\exp\left(-2\varepsilon^2\ell n/9s(x)\right) \\
&\leqslant 2k\exp\left(-2\varepsilon^2\ell/9\right)
\end{aligned}
$$

Let the candidate w be the winner of the election. The second inequality in the above derivation follows from the fact that, the function $\sum_{x \in \mathcal{C}} 2\exp\left(-2\varepsilon^2\ell n/9s(x)\right)$ is maximized in the domain, defined by the constraint: for every candidate $x \in \mathcal{C}$, $s(x) \in [0, n]$ and $\sum_{x \in \mathcal{C}} s(x) = kn$, by setting $s(x) = n$ for every $x \in \mathcal{C}'$ and $s(y) = 0$ for every $y \in \mathcal{C} \setminus \mathcal{C}'$, for any arbitrary subset $\mathcal{C}' \subset \mathcal{C}$ of cardinality k (due to Lemma 5.4). The rest of the proof follows by an argument analogous to the proof of Theorem 5.3 using Lemma 5.3. $\square$

Notice that, the sample complexity upper bound in Theorem 5.6 is independent of m for the plurality voting rule. Theorem 5.6 in turn implies the following Corollary which we consider to be of independent interest.

Corollary 5.2. *There is an algorithm to estimate the ℓ_∞ norm $\ell_\infty(\mu)$ of a distribution μ within an additive factor of ε by querying only $9\ln(2/\delta)/2\varepsilon^2$ many samples, if we are allowed to get i.i.d. samples from the distribution μ.*

Such a statement seems to be folklore in the statistics community [DKW56]. Recently in an independent and nearly simultaneous work, Waggoner [Wag15] obtained a sharp bound of $\frac{4}{\varepsilon^2}\ln(\frac{1}{\delta})$ for the sample complexity in Corollary 5.2.

We now turn our attention to the k-veto voting rule. For the k-veto voting rule, we have the following result for the (ε, δ)-WINNER DETERMINATION problem.

Theorem 5.7. *There is a (ε, δ)-WINNER DETERMINATION algorithm for the k-veto voting rule with sample complexity at most $9\ln(2k/\delta)/2\varepsilon^2$.*

Proof. We first observe that, since the margin of victory of the input election is at least εn, every candidate other than the winner must receive at least εn vetoes. Hence we have the following.

$$\varepsilon n(m-1) \geqslant kn \text{ i.e. } m-1 \leqslant k/\varepsilon$$

Let us sample ℓ votes uniformly at random from the set of votes with replacement. For a candidate x, let X_i^x be a random variable indicating whether x is among the bottom k candidates for the ith vote sample. Define $\hat{s}(x) = \frac{n}{\ell}\sum_{i=1}^{\ell} X_i^x$, and let $s(x)$ be the actual score of x. Then by the multiplicative Chernoff bound (Theorem 2.5), we have:

$$\Pr\left[|\hat{s}(x) - s(x)| > \varepsilon n/3\right] \leqslant 2\exp\left(-\frac{2\varepsilon^2 \ell n}{9s(x)}\right)$$

By union bound, we have the following,

$$\begin{aligned}
\Pr[\exists x \in \mathcal{C}, |\hat{s}(x) - s(x)| > \varepsilon n/3] &\leqslant \sum_{x \in \mathcal{C}} 2\exp\left(-2\varepsilon^2\ell n/9s(x)\right) \\
&\leqslant 2k\exp\left(-2\varepsilon^2\ell/9\right)
\end{aligned}$$

Let the candidate w be the winner of the election. The second inequality in the above derivation follows from the fact that, the function $\sum_{x \in \mathcal{C}} 2\exp\left(-2\varepsilon^2\ell n/9s(x)\right)$ is maximized in the do-

main, defined by the constraint: for every candidate $x \in \mathcal{C}$, $s(x) \in [0, n]$ and $\sum_{x \in \mathcal{C}} s(x) = kn$, by setting $s(x) = n$ for every $x \in \mathcal{C}'$ and $s(y) = 0$ for every $y \in \mathcal{C} \setminus \mathcal{C}'$, for any arbitrary subset $\mathcal{C}' \subset \mathcal{C}$ of cardinality k (due to Lemma 5.4). The rest of the proof follows by an argument analogous to the proof of Theorem 5.3 using Lemma 5.3. $\qquad \square$

5.2.2.3 Maximin Voting Rule

We now turn our attention to the maximin voting rule. The idea here is to sample enough number of votes such that we are able to estimate the weights of the edges in the weighted majority graph with certain level of accuracy which in turn leads us to predict winner.

Lemma 5.5. *Let $\mathcal{E} = (V, C)$ be any instance of a maximin election. If w and z are the candidates that receive highest and second highest maximin score respectively in $\mathcal{E}$ and $M_{maximin}(\mathcal{E})$ is the margin of victory of $\mathcal{E}$, then,*

$$2M_{maximin}(\mathcal{E}) \leqslant s(w) - s(z) \leqslant 4M_{maximin}(\mathcal{E})$$

Proof. Each vote change can increase the value of $s(z)$ by at most two and decrease the value of $s(w)$ by at most two. Hence, we have $s(w) - s(z) \leqslant 4M_{maximin}(\mathcal{E})$. Let x be the candidate that minimizes $D_{\mathcal{E}}(w, x)$, that is, $x \in \text{argmin}_{x \in C \setminus \{w\}}\{D_{\mathcal{E}}(w, x)\}$. Let $v \in V$ be a vote where w is preferred over x. We replace the vote v by the vote $v' = z \succ x \succ \cdots \succ w$. This vote change reduces the score of w by two and does not reduce the score of z. Hence, $s(w) - s(z) \geqslant 2M_{maximin}(\mathcal{E})$. $\qquad \square$

We now present our (ε, δ)-WINNER DETERMINATION algorithm for the maximin voting rule.

Theorem 5.8. *There is a (ε, δ)-WINNER DETERMINATION algorithm for the maximin voting rule with sample complexity $(9/2\varepsilon^2) \ln(2m/\delta)$.*

Proof. Let x and y be any two arbitrary candidates. We sample ℓ votes uniformly at random from the set of votes with replacement. Let X_i be a random variable defined as follows.

$$X_i = \begin{cases} 1, & \text{if } x \succ y \text{ in the } i^{th} \text{ sample} \\ -1, & \text{else} \end{cases}$$

Define $\hat{D}(x, y) = \frac{n}{\ell} \sum_{i=1}^{\ell} X_i$. We estimate $\hat{D}(x, y)$ within the closed ball of radius $\varepsilon n/2$ around $D(x, y)$ for every candidates $x, y \in \mathcal{C}$ and the rest of the proof follows from by an argument analogous to the proof of Theorem 5.4 using Lemma 5.5. $\qquad \square$

5.2.2.4 Copeland Voting Rule

Now we move on to the Copeland$^\alpha$ voting rule. The approach for the Copeland$^\alpha$ voting rule is similar to the maximin voting rule. However, it turns out that we need to estimate the edge weights of the weighted majority graph more accurately for the Copeland$^\alpha$ voting rule. Xia introduced the quantity called the *relative margin of victory* (see Section 5.1 in [Xia12]) which we will use crucially for showing sample complexity upper bound for the Copeland$^\alpha$ voting rule. Given an election, a candidate $x \in C$, and an integer (may be negative also) t, $s'_t(V, x)$ is defined as follows.

$$s'_t(V, x) = |\{y \in C : y \neq x, D(y, x) < 2t\}| + \alpha|\{y \in C : y \neq x, D(y, x) = 2t\}|$$

For every two distinct candidates x and y, the relative margin of victory, denoted by $RM(x, y)$, between x and y is defined as the minimum integer t such that, $s'_{-t}(V, x) \leqslant s'_t(V, y)$. Let w be the winner of the election $\mathcal{E}$. We define a quantity $\Gamma(\mathcal{E})$ to be $\min_{x \in C \setminus \{w\}} \{RM(w, x)\}$. Notice that, given an election $\mathcal{E}$, $\Gamma(\mathcal{E})$ can be computed in polynomial amount of time. Now we have the following lemma.

Lemma 5.6. *Suppose MOV* $\geqslant$ εn *and* w *be the winner of a Copeland$^\alpha$ election. Then,* $RM(w, x) \geqslant \varepsilon n/(2(\lceil \ln m \rceil + 1))$, *for every candidate* $x \neq w$.

Proof. Follows from Theorem 11 in [Xia12]. $\qquad \square$

Theorem 5.9. *There is a* (ε, δ)-WINNER DETERMINATION *algorithm for the Copeland$^\alpha$ voting rule with sample complexity* $(25/2\varepsilon^2) \ln^3(2m/\delta)$.

Proof. Let x and y be any two arbitrary candidates and w the Copeland$^\alpha$ winner of the election. We estimate $D(x, y)$ within the closed ball of radius $\varepsilon n/(5(\lceil \ln m \rceil + 1))$ around $D(x, y)$ for every candidates $x, y \in \mathcal{C}$ in a way analogous to the proof of Theorem 6.6. This needs $(25/2\varepsilon^2) \ln^3(2m/\delta)$ many samples. The rest of the proof follows from Lemma 5.6 by an argument analogous to the proof of Theorem 5.3. $\qquad \square$

5.2.2.5 Bucklin Voting Rule

For the Bucklin voting rule, we will estimate how many times each candidate occurs within the first k position for every $k \in [m]$. This eventually leads us to predict the winner of the election due to the following lemma.

Lemma 5.7. *Suppose MOV of a Bucklin election be at least εn. Let w be the winner of the election and x be any arbitrary candidate other than w. Suppose*

$$b_w = \min_i\{i : w \text{ is within top } i \text{ places in at least } n/2 + \varepsilon n/3 \text{ votes}\}$$

$$b_x = \min_i\{i : x \text{ is within top } i \text{ places in at least } n/2 - \varepsilon n/3 \text{ votes}\}$$

Then, $b_w < b_x$.

Proof. We prove it by contradiction. So, assume $b_w \geqslant b_x$. Now by changing $\varepsilon n/3$ votes, we can make the Bucklin score of w to be at least b_w. By changing another $\varepsilon n/3$ votes, we can make the Bucklin score of x to be at most b_x. Hence, by changing $2\varepsilon n/3$ votes, it is possible not to make w the unique winner which contradicts the fact that the MOV is at least εn. $\square$

Our (ε, δ)-WINNER DETERMINATION algorithm for the Bucklin voting rule is as follows.

Theorem 5.10. *There is a (ε, δ)-WINNER DETERMINATION algorithm for the Bucklin voting rule with sample complexity $(9/2\varepsilon^2)\ln(2m/\delta)$.*

Proof. Let x be any arbitrary candidate and $1 \leqslant k \leqslant m$. We sample ℓ votes uniformly at random from the set of votes with replacement. Let X_i be a random variable defined as follows.

$$X_i = \begin{cases} 1, & \text{if } x \text{ is within top } k \text{ places in } i^{th} \text{ sample} \\ 0, & \text{else} \end{cases}$$

Let $\hat{s}_k(x)$ be the estimate of the number of times the candidate x has been placed within top k positions. That is, $\hat{s}_k(x) = \frac{n}{\ell}\sum_{i=1}^{\ell} X_i$. Let $s_k(x)$ be the number of times the candidate x been placed in top k positions. Clearly, $E[\hat{s}_k(x)] = \frac{n}{\ell}\sum_{i=1}^{\ell} E[X_i] = s_k(x)$. We estimate $\hat{s}_k(x)$ within the closed ball of radius $\varepsilon n/3$ around $s_k(x)$ for every candidate $x \in \mathcal{C}$ and every integer $k \in [m]$, and the rest of the proof follows from by an argument analogous to the proof of Theorem 5.4 using Lemma 5.7. $\square$

5.2.2.6 Plurality with Runoff Voting Rule

Now we move on to the plurality with runoff voting rule. In this case, we first estimate the plurality score of each of the candidates. In the next round, we estimate the pairwise margin of victory of the two candidates that qualifies to the second round.

Lemma 5.8. *Suppose MOV $\geqslant \varepsilon n$, and w and r be the winner and runner up of a plurality with runoff election respectively, and x be any arbitrary candidate other than and r. Then, following holds. Let $s(.)$ denote plurality score of candidates. Then following holds.*

1. *$D(w, r) > 2\varepsilon n$.*

2. *For every candidate $x \in \mathcal{C} \setminus \{w, r\}$, $2s(w) > s(x) + s(r) + \varepsilon n$.*

3. *If $s(x) > s(r) - \varepsilon n/2$, then $D(w, x) > \varepsilon n/2$.*

Proof. If the first property does not hold, then by changing εn votes, we can make r winner. If the second property does not hold, then by changing εn votes, we can make both x and r qualify to the second round. If the third property does not hold, then by changing $\varepsilon n/2$ votes, the candidate x can be sent to the second round of the runoff election. By changing another $\varepsilon n/2$ votes, x can be made to win the election. This contradicts the MOV assumption. $\qquad \square$

Now we present our (ε, δ)-WINNER DETERMINATION algorithm for the plurality with runoff voting rule.

Theorem 5.11. *There is a (ε, δ)-WINNER DETERMINATION algorithm for the plurality with runoff voting rule with sample complexity $(27/\varepsilon^2) \ln(4/\delta)$.*

Proof. Let x be any arbitrary candidate. We sample ℓ votes uniformly at random from the set of votes with replacement. Let, X_i be a random variable defined as follows.

$$
X_i = \begin{cases} 1, & \text{if } x \text{ is at first position in the } i^{th} \text{ sample} \\ 0, & \text{else} \end{cases}
$$

The estimate of the plurality score of x be $\hat{s}(x)$. Then $\hat{s}(x) = \frac{n}{\ell} \sum_{i=1}^{\ell} X_i$. Let $s(x)$ be the actual plurality score of x. Then we have following,

$$
E[X_i] = \frac{s(x)}{n}, E[\hat{s}(x)] = \frac{n}{\ell} \sum_{i=1}^{\ell} E[X_i] = s(x)
$$

By Chernoff bound, we have the following,

$$\Pr[|\hat{s}(x) - s(x)| > \varepsilon n] \leqslant \frac{2}{\exp\{\varepsilon^2 \ell n / 3s(x)\}}$$

By union bound, we have the following,

$$\Pr[\exists x \in \mathcal{C}, |\hat{s}(x) - s(x)| > \varepsilon n] \leqslant \sum_{x \in \mathcal{C}} \frac{2}{\exp\{\varepsilon^2 \ell n / 3s(x)\}}$$
$$\leqslant \frac{2}{\exp\{\varepsilon^2 \ell / 3\}}$$

The last line follows from Lemma 5.4. Notice that, we do not need the random variables $\hat{s}(x)$ and $\hat{s}(y)$ to be independent for any two candidates x and y. Hence, we can use the same ℓ sample votes to estimate $\hat{s}(x)$ for every candidate x.

Now let y and z be the two candidates that go to the second round.

$$Y_i = \begin{cases} 1, & \text{if } y \succ z \text{ in the } i^{\text{th}} \text{ sample} \\ -1, & \text{else} \end{cases}$$

The estimate of $D(y, z)$ be $\hat{D}(y, z)$. Then $\hat{D}(y, z) = \frac{n}{\ell} \sum_{i=1}^{\ell} Y_i$. Then we have following,

$$E[Y_i] = \frac{D(y, z)}{n}, E[\hat{D}(y, z)] = \frac{n}{\ell} \sum_{i=1}^{\ell} E[Y_i] = D(y, z)$$

By Chernoff bound, we have the following,

$$\Pr[|\hat{D}(y, z) - D(y, z)| > \varepsilon n] \leqslant \frac{2}{\exp\{\varepsilon^2 \ell / 3\}}$$

Let A be the event that $\forall x \in \mathcal{C}, |\hat{s}(x) - s(x)| \leqslant \varepsilon n$ and $|\hat{D}(y, z) - D(y, z)| \leqslant \varepsilon n$. Now we have,

$$\Pr[A] \geqslant 1 - \left(\frac{2}{\exp\{\varepsilon^2 \ell / 3\}} + \frac{2}{\exp\{\varepsilon^2 \ell / 3\}}\right)$$

Since we do not need independence among the random variables $\hat{s}(a), \hat{s}(b), \hat{D}(w, x), \hat{D}(y, z)$ for any candidates $a, b, w, x, y,$ and z, we can use the same ℓ sampled votes. Now from Lemma 5.8, if $|\hat{s}(x) - s(x)| \leqslant \varepsilon n/3$ for every candidate x and $|\hat{D}(y, z) - D(y, z)| \leqslant \varepsilon n/3$ for every

candidates y and z, then the plurality with runoff winner of the sampled votes coincides with the actual runoff winner. The above event happens with probability at least $1 - \delta$ by choosing an appropriate $\ell = \left(27/\varepsilon^2\right) \ln\left(4/\delta\right)$. $\qquad\square$

5.2.2.7 STV Voting Rule

Now we move on the STV voting rule. The following lemma provides an upper bound on the number of votes that need to be changed to make some arbitrary candidate win the election. More specifically, given a sequence of m candidates $\{x_i\}_{i=1}^m$ with x_m not being the winner, Lemma 5.9 below proves an upper bound on the number of number of votes that need to be modified such that the candidate x_i gets eliminated at the ith round in the STV voting rule.

Lemma 5.9. *Suppose $\mathcal{V}$ be a set of votes and w be the winner of a STV election. Consider the following chain with candidates $x_1 \neq x_2 \neq \ldots \neq x_m$ and $x_m \neq w$.*

$$\mathcal{C} \supset \mathcal{C} \setminus \{x_1\} \supset \mathcal{C} \setminus \{x_1, x_2\} \supset \ldots \supset \{x_m\}$$

Let $s_{\mathcal{V}}(A, x)$ be the plurality score of a candidate x when all the votes in $\mathcal{V}$ are restricted to the set of candidates $A \subset \mathcal{C}$. Let us define $\mathcal{C}_{-i} = \mathcal{C} \setminus \{x_1, \ldots, x_i\}$ and $s_{\mathcal{V}}^*(A) := \min_{x \in A}\{s_{\mathcal{V}}(A, x)\}$. Then, we have the following.

$$\sum_{i=0}^{m-1} \left(s_{\mathcal{V}}\left(\mathcal{C}_{-i}, x_{i+1}\right) - s_{\mathcal{V}}^*\left(\mathcal{C}_{-i}\right)\right) \geqslant MOV$$

Proof. We will show that by changing $\sum_{i=0}^{m-1} \left(s_{\mathcal{V}}\left(\mathcal{C}_{-i}, x_{i+1}\right) - s_{\mathcal{V}}^*\left(\mathcal{C}_{-i}\right)\right)$ votes, we can make the candidate x_m winner. If x_1 minimizes $s_{\mathcal{V}}(\mathcal{C}, x)$ over $x \in \mathcal{C}$, then we do not change anything and define $\mathcal{V}_1 = \mathcal{V}$. Otherwise, there exist $s_{\mathcal{V}}(\mathcal{C}, x_1) - s_{\mathcal{V}}^*(\mathcal{C})$ many votes of following type.

$$x_1 \succ a_1 \succ a_2 \succ \ldots \succ a_{m-1}, a_i \in \mathcal{C}, \forall 1 \leqslant i \leqslant m-1$$

We replace $s_{\mathcal{V}}(\mathcal{C}, x_1) - s_{\mathcal{V}}^*(\mathcal{C})$ many votes of the above type by the votes as follows.

$$a_1 \succ x_1 \succ a_2 \succ \ldots \succ a_{m-1}$$

Let us call the new set of votes by $\mathcal{V}_1$. We claim that, $s_{\mathcal{V}}(\mathcal{C} \setminus x_1, x) = s_{\mathcal{V}_1}(\mathcal{C} \setminus x_1, x)$ for every candidate $x \in \mathcal{C} \setminus \{x_1\}$. Fix any arbitrary candidate $x \in \mathcal{C} \setminus \{x_1\}$. The votes in $\mathcal{V}_1$ that are same

as in $\mathcal{V}$ contributes same quantity to both side of the equality. Let v be a vote that has been changed as described above. If $x = a_1$ then, the vote v contributes one to both sides of the equality. If $x \neq a_1$, then the vote contributes zero to both sides of the equality. Hence, we have the claim. We repeat this process for $(m-1)$ times. Let $\mathcal{V}_i$ be the set of votes after the candidate x_i gets eliminated. Now in the above argument, by replacing $\mathcal{V}$ by $\mathcal{V}_{i-1}$, $\mathcal{V}_1$ by $\mathcal{V}_i$, the candidate set $\mathcal{C}$ by $\mathcal{C} \setminus \{x_1, \ldots, x_{i-1}\}$, and the candidate x_1 by the candidate x_i, we have the following.

$$s_{\mathcal{V}_{i-1}}(\mathcal{C}_{-i}, x) = s_{\mathcal{V}_i}(\mathcal{C}_{-i}, x) \forall x \in \mathcal{C} \setminus \{x_1, \ldots, x_i\}$$

Hence, we have the following.

$$s_{\mathcal{V}}(\mathcal{C}_{-i}, x) = s_{\mathcal{V}_i}(\mathcal{C}_{-i}, x) \forall x \in \mathcal{C} \setminus \{x_1, \ldots, x_i\}$$

In the above process, the total number of votes that are changed is $\sum_{i=0}^{m-1} (s_{\mathcal{V}}(\mathcal{C}_{-i}, x_{i+1}) - s_{\mathcal{V}}^*(\mathcal{C}_{-i}))$. $\qquad\square$

We now use Lemma 5.9 to prove the following sample complexity upper bound for the (ε, δ)-WINNER DETERMINATION problem for the STV voting rule.

Theorem 5.12. *There is a* (ε, δ)-WINNER DETERMINATION *algorithm for the STV voting rule with sample complexity* $(3m^2/\varepsilon^2)(m \ln 2 + \ln(2m/\delta))$.

Proof. We sample ℓ votes uniformly at random from the set of votes with replacement and output the STV winner of those ℓ votes say w' as the winner of the election. Let, w be the winner of the election. We will show that for $\ell = (3m^2/\varepsilon^2)(m \ln 2 + \ln(2m/\delta))$ for which $w = w'$ with probability at least $1 - \delta$. Let A be an arbitrary subset of candidates and x be any candidate in A. Let us define a random variables $X_i, 1 \leqslant i \leqslant \ell$ as follows.

$$X_i = \begin{cases} 1, & \text{if } x \text{ is at top } i^{\text{th}} \text{ sample when restricted to } A \\ 0, & \text{else} \end{cases}$$

Define another random variable $\hat{s}_{\mathcal{V}}(A, x) := \sum_{i=1}^{\ell} X_i$. Then we have, $E[\hat{s}_{\mathcal{V}}(A, x)] = s_{\mathcal{V}}(A, x)$. Now using Chernoff bound, we have the following,

$$\Pr[|\hat{s}_{\mathcal{V}}(A, x) - s_{\mathcal{V}}(A, x)| > \frac{\varepsilon n}{m}] \leqslant \frac{2}{\exp\{\varepsilon^2 \ell / 3m^2\}}$$

Let E be the event that $\exists A \subset \mathcal{C}$ and $\exists x \in A, |\hat{s}_\nu(A, x) - s_\nu(A, x)| > \frac{\varepsilon n}{m}$. By union bound, we have,

$$\Pr[\bar{E}] \quad \geqslant \quad 1 - \frac{m 2^{m+1}}{\exp\{\varepsilon^2 \ell / 3m^2\}}$$

The rest of the proof follows by an argument analogous to the proof of Theorem 5.3 using Lemma 5.9. □

5.3 Results for Estimating Margin of Victory

In this section we present our results for the (c, ε, δ)–MARGIN OF VICTORY problem.

5.3.1 Results on Lower Bounds

Our lower bounds for the sample complexity of the (c, ε, δ)–MARGIN OF VICTORY problem are derived from the information-theoretic lower bound for distinguishing two distributions.

Theorem 5.13. *The sample complexity of the (c, ε, δ)–MARGIN OF VICTORY problem for the plurality voting rule is at least $\left((1-c)^2/36\varepsilon^2\right) \ln \left(1/8e\sqrt{\pi}\delta\right)$ for any $c \in [0, 1)$.*

Proof. Consider two vote distributions X and Y, each over the candidate set $\{a, b\}$. In X, exactly $\frac{1}{2} + \frac{6\varepsilon + 2c/n}{1-c}$ fraction of voters prefer a to b and thus the margin of victory is $\frac{3\varepsilon + c/n}{1-c}n$. In Y, exactly $\frac{1}{2}$ fraction of voters prefer b to a and thus the margin of victory is one. Any (c, ε, δ)–MARGIN OF VICTORY algorithm $\mathcal{A}$ for the plurality voting rule gives us a distinguisher between X and Y with probability of error at most 2δ. This is so because, if the input to $\mathcal{A}$ is X then, the output of $\mathcal{A}$ is less than $c + 2\varepsilon n$ with probability at most δ, whereas, if the input to $\mathcal{A}$ is Y then, the output of $\mathcal{A}$ is more than $c + \varepsilon n$ with probability at most δ. Now since n can be arbitrarily large, we get the result. □

Theorem 5.1 immediately gives the following corollary.

Corollary 5.3. *For any $c \in [0, 1)$, every (c, ε, δ)–MARGIN OF VICTORY algorithm needs at least $\left((1-c)^2/36\varepsilon^2\right) \ln \left(1/8e\sqrt{\pi}\delta\right)$ many samples for all voting rules which reduce to the plurality rule for two candidates. In particular, the lower bound holds for scoring rules, approval, Bucklin, maximin, and Copeland$^\alpha$ voting rules.*

We note that the lower bound results in Theorem 5.1 and Corollary 5.1 do not assume anything about the sampling strategy or the computational complexity of the estimator.

5.3.2 Results on Upper Bounds

A natural approach for estimating the margin of victory of an election efficiently is to compute the margin of victory of a suitably small number of sampled votes. Certainly, it is not immediate that the samples chosen uniformly at random preserve the value of the margin of victory of the original election within some desired factor. Although it may be possible to formulate clever sampling strategies that tie into the margin of victory structure of the election, we will show that uniformly chosen samples are good enough to design algorithms for estimating the margin of victory for many common voting rules. Our proposal has the advantage that the sampling component of our algorithms are always easy to implement, and further, there is no compromise on the bounds in the sense that they are optimal for any constant number of candidates.

Our algorithms involve computing a quantity (which depends on the voting rule under consideration) based on the sampled votes, which we argue to be a suitable estimate of the margin of victory of the original election. This quantity is not necessarily the margin of victory of the sampled votes. For scoring rules, for instance, we will use the sampled votes to estimate candidate scores, and we use the difference between the scores of the top two candidates (suitably scaled) as an estimate for the margin of victory. We also establish a relationship between scores and the margin of victory to achieve the desired bounds on the estimate. The overall strategy is in a similar spirit for other voting rules as well, although the exact estimates may be different. We now turn to a more detailed description.

5.3.2.1 Scoring Rules and Approval Voting Rule

We begin with showing that the margin of victory of any scoring rule based election can be estimated quite accurately by sampling only $\frac{12}{\varepsilon^2} \ln \frac{2m}{\delta}$ many votes. An important thing to note is that, the sample complexity upper bound is independent of the score vector.

Theorem 5.14. *There is a polynomial time* $(1/3, \varepsilon, \delta)$*–*MOV *algorithm for the scoring rules with sample complexity at most* $(12/\varepsilon^2) \ln(2m/\delta)$.

Proof. Let $\alpha = (\alpha_1, \ldots, \alpha_m)$ be any arbitrary normalized score vector and $\mathcal{E} = (V, C)$ an election instance. We sample ℓ (the value of ℓ will be chosen later) votes uniformly at ran-

dom from the set of votes with replacement. For a candidate x, define a random variable $X_i(x) = \alpha_i/\alpha_1$ if x gets a score of α_i from the ith sample vote. Define $\bar{s}(x) = \frac{n\alpha_1}{\ell} \sum_{i=1}^{\ell} X_i(x)$ the estimate of $s(x)$, the score of x. Also define $\varepsilon' = \varepsilon/2$. Now using Chernoff bound (Theorem 2.5), we have the following.

$$\Pr\left[|\bar{s}(x) - s(x)| \geqslant \alpha_1 \varepsilon' n\right] \leqslant 2 \exp\left(-\frac{\varepsilon'^2 \ell}{3}\right)$$

We now use the union bound to get the following.

$$\Pr[\exists x \in C, |\bar{s}(x) - s(x)| > \alpha_1 \varepsilon' n] \leqslant 2m \exp\left(-\frac{\varepsilon'^2 \ell}{3}\right) \tag{5.1}$$

Define $\bar{M} \stackrel{\text{def}}{=} (\bar{s}(\bar{w}) - \bar{s}(\bar{z}))/1.5\alpha_1$ the estimate of the margin of victory of the election $\mathcal{E}$ (and thus the output of the algorithm), where $\bar{w} \in \text{argmax}_{x \in C}\{\bar{s}(x)\}$ and $\bar{z} \in \text{argmax}_{x \in C \setminus \{\bar{w}\}}\{\bar{s}(x)\}$. We claim that, if $\forall x \in C, |\bar{s}(x) - s(x)| \leqslant \varepsilon' n$, then $|\bar{M} - M_\alpha(\mathcal{E})| \leqslant M_\alpha(\mathcal{E})/3 + \varepsilon n$. This can be shown as follows.

$$\begin{aligned}
\bar{M} - M_\alpha(\mathcal{E}) &= \frac{\bar{s}(\bar{w}) - \bar{s}(\bar{z})}{1.5\alpha_1} - M_\alpha(\mathcal{E}) \\
&\leqslant \frac{s(w) - s(z)}{1.5\alpha_1} + \frac{2\varepsilon' n}{1.5} - M_\alpha(\mathcal{E}) \\
&\leqslant \frac{1}{3} M_\alpha(\mathcal{E}) + \varepsilon n
\end{aligned}$$

The second inequality follows from the fact that, $\bar{s}(\bar{w}) \leqslant s(\bar{w}) + \varepsilon' n \leqslant s(w) + \varepsilon' n$ and $\bar{s}(\bar{z}) \geqslant \bar{s}(z) \geqslant s(z) - \varepsilon' n$. The third inequality follows from Lemma 5.3. Similarly, we bound $M_\alpha(\mathcal{E}) - \bar{M}$ as follows.

$$\begin{aligned}
M_\alpha(\mathcal{E}) - \bar{M} &= M_\alpha(\mathcal{E}) - \frac{\bar{s}(w) - \bar{s}(z)}{1.5\alpha_1} \\
&\leqslant M_\alpha(\mathcal{E}) - \frac{s(w) - s(z)}{1.5\alpha_1} + \frac{2\varepsilon' n}{1.5} \\
&\leqslant \frac{1}{3} M_\alpha(\mathcal{E}) + \varepsilon n
\end{aligned}$$

This proves the claim. Now we bound the success probability of the algorithm as follows. Let A be the event that $\forall x \in C, |\bar{s}(x) - s(x)| \leqslant \varepsilon'n$.

$$
\begin{aligned}
\Pr &\left[|\bar{M} - M_\alpha(\mathcal{E})| \leqslant \frac{1}{3} M_\alpha(\mathcal{E}) + \varepsilon n \right] \\
&\geqslant \Pr \left[|\bar{M} - M_\alpha(\mathcal{E})| \leqslant \frac{1}{3} M_\alpha(\mathcal{E}) + \varepsilon n \middle| A \right] \Pr[A] \\
&= \Pr[A] \\
&\geqslant 1 - 2m \exp \left(-\varepsilon'^2 \ell / 3 \right)
\end{aligned}
$$

The third equality follows from Lemma 5.3 and the fourth inequality follows from inequality 5.1. Now by choosing $\ell = (12/\varepsilon^2) \ln(2m/\delta)$, we get a $(1/3, \varepsilon, \delta)$–MoV algorithm for the scoring rules. $\qquad\square$

Now we show an algorithm for the $(0, \varepsilon, \delta)$–MoV problem for the k-approval voting rule which not only provides more accurate estimate of the margin of victory, but also has a lower sample complexity. The following structural result will be used subsequently.

Lemma 5.10. *Let $\mathcal{E} = (V, C)$ be an arbitrary instance of a k-approval election. If w and z are the candidates that receive highest and second highest score respectively in $\mathcal{E}$ and $M_{k-approval}(\mathcal{E})$ is the margin of victory of $\mathcal{E}$, then,*

$$
2(M_{k-approval}(\mathcal{E}) - 1) < s(w) - s(z) \leqslant 2M_{k-approval}(\mathcal{E})
$$

Proof. We call a vote $v \in V$ *favorable* if w appears within the top k positions and z does not appear within top the k positions in v. We claim that the number of favorable votes must be at least $M_{k-approval}(\mathcal{E})$. Indeed, otherwise, we swap the positions of w and z in all the favorable votes while keeping the other candidates fixed. This makes the score of z at least as much as the score of w which contradicts the fact that the margin of victory is $M_{k-approval}(\mathcal{E})$. Now notice that the score of z must remain less than the score of w even if we swap the positions of w and z in $M_{k-approval}(\mathcal{E}) - 1$ many favorable votes, since the margin of victory is $M_{k-approval}(\mathcal{E})$. Each such vote change increases the score of z by one and reduces the score of w by one. Hence, $2(M_{k-approval}(\mathcal{E}) - 1) < s(w) - s(z)$. Again, since the margin of victory is $M_{k-approval}(\mathcal{E})$, there exists a candidate x other than w and $M_{k-approval}(\mathcal{E})$ many votes in V which can be modified such that x becomes a winner of

the modified election. Now each vote change can reduce the score of w by at most one and increase the score of x by at most one. Hence, $s(w) - s(x) \leqslant 2M_{k-\text{approval}}(\mathcal{E})$ and thus $s(w) - s(z) \leqslant 2M_{k-\text{approval}}(\mathcal{E})$ since $s(z) \geqslant s(x)$. $\qquad\square$

With Lemma 5.4 and 5.10 at hand, we now describe our margin of victory estimation algorithm for the k-approval voting rule.

Theorem 5.15. *There is a polynomial time $(0, \varepsilon, \delta)$–MoV algorithm for the k-approval voting rule with sample complexity at most $(12/\varepsilon^2)\ln(2k/\delta)$.*

Proof. Let $\mathcal{E} = (V, C)$ be an arbitrary k-approval election. We sample ℓ votes uniformly at random from V with replacement. For a candidate x, define a random variable $X_i(x)$ which takes value 1 if x appears among the top k candidates in the ith sample vote, and 0 otherwise. Define $\bar{s}(x) \stackrel{\text{def}}{=} \frac{n}{\ell} \sum_{i=1}^{\ell} X_i(x)$ the estimate of the score of the candidate x, and let $s(x)$ be the actual score of x. Also define $\varepsilon' = \frac{\varepsilon}{2}$. Then by the Chernoff bound (Theorem 2.5), we have:

$$\Pr\left[|\bar{s}(x) - s(x)| > \varepsilon'n\right] \leqslant 2\exp\left(-\frac{\varepsilon'^2 \ell n}{3s(x)}\right)$$

Now we apply the union bound to get the following.

$$\Pr[\exists x \in C, |\bar{s}(x) - s(x)| > \varepsilon'n]$$
$$\leqslant \sum_{x \in C} 2\exp\left(-\frac{\varepsilon'^2 \ell n}{3s(x)}\right)$$
$$\leqslant 2k\exp\left(-\varepsilon'^2 \ell/3\right) \tag{5.2}$$

The second inequality follows from Lemma 5.4 : The expression $\sum_{x \in C} 2\exp\left(-\frac{\varepsilon'^2 \ell n}{3s(x)}\right)$ is maximized subject to the constraints that $0 \leqslant s(x) \leqslant n, \forall x \in C$ and $\sum_{x \in C} s(x) = kn$, when $s(x) = n \forall x \in C'$ for any subset of candidates $C' \subseteq C$ with $|C'| = k$ and $s(x) = 0 \forall x \in C \setminus C'$.

Now to estimate the margin of victory of the given election $\mathcal{E}$, let $\bar{w}$ and $\bar{z}$ be candidates with maximum and second maximum estimated score respectively. That is, $\bar{w} \in \text{argmax}_{x \in C}\{\bar{s}(x)\}$ and $\bar{z} \in \text{argmax}_{x \in C \setminus \{\bar{w}\}}\{\bar{s}(x)\}$. We define $\bar{M} \stackrel{\text{def}}{=} (\bar{s}(\bar{w}) - \bar{s}(\bar{z}))/2$ the estimate of the margin of victory of the election $\mathcal{E}$ (and thus the output of the algorithm). Let A be the event that $\forall x \in C, |\bar{s}(x) - s(x)| \leqslant \varepsilon'n$. We bound the success probability of the algorithm as follows.

101

$$\Pr\left[|\bar{M} - M_{k-approval}(\mathcal{E})| \leqslant \varepsilon n\right]$$
$$\geqslant \Pr\left[|\bar{M} - M_{k-approval}(\mathcal{E})| \leqslant \varepsilon n \big| A\right] \Pr[A]$$
$$= \Pr[A]$$
$$\geqslant 1 - 2k \exp\left(-\varepsilon'^2 \ell / 3\right)$$

The second equality follows from Lemma 5.10 and an argument analogous to the proof of Theorem 5.14. The third inequality follows from inequality 5.2. Now by choosing $\ell = (12/\varepsilon^2) \ln(2k/\delta)$, we get a $(0, \varepsilon, \delta)$–MoV algorithm. □

Note that, the sample complexity upper bound matches with the lower bound proved in Corollary 5.3 for the k-approval voting rule when k is a constant, irrespective of the number of candidates. Next, we estimate the margin of victory of an approval election.

Theorem 5.16. *There is a polynomial time $(0, \varepsilon, \delta)$–MoV algorithm for the approval rule with sample complexity at most $(12/\varepsilon^2) \ln(2m/\delta)$.*

Proof. We estimate the approval score of every candidate within an additive factor of $\varepsilon n/2$ by sampling $(12/\varepsilon^2) \ln(2m/\delta)$ many votes uniformly at random with replacement and the result follows from an argument analogous to the proofs of Lemma 5.10 and Theorem 5.15. □

5.3.2.2 Bucklin Voting Rule

Now we consider the Bucklin voting rule. Given an election $\mathcal{E} = (V, C)$, a candidate $x \in C$, and an integer $\ell \in [m]$, we denote the number of votes in V in which x appears within the top ℓ positions by $n_\ell(x)$. We prove useful bounds on the margin of victory of any Bucklin election in Lemma 5.11.

Lemma 5.11. *Let $\mathcal{E} = (V, C)$ be an arbitrary instance of a Bucklin election, w the winner of $\mathcal{E}$, and $M_{Bucklin}(\mathcal{E})$ the margin of victory of $\mathcal{E}$. Let us define a quantity $\Delta(\mathcal{E})$ as follows.*

$$\Delta(\mathcal{E}) \stackrel{\text{def}}{=\!=} \min_{\substack{\ell \in [m-1]: n_\ell(w) > n/2, \\ x \in C \setminus \{w\}: n_\ell(x) \leqslant n/2}} \{n_\ell(w) - n_\ell(x) + 1\}$$

Then,

$$\frac{\Delta(\mathcal{E})}{2} \leqslant M_{Bucklin}(\mathcal{E}) \leqslant \Delta(\mathcal{E})$$

Proof. Pick any $\ell \in [m-1]$ and $x \in C \setminus \{w\}$ such that, $n_\ell(w) > n/2$ and $n_\ell(x) \leqslant n/2$. Now by changing $n_\ell(w) - \lfloor n/2 \rfloor$ many votes, we can ensure that w is not placed within the top ℓ positions in more than $n/2$ votes: choose $n_\ell(w) - \lfloor n/2 \rfloor$ many votes where w appears within top ℓ positions and swap w with candidates placed at the last position in those votes. Similarly, by changing $\lfloor n/2 \rfloor + 1 - n_\ell(x)$ many votes, we can ensure that x is placed within top ℓ positions in more than $n/2$ votes. Hence, by changing at most $n_\ell(w) - \lfloor n/2 \rfloor + \lfloor n/2 \rfloor + 1 - n_\ell(x) = n_\ell(w) - n_\ell(x) + 1$ many votes, we can make w not win the election. Hence, $M_{\texttt{Bucklin}}(\mathcal{E}) \leqslant n_\ell(w) - n_\ell(x) + 1$. Now since we have picked an arbitrary ℓ and an arbitrary candidate x, we have $M_{\texttt{Bucklin}}(\mathcal{E}) \leqslant \Delta(\mathcal{E})$.

For the other inequality, since the margin of victory is $M_{\texttt{Bucklin}}(\mathcal{E})$, there exists an $\ell' \in [m-1]$, a candidate $x \in C \setminus \{w\}$, and $M_{\texttt{Bucklin}}(\mathcal{E})$ many votes in V such that, we can change those votes in such a way that in the modified election, w is not placed within top ℓ' positions in more than $n/2$ votes and x is placed within top ℓ' positions in more than $n/2$ votes. Hence, we have the following.

$$M_{\texttt{Bucklin}}(\mathcal{E}) \geqslant n'_\ell(w) - \left\lfloor \frac{n}{2} \right\rfloor, M_{\texttt{Bucklin}}(\mathcal{E}) \geqslant \left\lfloor \frac{n}{2} \right\rfloor + 1 - n'_\ell(x)$$

$$\Rightarrow M_{\texttt{Bucklin}}(\mathcal{E}) \geqslant \max\{n_{\ell'}(w) - \left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1 - n_{\ell'}(x)\}$$

$$\Rightarrow M_{\texttt{Bucklin}}(\mathcal{E}) \geqslant \frac{n_{\ell'}(w) - \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + 1 - n_{\ell'}(x)}{2}$$

$$\geqslant \frac{\Delta(\mathcal{E})}{2}$$

$\square$

Notice that, given an election $\mathcal{E}$, $\Delta(\mathcal{E})$ can be computed in a polynomial amount of time. Lemma 5.7 leads us to the following result for the Bucklin voting rule.

Theorem 5.17. *There is a polynomial time* $(1/3, \varepsilon, \delta)$*–MOV algorithm for the Bucklin rule with sample complexity* $(12/\varepsilon^2) \ln(2m/\delta)$.

Proof. Similar to the proof of Theorem 5.15, we estimate, for every candidate $x \in C$ and for every integer $\ell \in [m]$, the number of votes where x appears within top ℓ positions within an approximation factor of $(0, \varepsilon/2)$. Next, we compute an estimate of $\bar{\Delta}(\mathcal{E})$ from the sampled votes and output the estimate for the margin of victory as $\bar{\Delta}(\mathcal{E})/1.5$. Using Lemma 5.11, we can argue the rest of the proof in a way that is analogous to the proofs of Theorem 5.5 and 5.6. $\square$

5.3.2.3 Maximin Voting Rule

Next, we present our $(1/3, \varepsilon, \delta)$–MOV algorithm for the maximin voting rule.

Theorem 5.18. *There is a polynomial time $(1/3, \varepsilon, \delta)$–MOV algorithm for the maximin rule with sample complexity $(24/\varepsilon^2) \ln(2m/\delta)$.*

Proof. Let $\mathcal{E} = (V, C)$ be an instance of maximin election. Let x and y be any two candidates. We sample ℓ votes uniformly at random from the set of all votes with replacement. Let $X_i(x, y)$ be a random variable defined as follows.

$$X_i(x, y) = \begin{cases} 1, & \text{if } x \succ y \text{ in the } i^{\text{th}} \text{ sample vote} \\ -1, & \text{else} \end{cases}$$

Define $\bar{D}_{\mathcal{E}}(x, y) = \frac{n}{\ell} \sum_{i=1}^{\ell} X_i(x, y)$. By using the Chernoff bound and union bound, we have the following.

$$\Pr\left[\exists x, y \in C, |\bar{D}_{\mathcal{E}}(x, y) - D_{\mathcal{E}}(x, y)| > \varepsilon n \right] \leqslant 2m^2 \exp\left(-\frac{\varepsilon^2 \ell}{3} \right)$$

We define $\bar{M} \overset{\text{def}}{=\joinrel=} (\bar{s}(\bar{w}) - \bar{s}(\bar{z}))/3$, the estimate of the margin of victory of $\mathcal{E}$, where $\bar{w} \in \text{argmax}_{x \in C}\{\bar{s}(x)\}$ and $\bar{z} \in \text{argmax}_{x \in C \setminus \{\bar{w}\}}\{\bar{s}(x)\}$. Now using Lemma 5.5, we can complete the rest of the proof in a way that is analogous to the proof of Theorem 5.14. □

5.3.2.4 Copeland$^\alpha$ Voting Rule

Now we present our result for the Copeland$^\alpha$ voting rule. The following lemma is immediate from Theorem 11 in [Xia12].

Lemma 5.12. $\Gamma(\mathcal{E}) \leqslant M_{Copeland^\alpha}(\mathcal{E}) \leqslant 2(\lceil \ln m \rceil + 1)\Gamma(\mathcal{E})$.

Proof. Follows from Theorem 11 in [Xia12]. □

Theorem 5.19. *For the Copeland$^\alpha$ voting rule, there is a polynomial time $(1 - O(1/\ln m), \varepsilon, \delta)$–MOV algorithm whose sample complexity is $(96/\varepsilon^2) \ln(2m/\delta)$.*

Proof. Let $\mathcal{E} = (V, C)$ be an instance of a Copeland$^\alpha$ election. For every $x, y \in C$, we compute $\bar{D}_{\mathcal{E}}(x, y)$, which is an estimate of $D_{\mathcal{E}}(x, y)$, within an approximation factor of $(0, \varepsilon')$, where $\varepsilon' = \varepsilon/4$. This can be achieved with an error probability at most δ by sampling

$(^{96}/_{\varepsilon^2}) \ln(^{2m}/_{\delta})$ many votes uniformly at random with replacement (the argument is same as the proof of Theorem 5.5). We define $\bar{s}'_t(V, x) = |\{y \in C : y \neq x, D_\varepsilon(y, x) < 2t\}| + \alpha|\{y \in C : y \neq x, D_\varepsilon(y, x) = 2t\}|$. We also define $\overline{RM}(x, y)$ between x and y to be the minimum integer t such that, $\bar{s}'_{-t}(V, x) \leqslant s'_t(V, y)$. Let $\bar{w}$ be the winner of the sampled election, $\bar{z} = \text{argmin}_{x \in C \setminus \{\bar{w}\}}\{\overline{RM}(w, x)\}$, w the winner of $\mathcal{E}$, and $z = \text{argmin}_{x \in C \setminus \{w\}}\{RM(w, x)\}$. Since, $\bar{D}_\varepsilon(x, y)$ is an approximation of $D_\varepsilon(x, y)$ within a factor of $(0, \varepsilon')$, we have the following for every candidate $x, y \in C$.

$$s'_t(V, x) - \varepsilon'n \leqslant \bar{s}'_t(V, x) \leqslant s'_t(V, x) + \varepsilon'n$$

$$RM(x, y) - 2\varepsilon'n \leqslant \overline{RM}(x, y) \leqslant RM(x, y) + 2\varepsilon'n \tag{5.3}$$

Define $\bar{\Gamma}(\mathcal{E}) = \overline{RM}(\bar{w}, \bar{z})$ to be the estimate of $\Gamma(\mathcal{E})$. We show the following claim.

Claim 5.1. *With the above definitions of $w, z, \bar{w}$, and $\bar{z}$, we have the following.*

$$\Gamma(\mathcal{E}) - 4\varepsilon'n \leqslant \bar{\Gamma}(\mathcal{E}) \leqslant \Gamma(\mathcal{E}) + 4\varepsilon'n$$

Proof. Below, we show the upper bound for $\bar{\Gamma}(\mathcal{E})$.

$$\begin{aligned}
\bar{\Gamma}(\mathcal{E}) = \overline{RM}(\bar{w}, \bar{z}) &\leqslant \overline{RM}(w, \bar{z}) + 2\varepsilon'n \\
&\leqslant \overline{RM}(w, z) + 2\varepsilon'n \\
&\leqslant RM(w, z) + 4\varepsilon'n \\
&= \Gamma(\mathcal{E}) + 4\varepsilon'n
\end{aligned}$$

The second inequality follows from the fact that $\bar{D}_\varepsilon(x, y)$ is an approximation of $D_\varepsilon(x, y)$ by a factor of $(0, \varepsilon')$. The third inequality follows from the definition of $\bar{z}$, and the fourth inequality uses inequality 5.3. Now we show the lower bound for $\bar{\Gamma}(\mathcal{E})$.

$$\begin{aligned}
\bar{\Gamma}(\mathcal{E}) = \overline{RM}(\bar{w}, \bar{z}) &\geqslant \overline{RM}(w, \bar{z}) - 2\varepsilon'n \\
&\geqslant RM(w, \bar{z}) - 4\varepsilon'n \\
&\geqslant RM(w, z) - 4\varepsilon'n \\
&= \Gamma(\mathcal{E}) - 4\varepsilon'n
\end{aligned}$$

105

The third inequality follows from inequality 5.3 and the fourth inequality follows from the definition of z. $\qquad\square$

We define $\bar{M}$, the estimate of $M_{Copeland^\alpha}(\mathcal{E})$, to be $\frac{4(\ln m+1)}{2\ln m+3}\bar{\Gamma}(\mathcal{E})$. The following argument shows that $\bar{M}$ is a $\left(1-O\left(\frac{1}{\ln m}\right),\varepsilon,\delta\right)$–estimate of $M_{Copeland^\alpha}(\mathcal{E})$.

$$
\begin{aligned}
&\bar{M}-M_{Copeland^\alpha}(\mathcal{E})\\
&=\frac{4(\ln m+1)}{2\ln m+3}\bar{\Gamma}(\mathcal{E})-M_{Copeland^\alpha}(\mathcal{E})\\
&\leqslant\frac{4(\ln m+1)}{2\ln m+3}\Gamma(\mathcal{E})-M_{Copeland^\alpha}(\mathcal{E})+\frac{16(\ln m+1)}{2\ln m+3}\varepsilon'n\\
&\leqslant\frac{4(\ln m+1)}{2\ln m+3}M_{Copeland^\alpha}(\mathcal{E})-M_{Copeland^\alpha}(\mathcal{E})+\varepsilon n\\
&\leqslant\frac{2\ln m+1}{2\ln m+3}M_{Copeland^\alpha}(\mathcal{E})+\varepsilon n\\
&\leqslant\left(1-O\left(\frac{1}{\ln m}\right)\right)M_{Copeland^\alpha}(\mathcal{E})+\varepsilon n
\end{aligned}
$$

The second inequality follows from Claim 5.1 and the third inequality follows from Lemma 5.12. Analogously, we have:

$$
\begin{aligned}
&M_{Copeland^\alpha}(\mathcal{E})-\bar{M}\\
&=M_{Copeland^\alpha}(\mathcal{E})-\frac{4(\ln m+1)}{2\ln m+3}\bar{\Gamma}(\mathcal{E})\\
&\leqslant M_{Copeland^\alpha}(\mathcal{E})-\frac{4(\ln m+1)}{2\ln m+3}\Gamma(\mathcal{E})+\frac{16(\ln m+1)}{2\ln m+3}\varepsilon'n\\
&\leqslant M_{Copeland^\alpha}(\mathcal{E})-\frac{2(\ln m+1)}{2\ln m+3}M_{Copeland^\alpha}(\mathcal{E})+\varepsilon n\\
&\leqslant\frac{2\ln m+1}{2\ln m+3}M_{Copeland^\alpha}(\mathcal{E})+\varepsilon n\\
&\leqslant\left(1-O\left(\frac{1}{\ln m}\right)\right)M_{Copeland^\alpha}(\mathcal{E})+\varepsilon n
\end{aligned}
$$

The second line follows Claim 5.1 and the third line follows from Lemma 5.12. $\qquad\square$

The approximation factor in Theorem 5.19 is weak when we have a large number of candidates. The main difficulty for showing a better approximation factor for the Copeland$^\alpha$ voting

rule is to find a polynomial time computable quantity (for example, $\Gamma(\mathcal{E})$ in Lemma 5.12) that exhibits tight bounds with margin of victory. We remark that, existence of such a quantity will not only imply a better estimation algorithm, but also, a better approximation algorithm (the best known approximation factor for finding the margin of victory for the Copeland$^\alpha$ voting rule is $O(\ln m)$ and it uses the quantity $\Gamma(\mathcal{E})$). However, we remark that Theorem 5.19 will be useful in applications, for example, post election audit and polling, where the number of candidates is often small.

5.4   Conclusion

In this work, we introduced the (ε, δ)-WINNER DETERMINATION problem and showed (often tight) bounds for the sample complexity for many common voting rules. We have also introduced the (c, ε, δ)–MARGIN OF VICTORY problem and presented efficient sampling based algorithms for solving it for many commonly used voting rules which are also often observes an optimal number of sample votes. We observe that predicting the winner of an elections needs least number of queries, whereas more involved voting rules like Borda and maximin need significantly more queries.

In the next chapter, we study the problem of finding a winner of an election when votes are arriving one by one in a sequential manner.

Chapter 6

Streaming Algorithms for Winner Determination

We give the first optimal bounds for returning the ℓ_1-heavy hitters in a data stream of insertions, together with their approximate frequencies, closing a long line of work on this problem. For a stream of m items in $\{1, 2, \ldots, n\}$ and parameters $0 < \varepsilon < \phi \leqslant 1$, let f_i denote the frequency of item i, i.e., the number of times item i occurs in the stream. With arbitrarily large constant probability, our algorithm returns all items i for which $f_i \geqslant \phi m$, returns no items j for which $f_j \leqslant (\phi - \varepsilon)m$, and returns approximations $\tilde{f}_i$ with $|\tilde{f}_i - f_i| \leqslant \varepsilon m$ for each item i that it returns. Our algorithm uses $O(\varepsilon^{-1} \log \phi^{-1} + \phi^{-1} \log n + \log \log m)$ bits of space, processes each stream update in $O(1)$ worst-case time, and can report its output in time linear in the output size. We also prove a lower bound, which implies that our algorithm is optimal up to a constant factor in its space complexity. A modification of our algorithm can be used to estimate the maximum frequency up to an additive εm error in the above amount of space, resolving Question 3 in the IITK 2006 Workshop on Algorithms for Data Streams for the case of ℓ_1-heavy hitters. We also introduce several variants of the heavy hitters and maximum frequency problems, inspired by rank aggregation and voting schemes, and show how our techniques can be applied in such settings. Unlike the traditional heavy hitters problem, some of these variants look at

---

A preliminary version of the work in this chapter was published as [BDW16]: Arnab Bhattacharyya, Palash Dey, and David P. Woodruff. An optimal algorithm for l1-heavy hitters in insertion streams and related problems. In Proc. 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 16, pages 385-400, New York, NY, USA, 2016. ACM.

comparisons between items rather than numerical values to determine the frequency of an item.

6.1 Introduction

The data stream model has emerged as a standard model for processing massive data sets. Because of the sheer size of the data, traditional algorithms are no longer feasible, e.g., it may be hard or impossible to store the entire input, and algorithms need to run in linear or even sublinear time. Such algorithms typically need to be both randomized and approximate. Moreover, the data may not physically reside on any device, e.g., if it is internet traffic, and so if the data is not stored by the algorithm, it may be impossible to recover it. Hence, many algorithms must work given only a single pass over the data. Applications of data streams include data warehousing [HSST05, BR99, FSG+98, HPDW01], network measurements [ABW03, CKMS08, DLOM02, EV03], sensor networks [BGS01, SBAS04], and compressed sensing [GSTV07, CRT06]. We refer the reader to recent surveys on the data stream model [Mut05, Nel12, Cor12].

One of the oldest and most fundamental problems in the area of data streams is the problem of finding the ℓ_1-heavy hitters (or simply, "heavy hitters"), also known as the top-k, most popular items, frequent items, elephants, or iceberg queries. Such algorithms can be used as subroutines in network flow identification at IP routers [EV03], association rules and frequent itemsets [AS94, SON95, Toi96, Hid99, HPY00], iceberg queries and iceberg datacubes [FSG+98, BR99, HPDW01]. The survey [CH08] presents an overview of the state-of-the-art for this problem, from both theoretical and practical standpoints.

We now formally define the heavy hitters problem that we focus on in this work:

Definition 6.1. ((ε, ϕ)-**Heavy Hitters Problem**) *In the* (ε, ϕ)-*Heavy Hitters Problem, we are given parameters* $0 < \varepsilon < \phi \leqslant 1$ *and a stream* $a_1, \ldots, a_m$ *of items* $a_j \in \{1, 2, \ldots, n\}$. *Let* f_i *denote the number of occurrences of item* i, *i.e., its frequency. The algorithm should make one pass over the stream and at the end of the stream output a set* $S \subseteq \{1, 2, \ldots, n\}$ *for which if* $f_i \geqslant \phi m$, *then* $i \in S$, *while if* $f_i \leqslant (\phi - \varepsilon)m$, *then* $i \notin S$. *Further, for each item* $i \in S$, *the algorithm should output an estimate* $\tilde{f}_i$ *of the frequency* f_i *which satisfies* $|f_i - \tilde{f}_i| \leqslant \varepsilon m$.

Note that other natural definitions of heavy hitters are possible and sometimes used. For example, ℓ_2-*heavy hitters* are those items i for which $f_i^2 \geqslant \phi^2 \sum_{j=1}^n f_j^2$, and more generally,

ℓ_p-*heavy hitters* are those items i for which $f_i^p \geqslant \phi^p \sum_{j=1}^n f_j^p$. It is in this sense that Definition 6.1 corresponds to ℓ_1-heavy hitters. While ℓ_p-heavy hitters for $p > 1$ relax ℓ_1-heavy hitters and algorithms for them have many interesting applications, we focus on the most direct and common formulation of the heavy hitters notion.

We are interested in algorithms which use as little space in bits as possible to solve the (ε, ϕ)-**Heavy Hitters Problem**. Further, we are also interested in minimizing the *update time* and *reporting time* of such algorithms. Here, the update time is defined to be the time the algorithm needs to update its data structure when processing a stream insertion. The reporting time is the time the algorithm needs to report the answer after having processed the stream. We allow the algorithm to be randomized and to succeed with probability at least $1 - \delta$ for $0 < \delta < 1$. We do not make any assumption on the ordering of the stream $a_1, \ldots, a_m$. This is desirable as often in applications one cannot assume a best-case or even a random order. We are also interested in the case when the length m of the stream is not known in advance, and give algorithms in this more general setting.

The first algorithm for the (ε, ϕ)-**Heavy Hitters Problem** was given by Misra and Gries [MG82], who achieved $O(\varepsilon^{-1}(\log n + \log m))$ bits of space for any $\phi > \varepsilon$. This algorithm was rediscovered by Demaine et al. [DLOM02], and again by Karp et al. [KSP03]. Other than these algorithms, which are deterministic, there are also a number of randomized algorithms, such as the CountSketch [CCFC04], Count-Min sketch [CM05], sticky sampling [MM02], lossy counting [MM02], space-saving [MAE05], sample and hold [EV03], multi-stage bloom filters [CFM09], and sketch-guided sampling [KX06]. Berinde et al. [BICS10] show that using $O(k\varepsilon^{-1}\log(mn))$ bits of space, one can achieve the stronger guarantee of reporting, for each item $i \in S$, $\tilde{f}_i$ with $|\tilde{f}_i - f_i| \leqslant \varepsilon/k F_1^{res(k)}$, where $F_1^{res(k)} < m$ denotes the sum of frequencies of items in $\{1, 2, \ldots, n\}$ excluding the frequencies of the k most frequent items.

We emphasize that prior to our work the best known algorithms for the (ε, ϕ)-**Heavy Hitters Problem** used $O(\varepsilon^{-1}(\log n + \log m))$ bits of space. Two previous lower bounds were known. The first is a lower bound of $\log\left(\binom{n}{1/\phi}\right) = \Omega(\phi^{-1}\log(\phi n))$ bits, which comes from the fact that the output set S can contain ϕ^{-1} items and it takes this many bits to encode them. The second lower bound is $\Omega(\varepsilon^{-1})$ which follows from a folklore reduction from the randomized communication complexity of the Index problem. In this problem, there are two players, Alice and Bob. Alice has a bit string x of length $(2\varepsilon)^{-1}$, while Bob has an index i. Alice creates a stream of length $(2\varepsilon)^{-1}$ consisting of one copy of each j for which $x_j = 1$ and copies of a dummy item to fill the rest of the stream. She runs the heavy hitters streaming

algorithm on her stream and sends the state of the algorithm to Bob. Bob appends $(2\varepsilon)^{-1}$ copies of the item i to the stream and continues the execution of the algorithm. For $\phi = 1/2$, it holds that $i \in S$. Moreover, f_i differs by an additive εm factor depending on whether $x_i = 1$ or $x_i = 0$. Therefore by the randomized communication complexity of the Index problem [KNR99], the $(\varepsilon, 1/2)$-heavy hitters problem requires $\Omega(\varepsilon^{-1})$ bits of space. Although this proof was for $\phi = 1/2$, no better lower bound is known for any $\phi > \varepsilon$.

Thus, while the upper bound for the (ε, ϕ)-**Heavy Hitters Problem** is $O(\varepsilon^{-1}(\log n + \log m))$ bits, the best known lower bound is only $\Omega(\phi^{-1} \log n + \varepsilon^{-1})$ bits. For constant ϕ, and $\log n \approx \varepsilon^{-1}$, this represents a nearly quadratic gap in upper and lower bounds. Given the limited resources of devices which typically run heavy hitters algorithms, such as internet routers, this quadratic gap can be critical in applications.

A problem related to the (ε, ϕ)-**Heavy Hitters Problem** is estimating the *maximum frequency* in a data stream, also known as the ℓ_∞-norm. In the IITK 2006 Workshop on Algorithms for Data Streams, Open Question 3 asks for an algorithm to estimate the maximum frequency of any item up to an additive εm error using as little space as possible. The best known space bound is still $O(\varepsilon^{-1} \log n)$ bits, as stated in the original formulation of the question (note that the "m" in the question there corresponds to the "n" here). Note that, if one can find an item whose frequency is the largest, up to an additive εm error, then one can solve this problem. The latter problem is independently interesting and corresponds to finding approximate plurality election winners in voting streams [DB15]. We refer to this problem as the ε-**Maximum** problem.

Finally, we note that there are many other variants of the (ε, ϕ)-**Heavy Hitters Problem** that one can consider. One simple variant of the above is to output an item of frequency within εm of the *minimum frequency* of any item in the universe. We refer to this as the ε-**Minimum** problem. This only makes sense for small universes, as otherwise outputting a random item typically works. This is useful when one wants to count the "number of dislikes", or in anomaly detection; see more motivation below. In other settings, one may not have numerical scores associated with the items, but rather, each stream update consists of a "ranking" or "total ordering" of all stream items. This may be the case in ranking aggregation on the web (see, e.g., [MBG04, MYCC07]) or in voting streams (see, e.g., [CS05, CP11, DB15, Xia12]). One may consider a variety of aggregation measures, such as the Borda score of an item i, which asks for the sum, over rankings, of the number of items $j \neq i$ for which i is ranked ahead of j in the ranking. Alternatively, one may consider the Maximin score of an item i,

Problem	Space complexity	
	Upper bound	Lower bound
(ε, ϕ)-Heavy Hitters	$O\left(\varepsilon^{-1}\log\phi^{-1} + \phi^{-1}\log n + \log\log m\right)$ [Theorem 6.2 and 6.7]	$\Omega\left(\varepsilon^{-1}\log\phi^{-1} + \phi^{-1}\log n + \log\log m\right)$ [Theorem 6.9 and 6.14]
ε-Maximum and ℓ_∞-approximation	$O\left(\varepsilon^{-1}\log\varepsilon^{-1} + \log n + \log\log m\right)$ [Theorem 6.1 and 6.7]	$\Omega\left(\varepsilon^{-1}\log\varepsilon^{-1} + \log n + \log\log m\right)$ [Theorem 6.10 and 6.14]
ε-Minimum	$O\left(\varepsilon^{-1}\log\log\varepsilon^{-1} + \log\log m\right)$ [Theorem 6.4 and 6.8]	$\Omega\left(\varepsilon^{-1} + \log\log m\right)$ [Theorem 6.11 and 6.14]
ε-Borda	$O\left(n(\log\varepsilon^{-1} + \log n) + \log\log m\right)$ [Theorem 6.5 and 6.8]	$\Omega\left(n(\log\varepsilon^{-1} + \log n) + \log\log m\right)$ [Theorem 6.12 and 6.14]
ε-Maximin	$O\left(n\varepsilon^{-2}\log^2 n + \log\log m\right)$ [Theorem 6.6 and 6.8]	$\Omega\left(n(\varepsilon^{-2} + \log n) + \log\log m\right)$ [Theorem 6.13]

Table 6.1: The bounds hold for constant success probability algorithms and for n sufficiently large in terms of ε. For the (ε, ϕ)-**Heavy Hitters** problem and the ε-**Maximum** problem, we also achieve $O(1)$ update time and reporting time which is linear in the size of the output. The upper bound for ε-**Borda** (resp. ε-**Maximin**) is for returning every item's Borda score (resp. Maximin score) up to an additive εmn (resp. additive εm), while the lower bound for ε-**Borda** (resp. ε-**Maximin**) is for returning only the approximate Borda score (resp. Maximin score) of an approximate maximum.

which asks for the minimum, over items $j \neq i$, of the number of rankings for which i is ranked ahead of j. For these aggregation measures, one may be interested in finding an item whose score is an approximate maximum. This is the analogue of the ε-**Maximum** problem above. Or, one may be interested in listing all items whose score is above a threshold, which is the analogue of the (ε, ϕ)-**Heavy Hitters Problem**.

We give more motivation of these variants of heavy hitters in this section below, and more precise definitions in Section 6.2.

6.1.1 Our Contribution

Our results are summarized in Table 6.1. We note that independently of this work and nearly parallelly, there have been improvements to the space complexity of the ℓ_2-heavy hitters problem in insertion streams [BCIW16] and to the time complexity of the ℓ_1-heavy hitters problem in turnstile[1] streams [LNNT16]. These works use very different techniques.

Our first contribution is an optimal algorithm and lower bound for the (ε, ϕ)-**Heavy Hitters Problem**. Namely, we show that there is a randomized algorithm with constant proba-

---

[1]In a turnstile stream, updates modify an underlying n-dimensional vector x initialized at the zero vector; each update is of the form $x \leftarrow x + e_i$ or $x \leftarrow x - e_i$ where e_i is the i'th standard unit vector. In an insertion stream, only updates of the form $x \leftarrow x + e_i$ are allowed.

bility of success which solves this problem using

$$O(\varepsilon^{-1} \log \phi^{-1} + \phi^{-1} \log n + \log\log m)$$

bits of space, and we prove a lower bound matching up to constant factors. In the unit-cost RAM model with $O(\log n)$ bit words, our algorithm has $O(1)$ update time and reporting time linear in the output size, under the standard assumptions that the length of the stream and universe size are at least $\mathrm{poly}(\varepsilon^{-1} \log(1/\phi))$. Furthermore, we can achieve nearly the optimal space complexity even when the length m of the stream is *not known in advance*. Although the results of [BICS10] achieve stronger error bounds in terms of the tail, which are useful for skewed streams, here we focus on the original formulation of the problem.

Next, we turn to the problem of estimating the maximum frequency in a data stream up to an additive εm. We give an algorithm using

$$O(\varepsilon^{-1} \log \varepsilon^{-1} + \log n + \log\log m)$$

bits of space, improving the previous best algorithms which required space at least $\Omega(\varepsilon^{-1} \log n)$ bits, and show that our bound is tight. As an example setting of parameters, if $\varepsilon^{-1} = \Theta(\log n)$ and $\log\log m = O(\log n)$, our space complexity is $O(\log n \log\log n)$ bits, improving the previous $\Omega(\log^2 n)$ bits of space algorithm. We also prove a lower bound showing our algorithm is optimal up to constant factors. This resolves Open Question 3 from the IITK 2006 Workshop on Algorithms for Data Streams in the case of insertion streams, for the case of "ℓ_1-heavy hitters". Our algorithm also returns the identity of the item with the approximate maximum frequency, solving the ε-**Maximum** problem.

We then focus on a number of variants of these problems. We first give nearly tight bounds for finding an item whose frequency is within εm of the minimum possible frequency. While this can be solved using our new algorithm for the $(\varepsilon, \varepsilon)$-**Heavy Hitters Problem**, this would incur $\Omega(\varepsilon^{-1} \log \varepsilon^{-1} + \log\log m)$ bits of space, whereas we give an algorithm using only $O(\varepsilon^{-1} \log\log(\varepsilon^{-1}) + \log\log m)$ bits of space. We also show a nearly matching $\Omega(\varepsilon^{-1} + \log\log m)$ bits of space lower bound. We note that for this problem, a dependence on n is not necessary since if the number of possible items is sufficiently large, then outputting the identity of a random item among the first say, $10\varepsilon^{-1}$ items, is a correct solution with large constant probability.

Finally, we study variants of heavy hitter problems that are ranking-based. In this setting, each stream update consists of a total ordering of the n universe items. For the ε-**Borda** problem, we give an algorithm using $O(n(\log \varepsilon^{-1} + \log\log n) + \log\log m)$ bits of space to report the Borda score of every item up to an additive εmn. We also show this is nearly optimal by proving an $\Omega(n \log \varepsilon^{-1} + \log\log m)$ bit lower bound for the problem, even in the case when one is only interested in outputting an item maximum Borda score up to an additive εmn. For the ε-**Maximin** problem, we give an algorithm using $O(n\varepsilon^{-2} \log^2 n + \log\log m)$ bits of space to report the maximin score of every item up to an additive εm, and prove an $\Omega(n\varepsilon^{-2} + \log\log m)$ bits of space lower bound even in the case when one is only interested in outputting the maximum maximin score up to an additive εm. This shows that finding heavy hitters with respect to the maximin score is significantly more expensive than with respect to the Borda score.

6.1.2 Motivation for Variants of Heavy Hitters

While the (ε, ϕ)-**Heavy Hitters** and ε-**Maximum** problem are very well-studied in the data stream literature, the other variants introduced are not. We provide additional motivation for them here.

For the ε-**Minimum** problem, in our formulation, an item with frequency zero, i.e., one that does not occur in the stream, is a valid solution to the problem. In certain scenarios, this might not make sense, e.g., if a stream containing only a small fraction of IP addresses. However, in other scenarios we argue this is a natural problem. For instance, consider an online portal where users register complaints about products. Here, minimum frequency items correspond to the "best" items. That is, such frequencies arise in the context of voting or more generally making a choice: in cases for which one does not have a strong preference for an item, but definitely does not like certain items, this problem applies, since the frequencies correspond to "number of dislikes".

The ε-**Minimum** problem may also be useful for anomaly detection. Suppose one has a known set of sensors broadcasting information and one observes the "From:" field in the broadcasted packets. Sensors which send a small number of packets may be down or defective, and an algorithm for the ε-**Minimum** problem could find such sensors.

Finding items with maximum and minimum frequencies in a stream correspond to finding winners under plurality and veto voting rules respectively in the context of voting[1] [BCE$^+$15]. The streaming aspect of voting could be crucial in applications like online polling [KTW11], recommender systems [RV97, HKTR04, AT05] where the voters are providing their votes in a streaming fashion and at every point in time, we would like to know the popular items. While in some elections, such as for political positions, the scale of the election may not be large enough to require a streaming algorithm, one key aspect of these latter voting-based problems is that they are rank-based which is useful when numerical scores are not available. Orderings naturally arise in several applications - for instance, if a website has multiple parts, the order in which a user visits the parts given by its clickstream defines a voting, and for data mining and recommendation purposes the website owner may be interested in aggregating the orderings across users. Motivated by this connection, we define similar problems for two other important voting rules, namely Borda and maximin. The Borda scoring method finds its applications in a wide range of areas of artificial intelligence, for example, machine learning [HHS94, CP06, VZ14, PPR15], image processing [LN06, MG09], information retrieval [Li14, AM01, NC06], etc. The Maximin score is often used when the spread between the best and worst outcome is very large (see, e.g., p. 373 of [MR91]). The maximin scoring method also has been used frequently in machine learning [WL04, JMP$^+$14], human computation [MPC12, MPC13], etc.

6.2 Problem Definitions

We now formally define the problems we study here. Suppose we have $0 < \varepsilon < \varphi < 1$.

Definition 6.2. (ε, φ)-LIST HEAVY HITTERS
Given an insertion-only stream of length m over a universe U of size n, find all items in U with frequency more than φm, along with their frequencies up to an additive error of εm, and report no items with frequency less than $(\varphi - \varepsilon)m$.

Definition 6.3. ε-MAXIMUM
Given an insertion-only stream of length m over a universe U of size n, find the maximum frequency up to an additive error of εm.

---

[1]In fact, the first work [Moo81] to formally pose the heavy hitters problem couched it in the context of voting.

Next we define the *minimum* problem for $0 < \varepsilon < 1$.

Definition 6.4. ε-MINIMUM

Given an insertion-only stream of length m over a universe $\mathcal{U}$ of size n, find the minimum frequency up to an additive error of εm.

Next we define related heavy hitters problems in the context of rank aggregation. The input is a stream of rankings (permutations) over an item set $\mathcal{U}$ for the problems below. The *Borda score* of an item i is the sum, over all rankings, of the number of items $j \neq i$ for which i is ranked ahead of j in the ranking.

Definition 6.5. (ε, φ)-LIST BORDA

Given an insertion-only stream over a universe $\mathcal{L}(\mathcal{U})$ where $|\mathcal{U}| = n$, find all items with Borda score more than $\varphi m n$, along with their Borda score up to an additive error of $\varepsilon m n$, and report no items with Borda score less than $(\varphi - \varepsilon) m n$.

Definition 6.6. ε-BORDA

Given an insertion-only stream over a universe $\mathcal{L}(\mathcal{U})$ where $|\mathcal{U}| = n$, find the maximum Borda score up to an additive error of $\varepsilon m n$.

The *maximin score* of an item i is the minimum, over all items $j \neq i$, of the number of rankings for which i is ranked ahead of j.

Definition 6.7. (ε, φ)-LIST MAXIMIN

Given an insertion-only stream over a universe $\mathcal{L}(\mathcal{U})$ where $|\mathcal{U}| = n$, find all items with maximin score more than φm along with their maximin score up to an additive error of εm, and report no items with maximin score less than $(\varphi - \varepsilon) m$.

Definition 6.8. ε-MAXIMIN

Given an insertion-only stream over a universe $\mathcal{L}(\mathcal{U})$ where $|\mathcal{U}| = n$, find the maximum maximin score up to an additive error of εm.

Notice that the maximum possible Borda score of an item is $m(n - 1) = \Theta(mn)$ and the maximum possible maximin score of an item is m. This justifies the approximation factors in Definition 6.5 to 6.8. We note that finding an item with maximum Borda score within additive $\varepsilon m n$ or maximum maximin score within additive εm corresponds to finding an approximate winner of an election (more precisely, what is known as an ε-winner) [DB15].

6.3   Our Algorithms

In this section, we present our upper bound results. All omitted proofs are in Appendix B. Before describing specific algorithms, we record some claims for later use. We begin with the following space efficient algorithm for picking an item uniformly at random from a universe of size n below.

Lemma 6.1. *Suppose m is a power of two[1]. Then there is an algorithm $\mathcal{A}$ for choosing an item with probability $1/m$ that has space complexity of $O(\log \log m)$ bits and time complexity of $O(1)$ in the unit-cost RAM model.*

Proof. We generate a $(\log_2 m)$-bit integer C uniformly at random and record the sum of the digits in C. Choose the item only if the sum of the digits is 0, i.e. if $C = 0$.   □

   We remark that the algorithm in Lemma 6.1 has optimal space complexity as shown in Proposition 6.1 in Appendix B.

   We remark that the algorithm in Lemma 6.1 has optimal space complexity as shown in Proposition 6.1 below which may be of independent interest. We also note that every algorithm needs to toss a fair coin at least $\Omega(\log m)$ times to perform any task with probability at least $1/m$.

Proposition 6.1. *Any algorithm that chooses an item from a set of size n with probability p for $0 < p \leqslant \frac{1}{n}$, in unit cost RAM model must use $\Omega(\log \log m)$ bits of memory.*

Proof. The algorithm generates t bits uniformly at random (the number of bits it generates uniformly at random may also depend on the outcome of the previous random bits) and finally picks an item from the say x. Consider a run $\mathcal{R}$ of the algorithm where it chooses the item x with *smallest number of random bits getting generated*; say it generates t random bits in this run $\mathcal{R}$. This means that in any other run of the algorithm where the item x is chosen, the algorithm must generate at least t many random bits. Let the random bits generated in $\mathcal{R}$ be $r_1, \cdots, r_t$. Let s_i be the memory content of the algorithm immediately after it generates i^{th} random bit, for $i \in [t]$, in the run $\mathcal{R}$. First notice that if $t < \log_2 n$, then the probability with which the item x is chosen is more than $\frac{1}{n}$, which would be a contradiction. Hence, $t \geqslant \log_2 n$. Now we claim that all the s_i's must be different. Indeed otherwise, let us assume

---

[1]In all our algorithms, whenever we pick an item with probability $p > 0$, we can assume, without loss of generality, that $1/p$ is a power of two. If not, then we replace p with p' where $1/p'$ is the largest power of two less than $1/p$. This does not affect correctness and performance of our algorithms.

$s_i = s_j$ for some $i < j$. Then the algorithm chooses the item x after generating $t - (j - i)$ many random bits (which is strictly less than t) when the random bits being generated are $r_1, \cdots, r_i, r_{j+1}, \cdots, r_t$. This contradicts the assumption that the run $\mathcal{R}$ we started with chooses the item x with smallest number of random bits generated. $\qquad\square$

Our second claim is a standard result for universal families of hash functions.

Lemma 6.2. *For $S \subseteq A$, $\delta \in (0, 1)$, and universal family of hash functions $\mathcal{H} = \{h | h : A \to [[\lceil |S|^2/\delta \rceil]]\}$:*

$$\Pr_{h \in_u \mathcal{H}}[\exists i \neq j \in S, h(i) = h(j)] \leqslant \delta$$

Proof. For every $i \neq j \in S$, since $\mathcal{H}$ is a universal family of hash functions, we have $\Pr_{h \in_u \mathcal{H}}[h(i) = h(j)] \leqslant \frac{1}{\lceil |S|^2/\delta \rceil}$. Now we apply the union bound to get $\Pr_{h \in_u \mathcal{H}}[\exists i \neq j \in S, h(i) = h(j)] \leqslant \frac{|S|^2}{\lceil |S|^2/\delta \rceil} \leqslant \delta$ $\qquad\square$

Our third claim is folklore and also follows from the celebrated DKW inequality [DKW56]. We provide a simple proof here that works for constant δ.

Lemma 6.3. *Let f_i and $\hat{f}_i$ be the frequencies of an item i in a stream $\mathcal{S}$ and in a random sample $\mathcal{T}$ of size r from $\mathcal{S}$, respectively. Then for $r \geqslant 2\varepsilon^{-2} \log(2\delta^{-1})$, with probability $1 - \delta$, for every universe item i simultaneously,*

$$\left| \frac{\hat{f}_i}{r} - \frac{f_i}{m} \right| \leqslant \varepsilon.$$

Proof for constant δ. This follows by Chebyshev's inequality and a union bound. Indeed, consider a given $i \in [n]$ with frequency f_i and suppose we sample each of its occurrences pairwise-independently with probability r/m, for a parameter r. Then the expected number $\mathbf{E}[\hat{f}_i]$ of sampled occurrences is $f_i \cdot r/m$ and the variance $\mathbf{Var}[\hat{f}_i]$ is $f_i \cdot r/m(1 - r/m) \leqslant f_i r/m$. Applying Chebyshev's inequality,

$$\Pr\left[\left| \hat{f}_i - \mathbf{E}[\hat{f}_i] \right| \geqslant \frac{r\varepsilon}{2} \right] \leqslant \frac{\mathbf{Var}[\hat{f}_i]}{(r\varepsilon/2)^2} \leqslant \frac{4f_i r}{mr^2\varepsilon^2}.$$

Setting $r = \frac{C}{\varepsilon^2}$ for a constant $C > 0$ makes this probability at most $\frac{4f_i}{Cm}$. By the union bound, if we sample each element in the stream independently with probability $\frac{r}{m}$, then the probability there exists an i for which $|\hat{f}_i - \mathbf{E}[\hat{f}_i]| \geqslant \frac{r\varepsilon}{2}$ is at most $\sum_{i=1}^n \frac{4f_i}{Cm} \leqslant \frac{4}{C}$, which for $C \geqslant 400$ is at most $\frac{1}{100}$, as desired. $\qquad\square$

For now, assume that the length of the stream is known in advance; we show in section 6.3.5 how to remove this assumption.

6.3.1 List Heavy Hitters Problem

For the LIST HEAVY HITTERS problem, we present two algorithms. The first is slightly suboptimal, but simple conceptually and already constitutes a very large improvement in the space complexity over known algorithms. We expect that this algorithm could be useful in practice as well. The second algorithm is more complicated, building on ideas from the first algorithm, and achieves the optimal space complexity upto constant factors.

We note that both algorithms proceed by sampling $O(\varepsilon^{-2}\ln(1/\delta))$ stream items and updating a data structure as the stream progresses. In both cases, the time to update the data structure is bounded by $O(1/\varepsilon)$, and so, under the standard assumption that the length of the stream is at least $\text{poly}(\ln(1/\delta)\varepsilon)$, the time to perform this update can be spread out across the next $O(1/\varepsilon)$ stream updates, since with large probability there will be no items sampled among these next $O(1/\varepsilon)$ stream updates. Therefore, we achieve worst-case[1] update time of $O(1)$.

6.3.1.1 A Simpler, Near-Optimal Algorithm

Theorem 6.1. *Assume the stream length is known beforehand. Then there is a randomized one-pass algorithm $\mathcal{A}$ for the (ε, φ)-LIST HEAVY HITTERS problem which succeeds with probability at least $1 - \delta$ using $O\left(\varepsilon^{-1}(\log \varepsilon^{-1} + \log\log \delta^{-1}) + \varphi^{-1}\log n + \log\log m\right)$ bits of space. Moreover, $\mathcal{A}$ has an update time of $O(1)$ and reporting time linear in its output size.*

Overview The overall idea is as follows. We sample $\ell = O(\varepsilon^{-2})$ many items from the stream uniformly at random as well as hash the id's (the word "id" is short for identifier) of the sampled elements into a space of size $O(\varepsilon^{-4})$. Now, both the stream length as well as the universe size are $\text{poly}(\varepsilon^{-1})$. From Lemma 6.3, it suffices to solve the heavy hitters problem on the sampled stream. From Lemma 6.2, because the hash function is chosen from a universal family, the sampled elements have distinct hashed id's. We can then feed these elements into a standard Misra-Gries data structure with ε^{-1} counters, incurring space $O(\varepsilon^{-1}\log \varepsilon^{-1})$.

---

[1] We emphasize that this is stronger than an amortized guarantee, as on every insertion, the cost will be $O(1)$.

Because we want to return the unhashed element id's for the heavy hitters, we also use $\log n$ space for recording the ϕ^{-1} top items according to the Misra-Gries data structure and output these when asked to report.

Algorithm 5 for (ε, φ)-LIST HEAVY HITTERS

Input: A stream S of length m over $\mathcal{U} = [n]$; let $f(x)$ be the frequency of $x \in \mathcal{U}$ in S
Output: A set $X \subseteq \mathcal{U}$ and a function $\hat{f} : X \to \mathbb{N}$ such that if $f(x) \geqslant \varphi m$, then $x \in X$ and
$f(x) - \varepsilon m \leqslant \hat{f}(x) \leqslant f(x) + \varepsilon m$ and if $f(y) \leqslant (\phi - \varepsilon)m$, then $y \notin X$ for every $x, y \in \mathcal{U}$

1: **Initialize:**
  2: $\ell \leftarrow {}^{6\log(6/\delta)}\!/_{\varepsilon^2}$
  3: Hash function h uniformly at random from a universal family
    $\mathcal{H} \subseteq \{h : [n] \to \lceil 4\ell^2/\delta \rceil\}$.
  4: An empty table $\mathcal{T}_1$ of (key, value) pairs of length ε^{-1}. Each key entry of
    $\mathcal{T}_1$ can store an integer in $[0, \lceil 400\ell^2/\delta \rceil]$ and each value entry can store
    an integer in $[0, 11\ell]$. $\triangleright$ The table $\mathcal{T}_1$ will be in sorted order by value
    throughout.
  5: An empty table $\mathcal{T}_2$ of length $^1\!/_\varphi$. Each entry of $\mathcal{T}_2$ can store an integer
    in $[0, n]$. $\triangleright$ The entries of $\mathcal{T}_2$ will correspond to ids of the keys in $\mathcal{T}_1$ of
    the highest $^1\!/_\varphi$ values
6:
7: **procedure** INSERT(x)
8:     With probability $p = {}^{6\ell}\!/_m$, continue. Otherwise,
9:     **return** .
10:     Perform Misra-Gries update using $h(x)$ maintaining $\mathcal{T}_1$ sorted by values.
11:     **if** The value of $h(x)$ is among the highest $^1\!/_\varphi$ valued items in $\mathcal{T}_1$ **then**
12:         **if** x_i is not in $\mathcal{T}_2$ **then**
13:             **if** $\mathcal{T}_2$ currently contains $^1\!/_\varphi$ many items **then**
14:                 For y in $\mathcal{T}_2$ such that $h(y)$ is not among the highest $^1\!/_\varphi$ valued items in $\mathcal{T}_1$,
replace y with x.
15:             **else**
16:                 We put x in $\mathcal{T}_2$.
17:             **end if**
18:         **end if**
19:         Ensure that elements in $\mathcal{T}_2$ are ordered according to corresponding values in $\mathcal{T}_1$.
20:     **end if**
21: **end procedure**
22:
23: **procedure** REPORT( )
24:
25:     **return** items in $\mathcal{T}_2$ along with their corresponding values in $\mathcal{T}_1$
26: **end procedure**

Proof of Theorem 6.1. The pseudocode of our (ε, φ)-LIST HEAVY HITTERS algorithm is in Algorithm 5. By Lemma 6.3, if we select a subset $\mathcal{S}$ of size at least $\ell = 6\varepsilon^{-2}\log(6\delta^{-1})$ uniformly at random from the stream, then $\Pr[\forall i \in \mathcal{U}, |(\hat{f}_i/|\mathcal{S}|) - (f_i/n)| \leqslant \varepsilon] \geqslant 1 - \delta/3$, where f_i and $\hat{f}_i$ are the frequencies of item i in the input stream and $\mathcal{S}$ respectively. First we show that with the choice of p in line 11 in Algorithm 5, the number of items sampled is at least ℓ and at most 11ℓ with probability at least $(1 - \delta/3)$. Let X_i be the indicator random variable of the event that the item x_i is sampled for $i \in [m]$. Then the total number of items sampled $X = \sum_{i=1}^{m} X_i$. We have $\mathbb{E}[X] = 6\ell$ since $p = 6\ell/m$. Now we have the following.

$$\Pr[X \leqslant \ell \text{ or } X \geqslant 11\ell] \leqslant \Pr[|X - \mathbb{E}[X]| \geqslant 5\ell] \leqslant \delta/3$$

The inequality follows from the Chernoff bound and the value of ℓ. From here onwards we assume that the number of items sampled is in $[\ell, 11\ell]$.

We use (a modified version of) the Misra-Gries algorithm [MG82] to estimate the frequencies of items in $\mathcal{S}$. The length of the table in the Misra-Gries algorithm is ε^{-1}. We pick a hash function h uniformly at random from a universal family $\mathcal{H} = \{h | h : [n] \to \lceil 4\ell^2/\delta \rceil\}$ of hash functions of size $|\mathcal{H}| = O(n^2)$. Note that picking a hash function h uniformly at random from $\mathcal{H}$ can be done using $O(\log n)$ bits of space. Lemma 6.2 shows that there are no collisions in $\mathcal{S}$ under this hash function h with probability at least $1 - \delta/3$. From here onwards we assume that there is no collision among the ids of the sampled items under the hash function h.

We modify the Misra-Gries algorithm as follows. Instead of storing the id of any item x in the Misra-Gries table (table $\mathcal{T}_1$ in line 5 in Algorithm 5) we only store the hash $h(x)$ of the id x. We also store the ids (not the hash of the id) of the items with highest $1/\varphi$ values in $\mathcal{T}_1$ in another table $\mathcal{T}_2$. Moreover, we always maintain the table $\mathcal{T}_2$ consistent with the table $\mathcal{T}_1$ in the sense that the ith highest valued key in $\mathcal{T}_1$ is the hash of the ith id in $\mathcal{T}_2$.

Upon picking an item x with probability p, we create an entry corresponding to $h(x)$ in $\mathcal{T}_1$ and make its value one if there is space available in $\mathcal{T}_1$; decrement the value of every item in $\mathcal{T}_1$ by one if the table is already full; increment the entry in the table corresponding to $h(x)$ if $h(x)$ is already present in the table. When we decrement the value of every item in $\mathcal{T}_1$, the table $\mathcal{T}_2$ remains consistent and we do not need to do anything else. Otherwise there are three cases to consider. Case 1: $h(x)$ is not among the $1/\varphi$ highest valued items in $\mathcal{T}_1$. In this case, we do not need to do anything else. Case 2: $h(x)$ was not among the $1/\varphi$ highest valued items in $\mathcal{T}_1$ but now it is among the $1/\varphi$ highest valued items in $\mathcal{T}_1$. In this case the last item y in $\mathcal{T}_2$ is no longer among the $1/\varphi$ highest valued items in $\mathcal{T}_1$. We replace y with x in $\mathcal{T}_2$. Case

3: $h(x)$ was among the $1/\varphi$ highest valued items in $\mathcal{T}_1$. When the stream finishes, we output the ids of all the items in table $\mathcal{T}_2$ along with the values corresponding to them in table $\mathcal{T}_1$. Correctness follows from the correctness of the Misra-Gries algorithm and the fact that there is no collision among the ids of the sampled items. $\qquad\square$

6.3.1.2 An Optimal Algorithm

Theorem 6.2. *Assume the stream length is known beforehand. Then there is a randomized one-pass algorithm $\mathcal{A}$ for the (ε, φ)-LIST HEAVY HITTERS problem which succeeds with constant probability using $O\left(\varepsilon^{-1} \log \phi^{-1} + \phi^{-1} \log n + \log \log m\right)$ bits of space. Moreover, $\mathcal{A}$ has an update time of $O(1)$ and reporting time linear in its output size.*

Note that in this section, for the sake of simplicity, we ignore floors and ceilings and state the results for a constant error probability, omitting the explicit dependence on δ.

Overview As in the simpler algorithm, we sample $\ell = O(\varepsilon^{-2})$ many stream elements and solve the $(\varepsilon/2, \phi)$-LIST HEAVY HITTERS problem on this sampled stream. Also, the Misra-Gries algorithm for $(\phi/2, \phi)$-LIST HEAVY HITTERS returns a candidate set of $O(\phi^{-1})$ items containing all items of frequency at least $\phi\ell$. It remains to count the frequencies of these $O(\phi^{-1})$ items with upto $\varepsilon\ell/2 = O(\varepsilon^{-1})$ *additive* error, so that we can remove those whose frequency is less than $(\phi - \varepsilon/2)\ell$.

Fix some item $i \in [n]$, and let f_i be i's count in the sampled stream. A natural approach to count f_i approximately is to increment a counter probabilistically, instead of deterministically, at every occurrence of i. Suppose that we increment a counter with probability $0 \leqslant p_i \leqslant 1$ whenever item i arrives in the stream. Let the value of the counter be $\hat{c}_i$, and let $\hat{f}_i = \hat{c}_i/p_i$. We see that $\mathbb{E}\left[\hat{f}_i\right] = f_i$ and $\mathsf{Var}[\hat{f}_i] \leqslant f_i/p_i$. It follows that if $p_i = \Theta(\varepsilon^2 f_i)$, then $\mathsf{Var}[\hat{f}_i] = O(\varepsilon^{-2})$, and hence, $\hat{f}_i$ is an unbiased estimator of f_i with additive error $O(\varepsilon^{-1})$ with constant probability. We call such a counter an *accelerated counter* as the probability of incrementing accelerates with increasing counts. For each i, we can maintain $O(\log \phi^{-1})$ accelerated counters independently and take their median to drive the probability of deviating by more than $O(\varepsilon^{-1})$ down to $O(\phi)$. So, with constant probability, the frequency for each of the $O(\phi^{-1})$ items in the Misra-Gries data structure is estimated within $O(\varepsilon^{-1})$ error, as desired.

However, there are two immediate issues with this approach. The first problem is that we may need to keep counts for $\Omega(\ell) = \Omega(\varepsilon^{-2})$ distinct items, which is too costly for our

Algorithm 6 for (ε, φ)-LIST HEAVY HITTERS

Input: A stream S of length m over universe $\mathcal{U} = [n]$; let $f(x)$ be the frequency of $x \in \mathcal{U}$ in S

Output: A set $X \subseteq \mathcal{U}$ and a function $\hat{f} : X \to \mathbb{N}$ such that if $f(x) \geqslant \varphi m$, then $x \in X$ and
$f(x) - \varepsilon m \leqslant \hat{f}(x) \leqslant f(x) + \varepsilon m$ and if $f(y) \leqslant (\varphi - \varepsilon)m$, then $y \notin X$ for every $x, y \in \mathcal{U}$

1: **Initialize:**
2:      $\ell \leftarrow 10^5 \varepsilon^{-2}$
3:      $s \leftarrow 0$
4:      Hash functions $h_1, \ldots, h_{200 \log(12\varphi^{-1})}$ uniformly at random from a
        universal family $\mathcal{H} \subseteq \{h : [n] \to [^{100}/_\varepsilon]\}$.
5:      An empty table $\mathcal{T}_1$ of (key, value) pairs of length $2\varphi^{-1}$. Each key entry
        of $\mathcal{T}_1$ can store an element of $[n]$ and each value entry can store an
        integer in $[0, 10\ell]$.
6:      An empty table $\mathcal{T}_2$ with $100\varepsilon^{-1}$ rows and $200 \log(12\varphi^{-1})$ columns.
        Each entry of $\mathcal{T}_2$ can store an integer in $[0, 100\varepsilon\ell]$.
7:      An empty 3-dimensional table $\mathcal{T}_3$ of size at most
        $100\varepsilon^{-1} \times 200 \log(12\varphi^{-1}) \times 4 \log(\varepsilon^{-1})$. Each entry of $\mathcal{T}_3$ can store an
        integer in $[0, 10\ell]$.   ▷ These are upper bounds; not all the allowed cells
        will actually be used.
8:
9: **procedure** INSERT(x)
10:      With probability $^\ell/_m$, increment s and continue. Else,
11:      **return**
12:      Perform Misra-Gries update on $\mathcal{T}_1$ with x.
13:      **for** $j \leftarrow 1$ to $200 \log(12\varphi^{-1})$ **do**
14:         $i \leftarrow h_j(x)$
15:         With probability ε, increment $\mathcal{T}_2[i, j]$
16:         $t \leftarrow \lfloor \log(10^{-6}\mathcal{T}_2[i, j]^2) \rfloor$ and $p \leftarrow \min(\varepsilon \cdot 2^t, 1)$
17:         **if** $t \geqslant 0$ **then**
18:            With probability p, increment $\mathcal{T}_3[i, j, t]$
19:         **end if**
20:      **end for**
21: **end procedure**
22:
23: **procedure** REPORT( )
24:      $X \leftarrow \emptyset$
25:      **for** each key x with nonzero value in $\mathcal{T}_1$ **do**
26:         **for** $j \leftarrow 1$ to $200 \log(12\varphi^{-1})$ **do**
27:            $\hat{f}_j(x) \leftarrow \sum_{t=0}^{4\log(\varepsilon^{-1})} \mathcal{T}_3[h(x), j, t] / \min(\varepsilon 2^t, 1)$
28:         **end for**
29:         $\hat{f}(x) \leftarrow \text{median}(\hat{f}_1, \ldots, \hat{f}_{10 \log \varphi^{-1}})$
30:         **if** $\hat{f}(x) \geqslant (\varphi - \varepsilon/2)s$ **then**
31:            $X \leftarrow X \cup \{x\}$
32:         **end if**
33:      **end for**
34:
35:      **return** $X, \hat{f}$
36: **end procedure**

123

purposes. To get around this, we use a hash function from a universal family to hash the universe to a space of size $u = \Theta(\varepsilon^{-1})$, and we work throughout with the hashed id's. We can then show that the space complexity for each iteration is $O(\varepsilon^{-1})$. Also, the accelerated counters now estimate frequencies of hashed id's instead of actual items, but because of universality, the expected frequency of any hashed id is $\ell/u = O(\varepsilon^{-1})$, our desired error bound.

The second issue is that we need a constant factor approximation of f_i, so that we can set p_i to $\Theta(\varepsilon^2 f_i)$. But because the algorithm needs to be one-pass, we cannot first compute p_i in one pass and then run the accelerated counter in another. So, we divide the stream into *epochs* in which f_i stays within a factor of 2, and use a different p_i for each epoch. In particular, set $p_i^t = \varepsilon \cdot 2^t$ for $0 \leqslant t \leqslant \log(p_i/\varepsilon)$. We want to keep a running estimate of i's count to within a factor of 2 to know if the current epoch should be incremented. For this, we subsample each element of the stream with probability ε independently and maintain exact counts for the observed hashed id's. It is easy to see that this requires only $O(\varepsilon^{-1})$ bits in expectation. Consider any $i \in [u]$ and the prefix of the stream upto $b \leqslant \ell$, and let $f_i(b)$ be i's frequency in the prefix, let $\bar{c}_i(b)$ be i's frequency among the samples in the prefix, and $\bar{f}_i(b) = \frac{\bar{c}_i(b)}{\varepsilon}$. We see that $\mathbb{E}\left[\bar{f}_i(b)\right] = f_i(b)$, and $\mathrm{Var}[\bar{f}_i(b)] \leqslant \frac{f_i(b)}{\varepsilon}$. By Chebyshev, for any fixed b, $\Pr[|\bar{f}_i(b) - f_i(b)| > f_i(b)/\sqrt{2}] \leqslant \frac{2}{f_i(b)\varepsilon}$, and hence, can show that $\bar{f}_i(b)$ is a $\sqrt{2}$-factor approximation of $f_i(b)$ with probability $1 - O((f_i(b)\varepsilon)^{-1})$. Now, let $p_i(b) = \Theta(\varepsilon^2 f_i(b))$, and for any epoch t, set $b_{i,t} = \min\{b : p_i(b) > p_i^{t-1}\}$. The last makes sense because $p_i(b)$ is non-decreasing with b. Also, note that $f_i(b_{i,t}) = \Omega(2^{t/2}/\varepsilon)$. So, by the union bound, the probability that there exists t for which $\bar{f}_i(b_{i,t})$ is not a $\sqrt{2}$-factor approximation of $f_i(b_{i,t})$ is at most $\sum_t \frac{1}{\Omega(f_i(b_{i,t})\varepsilon)} = \sum_t \frac{1}{\Omega(2^{t/2})}$, a small constant. In fact, it follows then that with constant probability, for all $b \in [\ell]$, $\bar{f}_i(b)$ is a 2-factor approximation of $f_i(b)$. Moreover, we show that for any $b \in [\ell]$, $\bar{f}_i(b)$ is a 4-factor approximation of $f_i(b)$ with constant probability. By repeating $O(\log \phi^{-1})$ times independently and taking the median, the error probability can be driven down to $O(\phi)$.

Now, for every hashed id $i \in [u]$, we need not one accelerated counter but $O(\log(\varepsilon f_i))$ many, one corresponding to each epoch t. When an element with hash id i arrives at position b, we decide, based on $\bar{f}_i(b)$, the epoch t it belongs to and then increment the t'th accelerated counter with probability p_i^t. The storage cost over all i is still $O(1/\varepsilon)$. Also, we iterate the whole set of accelerated counters $O(\log \phi^{-1})$ times, making the total storage cost $O(\varepsilon^{-1} \log \phi^{-1})$.

124

Let $\hat{c}_{i,t}$ be the count in the accelerated counter for hash id i and epoch t. Then, let $\hat{f}_i = \sum_t \hat{c}_{i,t}/p_i^t$. Clearly, $\mathbb{E}\left[\hat{f}_i\right] = f_i$. The variance is $O(\varepsilon^{-2})$ in each epoch, and so, $\mathrm{Var}[\hat{f}_i] = O(\varepsilon^{-2}\log\varepsilon^{-1})$, not $O(\varepsilon^{-2})$ which we wanted. This issue is fixed by a change in how the sampling probabilities are defined. We now go on to the formal proof.

Proof of Theorem 6.2. Pseudocode appears in algorithm 6. Note that the numerical constants are chosen for convenience of analysis and have not been optimized. Also, for the sake of simplicity, the pseudocode does not have the optimal reporting time, but it can be modified to achieve this; see the end of this proof for details.

By standard Chernoff bounds, with probability at least $99/100$, the length of the sampled stream $\ell/10 \leqslant s \leqslant 10\ell$. For $x \in [n]$, let $f_{\mathrm{samp}}(x)$ be the frequency of x in the sampled stream. By Lemma 6.3, with probability at least $9/10$, for all $x \in [n]$:

$$\left| \frac{f_{\mathrm{samp}}(x)}{s} - \frac{f(x)}{m} \right| \leqslant \frac{\varepsilon}{4}$$

Now, fix $j \in [10\log\phi^{-1}]$ and $x \in [n]$. Let $i = h_j(x)$ and $f_i = \sum_{y:h_j(y)=h_j(x)} f_{\mathrm{samp}}(y)$. Then, for a random $h_j \in \mathcal{H}$, the expected value of $\frac{f_i}{s} - \frac{f_{\mathrm{samp}}(x)}{s}$ is $\frac{\varepsilon}{100}$, since $\mathcal{H}$ is a universal mapping to a space of size $100\varepsilon^{-1}$. Hence, using Markov's inequality and the above:

$$\Pr\left[\left| \frac{f(x)}{m} - \frac{f_i}{s} \right| \geqslant \frac{\varepsilon}{2} \right] \leqslant \Pr\left[\left| \frac{f(x)}{m} - \frac{f_{\mathrm{samp}}}{s} \right| \geqslant \frac{\varepsilon}{4} \right] + \Pr\left[\left| \frac{f_{\mathrm{samp}}(x)}{m} - \frac{f_i}{s} \right| \geqslant \frac{\varepsilon}{4} \right] < \frac{1}{10} + \frac{1}{25} < \frac{3}{20}$$
(6.1)

In Lemma 6.4 below, we show that for each $j \in [200\log(12\phi^{-1})]$, with error probability at most $3/10$, $\hat{f}_j(x)$ (in line 27) estimates f_i with additive error at most $5000\varepsilon^{-1}$, hence estimating $\frac{f_i}{s}$ with additive error at most $\frac{\varepsilon}{2}$. Taking the median over $200\log(12\phi^{-1})$ repetitions (line 29) makes the error probability go down to $\frac{\phi}{6}$ using standard Chernoff bounds. Hence, by the union bound, with probability at least $2/3$, for each of the $2/\phi$ keys x with nonzero values in $\mathcal{T}_1$, we have an estimate of $\frac{f(x)}{m}$ within additive error ε, thus showing correctness.

Lemma 6.4. *Fix $x \in [n]$ and $j \in [200\log 12\phi^{-1}]$, and let $i = h_j(x)$. Then, $\Pr[|\hat{f}_j(x) - f_i| > 5000\varepsilon^{-1}] \leqslant 3/10$, where $\hat{f}_j$ is the quantity computed in line 27.*

Proof. Index the sampled stream elements $1, 2, \ldots, s$, and for $b \in [s]$, let $f_i(b)$ be the frequency of items with hash id i restricted to the first b elements of the sampled stream. Let $\bar{f}_i(b)$ denote the value of $\mathcal{T}_2[i, j] \cdot \varepsilon^{-1}$ after the procedure INSERT has been called for the first b items of the sampled stream.

Claim 6.1. *With probability at least 9/10, for all $b \in [s]$ such that $f_i(b) \geqslant 100\varepsilon^{-1}$, $\bar{f}_i(b)$ is within a factor of 4 of $f_i(b)$.*

Proof. Fix $b \in [s]$. Note that $\mathbb{E}\left[\bar{f}_i(b)\right] = f_i(b)$ as $\mathcal{T}_2$ is incremented with rate ε. $\mathrm{Var}[\bar{f}_i(b)] \leqslant f_i/\varepsilon$, and so by Chebyshev's inequality:

$$\Pr[|\bar{f}_i(b) - f_i(b)| > f_i(b)/2] < \frac{4}{f_i(b)\varepsilon}$$

We now break the stream into chunks, apply this inequality to each chunk and then take a union bound to conclude. Namely, for any integer $t \geqslant 0$, define b_t to be the first b such that $100\varepsilon^{-1}2^t \leqslant f_i(b) < 100\varepsilon^{-1}2^{t+1}$ if such a b exists. Then:

$$\Pr[\exists t \geqslant 0 : |\bar{f}_i(b_t) - f_i(b_t)| > f_i(b_t)/2] < \sum_t \frac{4}{100 \cdot 2^{t-1}}$$
$$< \frac{1}{10}$$

So, with probability at least 9/10, every $\bar{f}_i(b_t)$ and $f_i(b_t)$ are within a factor of 2 of each other. Since for every $b \geqslant b_0$, $f_i(b)$ is within a factor of 2 from some $f_i(b_t)$, the claim follows. $\qquad\square$

Assume the event in Claim 6.1 henceforth. Now, we are ready to analyze $\mathcal{T}_3$ and in particular, $\hat{f}_j(x)$. First of all, observe that if $t < 0$ in line 16, at some position b in the stream, then $\mathcal{T}_2[i, j]$ at that time must be at most 1000, and so by standard Markov and Chernoff bounds, with probability at least 0.85,

$$f_i(b) \begin{cases} < 4000\varepsilon^{-1}, & \text{if } t < 0 \\ > 100\varepsilon^{-1}, & \text{if } t \geqslant 0 \end{cases} \tag{6.2}$$

Assume this event. Then, $f_i - 4000\varepsilon^{-1} \leqslant \mathbb{E}\left[\hat{f}_j(x)\right] \leqslant f_i$.

Claim 6.2.

$$\mathrm{Var}(\hat{f}_j(x)) \leqslant 20000\varepsilon^{-2}$$

Proof. If the stream element at position b causes an increment in $\mathcal{T}_3$ with probability $\varepsilon 2^t$ (in line 18), then $1000 \cdot 2^{t/2} \leqslant \mathcal{T}_2[i, j] \leqslant 1000 \cdot 2^{(t+1)/2}$, and so, $\bar{f}_i(b) \leqslant 1000\varepsilon^{-1}2^{(t+1)/2}$. This must be the case for the highest $b = \bar{b}_t$ at which the count for i in $\mathcal{T}_3$ increments at the t'th

slot. The number of such occurrences of i is at most $f_i(\bar{b}_t) \leqslant 4\bar{f}_i(\bar{b}_t) \leqslant 4000\varepsilon^{-1}2^{(t+1)/2}$ by Claim 6.1 (which can be applied since $f_i(b) > 100\varepsilon^{-1}$ by Equation 6.2). So:

$$\mathsf{Var}[\hat{f}_j(x)] \leqslant \sum_{t \geqslant 0} \frac{f_i(\bar{b}_t)}{\varepsilon 2^t} \leqslant \sum_{t \geqslant 0} \frac{4000}{\varepsilon^2}2^{-t/3} \leqslant 20000\varepsilon^{-2}$$

Elements inserted with probability 1 obviously do not contribute to the variance. □

So, conditioning on the events mentioned, the probability that $\hat{f}_j(x)$ deviates from f_i by more than $5000\varepsilon^{-1}$ is at most $1/50$. Removing all the conditioning yields what we wanted:

$$\Pr[|\hat{f}_j(x) - f_i| > 5000\varepsilon^{-1}] \leqslant \frac{1}{50} + \frac{3}{20} + \frac{1}{10} \leqslant 0.3$$

□

We next bound the space complexity.

Claim 6.3. *With probability at least $2/3$, algorithm 6 uses $O(\varepsilon^{-1}\log\phi^{-1} + \phi^{-1}\log n + \log\log m)$ bits of storage, if $n = \omega(\varepsilon^{-1})$.*

Proof. The expected length of the sampled stream is $\ell = O(\varepsilon^{-2})$. So, the number of bits stored in $\mathcal{T}_1$ is $O(\phi^{-1}\log n)$. For $\mathcal{T}_2$, note that in lines 13-15, for any given j, $\mathcal{T}_2$ is storing a total of $\varepsilon\ell = O(\varepsilon^{-1})$ elements in expectation. So, for $k \geqslant 0$, there can be at most $O((\varepsilon 2^k)^{-1})$ hashed id's with counts between 2^k and 2^{k+1}. Summing over all k's and accounting for the empty cells gives $O(\varepsilon^{-1})$ bits of storage, and so the total space requirement of $\mathcal{T}_2$ is $O(\varepsilon^{-1}\log\phi^{-1})$. .

The probability that a hashed id i gets counted in table $\mathcal{T}_3$ is at most $10^{-6}\varepsilon^3\bar{f}_i^2(s)$ from line 16 and our definition of $\bar{f}_i$ above. Moreover, from Claim 6.1, we have that this is at most $16 \cdot 10^{-6}\varepsilon^3 f_i^2(s)$ if $f_i > 100\varepsilon^{-1}$. Therefore, if $f_i = 2^k \cdot 100\varepsilon^{-1}$ with $k \geqslant 0$, then the expected value of a cell in $\mathcal{T}_3$ with first coordinate i is at most $1600 \cdot 2^{2k}\varepsilon = 2^{O(k)}$. Taking into account that there are at most $O((\varepsilon 2^k)^{-1})$ many such id's i and that the number of epochs t associated with such an i is at most $\log(16 \cdot 10^{-6}\varepsilon^2 f_i^2) = O(\log(\varepsilon f_i)) = O(k)$ (from line 16), we get that

the total space required for $\mathcal{T}_3$ is:

$$\sum_{j=1}^{O(\log \phi^{-1})} \left(O(\varepsilon^{-1}) + \sum_{k=0}^{\infty} O((\varepsilon 2^k)^{-1}) \cdot O(k) \cdot O(k) \right)$$
$$= O(\varepsilon^{-1} \log \phi^{-1})$$

where the first $O(\varepsilon^{-1})$ term inside the summation is for the i's with $f_i < 100\varepsilon^{-1}$. Since we have an expected space bound, we obtain a worst-case space bound with error probability $1/3$ by a Markov bound.

The space required for sampling is an additional $O(\log \log m)$, using Lemma 6.1. $\qquad \square$

We note that the space bound can be made worst case by aborting the algorithm if it tries to use more space.

The only remaining aspect of Theorem 6.2 is the time complexity. As observed in section 6.3.1, the update time can be made $O(1)$ per insertion under the standard assumption of the stream being sufficiently long. The reporting time can also be made linear in the output by changing the bookkeeping a bit. Instead of computing $\hat{f}_j$ and $\hat{f}$ at reporting time, we can maintain them after every insertion. Although this apparently makes INSERT costlier, this is not true in fact because we can spread the cost over future stream insertions. The space complexity grows by a constant factor. $\qquad \square$

6.3.2 ε-Maximum Problem

By tweaking Algorithm 5 slightly, we get the following result for the ε-MAXIMUM problem.

Theorem 6.3. *Assume the length of the stream is known beforehand. Then there is a randomized one-pass algorithm $\mathcal{A}$ for the ε-MAXIMUM problem which succeeds with probability at least $1 - \delta$ using $O\left(\min\{^1/_\varepsilon, n\}(\log ^1/_\varepsilon + \log\log ^1/_\delta) + \log n + \log\log m\right)$ bits of space. Moreover, the algorithm $\mathcal{A}$ has an update time of $O(1)$.*

Proof. Instead of maintaining the table $\mathcal{T}_2$ in Algorithm 5, we just store the actual id of the item with maximum frequency in the sampled items. $\qquad \square$

6.3.3 ε-Minimum Problem

Theorem 6.4. *Assume the length of the stream is known beforehand. Then there is a randomized one-pass algorithm $\mathcal{A}$ for the ε-MINIMUM problem which succeeds with probability at least $1 - \delta$ using $O\left((1/\varepsilon) \log \log(1/\varepsilon\delta) + \log \log m\right)$ bits of space. Moreover, the algorithm $\mathcal{A}$ has an update time of $O(1)$.*

Overview   Pseudocode is provided in algorithm 7. The idea behind our ε-Minimum problem is as follows. It is most easily explained by looking at the REPORT(x) procedure starting in line 13. In lines 14-15 we ask, is the universe size $|U|$ significantly larger than $1/\varepsilon$? Note that if it is, then outputting a random item from $|U|$ is likely to be a solution. Otherwise $|U|$ is $O(1/\varepsilon)$.

The next point is that if the number of distinct elements in the stream were smaller than $1/(\varepsilon \log(1/\varepsilon))$, then we could just store all the items together with their frequencies with $O(1/\varepsilon)$ bits of space. Indeed, we can first sample $O(1/\varepsilon^2)$ stream elements so that all relative frequencies are preserved up to additive ε, thereby ensuring each frequency can be stored with $O(\log(1/\varepsilon)$ bits. Also, since the universe size is $O(1/\varepsilon)$, the item identifiers can also be stored with $O(\log(1/\varepsilon)$ bits. So if this part of the algorithm starts taking up too much space, we stop, and we know the number of distinct elements is at least $1/(\varepsilon \log(1/\varepsilon))$, which means that the minimum frequency is at most $O(m\varepsilon \log(1/\varepsilon))$. This is what is being implemented in steps 9-10 and 18-19 in the algorithm.

We can also ensure the minimum frequency is at least $\Omega(m\varepsilon/\log(1/\varepsilon))$. Indeed, by randomly sampling $O((\log(1/\varepsilon)/\varepsilon)$ stream elements, and maintaining a bit vector for whether or not each item in the universe occurs - which we can with $O(1/\varepsilon)$ bits of space since $|U| = O(1/\varepsilon)$ - any item with frequency at least $\Omega(\varepsilon m/\log(1/\varepsilon))$ will be sampled and so if there is an entry in the bit vector which is empty, then we can just output that as our solution. This is what is being implemented in steps 8 and 16-17 of the algorithm.

Finally, we now know that the minimum frequency is at least $\Omega(m\varepsilon/\log(1/\varepsilon))$ and at most $O(m\varepsilon \log(1/\varepsilon))$. At this point if we randomly sample $O((\log^6 1/\varepsilon)/\varepsilon)$ stream elements, then by Chernoff bounds all item frequencies are preserved up to a relative error factor of $(1 \pm 1/\log^2(1/\varepsilon))$, and in particular the relative minimum frequency is guaranteed to be preserved up to an additive ε. At this point we just maintain the exact counts in the sampled stream but truncate them once they exceed $\mathrm{poly}(\log(1/\varepsilon)))$ bits, since we know such counts

do not correspond to the minimum. Thus we only need $O(\log\log(1/\varepsilon))$ bits to represent their counts. This is implemented in step 11 and step 20 of the algorithm.

Algorithm 7 for ε-MINIMUM

Input: A stream $\mathcal{S} = (x_i)_{i\in[m]} \in \mathcal{U}^m$ of length m over $\mathcal{U}$; let $f(x)$ be the frequency of $x \in \mathcal{U}$ in $\mathcal{S}$

Output: An item $x \in \mathcal{U}$ such that $f(x) \leqslant f(y) + \varepsilon m$ for every $y \in \mathcal{U}$

1: **Initialize:**

    2: $\ell_1 \leftarrow {\log(6/\varepsilon\delta)}/{\varepsilon}$, $\ell_2 \leftarrow {\log(6/\delta)}/{\varepsilon^2}$, $\ell_3 \leftarrow {\log^6(6/\delta\varepsilon)}/{\varepsilon}$

    3: $p_1 \leftarrow {6\ell_1}/{m}$, $p_2 \leftarrow {6\ell_2}/{m}$, $p_3 \leftarrow {6\ell_3}/{m}$

    4: $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3 \leftarrow \emptyset$

    5: $\mathcal{B}_1 \leftarrow$ the bit vector for $\mathcal{S}_1$

6:

7: **procedure** INSERT(x)

8:    Put x in $\mathcal{S}_1$ with probability p_1 by updating the bit vector $\mathcal{B}_1$

9:    **if** the number of distinct items in the stream so far is at most ${1}/{(\varepsilon\log(1/\varepsilon))}$ **then**

10:        Pick x with probability p_2 and put the id of x in $\mathcal{S}_2$ and initialize the corresponding counter to 1 if $x \notin \mathcal{S}_2$ and increment the counter corresponding to x by 1.

11:    **end if**

12:    Pick x with probability p_3, put the id of x in $\mathcal{S}_3$ and initialize the corresponding counter to 1 if $x_i \notin \mathcal{S}_3$ and increment the counter corresponding to x_i by 1. Truncate counters of $\mathcal{S}_3$ at $2\log^7(2/\varepsilon\delta)$.

13: **end procedure**

14:

15: **procedure** REPORT( )

16:    **if** $|\mathcal{U}| \geqslant {1}/{((1-\delta)\varepsilon)}$ **then**

17:

18:        **return** an item x from the first ${1}/{((1-\delta)\varepsilon)}$ items in $\mathcal{U}$ (ordered arbitrarily) uniformly at random

19:    **end if**

20:    **if** $\mathcal{S}_1 \neq \mathcal{U}$ **then**

21:

22:        **return** any item from $\mathcal{U} \setminus \mathcal{S}_1$

23:    **end if**

24:    **if** the number of distinct items in the stream is at most ${1}/{(\varepsilon\log(1/\varepsilon))}$ **then**

25:

26:        **return** an item in $\mathcal{S}_2$ with minimum counter value in $\mathcal{S}_2$

27:    **end if**

28:

29:    **return** the item with minimum frequency in $\mathcal{S}_3$

30: **end procedure**

Proof of Theorem 6.4. The pseudocode of our ε-MINIMUM algorithm is in Algorithm 7. If the size of the universe $|\mathcal{U}|$ is at least $1/((1-\delta)\varepsilon)$, then we return an item x chosen from $\mathcal{U}$ uniformly at random. Note that there can be at most $1/\varepsilon$ many items with frequency at least εm. Hence every item x among other remaining $\delta/((1-\delta)\varepsilon)$ many items has frequency less than εm and thus is a correct output of the instance. Thus the probability that we answer correctly is at least $(1-\delta)$. From here on, let us assume $|\mathcal{U}| < 1/((1-\delta)\varepsilon)$.

Now, by the value of p_j, it follows from the proof of Theorem 6.1 that we can assume $\ell_j < |\mathcal{S}_j| < 11\ell_j$ for $j = 1, 2, 3$ which happens with probability at least $(1-(\delta/3))$. We first show that every item in $\mathcal{U}$ with frequency at least εm is sampled in $\mathcal{S}_1$ with probability at least $(1-(\delta/6))$. For that, let X_i^j be the indicator random variable for the event that the jth sample in $\mathcal{S}_1$ is item i where $i \in \mathcal{U}$ is an item with frequency at least εm. Let $\mathcal{H} \subset \mathcal{U}$ be the set of items with frequencies at least εm. Then we have the following.

$$\Pr[X_i^j = 0] = 1 - \varepsilon \Rightarrow \Pr[X_i^j = 0 \; \forall j \in \mathcal{S}_1] \leqslant (1-\varepsilon)^{\ell_1} \leqslant \exp\{-\varepsilon\ell_1\} = \varepsilon\delta/6$$

Now applying union bound we get the following.

$$\Pr[\exists i \in \mathcal{H}, X_i^j = 0 \; \forall j \in \mathcal{S}_1] \leqslant (1/\varepsilon)\varepsilon\delta/6 \leqslant \delta/6$$

Hence with probability at least $(1-(\delta/3)-(\delta/6)) \geqslant (1-\delta)$, the output at line 22 is correct. Now we show below that if the frequency of any item $x \in \mathcal{U}$ is at most $\varepsilon \ln(6/\delta)/\ln(6/\varepsilon\delta)$, then $x \in \mathcal{S}_1$ with probability at least $(1-(\delta/6))$.

$$\Pr[x \notin \mathcal{S}_1] = (1 - \varepsilon \ln(6/\delta)/\ln(6/\varepsilon\delta))^{\ln(6/\varepsilon\delta)/\varepsilon} \leqslant \delta/6$$

Hence from here onwards we assume that the frequency of every item in $\mathcal{U}$ is at least $\varepsilon m \ln(6/\delta)/\ln(6/\varepsilon\delta)$.

If the number of distinct elements is at most $1/(\varepsilon \ln(1/\varepsilon))$, then line 26 outputs the minimum frequency item up to an additive factor of εm due to Chernoff bound. Note that we need only $O(\ln(1/((1-\delta)\varepsilon)))$ bits of space for storing ids. Hence $\mathcal{S}_2$ can be stored in space $O((1/\varepsilon \ln(1/\varepsilon)) \ln(1/((1-\delta)\varepsilon) \ln\ln(1/\delta)) = O(1/\varepsilon \ln\ln(1/\delta))$.

Now we can assume that the number of distinct elements is at least $1/(\varepsilon \ln(1/\varepsilon))$. Hence if $f(t)$ is the frequency of the item t with minimum frequency, then we have $m\varepsilon/\ln(1/\varepsilon) \leqslant f(t) \leqslant m\varepsilon \ln(1/\varepsilon)$.

Let f_i be the frequency of item $i \in \mathcal{U}$, e_i be the counter value of i in $\mathcal{S}_3$, and $\hat{f}_i = {}^{e_i m}/\ell_3$. Now again by applying Chernoff bound we have the following for any fixed $i \in \mathcal{U}$.

$$
\begin{aligned}
\Pr[|f_i - \hat{f}_i| > {}^{f_i}/\ln^2(1/\varepsilon)] &\leqslant 2\exp\{-\ell_3 f_i/(m\ln^4(1/\varepsilon))\} \\
&\leqslant 2\exp\{-f_i \ln^2(6/\varepsilon\delta)/(\varepsilon m)\} \\
&\leqslant {}^{\varepsilon\delta}/6.
\end{aligned}
$$

Now applying union bound we get the following using the fact that $|\mathcal{U}| \leqslant {}^1/\varepsilon(1-\delta)$.

$$\Pr[\forall i \in \mathcal{U}, |f_i - \hat{f}_i| \leqslant {}^{f_i}/\ln^2(1/\varepsilon)] > 1 - {}^\delta/6$$

Again by applying Chernoff bound and union bound we get the following.

$$\Pr[\forall i \in \mathcal{U} \text{ with } f_i > 2m\varepsilon\ln(1/\varepsilon), |f_i - \hat{f}_i| \leqslant {}^{f_i}/2] > 1 - {}^\delta/6$$

Hence the items with frequency more than $2m\varepsilon\ln(1/\varepsilon)$ are approximated up to a multiplicative factor of $1/2$ from below in $\mathcal{S}_3$. The counters of these items may be truncated. The other items with frequency at most $2m\varepsilon\ln(1/\varepsilon)$ are be approximated up to $(1 \pm {}^1/\ln^2(1/\varepsilon))$ relative error and thus up to an additive error of ${}^{\varepsilon m}/3$. The counters of these items would not get truncated. Hence the item with minimum counter value in $\mathcal{S}_3$ is the item with minimum frequency up to an additive εm.

We need $O(\ln(1/\varepsilon\delta))$ bits of space for the bit vector $\mathcal{B}_1$ for the set $\mathcal{S}_1$. We need $O(\ln^2(1/\varepsilon\delta))$ bits of space for the set $\mathcal{S}_2$ and $O((1/\varepsilon)\ln\ln(1/\varepsilon\delta))$ bits of space for the set $\mathcal{S}_3$ (by the choice of truncation threshold). We need an additional $O(\ln\ln m)$ bits of space for sampling using Lemma 6.1. Moreover, using the data structure of Section 3.3 of [DLOM02] Algorithm 7 can be performed in $O(1)$ time. Alternatively, we may also use the strategy described in section 6.3.1 of spreading update operations over several insertions to make the cost per insertion be $O(1)$. $\qquad\square$

6.3.4 Problems for the Borda and Maximin Voting Rules

Theorem 6.5. *Assume the length of the stream is known beforehand. Then there is a randomized one-pass algorithm $\mathcal{A}$ for (ε, φ)-LIST BORDA problem which succeeds with probability at least $1 - \delta$ using $O\left(n\left(\log n + \log\frac{1}{\varepsilon} + \log\log\frac{1}{\delta}\right) + \log\log m\right)$ bits of space.*

Proof. Let $\ell = 6\varepsilon^{-2}\log(6n\delta^{-1})$ and $p = {}^{6\ell}/{}_m$. On each insertion of a vote v, select v with probability p and store for every $i \in [n]$, the number of candidates that candidate i beats in the vote v. Keep these exact counts in a counter of length n.

Then it follows from the proof of Theorem 6.1 that $\ell \leqslant |\mathcal{S}| \leqslant 11\ell$ with probability at least $(1 - \delta/3)$. Moreover, from a straightforward application of the Chernoff bound (see [DB15]), it follows that if $\hat{s}(i)$ denotes the Borda score of candidate i restricted to the sampled votes, then:

$$\Pr\left[\forall i \in [n], \left|\frac{m}{|\mathcal{S}|}\hat{s}(i) - s(i)\right| < \varepsilon m n\right] > 1 - \delta$$

The space complexity for exactly storing the counts is $O(n\log(n\ell)) = O(n(\log n + \log \varepsilon^{-1} + \log\log\delta^{-1}))$ and the space for sampling the votes is $O(\log\log m)$ by Lemma 6.1. $\qquad\square$

Theorem 6.6. *Assume the length of the stream is known beforehand. Then there is a randomized one-pass algorithm $\mathcal{A}$ for (ε, φ)-LIST MAXIMIN problem which succeeds with probability at least $1 - \delta$ using $O\left(n\varepsilon^{-2}\log^2 n + n\varepsilon^{-2}\log n\log\delta^{-1} + \log\log m\right)$ bits of space.*

Proof. Let $\ell = ({}^8/{}_{\varepsilon^2})\ln({}^{6n}/{}_\delta)$ and $p = {}^{6\ell}/{}_m$. We put the current vote in a set $\mathcal{S}$ with probability p. Then it follows from the proof of Theorem 6.1 that $\ell \leqslant |\mathcal{S}| \leqslant 11\ell$ with probability at least $(1 - {}^\delta/{}_3)$. Suppose $|\mathcal{S}| = \ell_1$; let $\mathcal{S} = \{v_i : i \in [\ell_1]\}$ be the set of votes sampled. Let $D_\mathcal{E}(x, y)$ be the total number of votes in which x beats y and $D_\mathcal{S}(x, y))$ be the number of such votes in $\mathcal{S}$. Then by the choice of ℓ and the Chernoff bound (see [DB15]), it follows that $|D_\mathcal{S}(x, y){}^m/{}_{\ell_1} - D_\mathcal{E}(x, y)| \leqslant {}^{\varepsilon m}/{}_2$ for every pair of candidates $x, y \in \mathcal{U}$. Note that each vote can be stored in $O(n\log n)$ bits of space. Hence simply finding $D_\mathcal{S}(x, y)$ for every $x, y \in \mathcal{U}$ by storing $\mathcal{S}$ and returning all the items with maximin score at least $(\varphi - \varepsilon/2)\ell_1$ in $\mathcal{S}$ requires $O\left(n\varepsilon^{-2}\log n(\log n + \log\delta^{-1}) + \log\log m\right)$ bits of memory, with the additive $O(\log\log m)$ due to Lemma 6.1. $\qquad\square$

6.3.5 Algorithms with Unknown Stream Length

Now we consider the case when the length of the stream is not known beforehand. We present below an algorithm for (ε, φ)-LIST HEAVY HITTERS and ε-MAXIMUM problems in the setting where the length of the stream is not known beforehand.

Theorem 6.7. *There is a randomized one-pass algorithm for (ε, φ)-LIST HEAVY HITTERS and ε-MAXIMUM problems with space complexity $O\left(\varepsilon^{-1}\log\varepsilon^{-1} + \varphi^{-1}\log n + \log\log m\right)$ bits and update time $O(1)$ even when the length of the stream is not known beforehand.*

Proof. We describe below a randomized one-pass algorithm for the $(8\varepsilon, \varphi)$-LIST HEAVY HIT-TERS problem. We may assume that the length of the stream is at least $1/\varepsilon^2$; otherwise, we use the algorithm in Theorem 6.1 and get the result. Now we guess the length of the stream to be $1/\varepsilon^2$, but run an instance $\mathcal{I}_1$ of Algorithm 5 with $\ell = \log(6/\delta)/\varepsilon^3$ at line 2. By the choice of the size of the sample (which is $\Theta(\log(1/\delta)/\varepsilon^3)$), $\mathcal{I}_1$ outputs correctly with probability at least $(1 - \delta)$, if the length of the stream is in $[1/\varepsilon^2, 1/\varepsilon^3]$. If the length of the stream exceeds $1/\varepsilon^2$, we run another instance $\mathcal{I}_2$ of Algorithm 5 with $\ell = \log(6/\delta)/\varepsilon^3$ at line 2. Again by the choice of the size of the sample, $\mathcal{I}_2$ outputs correctly with probability at least $(1 - \delta)$, if the length of the stream is in $[1/\varepsilon^3, 1/\varepsilon^4]$. If the stream length exceeds $1/\varepsilon^3$, we discard $\mathcal{I}_1$, free the space it uses, and run an instance $\mathcal{I}_3$ of Algorithm 5 with $\ell = \log(6/\delta)/\varepsilon^3$ at line 2 and so on. At any point of time, we have at most two instances of Algorithm 5 running. When the stream ends, we return the output of the older of the instances we are currently running. We use the approximate counting method of Morris [Mor78] to approximately count the length of the stream. We know that the Morris counter outputs correctly with probability $(1 - 2^{-k/2})$ using $O(\log\log m + k)$ bits of space at any point in time [Fla85]. Also, since the Morris counter increases only when an item is read, it outputs correctly up to a factor of four at every position if it outputs correctly at positions $1, 2, 4, \ldots, 2^{\lfloor \log_2 m \rfloor}$; call this event E. Then we have $\Pr(E) \geqslant 1 - \delta$ by choosing $k = 2\log_2(\log_2 m/\delta)$ and applying union bound over the positions $1, 2, 4, \ldots, 2^{\lfloor \log_2 m \rfloor}$. The correctness of the algorithm follows from the correctness of Algorithm 5 and the fact that we are discarding at most εm many items in the stream (by discarding a run of an instance of Algorithm 5). The space complexity and the $O(1)$ update time of the algorithm follow from Theorem 6.1, the choice of k above, and the fact that we have at most two instances of Algorithm 5 currently running at any point of time.

The algorithm for the ε-MAXIMUM problem is same as the algorithm above except we use the algorithm in Theorem 6.3 instead of Algorithm 5. $\qquad\square$

Note that this proof technique does not seem to apply to our optimal algorithm 6. Similarly to Theorem 6.7, we get the following result for the ε-MINIMUM, (ε, ϕ)-BORDA, and (ε, ϕ)-MAXIMIN problems.

Theorem 6.8. *There are randomized one-pass algorithms for ε-MINIMUM, (ε, ϕ)-BORDA, and (ε, ϕ)-MAXIMIN problems with space complexity $O((1/\varepsilon)\log\log(1/\varepsilon\delta) + \log\log m)$, $O\left(n\left(\log n + \log\frac{1}{\varepsilon} + \log\log\frac{1}{\delta}\right) + \log\log m\right)$, and $O\left(n\varepsilon^{-2}\log^2 n + n\varepsilon^{-2}\log n\log(1/\delta) + \log\log m\right)$ bits respectively even when the length of the stream is not known beforehand. Moreover, the update time for ε-MINIMUM is $O(1)$.*

6.4 Results on Space Complexity Lower Bounds

In this section, we prove space complexity lower bounds for the ε-HEAVY HITTERS, ε-MINIMUM, ε-BORDA, and ε-MAXIMIN problems. We present reductions from certain communication problems for proving space complexity lower bounds. Let us first introduce those communication problems with necessary results.

6.4.1 Communication Complexity

Definition 6.9. *(INDEXING$_{m,t}$)*
Let t *and* m *be positive integers. Alice is given a string* $x = (x_1, \cdots, x_t) \in [m]^t$. *Bob is given an index* $i \in [t]$. *Bob has to output* x_i.

The following is a well known result [KN97].

Lemma 6.5. $\mathcal{R}_\delta^{1\text{-}way}(\text{INDEXING}_{m,t}) = \Omega(t \log m)$ *for constant* $\delta \in (0, 1)$.

Definition 6.10. *(AUGMENTED-INDEXING$_{m,t}$)*
Let t *and* m *be positive integers. Alice is given a string* $x = (x_1, \cdots, x_t) \in [m]^t$. *Bob is given an integer* $i \in [t]$ *and* $(x_1, \cdots, x_{i-1})$. *Bob has to output* x_i.

The following communication complexity lower bound result is due to [EJS10] by a simple extension of the arguments of Bar-Yossef et al [BYJKS02].

Lemma 6.6. $\mathcal{R}_\delta^{1\text{-}way}(\text{AUGMENTED-INDEXING}_{m,t}) = \Omega((1 - \delta)t \log m)$ *for any* $\delta < 1 - \frac{3}{2m}$.

[SW15] defines a communication problem called PERM, which we generalize to ε-PERM as follows.

Definition 6.11. *(ε-PERM)*
Alice is given a permutation σ *over* $[n]$ *which is partitioned into* $1/\varepsilon$ *many contiguous blocks. Bob is given an index* $i \in [n]$ *and has to output the block in* σ *where* i *belongs.*

Our lower bound for ε-PERM matches the lower bound for PERM in Lemma 1 in [SW15] when $\varepsilon = 1/n$. For the proof, the reader may find useful some information theory facts described in Appendix A.

Lemma 6.7. $\mathcal{R}_\delta^{1\text{-}way}(\varepsilon - \text{PERM}) = \Omega(n \log(1/\varepsilon))$, *for any constant* $\delta < 1/10$.

Proof. Let us assume σ, the permutation Alice has, is uniformly distributed over the set of all permutations. Let τ_j denotes the block the item j is in for $j \in [n]$, $\tau = (\tau_1, \ldots, \tau_n)$, and $\tau_{<j} = (\tau_1, \ldots, \tau_{j-1})$. Let $M(\tau)$ be Alice's message to Bob, which is a random variable depending on the randomness of σ and the private coin tosses of Alice. Then we have $\mathcal{R}^{1-way}(\varepsilon - \text{PERM}) \geqslant H(M(\tau)) \geqslant I(M(\tau); \tau)$. Hence it is enough to lower bound $I(M(\tau); \tau)$. Then we have the following by chain rule.

$$
\begin{aligned}
I(M(\tau); \tau) &= \sum_{j=1}^{n} I(M(\tau); \tau_j | \tau_{<j}) \\
&= \sum_{j=1}^{n} H(\tau_j | \tau_{<j}) - H(\tau_j | M(\tau), \tau_{<j}) \\
&\geqslant \sum_{j=1}^{n} H(\tau_j | \tau_{<j}) - \sum_{j=1}^{n} H(\tau_j | M(\tau)) \\
&= H(\tau) - \sum_{j=1}^{n} H(\tau_j | M(\tau))
\end{aligned}
$$

The number of ways to partition n items into $1/\varepsilon$ blocks is $n!/((\varepsilon n)!)^{(1/\varepsilon)}$ which is $\Omega\left((n/e)^n / (\varepsilon n/e)^n\right)$. Hence we have $H(\tau) = n \log(1/\varepsilon)$. Now we consider $H(\tau_j | M(\tau))$. By the correctness of the algorithm, Fano's inequality, we have $H(\tau_j | M(\tau)) \leqslant H(\delta) + (1/10) \log_2((1/\varepsilon) - 1) \leqslant (1/2) \log(1/\varepsilon)$. Hence we have the following.

$$
I(M(\tau); \tau) \geqslant (n/2) \log(1/\varepsilon)
$$

$\square$

Finally, we consider the GREATER-THAN problem.

Definition 6.12. *(GREATER-THAN$_n$)*
Alice is given an integer $x \in [n]$ and Bob is given an integer $y \in [n], y \neq x$. Bob has to output 1 if $x > y$ and 0 otherwise.

The following result is due to [Smi88, MNSW98]. We provide a simple proof of it that seems to be missing[1] in the literature.

Lemma 6.8. $\mathcal{R}_\delta^{1-way}(\text{GREATER-THAN}_n) = \Omega(\log n)$, *for every $\delta < 1/4$.*

---

[1]A similar proof appears in [KNR99] but theirs gives a weaker lower bound.

Proof. We reduce the AUGMENTED-INDEXING$_{2,\lceil \log n \rceil + 1}$ problem to the GREATER-THAN$_n$ problem thereby proving the result. Alice runs the GREATER-THAN$_n$ protocol with its input number whose representation in binary is $a = (x_1 x_2 \cdots x_{\lceil \log n \rceil} 1)_2$. Bob participates in the GREATER-THAN$_n$ protocol with its input number whose representation in binary is $b = (x_1 x_2 \cdots x_{i-1} 1 \underbrace{0 \cdots 0}_{(\lceil \log n \rceil - i + 1) \ 0's})_2$. Now $x_i = 1$ if and only if $a > b$. $\square$

6.4.2 Reductions from Problems in Communication Complexity

We observe that a trivial $\Omega((1/\varphi) \log n)$ bits lower bound for (ε, φ)-LIST HEAVY HITTERS, (ε, φ)-LIST BORDA, (ε, φ)-LIST MAXIMIN follows from the fact that any algorithm may need to output $1/\phi$ many items from the universe. Also, there is a trivial $\Omega(n \log n)$ lower bound for (ε, φ)-LIST BORDA and (ε, φ)-LIST MAXIMIN because each stream item is a permutation on $[n]$, hence requiring $\Omega(n \log n)$ bits to read.

We show now a space complexity lower bound of $\Omega(\frac{1}{\varepsilon} \log \frac{1}{\phi})$ bits for the ε-HEAVY HITTERS problem.

Theorem 6.9. *Suppose the size of universe n is at least $1/(\varepsilon \phi^\mu)$ for any constant $\mu > 0$ and that $\phi > 2\varepsilon$. Any randomized one pass (ε, ϕ)-HEAVY HITTERS algorithm with success probability at least $(1 - \delta)$ must use $\Omega((1/\varepsilon) \log 1/\phi)$ bits of space, for constant $\delta \in (0, 1)$.*

Proof. Let $\mu > 0$ be any constant. Without loss of generality, we can assume $\mu \leqslant 1$. We will show that, when $n \geqslant 1/(\varepsilon \phi^\mu)$, any ε-HEAVY HITTERS algorithm must use $\Omega((1/\varepsilon) \log 1/\phi)$ bits of memory, thereby proving the result. Consider the INDEXING$_{1/\phi^\mu, 1/\varepsilon}$ problem where Alice is given a string $x = (x_1, x_2, \cdots, x_{1/\varepsilon}) \in [1/\phi^\mu]^{1/\varepsilon}$ and Bob is given an index $i \in [1/\varepsilon]$. The stream we generate is over $[1/\phi^\mu] \times [1/\varepsilon] \subseteq \mathcal{U}$ (this is possible since $|\mathcal{U}| \geqslant 1/(\varepsilon \phi^\mu)$). Alice generates a stream of length $m/2$ in such a way that the frequency of every item in $\{(x_j, j) : j \in [1/\varepsilon]\}$ is at least $\lfloor \varepsilon m/2 \rfloor$ and the frequency of any other item is 0. Alice now sends the memory content of the algorithm to Bob. Bob resumes the run of the algorithm by generating another stream of length $m/2$ in such a way that the frequency of every item in $\{(j, i) : j \in [1/\phi^\mu]\}$ is at least $\lfloor \phi^\mu m/2 \rfloor$ and the frequency of any other item is 0. The frequency of the item (x_i, i) is at least $\lfloor \varepsilon m/2 + \phi^\mu m/2 \rfloor$ whereas the frequency of every other item is at most $\lfloor \phi^\mu m/2 \rfloor$. Hence from the output of the $(\varepsilon/5, \phi/2)$-HEAVY HITTERS algorithm Bob knows i with probability at least $(1 - \delta)$. Now the result follows from Lemma 6.5. $\square$

We now use the same idea as in the proof of Theorem 6.9 to prove an $\Omega(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ space complexity lower bound for the ε-Maximum problem.

Theorem 6.10. *Suppose the size of universe n is at least $\frac{1}{\varepsilon^{1+\mu}}$ for any constant $\mu > 0$. Any randomized one pass ε-MAXIMUM algorithm with success probability at least $(1 - \delta)$ must use $\Omega(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ bits of space, for constant $\delta \in (0,1)$.*

Proof. Let $\mu > 0$ be any constant. Without loss of generality, we can assume $\mu \leqslant 1$. We will show that, when $n \geqslant \frac{1}{\varepsilon^{1+\mu}}$, any ε-MAXIMUM algorithm must use $\Omega(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ bits of memory, thereby proving the result. Consider the INDEXING$_{1/\varepsilon^\mu, 1/\varepsilon}$ problem where Alice is given a string $x = (x_1, x_2, \cdots, x_{1/\varepsilon}) \in [1/\varepsilon^\mu]^{1/\varepsilon}$ and Bob is given an index $i \in [1/\varepsilon]$. The stream we generate is over $[1/\varepsilon^\mu] \times [1/\varepsilon] \subseteq \mathcal{U}$ (this is possible since $|\mathcal{U}| \geqslant \frac{1}{\varepsilon^{1+\mu}}$). Alice generates a stream of length $m/2$ in such a way that the frequency of every item in $\{(x_j, j) : j \in [1/\varepsilon]\}$ is at least $\lfloor \varepsilon m/2 \rfloor$ and the frequency of any other item is 0. Alice now sends the memory content of the algorithm to Bob. Bob resumes the run of the algorithm by generating another stream of length $m/2$ in such a way that the frequency of every item in $\{(j, i) : j \in [1/\varepsilon^\mu]\}$ is at least $\lfloor \varepsilon^\mu m/2 \rfloor$ and the frequency of any other item is 0. The frequency of the item (x_i, i) is at least $\lfloor \varepsilon m/2 + \varepsilon^\mu m/2 \rfloor$ where as the frequency of every other item is at most $\lfloor \varepsilon^\mu m/2 \rfloor$. Hence the $\varepsilon/5$-MAXIMUM algorithm must output (x_i, i) with probability at least $(1 - \delta)$. Now the result follows from Lemma 6.5. $\qquad \square$

For ε-MINIMUM, we prove a space complexity lower bound of $\Omega(1/\varepsilon)$ bits.

Theorem 6.11. *Suppose the universe size n is at least $1/\varepsilon$. Then any randomized one pass ε-MINIMUM algorithm must use $\Omega(1/\varepsilon)$ bits of space.*

Proof. We reduce from INDEXING$_{2, 5/\varepsilon}$ to ε-MINIMUM thereby proving the result. Let the inputs to Alice and Bob in INDEXING$_{2, 5/\varepsilon}$ be $(x_1, \ldots, x_{5/\varepsilon}) \in \{0,1\}^{5/\varepsilon}$ and an index $i \in [5/\varepsilon]$ respectively. Alice and Bob generate a stream $\mathcal{S}$ over the universe $[(5/\varepsilon) + 1]$. Alice puts two copies of item j in $\mathcal{S}$ for every $j \in \mathcal{U}$ with $x_j = 1$ and runs the ε-MINIMUM algorithm. Alice now sends the memory content of the algorithm to Bob. Bob resumes the run of the algorithm by putting two copies of every item in $\mathcal{U} \setminus \{i, (5/\varepsilon) + 1\}$ in the stream $\mathcal{S}$. Bob also puts one copy of $(5/\varepsilon) + 1$ in $\mathcal{S}$. Suppose the size of the support of $(x_1, \ldots, x_{5/\varepsilon})$ be ℓ. Since $1/(2\ell + (2/\varepsilon) - 1) > \varepsilon/5$, we have the following. If $x_i = 0$, then the ε-MINIMUM algorithm must output i with probability at least $(1 - \delta)$. If $x_i = 1$, then the ε-MINIMUM algorithm must output $(5/\varepsilon) + 1$ with probability at least $(1 - \delta)$. Now the result follows from Lemma 6.5. $\qquad \square$

We show next a $\Omega(n\log(1/\varepsilon))$ bits space complexity lower bound for ε-BORDA.

Theorem 6.12. *Any one pass algorithm for ε-BORDA must use $\Omega(n\log(1/\varepsilon))$ bits of space.*

Proof. We reduce ε-PERM to ε-BORDA. Suppose Alice has a permutation σ over $[n]$ and Bob has an index $i \in [n]$. The item set of our reduced election is $\mathcal{U} = [n] \sqcup \mathcal{D}$, where $\mathcal{D} = \{d_1, d_2, \ldots, d_{2n}\}$. Alice generates a vote $\mathfrak{v}$ over the item set $\mathcal{U}$ from σ as follows. The vote $\mathfrak{v}$ is $\mathcal{B}_1 \succ \mathcal{B}_2 \succ \cdots \succ \mathcal{B}_{1/\varepsilon}$ where $\mathcal{B}_j$ for $j = 1, \ldots, 1/\varepsilon$ is defined as follows.

$$\mathcal{B}_j = d_{(j-1)2\varepsilon n+1} \succ d_{(j-1)2\varepsilon n+2} \succ \cdots \succ d_{(2j-1)\varepsilon n}$$
$$\succ \sigma_{j\varepsilon n+1} \succ \cdots \succ \sigma_{(j+1)\varepsilon n} \succ d_{(2j-1)\varepsilon+1} \succ \cdots \succ d_{2j\varepsilon n}$$

Alice runs the ε-BORDA algorithm with the vote $\mathfrak{v}$ and sends the memory content to Bob. Let $\mathcal{D}_{-i} = \mathcal{D} \setminus \{i\}$, $\overrightarrow{\mathcal{D}_{-i}}$ be an arbitrary but fixed ordering of the items in $\mathcal{D}_{-i}$, and $\overleftarrow{\mathcal{D}_{-i}}$ be the reverse ordering of $\overrightarrow{\mathcal{D}_{-i}}$. Bob resumes the algorithm by generating two votes each of the form $i \succ \overrightarrow{\mathcal{D}_{-i}}$ and $i \succ \overleftarrow{\mathcal{D}_{-i}}$. Let us call the resulting election $\mathcal{E}$. The number of votes m in $\mathcal{E}$ is 5. The Borda score of the item i is at least $12n$. The Borda score of every item $x \in \mathcal{U}$ is at most $9n$. Hence for $\varepsilon < 1/15$, the ε-BORDA algorithm must output the item i. Moreover, it follows from the construction of $\mathfrak{v}$ that an εmn additive approximation of the Borda score of the item i reveals the block where i belongs in the ε-PERM instance. $\qquad\square$

We next give a nearly-tight lower bound for the ε-MAXIMIN problem.

Theorem 6.13. *Any one-pass algorithm for ε-MAXIMIN requires $\Omega(n/\varepsilon^2)$ memory bits of storage.*

Proof. We reduce from INDEXING. Let $\gamma = 1/\varepsilon^2$. Suppose Alice has a string y of length $(n-\gamma)\cdot\gamma$, partitioned into $n-\gamma$ blocks of length γ each. Bob has an index $\ell = i+(j-\gamma-1)\cdot\gamma$ where $i \in [\gamma], j \in \{\gamma+1, \ldots, n\}$. The INDEXING problem is to return y_ℓ for which there is a $\Omega(|y|) = \Omega(n/\varepsilon^2)$ lower bound (Lemma 6.5).

The initial part of the reduction follows the construction in the proof of Theorem 6 in [VWWZ15], which we encapsulate in the following lemma.

Lemma 6.9 (Theorem 6 in [VWWZ15]). *Given y, Alice can construct a matrix $P \in \{0,1\}^{n\times\gamma}$ using public randomness, such that if P^i and P^j are the i'th and j'th rows of P respectively, then with probability at least 2/3, $\Delta(P^i, P^j) \geqslant \frac{\gamma}{2} + \sqrt{\gamma}$ if $y_\ell = 1$ and $\Delta(a, b) \leqslant \frac{\gamma}{2} - \sqrt{\gamma}$ if $y_\ell = 0$.*

Let Alice construct P according to Lemma 6.9 and then adjoin the bitwise complement of the matrix P below P to form the matrix $P' \in \{0, 1\}^{2n \times \gamma}$; note that each column of P' has exactly n 1's and n 0's. Now, we interpret each row of P as a candidate and each column of P as a vote in the following way: for each $v \in [\gamma]$, vote v has the candidates in $\{c : P'_{c,v} = 1\}$ in ascending order in the top n positions and the rest of the candidates in ascending order in the bottom n positions. Alice inserts these γ votes into the stream and sends the state of the ε-MAXIMIN algorithm to Bob as well as the Hamming weight of each row in P'. Bob inserts γ more votes, in each of which candidate i comes first, candidate j comes second, and the rest of the $2n - 2$ candidates are in arbitrary order.

Note that because of Bob's votes, the maximin score of j is the number of votes among the ones casted by Alice in which j defeats i. Since $i < j$, in those columns v where $P_{i,v} = P_{j,v}$, candidate i beats candidate j. Thus, the set of votes in which j defeats i is $\{v \mid P_{i,v} = 0, P_{j,v} = 1\}$. The size of this set is $\frac{1}{2} \left(\Delta(P^i, P^j) + |P^j| - |P^i| \right)$. Therefore, if Bob can estimate the maximin score of j upto $\sqrt{\gamma}/4$ additive error, he can find $\Delta(P^i, P^j)$ upto $\sqrt{\gamma}/2$ additive error as Bob knows $|P^i|$ and $|P^j|$. This is enough, by Lemma 6.9, to solve the INDEXING problem with probability at least $2/3$. $\qquad\square$

Finally, we show a space complexity lower bound that depends on the length of the stream m.

Theorem 6.14. *Any one pass algorithm for ε-HEAVY HITTERS, ε-MINIMUM, ε-BORDA, and ε-MAXIMIN must use $\Omega(\log \log m)$ memory bits, even if the stream is over a universe of size 2, for every $\varepsilon < \frac{1}{4}$.*

Proof. It is enough to prove the result only for ε-HEAVY HITTERS since the other three problems reduce to ε-HEAVY HITTERS for a universe of size 2. Suppose we have a randomized one pass ε-HEAVY HITTERS algorithm which uses $s(m)$ bits of space. Using this algorithm, we will show a communication protocol for the GREATER-THAN$_m$ problem whose communication complexity is $s(2^m)$ thereby proving the statement. The universal set is $\mathcal{U} = \{0, 1\}$. Alice generates a stream of 2^x many copies of the item 1. Alice now sends the memory content of the algorithm. Bob resumes the run of the algorithm by generating a stream of 2^y many copies of the item 0. If $x > y$, then the item 1 is the only ε-winner; whereas if $x < y$, then the item 0 is the only ε-winner. $\qquad\square$

6.5   Conclusion

In this work, we not only resolve a long standing fundamental open problem in the data streaming literature namely heavy hitters but also provide an optimal algorithm for a substantial generalization of heavy hitters by introducing another parameter ϕ. We also initiate a promising direction of research on finding a winner of a stream of votes.

In the next chapter, we study the scenario when voters are allowed to have incomplete preferences in the form of a partial order.

Chapter 7

Kernelization for Possible Winner and Coalitional Manipulation

In the SMALL CAPS POSSIBLE WINNER problem in computational social choice theory, we are given a set of partial preferences and the question is whether a distinguished candidate could be made winner by extending the partial preferences to linear preferences. Previous work has provided, for many common voting rules, fixed parameter tractable algorithms for the POSSIBLE WINNER problem, with number of candidates as the parameter. However, the corresponding kernelization question is still open and in fact, has been mentioned as a key research challenge [BCF+14a]. In this work, we settle this open question for many common voting rules.

We show that the POSSIBLE WINNER problem for maximin, Copeland, Bucklin, ranked pairs, and a class of scoring rules that includes the Borda voting rule do not admit a polynomial kernel with the number of candidates as the parameter. We show however that the COALITIONAL MANIPULATION problem which is an important special case of the POSSIBLE WINNER problem does admit a polynomial kernel for maximin, Copeland, ranked pairs, and a class of scoring rules that includes the Borda voting rule, when the number of

manipulators is polynomial in the number of candidates. A significant conclusion of this work is that the POSSIBLE WINNER problem is harder than the COALITIONAL MANIPULATION problem since the COALITIONAL MANIPULATION problem admits a polynomial kernel whereas the POSSIBLE WINNER problem does not admit a polynomial kernel.

7.1   Introduction

Usually, in a voting setting, it is assumed that the votes are complete orders over the candidates. However, due to many reasons, for example, lack of knowledge of voters about some candidates, a voter may be indifferent between some pairs of candidates. Hence, it is both natural and important to consider scenarios where votes are partial orders over the candidates. When votes are only partial orders over the candidates, the winner cannot be determined with certainty since it depends on how these partial orders are extended to linear orders. This leads to a natural computational problem called the POSSIBLE WINNER [KL05] problem: given a set of partial votes P and a distinguished candidate c, is there a way to extend the partial votes to linear ones to make c win? The POSSIBLE WINNER problem has been studied extensively in the literature [LPR$^+$07, PRVW07, Wal07, XC11, BHN09, BD09, CLMM10, BBN10, BRR11, LPR$^+$12, FRRS14] following its definition in [KL05]. Betzler et al. [BD09] and Baumeister et al. [BR12] show that the POSSIBLE WINNER winner problem is NP-complete for all scoring rules except for the plurality and veto voting rules; the POSSIBLE WINNER winner problem is in P for the plurality and veto voting rules. Moreover the POSSIBLE WINNER problem is known to be NP-complete for many common voting rules, for example, scoring rules, maximin, Copeland, Bucklin etc. even when the maximum number of undetermined pairs of candidates in every vote is bounded above by small constants [XC11]. Walsh showed that the POSSIBLE WINNER problem can be solved in polynomial time for all the voting rules mentioned above when we have a constant number of candidates [Wal07]. An important special case of the POSSIBLE WINNER problem is the COALITIONAL MANIPULATION problem [BTT89] where only two kinds of partial votes are allowed - complete preference and empty preference. The set of empty votes is called the manipulators' vote and is denoted by M. The COALITIONAL MANIPULATION problem is NP-complete for maximin, Copeland, and ranked pairs voting rules even when $|M| \geqslant 2$ [FHS08, FHS10, XZP$^+$09]. The COALITIONAL MANIPULATION problem is in P for the Bucklin voting rule [XZP$^+$09]. We refer to [XC11, Wal07, XZP$^+$09] for detailed overviews.

7.1.1 Our Contribution

Discovering kernelization algorithms is currently an active and interesting area of research in computational social choice theory [BBN10, Bet10, BFG$^+$09, BCH$^+$14, FVBNS13, BCF$^+$14b, BBCN12, DS12]. Betzler et al. [BHN09] showed that the POSSIBLE WINNER problem admits fixed parameter tractable algorithm when parameterized by the total number of candidates for scoring rules, maximin, Copeland, Bucklin, and ranked pairs voting rules. Yang et al. [YG13, Yan14] provides efficient fixed parameter tractable algorithms for the COALITIONAL MANIPULATION problem for the Borda, maximin, and Copeland voting rules. A natural and practical follow-up question is whether the POSSIBLE WINNER and COALITIONAL MANIPULATION problems admit a polynomial kernel when parameterized by the number of candidates. This question has been open ever since the work of Betzler et al. and in fact, has been mentioned as a key research challenge in parameterized algorithms for computational social choice theory [BCF$^+$14a]. Betzler et al. showed non-existence of polynomial kernel for the POSSIBLE WINNER problem for the k-approval voting rule when parameterized by (t, k), where t is the number of partial votes [Bet10]. The NP-complete reductions for the POSSIBLE WINNER problem for scoring rules, maximin, Copeland, Bucklin, and ranked pairs voting rules given by Xia et al. [XZP$^+$09] are from the EXACT 3 SET COVER problem. Their results do not throw any light on the existence of a polynomial kernel since EXACT 3 SET COVER has a trivial $O(m^3)$ kernel where m is the size of the universe. In this work, we show that there is no polynomial kernel (unless CoNP $\subseteq$ NP/Poly) for the POSSIBLE WINNER problem, when parameterized by the total number of candidates, with respect to maximin, Copeland, Bucklin, and ranked pairs voting rules, and a class of scoring rules that includes the Borda voting rule. These hardness results are shown by a parameter-preserving many-to-one reduction from the SMALL UNIVERSE SET COVER problem for which there does not exist any polynomial kernel parameterized by universe size unless CoNP $\subseteq$ NP/Poly [DLS09].

On the other hand, we show that the COALITIONAL MANIPULATION problem admits a polynomial kernel for maximin, Copeland, and ranked pairs voting rules, and a class of scoring rules that includes the Borda voting rule when we have $poly(m)$ number of manipulators – specifically, we exhibit an $O(m^2|M|)$ kernel for maximin and Copeland voting rules, and an $O(m^4|M|)$ kernel for the ranked pairs voting rule, where m is the number of candidates and M is the set of manipulators. The COALITIONAL MANIPULATION problem for the Bucklin voting rule is in P [XZP$^+$09] and thus the kernelization question does not arise.

A significant conclusion of our work is that, although the POSSIBLE WINNER and COALITIONAL MANIPULATION problems are both NP-complete for many voting rules, the POSSIBLE WINNER problem is harder than the COALITIONAL MANIPULATION problem since the COALITIONAL MANIPULATION problem admits a polynomial kernel whereas the POSSIBLE WINNER problem does not admit a polynomial kernel.

7.2   Problem Definitions

The POSSIBLE WINNER problem with respect to r is the following:

POSSIBLE WINNER $[r]$ **Parameter:** m

Input: A set $\mathcal{C}$ of candidates and votes $\mathcal{V}$, where each vote is a partial order over $\mathcal{C}$, and a candidate $c \in \mathcal{C}$.

Question: Is there a linear extension of $\mathcal{V}$ for which c is the unique winner with respect to r?

An important special case of the POSSIBLE WINNER problem is the COALITIONAL MANIPULATION problem where every partial vote is either an empty order or a complete order. We call the complete votes the votes of the non-manipulators and the empty votes the votes of the manipulators. Formally the COALITIONAL MANIPULATION problem is defined as follows.

COALITIONAL MANIPULATION $[r]$ **Parameter:** m

Input: A set $\mathcal{C}$ of candidates, a set $\mathcal{V}$ of complete votes, an integer t corresponding to the number of manipulators, and a candidate $c \in \mathcal{C}$.

Question: Does there exist a set of votes $\mathcal{V}'$ of size t such that c is the unique winner with respect to r for the voting profile $(\mathcal{V}, \mathcal{V}')$?

When the voting rule r is clear from the context, we often refer to the problems as POSSIBLE WINNER and COALITIONAL MANIPULATION without any further qualification. We note that one might also consider the variant of the problem where the designated candidate c is only required to be a co-winner, instead of being the unique winner. All the results in this work can be easily adapted to this variant as well.

We now briefly describe the framework in which we analyze the computational complexity of POSSIBLE WINNER and COALITIONAL MANIPULATION problems.

7.3 Kernelization Hardness of the Possible Winner Problem

In this section, we show non-existence of polynomial kernels for the POSSIBLE WINNER problem for the maximin, Copeland, Bucklin, and ranked pairs voting rules, and a class of scoring rules that includes the Borda voting rule. We do this by demonstrating polynomial parameter transformations from the SMALL UNIVERSE SET COVER problem, which is the classic SET COVER problem, but now parameterized by the size of the universe and the budget.

SMALL UNIVERSE SET COVER $\qquad\qquad$ **Parameter:** $m + k$

Input: A set $\mathcal{U} = \{u_1, \ldots, u_m\}$ and a family $\mathcal{F} = \{S_1, \ldots, S_t\}$.

Question: Is there a subfamily $\mathcal{H} \subseteq \mathcal{F}$ of size at most k such that every element of the universe belongs to at least one $H \in \mathcal{H}$?

It is well-known [DLS09] that RED-BLUE DOMINATING SET parameterized by k and the number of non-terminals does not admit a polynomial kernel unless $\text{CoNP} \subseteq \text{NP/Poly}$. It follows, by the duality between dominating set and set cover, that SET COVER when parameterized by the solution size and the size of the universe (in other words, the SMALL UNIVERSE SET COVER problem defined above) does not admit a polynomial kernel unless $\text{CoNP} \subseteq \text{NP/Poly}$.

We now consider the POSSIBLE WINNER problem parameterized by the number of candidates for the maximin, Copeland, Bucklin, and ranked pairs voting rules, and a class of scoring rules that includes the Borda rule, and establish that they do not admit a polynomial kernel unless $\text{CoNP} \subseteq \text{NP/Poly}$, by polynomial parameter transformations from SMALL UNIVERSE SET COVER.

7.3.1 Results for the Scoring Rules

We begin with proving the hardness of kernelization for the POSSIBLE WINNER problem for a class of scoring rules that includes the Borda voting rule. For that, we use the following lemma which has been used before [BRR11].

Lemma 7.1. *Let* $\mathcal{C} = \{c_1, \ldots, c_m\} \uplus D, (|D| > 0)$ *be a set of candidates, and* $\vec{\alpha}$ *a normalized score vector of length* $|\mathcal{C}|$. *Then, for any given* $\mathbf{X} = (X_1, \ldots, X_m) \in \mathbb{Z}^m$, *there exists* $\lambda \in \mathbb{R}$ *and a voting profile such that the* $\vec{\alpha}$-*score of* c_i *is* $\lambda + X_i$ *for all* $1 \leqslant i \leqslant m$, *and the score of candidates* $d \in D$ *is less than* λ. *Moreover, the number of votes is* $O(\text{poly}(|\mathcal{C}| \cdot \sum_{i=1}^{m} |X_i|))$.

With the above lemma at hand, we now show hardness of polynomial kernel for the class of strict scoring rules.

Theorem 7.1. *The* POSSIBLE WINNER *problem for any strict scoring rule, when parameterized by the number of candidates, does not admit a polynomial kernel unless* CoNP $\subseteq$ NP/Poly.

Proof. Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of SMALL UNIVERSE SET COVER, where $\mathcal{U} = \{u_1, \ldots, u_m\}$ and $\mathcal{F} = \{S_1, \ldots, S_t\}$. We use T_i to denote $\mathcal{U} \setminus S_i$ for $i \in [t]$. We let $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{2m+3})$ denote the score vector of length t, and let δ_i denote the difference $(\alpha_i - \alpha_{i+1})$ for $i \in [2m+2]$. Note that for a strict scoring rule, all the δ_i's will be strictly positive. We now construct an instance $(\mathcal{C}, V, c)$ of POSSIBLE WINNER as follows.

Candidates $\mathcal{C} = \mathcal{U} \uplus \mathcal{V} \uplus \{w, c, d\}$, where $\mathcal{V} := \{v_1, \ldots, v_m\}$.

Partial Votes, P The first part of the voting profile comprises t partial votes, and will be denoted by P. Let V_j denote the set $\{v_1, \ldots, v_j\}$. For each $i \in [t]$, we first consider a profile built on a total order η_i:

$$\eta_i := d \succ S_i \succ V_j \succ w \succ \text{others}, \text{where } j = m - |S_i|.$$

Now we obtain a partial order λ_i based on η_i for every $i \in [t]$ as follows:

$$\lambda_i := \eta_i \setminus (\{w\} \times (\{d\} \uplus S_i \uplus V_j))$$

That is, λ_i is the partial vote where the order between the candidates x and y for $x \neq w$ or $y \notin \{d\} \uplus S_i \uplus V_j$, is same as the order between x and y in η_i. Whereas, if $x = w$ and $y \in \{d\} \uplus S_i \uplus V_j$, then the order between x and y is unspecified. Let P' be the set of votes $\{\eta_i \mid i \in [t]\}$ and P be the set of votes $\{\lambda_i \mid i \in [t]\}$.

Complete Votes, Q We now add complete votes, which we denote by Q, such that $s(c) = s(u_i), s(d) - s(c) = (k-1)\delta_1, s(c) - s(w) = k(\delta_2 + \delta_3 + \cdots + \delta_{m+1}) + \delta_1, s(c) > s(v_i) + 1$ for all $i \in [t]$, where $s(a)$ is the score of candidate a from the combined voting profile $P' \uplus Q$. From the proof of Lemma 7.1, we can see that such votes can always be constructed. In particular, also note that the voting profile Q consists of complete votes. Note that the number of candidates is $2m + 3$, which is polynomial in the size of the universe, as desired.

We now claim that the candidate c is a possible winner for the voting profile $P \uplus Q$ with respect to the strict scoring rule $\vec{\alpha}$ if and only if $(\mathcal{U}, \mathcal{F})$ is a YES instance of Set Cover.

In the forward direction, suppose, without loss of generality, that $S_1, \ldots, S_k$ form a set cover. Then we propose the following extension for the partial votes $\lambda_1, \ldots, \lambda_k$:

$$w > d > S_i > V_j > \text{others},$$

and the following extension for the partial votes $\lambda_{k+1}, \ldots, \lambda_t$:

$$d > S_i > V_j > w > \text{others}$$

For $i \in [k]$, the position of d in the extension of λ_i proposed above is one lower than its original position in η_i. Therefore, the score of d decreases by $k\delta_1$ making the final score of d less than the score of c. Similarly, since $S_1, \ldots, S_k$ form a set cover, the score of u_i decreases by at least $\min_{i=2}^{m}\{\delta_i\}$ for every $i \in [m]$, which is strictly positive as the scoring rule is strict. Finally, the score of w increase by at most $k(\delta_2 + \delta_3 + \cdots + \delta_{m+1})$, since there are at most k votes where the position of w in the extension of λ_i improved from it's original position in η_i for $i \in [t]$. Therefore, the score of c is greater than any other candidate, implying that c is a possible winner.

For the reverse direction, notice that there must be at least k extensions where d is in the second position, since the score of d is $(k-1)\delta_1$ more than the score of c. In these extensions, observe that w will be at the first position. On the other hand, placing w in the first position causes its score to increase by $(\delta_2 + \delta_3 + \cdots + \delta_{m+1})$, therefore, if w is in the first position in ℓ extensions, its score increases by $\ell(\delta_2 + \delta_3 + \cdots + \delta_{m+1})$. Since the score difference between w and c is only $k(\delta_2 + \delta_3 + \cdots + \delta_{m+1}) + 1$, we can afford to have w in the first position in *at most* k votes. Therefore, apart from the extensions where d is in the second position, in all remaining extensions, w appears after V_j, and therefore the candidates from S_i continue to be in their original positions. Moreover, there must be exactly k votes where d is at the second position. We now claim that the sets corresponding to the k votes where d is at the second position form a set cover. Indeed, if not, suppose the element u_i is not covered. It is easily checked that the score of such a u_i remains unchanged in this extension, and therefore its score is equal to c, contradicting our assumption that we started with an extension for which c was a winner. $\qquad\square$

The proof of Theorem 7.1 can be generalized to a wider class of scoring rules as stated in the following corollary.

Corollary 7.1. *Let r be a positional scoring rule such that there exists a polynomial function $f : \mathbb{N} \to \mathbb{N}$, such that for every $m \in \mathbb{N}$, there exists an index l in the $f(m)$ length score vector $\vec{\alpha}$ satisfying following,*

$$\alpha_i - \alpha_{i+1} > 0 \; \forall l \leqslant i \leqslant l + m$$

Then the POSSIBLE WINNER *problem for r, when parameterized by the number of candidates, does not admit a polynomial kernel unless* CoNP $\subseteq$ NP/Poly.

7.3.2 Results for the Maximin Voting Rule

We will need the following lemma in subsequent proofs. The lemma has been used before [McG53, XC11].

Lemma 7.2. *Let $f : \mathcal{C} \times \mathcal{C} \longrightarrow \mathbb{Z}$ be a function such that*

1. *$\forall a, b \in \mathcal{C}, f(a, b) = -f(b, a)$.*

2. *$\forall a, b, c, d \in \mathcal{C}, f(a, b) + f(c, d)$ is even.*

Then we can construct in time $O\left(|\mathcal{C}| \sum_{\{a,b\} \in \mathcal{C} \times \mathcal{C}} |f(a, b)|\right)$ an election E with n votes over the candidate set $\mathcal{C}$ such that for all $a, b \in \mathcal{C}$, $D_E(a, b) = f(a, b)$.

We now describe the reduction for the POSSIBLE WINNER problem for the maximin voting rule parameterized by the number of candidates.

Theorem 7.2. *The* POSSIBLE WINNER *problem for the maximin voting rule, when parameterized by the number of candidates, does not admit a polynomial kernel unless* CoNP $\subseteq$ NP/Poly.

Proof. Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of SMALL UNIVERSE SET COVER, where $\mathcal{U} = \{u_1, \ldots, u_m\}$ and $\mathcal{F} = \{S_1, \ldots, S_t\}$. We use T_i to denote $\mathcal{U} \setminus S_i$. We now construct an instance $(\mathcal{C}, V, c)$ of the POSSIBLE WINNER as follows.

Candidates $C := \mathcal{U} \uplus W \uplus \{c, d, x\} \uplus L$, where $W := \{w_1, w_2, \ldots, w_m, w_x\}, L := \{l_1, l_2, l_3\}$.

Partial Votes, P The first part of the voting profile comprises t partial votes, and will be denoted by P. For each $i \in [t]$, we first consider a profile built on a total order η_i. We denote the order $w_1 \succ \cdots \succ w_m \succ w_x$ by $\vec{W}$. From this point onwards, whenever we place a set of candidates in some position of a partial order, we mean that the candidates in the set can

be ordered arbitrarily. For example, the candidates in S_i can be ordered arbitrarily among themselves in the total order η_i below for every $i \in [t]$.

$$\eta_i := L \succ \vec{W} \succ x \succ S_i \succ d \succ c \succ T_i$$

Now we obtain a partial order λ_i based on η_i for every $i \in [t]$ as follows:

$$\lambda_i := \eta_i \setminus (W \times (\{c, d, x\} \uplus \mathcal{U}))$$

The profile P consists of $\{\lambda_i \mid 1 \in [t]\}$.

Complete Votes, Q We now describe the remaining votes in the profile, which are linear orders designed to achieve specific pairwise difference scores among the candidates. This profile, denoted by Q, is defined according to Lemma 7.2 to contain votes such that the pairwise score differences of $P \cup Q$ satisfy the following.

▷ $D(c, w_1) = -2k$.

▷ $D(c, l_1) = -t$.

▷ $D(d, w_1) = -2k - 2$.

▷ $D(x, w_x) = -2k - 2$.

▷ $D(w_i, u_i) = -2t \ \forall \ i \in [m]$.

▷ $D(a_i, l_1) = D(w_x, l_1) = -4t$.

▷ $D(l_1, l_2) = D(l_2, l_3) = D(l_3, l_1) = -4t$.

▷ $D(l, r) \leqslant 1$ for all other pairs $(l, r) \in C \times C$.

We note that the for all $c, c' \in \mathcal{C}$, the difference $|D(c, c') - D_P(c, c')|$ is always even, as long as t is even and the number of sets in $\mathcal{F}$ that contain any element $u \in \mathcal{U}$ is always even. Note that the latter can always be ensured without loss of generality: indeed, if $u \in \mathcal{U}$ occurs in an odd number of sets, then we can always add the set $\{u\}$ if it is missing and remove it if it is present, flipping the parity in the process. In case $\{u\}$ is the only set containing the element u, then we remove the set from both $\mathcal{F}$ and $\mathcal{U}$ and decrease k by one. The number of sets t can

be assumed to be even by adding a dummy element and a dummy pair of sets that contains the said element. It is easy to see that these modifications always preserve the instance. Thus, the constructed instance of POSSIBLE WINNER is $(\mathcal{C}, V, c)$, where $V := P \cup Q$. We now turn to the proof of correctness.

In the forward direction, let $\mathcal{H} \subseteq \mathcal{F}$ be a set cover of size at most k. Without loss of generality, let $|\mathcal{H}| = k$ (since a smaller set cover can always be extended artificially) and let $\mathcal{H} = \{S_1, \dots, S_k\}$ (by renaming).

If $i \leqslant k$, let:
$$\lambda_i^* := L \succ x \succ S_i \succ d \succ c \succ \vec{W} \succ T_i$$

If $k < i \leqslant t$, let:
$$\lambda_i^* := L \succ \vec{W} \succ x \succ S_i \succ d \succ c \succ T_i$$

Clearly λ_i^* extends λ_i for every $i \in [t]$. Let V^* denote the extended profile consisting of the votes $\{\lambda_i^* \mid i \in [t]\} \cup Q$. We now claim that c is the unique winner with respect to the maximin voting rule in V^*.

Since there are k votes in V^* where c is preferred over w_1 and $(t - k)$ votes where w_1 is preferred to c, we have:

$$
\begin{aligned}
D_{V^*}(c, w_1) &= D_V(c, w_1) + k - (t - k) \\
&= -2k + k - (t - k) = -t
\end{aligned}
$$

It is easy to check that maximin score of c is $-t$. Also, it is straightforward to verify the following table.

Therefore, c is the unique winner for the profile V^*.

We now turn to the reverse direction. Let P^* be an extension of P such that $V^* := P^* \cup Q$ admits c as a unique winner with respect to the maximin voting rule. We first argue that P^* must admit a certain structure, which will lead us to an almost self-evident set cover for $\mathcal{U}$.

Let us denote by P_C^* the set of votes in P^* which are consistent with $c \succ w_1$, and let P_W^* be the set of votes in P^* which are consistent with $w_1 \succ c$. We first argue that P_C^* has at most k votes.

Claim 7.1. *Let P_C^* be as defined above. Then $|P_C^*| \leqslant k$.*

Candidate	maximin score
$w_i, \forall i \in \{1, 2, \ldots, m\}$	$< -t$
$u_i, \forall i \in \{1, 2, \ldots, m\}$	$\leqslant -4t$
w_x	$\leqslant -4t$
l_1, l_2, l_3	$\leqslant -4t$
x	$\leqslant -t - 2$
d	$\leqslant -t - 2$

Proof. Suppose, for the sake of contradiction, that more than k extensions are consistent with $c \succ w_1$. Then we have:

$$
\begin{aligned}
D_{V^*}(c, w_1) &\geqslant D_V(c, w_1) + k + 1 - (t - k - 1) \\
&= -2k + 2k - t + 2 = -t + 2
\end{aligned}
$$

Since $D_{V^*}(c, l_1) = -t$, the maximin score of c is $-t$. On the other hand, we also have that the maximin score of d is given by $D_{V^*}(d, w_1)$, which is now at least $(-t)$:

$$
\begin{aligned}
D_{V^*}(d, w_1) &\geqslant D_V(d, w_1) + k + 1 - (t - k - 1) \\
&= -2k - 2 + 2k - t + 2 = -t
\end{aligned}
$$

Therefore, c is no longer the unique winner in V^* with respect to the maximin voting rule, which is the desired contradiction. $\qquad \square$

We next propose that a vote that is consistent with $w_1 \succ c$ must be consistent with $w_x \succ x$.

Claim 7.2. *Let P_W^* be as defined above. Then any vote in P_W^* must respect $w_x \succ x$.*

Proof. Suppose there are r votes in P_C^*, and suppose that in at least one vote in P_W^* where $x \succ w_x$. Notice that any vote in P_C^* is consistent with $x \succ w_x$. Now we have:

$$
\begin{aligned}
D_{V^*}(c, w_1) &= D_V(c, w_1) + r - (t - r) \\
&= -2k + 2r - t \\
&= -t - 2(k - r)
\end{aligned}
$$

And further:

$$
\begin{aligned}
D_{V^*}(x, w_x) &\geqslant D_V(x, w_x) + (r + 1) - (t - r - 1) \\
&= -2k - 2 + 2r - t + 2 \\
&= -t - 2(k - r)
\end{aligned}
$$

It is easy to check that the maximin score of c in V^* is at most $-t - 2(k - r)$, witnessed by $D_{V^*}(c, w_1)$, and the maximin score of x is at least $-t - 2(k - r)$, witnessed by $D_{V^*}(x, w_x)$. Therefore, c is no longer the unique winner in V^* with respect to the maximin voting rule, and we have a contradiction. $\square$

We are now ready to describe a set cover of size at most k for $\mathcal{U}$ based on V^*. Define $J \subseteq [t]$ as being the set of all indices i for which the extension of λ_i in V^* belongs to P_C^*. Consider:

$$
\mathcal{H} := \{S_i \mid i \in J\}.
$$

The set $\mathcal{H}$ is our proposed set cover. Clearly, $|\mathcal{H}| \leqslant k$. It remains to show that $\mathcal{H}$ is a set cover.

We assume, for the sake of contradiction, that there is an element $u_i \in \mathcal{U}$ that is not covered by $\mathcal{H}$. This means that we have $u_i \in T_i$ for all $i \in J$, and thus $w_i \succ u_i$ in the corresponding extensions of λ_i in V^*. Further, for all $i \notin J$, we have that the extension of λ_i in V^* is consistent with:

$$
w_1 \succ \cdots \succ w_i \succ \cdots \succ w_x \succ x \succ S_i \succ c \succ T_i,
$$

implying again that $w_i \succ u_i$ in these votes. Therefore, we have:

$$
D_{V^*}(w_i, u_i) = D_V(w_i, u_i) + k + (t - k) = -2t + t = -t.
$$

We know that the maximin score of c is less than or equal to $-t$, since $D_{V^*}(c, l_1) = -t$, and we now have that the maximin score of w_i is $-t$. This overrules c as the unique winner in V^*, contradicting our assumption to that effect. This completes the proof. $\qquad\square$

7.3.3   Results for the Copeland Voting Rule

We now describe the result for the POSSIBLE WINNER problem for the Copeland voting rule parameterized by the number of candidates.

Theorem 7.3. *The* POSSIBLE WINNER *problem for the Copeland voting rule, when parameterized by the number of candidates, does not admit a polynomial kernel unless* CoNP $\subseteq$ NP/Poly.

Proof. Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of SMALL UNIVERSE SET COVER, where $\mathcal{U} = \{u_1, \ldots, u_m\}$ and $\mathcal{F} = \{S_1, \ldots, S_t\}$. For the purpose of this proof, we assume (without loss of generality) that $m \geqslant 6$. We now construct an instance $(\mathcal{C}, V, c)$ of POSSIBLE WINNER as follows.

Candidates $\mathcal{C} := \mathcal{U} \uplus \{z, c, d, w\}$.

Partial Votes, P The first part of the voting profile comprises of m partial votes, and will be denoted by P. For each $i \in [t]$, we first consider a profile built on a total order:

$$\eta_i := \mathcal{U} \setminus S_i \succ z \succ c \succ d \succ S_i \succ w$$

Now we obtain a partial order λ_i based on η_i as follows for each $i \in [t]$:

$$\lambda_i := \eta_i \setminus (\{z, c\} \times (S_i \uplus \{d, w\}))$$

The profile P consists of $\{\lambda_i \mid i \in [t]\}$.

Complete Votes, Q We now describe the remaining votes in the profile, which are linear orders designed to achieve specific pairwise difference scores among the candidates. This profile, denoted by Q, is defined according to Lemma 7.2 to contain votes such that the pairwise score differences of $P \cup Q$ satisfy the following.

▷ $D(c, d) = t - 2k + 1$

▷ $D(z, w) = t - 2k - 1$

▷ $D(c, u_i) = t - 1$

▷ $D(c, z) = t + 1$

▷ $D(c, w) = -t - 1$

▷ $D(u_i, d) = D(z, u_i) = t + 1 \ \forall \ i \in [m]$

▷ $D(z, d) = t + 1$

▷ $D(u_i, u_j) = t + 1 \ \forall \ j \in [i + 1 \ (\text{mod} *)m, i + \lfloor m/2 \rfloor \ (\text{mod} *)m]$

We note that the difference $|D(c, c') - D_P(c, c')|$ is always even for all $c, c' \in \mathcal{C}$, as long as t is odd and the number of sets in $\mathcal{F}$ that contain any element $a \in \mathcal{U}$ is always odd. Note that the latter can always be ensured without loss of generality: indeed, if $a \in \mathcal{U}$ occurs in an even number of sets, then we can always add the set $\{a\}$ if it is missing and remove it if it is present, flipping the parity in the process. In case $\{a\}$ is the only set containing the element a, then we remove the set from both $\mathcal{F}$ and $\mathcal{U}$ and decrease k by one. The number of sets t can be assumed to be odd by adding a dummy element in $\mathcal{U}$, adding a dummy set that contains the said element in $\mathcal{F}$, and incrementng k by one. It is easy to see that these modifications always preserve the instance.

Thus the constructed instance of POSSIBLE WINNER is $(\mathcal{C}, V, c)$, where $V := P \cup Q$. We now turn to the proof of correctness.

In the forward direction, let $\mathcal{H} \subseteq \mathcal{F}$ be a set cover of size at most k. Without loss of generality, let $|\mathcal{H}| = k$ (since a smaller set cover can always be extended artificially) and let $\mathcal{H} = \{S_1, \dots, S_k\}$ (by renaming).

If $i \leqslant k$, let:
$$\lambda_i^* := \mathcal{U} \setminus S_i \succ z \succ c \succ d \succ S_i \succ w$$

If $k < i \leqslant t$, let:
$$\lambda_i^* := \mathcal{U} \setminus S_i \succ d \succ S_i \succ w \succ z \succ c$$

Clearly λ_i^* extends λ_i for every $i \in [t]$. Let V^* denote the extended profile consisting of the votes $\{\lambda_i^* \mid i \in [t]\} \cup Q$. We now claim that c is the unique winner with respect to the Copeland voting rule in V^*.

First, consider the candidate z. For every $i \in [m]$, between z and u_i, even if z loses to u_i in λ_j^*, for every $j \in [t]$, because $D(z, u_i) = t + 1$, z wins the pairwise election between z

and u_i. The same argument holds between z and d. Therefore, the Copeland score of z, no matter how the partial votes were extended, is at least $(m+1)$.

Further, note that all other candidates (apart from c) have a Copeland score of less than $(m+1)$, because they are guaranteed to lose to at least three candidates (assuming $m \geqslant 6$). In particular, observe that u_i loses to at least $\lfloor m/2 \rfloor$ candidates, and d loses to u_i (merely by its position in the extended votes), and w loses to u_i (because of way the scores were designed) for every $i \in [m]$. Therefore, the Copeland score of all candidates in $\mathcal{C} \setminus \{z, c\}$ is strictly less than the Copeland score of z, and therefore they cannot be possible (co-)winners.

Now we restrict our attention to the contest between z and c. First note that c beats u_i for every $i \in [m]$: since the sets of $\mathcal{H}$ form a set cover, u_i is placed in a position after c in some λ_j^* for $j \in [k]$. Since the difference of score between c and u_i was $(t-1)$, even if c suffered defeat in every other extension, we have the pairwise score of c and u_i being at least $t - 1 - (t-1) + 1 = 1$, which implies that c defeats every u_i in their pairwise election. Note that c also defeats d by getting ahead of d in k votes, making its final score $t - 2k + 1 + k - (t-k) = 1$. Finally, c is defeated by w, simply by the preset difference score. Therefore, the Copeland score of c is $(m+2)$.

Now all that remains to be done is to rule z out of the running. Note that z is defeated by w in their pairwise election: this is because z defeats w in k of the extended votes, and is defeated by w in the remaining. This implies that its final pairwise score with respect to w is at most $t - 2k - 1 + k - (t-k) = -1$. Also note that z loses to c because of its predefined difference score. Thus, the Copeland score of z in the extended vote is exactly $(m+1)$, and thus c is the unique winner of the extended vote.

We now turn to the reverse direction. Let P^* be an extension of P such that $V^* := P^* \cup Q$ admits c as a unique winner with respect to the Copeland voting rule. As with the argument for the maximin voting rule, we first argue that P^* must admit a certain structure, which will lead us to an almost self-evident set cover for $\mathcal{U}$.

Let us denote by P_C^* the set of votes in P^* which are consistent with $c \succ d$, and let P_W^* be the set of votes in P^* which are consistent with $w \succ z$. Note that the votes in P_C^* necessarily have the form:

$$\lambda_i^* := \mathcal{U} \setminus S_i \succ z \succ c \succ d \succ S_i \succ w$$

and those in P_W^* have the form:

$$\lambda_i^* := \mathcal{U} \setminus S_i \succ d \succ S_i \succ w \succ z \succ c$$

It is easy to check that this structure is directly imposed by the relative orderings that are fixed by the partial orders.

Before we argue the details of the scores, let us recall that in any extension of P, z loses to c and z wins over d and all candidates in $\mathcal{U}$. Thus the Copeland score of z is at least $(m+1)$. On the other hand, in any extension of P, c loses to w, and therefore the Copeland score of c is at most $(m+2)$. (These facts follow from the analysis in the forward direction.)

Thus, we have the following situation. If z wins over w, then c cannot be the unique winner in the extended vote, because the score of z goes up to $(m+2)$. Similarly, c cannot afford to lose to any of $\mathcal{U} \cup \{d\}$, because that will cause its score to drop below $(m+2)$, resulting in either a tie with z, or defeat. These facts will successively lead us to the correctness of the reverse direction.

Now let us return to the sets P_C^* and P_W^*. If P_C^* has more than k votes, then z wins over w: the final score of z is at least $t - 2k - 1 + (k+1) - (t - k - 1) = 1$, and we have a contradiction. If P_C^* has fewer than k votes, then c loses to d, with a score of at most $t - 2k + 1 + (k - 1) - (t - k + 1) = -1$, and we have a contradiction. Hence, P_C^* must have exactly k votes.

Finally, suppose the sets corresponding to the votes of P_C^* do not form a set cover. Consider an element $u_i \in \mathcal{U}$ not covered by the union of these sets. Observe that c now loses the pairwise election between itself and u_i and is no longer in the running for being the unique winner in the extended vote. Therefore, the sets corresponding to the votes of P_C^* form a set cover of size exactly k, as desired. $\qquad\square$

7.3.4 Results for the Bucklin Voting Rule

We now describe the result for the POSSIBLE WINNER problem for the Bucklin voting rule parameterized by the number of candidates.

Theorem 7.4. *The* POSSIBLE WINNER *problem for the Bucklin voting rule, when parameterized by the number of candidates, does not admit a polynomial kernel unless* CoNP $\subseteq$ NP/Poly.

Proof. Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of SMALL UNIVERSE SET COVER, where $\mathcal{U} = \{u_1, \ldots, u_m\}$ and $\mathcal{F} = \{S_1, \ldots, S_t\}$. Without loss of generality, we assume that $t > k + 1$, and that every set in $\mathcal{F}$ has at least two elements. We now construct an instance $(\mathcal{C}, V, c)$ of POSSIBLE WINNER as follows.

Candidates $\mathcal{C} := \mathcal{U} \uplus \{z, c, a\} \uplus W \uplus D_1 \uplus D_2 \uplus D_3$, where D_1, D_2, and D_3 are sets of *"dummy candidates"* such that $|D_1| = m$, $|D_2| = 2m$, and $|D_3| = 2m$. $W := \{w_1, w_2, \ldots, w_{2m}\}$.

Partial Votes, P The first part of the voting profile comprises of t partial votes, and will be denoted by P. For each $i \in [t]$, we first consider a profile built on a total order:

$$\eta_i := \mathcal{U} \setminus S_i \succ S_i \succ w_{i \pmod{*} m} \succ w_{i+1 \pmod{*} m} \succ z \succ c \succ D_3 \succ \text{others}$$

Now we obtain a partial order λ_i based on η_i for every $i \in [t]$ as follows:

$$\lambda_i := \eta_i \setminus \left(\left(\{w_{i \pmod{*} m}, w_{i+1 \pmod{*} m}, z, c\} \uplus D_3 \right) \times S_i \right)$$

The profile P consists of $\{\lambda_i \mid i \in [t]\}$.

Complete Votes, Q

$$
\begin{array}{rcl}
t - k - 1 & : & D_1 \succ z \succ c \succ \text{others} \\
1 & : & D_1 \succ c \succ a \succ z \succ \text{others} \\
k - 1 & : & D_2 \succ \text{others}
\end{array}
$$

We now show that $(\mathcal{U}, \mathcal{F}, k)$ is a YES instance if and only if $(\mathcal{C}, V, c)$ is a YES instance. Suppose $\{S_j : j \in J\}$ forms a set cover. Then consider the following extension of P:

$$(\mathcal{U} \setminus S_j) \succ w_{j \pmod{*} m} \succ w_{j+1 \pmod{*} m} \succ z \succ c \succ D_3 \succ S_j \succ \text{others, for } j \in J$$

$$(\mathcal{U} \setminus S_j) \succ S_j \succ w_{j \pmod{*} m} \succ w_{j+1 \pmod{*} m} \succ z \succ c \succ D_3 \succ \text{others, for } j \notin J$$

We claim that in this extension, c is the unique winner with Bucklin score $(m + 2)$. First, let us establish the score of c. The candidate c is already within the top $(m + 1)$ choices in $(t - k)$ of the complete votes. In all the sets that form the set cover, c is ranked within the first $(m + 2)$ votes in the proposed extension of the corresponding vote (recall that every set has at least two elements). Therefore, there are a total of t votes where c is ranked within the top $(m + 2)$ preferences. Further, consider a candidate $v \in \mathcal{U}$. Such a candidate is not within the top $(m + 2)$ choices of any of the complete votes. Let S_i be the set that covers the element v. Note that in the extension of the vote λ_i, v is not ranked among the top $(m + 2)$ spots, since

there are at least m candidates from D_3 getting in the way. Therefore, v has strictly fewer than t votes where it is ranked among the top $(m + 2)$ spots, and thus has a Bucklin score more than c.

Now the candidate z is within the top $(m+2)$ ranks of at most $(t-k-1)$ votes among the complete votes. In the votes corresponding to the sets *not* in the set cover, z is placed beyond the first $(m + 2)$ spots. Therefore, the number of votes where z is among the top $(m + 2)$ candidates is at most $(t - 1)$, which makes its Bucklin score strictly more than $(m + 2)$.

The candidates from W are within the top $(m + 2)$ positions only in a constant number of votes. The candidates $D_1 \cup \{a\}$ have $(t - k)$ votes (among the complete ones) in which they are ranked among the top $(m + 2)$ preferences, but in all extensions, these candidates have ranks below $(m + 2)$. Finally, the candidates in D_3 do not feature in the top $(m + 2)$ positions of any of the complete votes, and similarly, the candidates in D_2 do not feature in the top $(m + 2)$ positions of any of the extended votes. Therefore, the Bucklin scores of all these candidates is easily seen to be strictly more than $(m + 2)$, concluding the argument in the forward direction.

Now consider the reverse direction. Suppose $(\mathcal{C}, V, c)$ is a YES instance. For the same reasons described in the forward direction, observe that only the following candidates can win depending upon how the partial preferences get extended - either one of the candidates in $\mathcal{U}$, or one of z or c. Note that the Bucklin score of z in any extension is at most $(m + 3)$. Therefore, the Bucklin score of c has to be $(m + 2)$ or less. Among the complete votes Q, there are $(t - k)$ votes where the candidate c appears in the top $(m + 2)$ positions. To get majority within top $(m + 2)$ positions, c should be within top $(m + 2)$ positions for at least k of the extended votes in P. Let us call these set of votes P'. Now notice that whenever c comes within top $(m + 2)$ positions in a valid extension of P, the candidate z also comes within top $(m + 2)$ positions in the same vote. However, the candidate z is already ranked among the top $(m + 2)$ candidates in $(t - k - 1)$ complete votes. Therefore, z can appear within top $(m + 2)$ positions in *at most* k extensions (since c is the unique winner), implying that $|P'| = k$. Further, note that the Bucklin score of c cannot be strictly smaller than $(m + 2)$ in any extension. Indeed, candidate c features in only one of the complete votes within the top $(m + 1)$ positions, and it would have to be within the top $(m + 1)$ positions in at least $(t - 1)$ extensions. However, as discussed earlier, this would give z exactly the same mileage, and therefore its Bucklin score would be $(m - 1)$ or even less; contradicting our assumption that c is the unique winner.

Now we claim that the S_i's corresponding to the votes in P' form a set cover for $\mathcal{U}$. If not, there is an element $x \in \mathcal{U}$ that is uncovered. Observe that x appears within top m positions in all the extensions of the votes in P', by assumption. Further, in all the remaining extensions, since z is not present among the top $(m + 2)$ positions, we only have room for two candidates from W. The remaining positions must be filled by all the candidates corresponding to elements of $\mathcal{U}$. Therefore, x appears within the top $(m + 2)$ positions of all the extended votes. Since these constitute half the total number of votes, we have that x ties with c in this situation, a contradiction. $\qquad\square$

7.3.5   Results for the Ranked Pairs Voting Rule

We now describe the reduction for POSSIBLE WINNER parameterized by the number of candidates, for the ranked pairs voting rule.

Theorem 7.5. *The* POSSIBLE WINNER *problem for the ranked pairs voting rule, when parameterized by the number of candidates, does not admit a polynomial kernel unless* CoNP $\subseteq$ NP/Poly.

Proof. Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of SMALL UNIVERSE SET COVER, where $\mathcal{U} = \{u_1, \ldots, u_m\}$ and $\mathcal{F} = \{S_1, \ldots, S_t\}$. Without loss of generality, we assume that t is even. We now construct an instance $(\mathcal{C}, V, c)$ of POSSIBLE WINNER as follows.

Candidates $\mathcal{C} := \mathcal{U} \uplus \{a, b, c, w\}$.

Partial Votes, P The first part of the voting profile comprises of t partial votes, and will be denoted by P. For each $i \in [t]$, we first consider a profile built on a total order:

$$\eta_i := \mathcal{U} \setminus S_i \succ S_i \succ b \succ a \succ c \succ \text{others}$$

Now we obtain a partial order λ_i based on η_i for every $i \in [t]$ as follows:

$$\lambda_i := \eta_i \setminus (\{a, c\} \times (S_i \uplus \{b\}))$$

The profile P consists of $\{\lambda_i \mid i \in [t]\}$.

Complete Votes, Q We add complete votes such that along with the already determined pairs from the partial votes P, we have the following.

$\triangleright$ $D(u_i, c) = 2 \; \forall \, i \in [m]$

$\triangleright$ $D(c, b) = 4t$

$\triangleright$ $D(c, w) = t + 2$

$\triangleright$ $D(b, a) = 2k + 4$

$\triangleright$ $D(w, a) = 4t$

$\triangleright$ $D(a, c) = t + 2$

$\triangleright$ $D(w, u_i) = 4t \; \forall \, i \in [m]$

We now show that $(\mathcal{U}, \mathcal{F}, k)$ is a YES instance if and only if $(\mathcal{C}, V, c)$ is a YES instance. Suppose $\{S_j : j \in J\}$ forms a set cover. Then consider the following extension of P :

$$\mathcal{U} \setminus S_j \succ a \succ c \succ S_j \succ b \succ \text{others } \forall j \in J$$

$$\mathcal{U} \setminus S_j \succ S_j \succ b \succ a \succ c \succ \text{others } \forall j \notin J$$

We claim that the candidate c is the unique winner in this extension. Note that the pairs $(w \succ a)$ and $(w \succ u_i)$ for every $i \in [t]$ get locked first (since these differences are clearly the highest and unchanged). The pair (c, b) gets locked next, with a difference score of $(3t + 2k)$. Now since the votes in which $c \succ b$ are based on a set cover of size at most k, the pairwise difference between b and a becomes at least $2k + 4 - k + (t - k) = t + 4$. Therefore, the next pair to get locked is $b \succ a$. Finally, for every element $u_i \in \mathcal{U}$, the difference $D(u_i, c)$ is at most $2 + (t - 1) = t + 1$, since there is at least one vote where $c \succ u_i$ (given that we used a set cover in the extension). It is now easy to see that the next highest pairwise difference is between c and w, so the ordering $c \succ w$ gets locked, and at this point, by transitivity, c is superior to w, b, a and all u_i. It follows that c wins the election irrespective the sequence in which pairs are considered subsequently.

Now suppose $(\mathcal{C}, V, c)$ is a YES instance. Notice that, irrespective of the extension of the votes in P, $c \succ b, w \succ a, w \succ u_i \; \forall i \in [m]$ are locked first. Now if $b \succ c$ in all the extended votes, then it is easy to check that $b \succ a$ gets locked next, with a difference score of $2k + 4 + t$; leaving us with $D(u_i, c) = t + 2 = D(c, w)$, where $u_i \succ c$ could be a potential lock-in. This implies the possibility of a u_i being a winner in some choice of tie-breaking, a contradiction to the assumption that c is the unique winner. Therefore, there are at least some votes in the

161

extended profile where $c \succ b$. We now claim that there are at most k such votes. Indeed, if there are more, then $D(b, a) = 2k+4-(k+1)+(t-k-1) = t+2$. Therefore, after the forced lock-ins above, we have $D(b, a) = D(c, w) = D(a, c) = t + 2$. Here, again, it is possible for $a \succ c$ to be locked in before the other choices, and we again have a contradiction.

Finally, we have that $c \succ b$ in at most k many extensions in P. Call the set of indices of these extensions J. We claim that $\{S_j : j \in J\}$ forms a set cover. If not, then suppose an element $u_i \in \mathcal{U}$ is not covered by $\{S_j : j \in J\}$. Then the candidate u_i comes before c in all the extensions which makes $D(u_i, c)$ become $(t + 2)$, which in turn ties with $D(c, w)$. This again contradicts the fact that c is the unique winner. Therefore, if there is an extension that makes c the unique winner, then we have the desired set cover. $\qquad\square$

7.4 Polynomial Kernels for the Coalitional Manipulation Problem

We now describe a kernelization algorithm for every scoring rule which satisfies certain properties mentioned in Theorem 7.6 below. Note that the Borda voting rule satisfies these properties.

Theorem 7.6. *For $m \in \mathbb{N}$, let $(\alpha_1, \dots, \alpha_m)$ and $(\alpha'_1, \dots, \alpha'_{m+1})$ be the normalized score vectors for a scoring rule r for an election with m and $(m+1)$ candidates respectively. Let $\alpha'_1 = \mathrm{poly}(m)$ and $\alpha_i = \alpha'_{i+1}$ for every $i \in [m]$. Then the* COALITIONAL MANIPULATION *problem for r admits a polynomial kernel when the number of manipulators is* $\mathrm{poly}(m)$.

Proof. Let c be the candidate whom the manipulators aim to make winner. Let M be the set of manipulators and $\mathcal{C}$ the set of candidates. Let $s_{NM}(x)$ be the score of candidate x from the votes of the non-manipulators. Without loss of generality, we assume that, all the manipulators place c at top position in their votes. Hence, the final score of c is $s_{NM}(c) + |M|\alpha_1$, which we denote by $s(c)$. Now if $s_{NM}(x) \geqslant s(c)$ for any $x \neq c$, then c cannot win and we output *no*. Hence, we assume that $s_{NM}(x) < s(c)$ for all $x \neq c$. Now let us define $s^*_{NM}(x)$ as follows.

$$s^*_{NM}(x) := \max\{s_{NM}(x), s_{NM}(c)\}$$

Also define $s^*_{NM}(c)$ as follows.

$$s^*_{NM}(c) := s_{NM}(c) - |M|(\alpha'_1 - \alpha_1)$$

We define a COALITIONAL MANIPULATION instance with $(m+1)$ candidates as $(\mathcal{C}', NM, M, c)$, where $\mathcal{C}' = \mathcal{C} \uplus \{d\}$ is the set of candidates, M is the set of manipulators, c is the distinguished candidate, and NM is the non-manipulators' vote such that it generates score of $x \in \mathcal{C}$ to be $K + (s^*_{NM}(x) - s_{NM}(c))$, where $K \in \mathbb{N}$ is same for $x \in \mathcal{C}$, and the score of d is less than $K - \alpha'_1 |M|$. The existence of such a voting profile NM of size $\text{poly}(m)$ is due to Lemma 7.1 and the fact that $\alpha'_1 = \text{poly}(m)$. Hence, once we show the equivalence of these two instances, we have a kernel whose size is polynomial in m. The equivalence of the two instances is due to the following facts: (1) The new instance has $(m+1)$ candidates and c is always placed at the top position without loss of generality. The candidate c recieves $|M|(\alpha'_1 - \alpha_1)$ score more than the initial instance and this is compensated in $s^*_{NM}(c)$. (2) The relative score difference from the final score of c to the current score of every $x \in \mathcal{C} \setminus \{c\}$ is same in both the instances. (3) In the new instance, we can assume without loss of generality that the candidate d will be placed in the second position in all the manipulators' votes. $\qquad\square$

We now move on to the voting rules that are based on the weighted majority graph. The reduction rules modify the weighted majority graph maintaining the property that there exists a set of votes that can realize the modified weighted majority graph. In particular, the final weighted majority graph is realizable with a set of votes.

Theorem 7.7. *The* COALITIONAL MANIPULATION *problem for the maximin voting rule admits a polynomial kernel when the number of manipulators is* $\text{poly}(m)$.

Proof. Let c be the distinguished candidate of the manipulators. Let M be the set of all manipulators. We can assume that $|M| \geqslant 2$ since for $|M| = 1$, the problem is in P [BTT89]. Define s to be $\min_{x \in C \setminus \{c\}} D_{(V \setminus M)}(c, x)$. So, s is the maximin score of the candidate c from the votes except from M. Since the maximin voting rule is monotone, we can assume that the voters in M put the candidate c at top position of their preferences. Hence, c's final maximin score will be $s + |M|$. This provides the following reduction rule.

Reduction rule 7.1. *If* $s + |M| \geqslant 0$, *then output* YES.

In the following, we will assume $s + |M|$ is negative. Now we propose the following reduction rules on the weighted majority graph.

Reduction rule 7.2. *If* $D_{(V \setminus M)}(c_i, c_j) < 0$ *and* $D_{(V \setminus M)}(c_i, c_j) > 2|M| + s$, *then make* $D_{(V \setminus M)}(c_i, c_j)$ *either* $2|M| + s + 1$ *or* $2|M| + s + 2$ *whichever keeps the parity of* $D_{(V \setminus M)}(c_i, c_j)$ *unchanged.*

If $D_{(v\setminus M)}(c_i, c_j) > 2|M| + s$, then $D_v(c_i, c_j) > |M| + s$ irrespective of the way the manipulators vote. Hence, given any votes of the manipulators, whether or not the maximin score of c_i and c_j will exceed the maximin score of c does not gets affected by this reduction rule. Hence, Reduction rule 7.1 is sound.

Reduction rule 7.3. *If $D_{(v\setminus M)}(c_i, c_j) < s$, then make $D_{(v\setminus M)}(c_i, c_j)$ either $s - 1$ or $s - 2$ whichever keeps the parity of $D_{(v\setminus M)}(c_i, c_j)$ unchanged.*

The argument for the correctness of Reduction rule 7.3 is similar to the argument for Reduction rule 7.1. Here onward, we may assume that whenever $D_{(v\setminus M)}(c_i, c_j) < 0$, $s - 2 \leqslant D_{(v\setminus M)}(c_i, c_j) \leqslant 2|M| + s + 2$

Reduction rule 7.4. *If $s < -4|M|$ then subtract $s + 5|M|$ from $D_{(v\setminus M)}(x, y)$ for every $x, y \in \mathcal{C}, x \neq y$.*

The correctness of Reduction rule 7.4 follows from the fact that it adds linear fixed offsets to all the edges of the weighted majority graph. Hence, if there a voting profile of the voters in M that makes the candidate c win in the original instance, the same voting profile will make c win the election in the reduced instance and vice versa.

Now we have a weighted majority graph with $O(|M|)$ weights for every edge. Also, all the weights have uniform parity and thus the result follows from Lemma 7.2. $\qquad\square$

We next present a polynomial kernel for the COALITIONAL MANIPULATION problem for the Copeland voting rule.

Theorem 7.8. *The COALITIONAL MANIPULATION problem for the Copeland voting rule admits a polynomial kernel when the number of manipulators is* $\mathrm{poly}(m)$.

Proof. We apply the following reduction rule.

Reduction rule 7.1. *If $D_{(v\setminus M)}(x, y) > |M|$ for $x, y \in \mathcal{C}$, then make $D_{(v\setminus M)}(x, y)$ either $|M| + 1$ or $|M| + 2$ whichever keeps the parity of $D_{(v\setminus M)}(x, y)$ unchanged.*

Given any votes of M, we have $D_v(x, y) > 0$ in the original instance if and only if $D_v(x, y) > 0$ in the reduced instance for every $x, y \in \mathcal{C}$. Hence, each candidate has the same Copeland score and thus the reduction rule is correct.

Now we have a weighted majority graph with $O(|M|)$ weights for every edges. Also, all the weights have uniform parity. From Lemma 7.2, we can realize the weighted majority graph using $O(m^2|M|)$ votes. $\qquad\square$

Now we move on to the ranked pairs voting rule.

Theorem 7.9. *The* COALITIONAL MANIPULATION *problem for the ranked pairs voting rule admits a polynomial kernel when the number of manipulators is* $\mathtt{poly}(m)$.

Proof. Consider all non-negative $D_{(\mathcal{V}\setminus M)}(c_i, c_j)$ and arrange them in non-decreasing order. Let the ordering be $x_1, x_2, \ldots, x_l$ where $l = \binom{m}{2}$. Now keep applying following reduction rule till possible. Define $x_0 = 0$.

Reduction rule 7.1. *If there exist any* i *such that,* $x_i - x_{i-1} > |M| + 2$, *subtract an even offset to all* $x_i, x_{i+1}, \ldots, x_l$ *such that* x_i *becomes either* $(x_{i-1} + |M| + 1)$ *or* $(x_{i-1} + |M| + 2)$.

The reduction rule is correct since for any set of votes by M, for any four candidates $a, b, x, y \in \mathcal{C}$, $D(a, b) > D(x, y)$ in the original instance if and only if $D(a, b) > D(x, y)$ in the reduced instance. Now we have a weighted majority graph with $O(m^2|M|)$ weights for every edges. Also, all the weights have uniform parity and hence can be realized with $O(m^4|M|)$ votes by Lemma 7.2. $\qquad\square$

7.5 Conclusion

Here we showed that the POSSIBLE WINNER problem does not admit a polynomial kernel for many common voting rules under the complexity theoretic assumption that $\mathsf{CoNP} \subseteq \mathsf{NP/Poly}$ is not true. We also showed the existence of polynomial kernels for the COALITIONAL MANIPULATION problem for many common voting rules. This shows that the POSSIBLE WINNER problem is a significantly harder problem than the COALITIONAL MANIPULATION problem, although both the problems are NP-complete.

With this, we conclude the winner determination part of the thesis. We now move on to the last part of the thesis which study computational complexity of various form of election control.

Part III

Election Control

In this part of the thesis, we present our results on the computational complexity of various problems in the context of election control. We have the following chapters in this part.

▷ In Chapter 8 – Manipulation with Partial Votes – we show that manipulating an election becomes a much harder problem when the manipulators only have a partial knowledge about the votes of the other voters. Hence, manipulating an election, although feasible in theory, may often be harder in practice.

▷ In Chapter 9 – Manipulation Detection – we initiate the work on detecting possible instances of manipulation behavior in elections. Our work shows that detecting possible instances of manipulation may often be a much easier problem than manipulating the election itself.

▷ In Chapter 10 – Frugal Bribery – we show that the computational problem of bribery in an election remains an intractable problem even with a much weaker notion of bribery which we call frugal bribery. Hence, our results strengthen the intractability results from the literature on bribery.

Chapter 8

Manipulation with Partial Votes

The Coalitional Manipulation problem has been studied extensively in the literature for many voting rules. However, most studies have focused on the complete information setting, wherein the manipulators know the votes of the non-manipulators. While this assumption is reasonable for purposes of showing intractability, it is unrealistic for algorithmic considerations. In most real-world scenarios, it is impractical for the manipulators to have accurate knowledge of all the other votes. In this work, we investigate manipulation with incomplete information. In our framework, the manipulators know a partial order for each voter that is consistent with the true preference of that voter. In this setting, we formulate three natural computational notions of manipulation, namely weak, opportunistic, and strong manipulation. We say that an extension of a partial order is *viable* if there exists a manipulative vote for that extension. We propose the following notions of manipulation when manipulators have incomplete information about the votes of other voters.

1. WEAK MANIPULATION: the manipulators seek to vote in a way that makes their preferred candidate win in *at least one extension* of the partial votes of the non-manipulators.

2. OPPORTUNISTIC MANIPULATION: the manipulators seek to vote in a way that makes their preferred candidate win *in every viable extension* of the partial votes of the non-manipulators.

3. STRONG MANIPULATION: the manipulators seek to vote in a way that makes their pre-
   ferred candidate win *in every extension* of the partial votes of the non-manipulators.

We consider several scenarios for which the traditional manipulation problems are easy (for instance, Borda with a single manipulator). For many of them, the corresponding manipulative questions that we propose turn out to be computationally intractable. Our hardness results often hold even when very little information is missing, or in other words, even when the instances are very close to the complete information setting. Our results show that the impact of paucity of information on the computational complexity of manipulation crucially depends on the notion of manipulation under consideration. Our overall conclusion is that computational hardness continues to be a valid obstruction to manipulation, in the context of a more realistic model.

8.1 Introduction

A central issue in voting is the possibility of *manipulation*. For many voting rules, it turns out that even a single vote, if cast differently, can alter the outcome. In particular, a voter manipulates an election if, by misrepresenting her preference, she obtains an outcome that she prefers over the "honest" outcome. In a cornerstone impossibility result, Gibbard and Satterthwaite [Gib73, Sat75] show that every unanimous and non-dictatorial voting rule with three candidates or more is manipulable. We refer to [BCE$^+$15] for an excellent introduction to various strategic issues in computational social choice theory.

Considering that voting rules are indeed susceptible to manipulation, it is natural to seek ways by which elections can be protected from manipulations. The works of Bartholdi et al. [BTT89, BO91] approach the problem from the perspective of computational intractability. They exploit the possibility that voting rules, despite being vulnerable to manipulation in theory, may be hard to manipulate in practice. Indeed, a manipulator is faced with the following decision problem: given a collection of votes $\mathcal{P}$ and a distinguished candidate c, does there exist a vote v that, when tallied with $\mathcal{P}$, makes c win for a (fixed) voting rule r? The manipulation problem has subsequently been generalized to the problem of COALITIONAL MANIPULATION by Conitzer et al. [CSL07], where one or more manipulators collude together and try to make a distinguished candidate win the election. The manipulation problem, fortunately, turns out to be NP-hard in several settings. This established the success of the approach of demonstrating a computational barrier to manipulation.

However, despite having set out to demonstrate the hardness of manipulation, the initial results in [BTT89] were to the contrary, indicating that many voting rules are in fact easy to manipulate. Moreover, even with multiple manipulators involved, popular voting rules like plurality, veto, k-approval, Bucklin, and Fallback continue to be easy to manipulate [XZP$^+$09]. While we know that the computational intractability may not provide a strong barrier [PR06, PR07, FKN08, XC08a, XC08b, FHH10, Wal10, Wal11b, IKM12, Dey15, DMN15b, DMN16c, DMN15a, DN14, DN15a] even for rules for which the coalitional manipulation problem turns out to be NP-hard, in all other cases the possibility of manipulation is a much more serious concern.

8.1.1 Motivation and Problem Formulation

In our work, we propose to extend the argument of computational intractability to address the cases where the approach appears to fail. We note that most incarnations of the manipulation problem studied so far are in the complete information setting, where the manipulators have complete knowledge of the preferences of the truthful voters. While these assumptions are indeed the best possible for the computationally negative results, we note that they are not reflective of typical real-world scenarios. Indeed, concerns regarding privacy of information, and in other cases, the sheer volume of information, would be significant hurdles for manipulators to obtain complete information. Motivated by this, we consider the manipulation problem in a natural *partial information* setting. In particular, we model the partial information of the manipulators about the votes of the non-manipulators as partial orders over the set of candidates. A partial order over the set of candidates will be called a partial vote. Our results show that several of the voting rules that are easy to manipulate in the complete information setting become intractable when the manipulators know only partial votes. Indeed, for many voting rules, we show that even if the ordering of a small number of pairs of candidates is missing from the profile, manipulation becomes an intractable problem. Our results therefore strengthen the view that manipulation may not be practical if we limit the information the manipulators have at their disposal about the votes of other voters [CWX11].

We introduce three new computational problems that, in a natural way, extend the question of manipulation to the partial information setting. In these problems, the input is a set of partial votes $\mathcal{P}$ corresponding to the votes of the non-manipulators, a non-empty set of manipulators M, and a preferred candidate c. The task in the WEAK MANIPULATION (WM) problem is to determine if there is a way to cast the manipulators' votes such that c wins

the election for at least one extension of the partial votes in $\mathcal{P}$. On the other hand, in the STRONG MANIPULATION (SM) problem, we would like to know if there is a way of casting the manipulators' votes such that c wins the election in *every extension* of the partial votes in $\mathcal{P}$.

We also introduce the problem of OPPORTUNISTIC MANIPULATION (OM), which is an "intermediate" notion of manipulation. Let us call an extension of a partial profile *viable* if it is possible for the manipulators to vote in such a way that the manipulators' desired candidate wins in that extension. In other words, a viable extension is a YES-instance of the standard COALITIONAL MANIPULATION problem. We have an opportunistic manipulation when it is possible for the manipulators to cast a vote which makes c win the election in *all* viable extensions. Note that any YES-instance of STRONG MANIPULATION is also a YES-instance of OPPORTUNISTIC MANIPULATION, but this may not be true in the reverse direction. As a particularly extreme example, consider a partial profile where there are no viable extensions: this would be a NO-instance for STRONG MANIPULATION, but a (vacuous) YES-instance of OPPORTUNISTIC MANIPULATION. The OPPORTUNISTIC MANIPULATION problem allows us to explore a more relaxed notion of manipulation: one where the manipulators are obliged to be successful only in extensions where it is possible to be successful. Note that the goal with STRONG MANIPULATION is to be successful in all extensions, and therefore the only interesting instances are the ones where all extensions are viable.

It is easy to see that YES instance of STRONG MANIPULATION is also a YES instance of OPPORTUNISTIC MANIPULATION and WEAK MANIPULATION. Beyond this, we remark that all the three problems are questions with different goals, and neither of them render the other redundant. We refer the reader to Figure 8.1 for a simple example distinguishing these scenarios.

All the problems above generalize COALITIONAL MANIPULATION, and hence any computational intractability result for COALITIONAL MANIPULATION immediately yields a corresponding intractability result for WEAK MANIPULATION, STRONG MANIPULATION, and OPPORTUNISTIC MANIPULATION under the same setting. For example, it is known that the COALITIONAL MANIPULATION problem is intractable for the maximin voting rule when we have at least two manipulators [XZP^{+}09]. Hence, the WEAK MANIPULATION, STRONG MANIPULATION, and OPPORTUNISTIC MANIPULATION problems are intractable for the maximin voting rule when we have at least two manipulators.
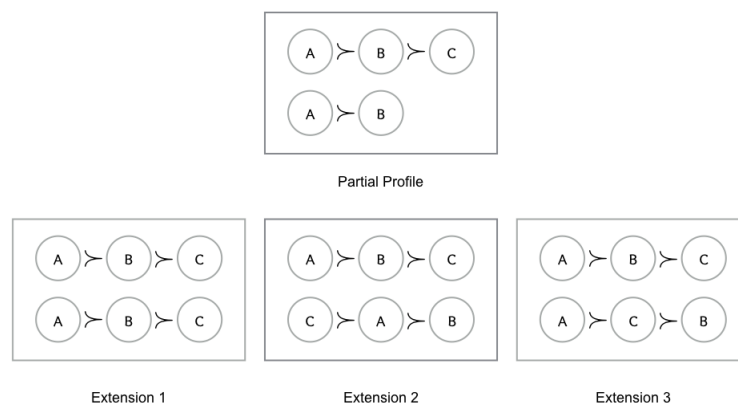
Figure 8.1: An example of a partial profile. Consider the plurality voting rule with one manipulator. If the favorite candidate is A, then the manipulator simply has to place A on the top his vote to make A win in *any* extension. If the favorite candidate is B, there is *no vote* that makes B win in any extension. Finally, if the favorite candidate is C, then with a vote that places C on top, the manipulator can make C win in the only viable extension (Extension 2).

8.1.2 Related Work

A notion of manipulation under partial information has been considered by Conitzer et al. [CWX11]. They focus on whether or not there exists a dominating manipulation and show that this problem is NP-hard for many common voting rules. Given some partial votes, a dominating manipulation is a non-truthful vote that the manipulator can cast which makes the winner at least as preferable (and sometimes more preferable) as the winner when the manipulator votes truthfully. The dominating manipulation problem and the WEAK MANIPU-LATION, OPPORTUNISTIC MANIPULATION, and STRONG MANIPULATION problems do not seem to have any apparent complexity-theoretic connection. For example, the dominating manipulation problem is NP-hard for all the common voting rules except plurality and veto, whereas, the STRONG MANIPULATION problem is easy for most of the cases (see Table 8.1). However, the results in [CWX11] establish the fact that it is indeed possible to make manipulation intractable by restricting the amount of information the manipulators possess about the votes of the other voters. Elkind and Erdélyi [EE12] study manipulation under voting rule uncertainty. However, in our work, the voting rule is fixed and known to the manipulators.

Two closely related problems that have been extensively studied in the context of incomplete votes are POSSIBLE WINNER and NECESSARY WINNER [KL05]. In the POSSIBLE WINNER problem, we are given a set of partial votes $\mathcal{P}$ and a candidate c, and the question is whether

there exists an extension of $\mathcal{P}$ where c wins, while in the NECESSARY WINNER problem, the question is whether c is a winner in every extension of $\mathcal{P}$. Following the work in [KL05], a number of special cases and variants of the POSSIBLE WINNER problem have been studied in the literature [CLMM10, BBF10, BRR11, BRR$^+$12, GNNW14, XC11, DL13a, NW14, BFLR12, ML15]. The flavor of the WEAK MANIPULATION problem is clearly similar to POSSIBLE WINNER. However, we emphasize that there are subtle distinctions between the two problems. A more elaborate comparison is made in the next section.

8.1.3 Our Contribution

Our primary contribution in this work is to propose and study three natural and realistic generalizations of the computational problem of manipulation in the incomplete information setting. We summarize the complexity results in this work in Table 8.1. Our results provides the following interesting insights on the impact of lack of information on the computational difficulty of manipulation. We note that the number of undetermined pairs of candidates per vote are small constants in all our hardness results.

▷ We observe that the computational problem of manipulation for the plurality and veto voting rules remains polynomial time solvable even with lack of information, irrespective of the notion of manipulation under consideration [Proposition 8.1, Theorem 8.11 and 8.15, and observation 8.4]. We note that the plurality and veto voting rule also remain vulnerable under the notion of dominating manipulation [CWX11].

▷ The impact of absence of information on the computational complexity of manipulation is more dynamic for the k-approval, k-veto, Bucklin, Borda, and maximin voting rules. Only the WEAK MANIPULATION and OPPORTUNISTIC MANIPULATION problems are computationally intractable for the k-approval [Theorem 8.1 and 8.5], k-veto [Theorem 8.2 and 8.6], Bucklin [Theorem 8.3 and 8.10], Borda [Observation 8.3 and Theorem 8.7], and maximin [Observation 8.3 and Theorem 8.8] voting rules, whereas the STRONG MANIPULATION problem remains computationally tractable [Theorem 8.11 to 8.14].

▷ Table 8.1 shows an interesting behavior of the fallback voting rule. The Fallback voting rule is the only voting rule among the voting rules we study here for which the WEAK MANIPULATION problem is NP-hard [Theorem 8.3] but both the OPPORTUNISTIC MANIPULATION and STRONG MANIPULATION problems are polynomial time solvable

[Theorem 8.13 and observation 8.4]. This is because the OPPORTUNISTIC MANIPULATION problem can be solved for the fallback voting rule by simply making manipulators vote for their desired candidate.

▷ Our results show that absence of information makes all the three notions of manipulations intractable for the Copeland$^\alpha$ voting rule for every rational $\alpha \in [0, 1] \setminus \{0.5\}$ for the WEAK MANIPULATION problem [Observation 8.3] and for every $\alpha \in [0, 1]$ for the OPPORTUNISTIC MANIPULATION and STRONG MANIPULATION problems [Theorem 8.4 and 8.9].

Our results (see Table 8.1) show that whether lack of information makes the manipulation problems harder, crucially depends on the notion of manipulation applicable to the situation under consideration. All the three notions of manipulations are, in our view, natural extension of manipulation to the incomplete information setting and tries to capture different behaviors of manipulators. For example, the WEAK MANIPULATION problem may be applicable to an optimistic manipulator whereas for an pessimistic manipulator, the STRONG MANIPULATION problem may make more sense.

8.1.4 Problem Definitions

We now formally define the three problems that we consider in this work, namely WEAK MANIPULATION, OPPORTUNISTIC MANIPULATION, and STRONG MANIPULATION. Let r be a fixed voting rule. We first introduce the WEAK MANIPULATION problem.

Definition 8.1. r-**WEAK MANIPULATION**
Given a set of partial votes $\mathcal{P}$ over a set of candidates $\mathcal{C}$, a positive integer ℓ (> 0) denoting the number of manipulators, and a candidate c, do there exist votes $\succ_1, \ldots, \succ_\ell \in \mathcal{L}(\mathcal{C})$ such that there exists an extension $\succ \in \mathcal{L}(\mathcal{C})^{|\mathcal{P}|}$ of $\mathcal{P}$ with $r(\succ, \succ_1, \ldots, \succ_\ell) = c$?

To define the OPPORTUNISTIC MANIPULATION problem, we first introduce the notion of an (r, c)-opportunistic voting profile, where r is a voting rule and c is any particular candidate.

Definition 8.2. (r, c)-*Opportunistic Voting Profile*
Let ℓ be the number of manipulators and $\mathcal{P}$ a set of partial votes. An ℓ-voter profile $(\succ_i)_{i \in [\ell]} \in \mathcal{L}(\mathcal{C})^\ell$ is called an (r, c)-opportunistic voting profile if for each extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ for which there exists an ℓ-vote profile $(\succ'_i)_{i \in [\ell]} \in \mathcal{L}(\mathcal{C})^\ell$ with $r\left(\overline{\mathcal{P}} \cup (\succ'_i)_{i \in [\ell]}\right) = c$, we have $r\left(\overline{\mathcal{P}} \cup (\succ_i)_{i \in [\ell]}\right) = c$.

	WM, $\ell=1$	WM	OM, $\ell=1$	OM	SM, $\ell=1$	SM
Plurality	P	P	P	P	P	P
Veto	P	P	P	P	P	P
k-Approval	NP-complete	NP-complete	coNP-hard	coNP-hard	P	P
k-Veto	NP-complete	NP-complete	coNP-hard	coNP-hard	P	P
Bucklin	NP-complete	NP-complete	coNP-hard	coNP-hard	P	P
Fallback	NP-complete	NP-complete	P	P	P	P
Borda	NP-complete	NP-complete	coNP-hard	coNP-hard	P	NP-hard
maximin	NP-complete	NP-complete	coNP-hard	coNP-hard	P	NP-hard
Copeland$^\alpha$	NP-complete	NP-complete	coNP-hard	coNP-hard	coNP-hard	NP-hard

Table 8.1: Summary of Results (ℓ denotes the number of manipulators). The results in white follow immediately from the literature (Observation 8.1 to 8.3). Our results for the Copeland$^\alpha$ voting rule hold for every rational $\alpha \in [0,1] \setminus \{0.5\}$ for the WEAK MANIPULATION problem and for every $\alpha \in [0,1]$ for the OPPORTUNISTIC MANIPULATION and STRONG MANIPULATION problems.

In other words, an ℓ-vote profile is (r, c)-opportunistic with respect to a partial profile if, when put together with the truthful votes of any extension, c wins if the extension is viable to begin with. We are now ready to define the OPPORTUNISTIC MANIPULATION problem.

Definition 8.3. r-OPPORTUNISTIC MANIPULATION

Given a set of partial votes $\mathcal{P}$ over a set of candidates $\mathcal{C}$, a positive integer ℓ (> 0) denoting the number of manipulators, and a candidate c, does there exist an (r, c)-opportunistic ℓ-vote profile?

We finally define the STRONG MANIPULATION problem.

Definition 8.4. r-STRONG MANIPULATION

Given a set of partial votes $\mathcal{P}$ over a set of candidates $\mathcal{C}$, a positive integer ℓ (> 0) denoting the number of manipulators, and a candidate c, do there exist votes $(\succ_i)_{i \in \ell} \in \mathcal{L}(\mathcal{C})^\ell$ such that for every extension $\succ \in \mathcal{L}(\mathcal{C})^{|\mathcal{P}|}$ of $\mathcal{P}$, we have $r(\succ, (\succ_i)_{i \in [\ell]}) = c$?

We use $(\mathcal{P}, \ell, c)$ to denote instances of WEAK MANIPULATION, OPPORTUNISTIC MANIPULATION, and STRONG MANIPULATION, where $\mathcal{P}$ denotes a profile of partial votes, ℓ denotes the number of manipulators, and c denotes the desired winner.

For the sake of completeness, we provide the definitions of the COALITIONAL MANIPULATION and POSSIBLE WINNER problems below.

Definition 8.5. r-COALITIONAL MANIPULATION

Given a set of complete votes $\succ$ over a set of candidates $\mathcal{C}$, a positive integer ℓ (> 0) denoting the number of manipulators, and a candidate c, do there exist votes $(\succ_i)_{i \in \ell} \in \mathcal{L}(\mathcal{C})^\ell$ such that $r\left(\succ, (\succ_i)_{i \in [\ell]}\right) = c$?

Definition 8.6. r-POSSIBLE WINNER

Given a set of partial votes $\mathcal{P}$ and a candidate c, does there exist an extension $\succ$ of the partial votes in $\mathcal{P}$ to linear votes such that $r(\succ) = c$?

8.1.5 Comparison with Possible Winner and Coalitional Manipulation Problems

For any fixed voting rule, the WEAK MANIPULATION problem with ℓ manipulators reduces to the POSSIBLE WINNER problem. This is achieved by simply using the same set as truthful votes and introducing ℓ empty votes. We summarize this in the observation below.

Observation 8.1. *The* WEAK MANIPULATION *problem many-to-one reduces to the* POSSIBLE WINNER *problem for every voting rule.*

Proof. Let $(\mathcal{P}, \ell, c)$ be an instance of WEAK MANIPULATION. Let $\mathcal{Q}$ be the set consisting of ℓ many copies of partial votes $\{\emptyset\}$. Clearly the WEAK MANIPULATION instance $(\mathcal{P}, \ell, c)$ is equivalent to the POSSIBLE WINNER instance $(\mathcal{P} \cup \mathcal{Q})$. □

However, whether the POSSIBLE WINNER problem reduces to the WEAK MANIPULATION problem or not is not clear since in any WEAK MANIPULATION problem instance, there must exist at least one manipulator and a POSSIBLE WINNER instance may have no empty vote. From a technical point of view, the difference between the WEAK MANIPULATION and POSSIBLE WINNER problems may look marginal; however we believe that the WEAK MANIPULATION problem is a very natural generalization of the COALITIONAL MANIPULATION problem in the partial information setting and thus worth studying. Similarly, it is easy to show, that the COALITIONAL MANIPULATION problem with ℓ manipulators reduces to WEAK MANIPULATION, OPPORTUNISTIC MANIPULATION, and STRONG MANIPULATION problems with ℓ manipulators, since the former is a special case of the latter ones.

Observation 8.2. *The* COALITIONAL MANIPULATION *problem with ℓ manipulators many-to-one reduces to* WEAK MANIPULATION, OPPORTUNISTIC MANIPULATION, *and* STRONG MANIPULATION *problems with ℓ manipulators for all voting rules and for all positive integers ℓ.*

Proof. Follows from the fact that every instance of the COALITIONAL MANIPULATION problem is also an equivalent instance of the WEAK MANIPULATION, OPPORTUNISTIC MANIPULATION, and STRONG MANIPULATION problems. □

Finally, we note that the COALITIONAL MANIPULATION problem with ℓ manipulators can be reduced to the WEAK MANIPULATION problem with just one manipulator, by introducing $\ell - 1$ empty votes. These votes can be used to witness a good extension in the forward direction. In the reverse direction, given an extension where the manipulator is successful, the extension can be used as the manipulator's votes. This argument leads to the following observation.

Observation 8.3. *The* COALITIONAL MANIPULATION *problem with ℓ manipulators many-to-one reduces to the* WEAK MANIPULATION *problem with one manipulator for every voting rule and for every positive integer ℓ.*

Proof. Let $(\mathcal{P}, \ell, c)$ be an instance of COALITIONAL MANIPULATION. Let $\mathcal{Q}$ be the set of consisting of $\ell - 1$ many copies of partial vote $\{c \succ \text{others}\}$. Clearly the WEAK MANIPULATION instance $(\mathcal{P} \cup \mathcal{Q}, 1, c)$ is equivalent to the COALITIONAL MANIPULATION instance $(\mathcal{P}, \ell, 1)$. $\qquad\square$

This observation can be used to derive the hardness of WEAK MANIPULATION even for one manipulator whenever the hardness for COALITIONAL MANIPULATION is known for any fixed number of manipulators (for instance, this is the case for the voting rules such as Borda, maximin and Copeland). However, determining the complexity of WEAK MANIPULATION with one manipulator requires further work for voting rules where COALITIONAL MANIPULATION is polynomially solvable for any number of manipulators (such as k-approval, Plurality, Bucklin, and so on).

8.2 Hardness Results for WEAK MANIPULATION, OPPORTUNISTIC MANIPULATION, and STRONG MANIPULATION Problems

In this section, we present our hardness results. While some of our reductions are from the POSSIBLE WINNER problem, the other reductions in this section are from the EXACT COVER BY 3-SETS problem, also referred to as X3C. This is a well-known NP-complete [GJ79] problem, and is defined as follows.

Definition 8.7 (Exact Cover by 3-Sets (X3C))**.** *Given a set* $\mathcal{U}$ *and a collection* $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$ *of* t *subsets of* $\mathcal{U}$ *with* $|S_i| = 3 \; \forall i = 1, \ldots, t$, *does there exist a* $\mathcal{T} \subset \mathcal{S}$ *with* $|\mathcal{T}| = \frac{|\mathcal{U}|}{3}$ *such that* $\cup_{X \in \mathcal{T}} X = \mathcal{U}$?

We use $\overline{\text{X3C}}$ to refer to the complement of X3C, which is to say that an instance of $\overline{\text{X3C}}$ is a YES instance if and only if it is a NO instance of X3C. The rest of this section is organized according to the problems being addressed.

8.2.1 Weak Manipulation Problem

To begin with, recall that the COALITIONAL MANIPULATION problem is NP-complete for the Borda [DKNW11, BNW11], maximin [XZP+09], and Copeland$^\alpha$ [FHS08, FHHR09, FHS10] voting rules for every rational $\alpha \in [0, 1] \backslash \{0.5\}$, when we have two manipulators. Therefore, it

follows from Observation 8.3 that the WEAK MANIPULATION problem is NP-complete for the Borda, maximin, and Copeland$^\alpha$ voting rules for every rational $\alpha \in [0, 1] \setminus \{0.5\}$, even with one manipulator.

For the k-approval and k-veto voting rules, we reduce from the corresponding POSSIBLE WINNER problems. While it is natural to start from the same voting profile, the main challenge is in undoing the advantage that the favorite candidate receives from the manipulator's vote, in the reverse direction.

8.2.1.1 Result for the k-Approval Voting Rule

We begin with proving that the WEAK MANIPULATION problem is NP-complete for the k-approval voting rule even with one manipulator and at most 4 undetermined pairs per vote.

Theorem 8.1. *The* WEAK MANIPULATION *problem is* NP-*complete for the* k-*approval voting rule even with one manipulator for any constant* $k > 1$, *even when the number of undetermined pairs in each vote is no more than* 4.

Proof. For simplicity of presentation, we prove the theorem for 2-approval. We reduce from the POSSIBLE WINNER problem for 2-approval which is NP-complete [XC11], even when the number of undetermined pairs in each vote is no more than 4. Let $\mathcal{P}$ be the set of partial votes in a POSSIBLE WINNER instance, and let $\mathcal{C} = \{c_1, \ldots, c_m, c\}$ be the set of candidates, where the goal is to check if there is an extension of $\mathcal{P}$ that makes c win. For developing the instance of WEAK MANIPULATION, we need to "reverse" any advantage that the candidate c obtains from the vote of the manipulator. Notice that the most that the manipulator can do is to increase the score of c by one. Therefore, in our construction, we *"artificially"* increase the score of all the other candidates by one, so that despite of the manipulator's vote, c will win the new election if and only if it was a possible winner in the POSSIBLE WINNER instance. To this end, we introduce $(m + 1)$ many *dummy* candidates $d_1, \ldots, d_{m+1}$ and the complete votes:

$$w_i = c_i \succ d_i \succ \text{others, for every } i \in \{1, \ldots, m\}$$

Further, we extend the given partial votes of the POSSIBLE WINNER instance to force the dummy candidates to be preferred least over the rest - by defining, for every $v_i \in \mathcal{P}$, the corresponding partial vote v_i' as follows.

$$v_i' = v_i \cup \{\mathcal{C} \succ \{d_1, \ldots, d_{m+1}\}\}.$$

179

This ensures that all the dummy candidates do not receive any score from the modified partial votes corresponding to the partial votes of the POSSIBLE WINNER instance. Notice that since the number of undetermined pairs in v_i is no more than 4, the number of undetermined pairs in v_i' is also no more than 4. Let $(\mathcal{C}', \mathcal{Q}, c)$ denote this constructed WEAK MANIPULATION instance. We claim that the two instances are equivalent.

In the forward direction, suppose c is a possible winner with respect to $\mathcal{P}$, and let $\overline{\mathcal{P}}$ be an extension where c wins. Then it is easy to see that the manipulator can make c win in some extension by placing c and d_{m+1} in the first two positions of her vote (note that the partial score of d_{m+1} is zero in $\mathcal{Q}$). Indeed, consider the extension of $\mathcal{Q}$ obtained by mimicking the extension $\overline{\mathcal{P}}$ on the "common" partial votes, $\{v_i' \mid v_i \in \mathcal{P}\}$. Notice that this is well-defined since v_i and v_i' have exactly the same set of incomparable pairs. In this extension, the score of c is strictly greater than the scores of all the other candidates, since the scores of all candidates in $\mathcal{C}$ is exactly one more than their scores in $\mathcal{P}$, and all the dummy candidates have a score of at most one.

In the reverse direction, notice that the manipulator puts the candidates c and d_{m+1} in the top two positions without loss of generality. Now suppose the manipulator's vote $c \succ d_{m+1} \succ$ others makes c win the election for an extension $\overline{\mathcal{Q}}$ of $\mathcal{Q}$. Then consider the extension $\overline{\mathcal{P}}$ obtained by restricting $\overline{\mathcal{Q}}$ to $\mathcal{C}$. Notice that the score of each candidate in $\mathcal{C}$ in this extension is one less than their scores in $\mathcal{Q}$. Therefore, the candidate c wins this election as well, concluding the proof.

The above proof can be imitated for any other constant values of k by reducing it from the POSSIBLE WINNER problem for k-approval and introducing $(m+1)(k-1)$ dummy candidates. □

8.2.1.2 Result for the k-Veto Voting Rule

Theorem 8.2. *The* WEAK MANIPULATION *problem for the k-veto voting rule is* NP-*complete even with one manipulator for any constant* $k > 1$.

Proof. We reduce from the POSSIBLE WINNER problem for the k-veto voting rule which is known to be NP-complete [BD09]. Let $\mathcal{P}$ be the set of partial votes in a POSSIBLE WINNER problem instance, and let $\mathcal{C} = \{c_1, \ldots, c_m, c\}$ be the set of candidates, where the goal is to check if there is an extension that makes c win with respect to k-veto. We assume without loss of generality that c's position is fixed in all the partial votes (if not, then we fix the position of c as high as possible in every vote).

We introduce $k+1$ many *dummy* candidates $d_1, \ldots, d_k, d$. The role of the first k dummy candidates is to ensure that the manipulator is forced to place them at the "bottom k" positions of her vote, so that all the original candidates get the same score from the additional vote of the manipulator. The most natural way of achieving this is to ensure that the dummy candidates have the same score as c in any extension (note that we know the score of c since c's position is fixed in all the partial votes). This would force the manipulator to place these k candidates in the last k positions. Indeed, doing anything else will cause these candidates to tie with c, even when there is an extension of $\mathcal{P}$ that makes c win.

To this end, we begin by placing the dummy candidates in the top k positions in all the partial votes. Formally, we modify every partial vote as follows:

$$w = d_i \succ \text{others}, \text{ for every } i \in \{1, \ldots, k\}$$

At this point, we know the scores of c and d_i, for every $i \in \{1, \ldots, k\}$. Using Lemma 7.1, we add complete votes such that the final score of c is the same with the score of every d_i and the score of c is strictly more than the score of d. The relative score of every other candidate remains the same. This completes the description of the construction. We denote the augmented set of partial votes by $\overline{\mathcal{P}}$.

We now argue the correctness. In the forward direction, if there is an extension of the votes that makes c win, then we repeat this extension, and the vote of the manipulator puts the candidate d_i at the position $m+i+2$; and all the other candidates in an arbitrary fashion. Formally, we let the manipulator's vote be:

$$\mathfrak{v} = c \succ c_1 \succ \cdots \succ c_m \succ d \succ d_1 \succ \cdots \succ d_k.$$

By construction c wins the election in this particular setup. In the reverse direction, consider a vote of the manipulator and an extension $\overline{\mathcal{Q}}$ of $\overline{\mathcal{P}}$ in which c wins. Note that the manipulator's vote necessarily places the candidates d_i in the bottom k positions — indeed, if not, then c cannot win the election by construction. We extend a partial vote $w \in \mathcal{P}$ by mimicking the extension of the corresponding partial vote $w' \in \overline{\mathcal{P}}$, that is, we simply project the extension of w' on the original set of candidates $\mathcal{C}$. Let $\mathcal{Q}$ denote this proposed extension of $\mathcal{P}$. We claim that c wins the election given by $\mathcal{Q}$. Indeed, suppose not. Let c_i be a candidate whose score is at least the score of c in the extension $\mathcal{Q}$. Note that the scores of c_i and c in the extension $\overline{\mathcal{Q}}$ are exactly the same as their scores in $\mathcal{Q}$, except for a constant offset — importantly, their

scores are offset by the same amount. This implies that the score of c_i is at least the score of c in $\overline{Q}$ as well, which is a contradiction. Hence, the two instances are equivalent. $\qquad\square$

8.2.1.3 Result for the Bucklin Voting Rule

We next prove, by a reduction from X3C, that the WEAK MANIPULATION problem for the Bucklin and simplified Bucklin voting rules is NP-complete even with one manipulator and at most 16 undetermined pairs per vote.

Theorem 8.3. *The* WEAK MANIPULATION *problem is* NP-*complete for Bucklin, simplified Bucklin, Fallback, and simplified Fallback voting rules, even when we have only one manipulator and the number of undetermined pairs in each vote is no more than* 16.

Proof. We reduce the X3C problem to WEAK MANIPULATION for simplified Bucklin. Let $(\mathcal{U} = \{u_1, \ldots, u_m\}, \mathcal{S} := \{S_1, S_2, \ldots, S_t\})$ be an instance of X3C, where each S_i is a subset of $\mathcal{U}$ of size three. We construct a WEAK MANIPULATION instance based on $(\mathcal{U}, \mathcal{S})$ as follows.

Candidate set: $\mathcal{C} = \mathcal{W} \cup \mathcal{X} \cup \mathcal{D} \cup \mathcal{U} \cup \{c, w, a, b\}$, where $|\mathcal{W}| = m - 3, |\mathcal{X}| = 4, |\mathcal{D}| = m + 1$

We first introduce the following partial votes $\mathcal{P}$ in correspondence with the sets in the family as follows.

$$\mathcal{W} \succ \mathcal{X} \succ S_i \succ c \succ (\mathcal{U} \setminus S_i) \succ \mathcal{D} \setminus (\{\mathcal{X} \times (\{c\} \cup S_i)\}), \forall i \leqslant t$$

Notice that the number of undetermined pairs in every vote in $\mathcal{P}$ is 16. We introduce the following additional complete votes $\mathcal{Q}$:

▷ t copies of $\mathcal{U} \succ c \succ$ others

▷ $m/3 - 1$ copies of $\mathcal{U} \succ a \succ c \succ$ others

▷ $m/3 + 1$ copies of $\mathcal{D} \succ b \succ$ others

The total number of voters, including the manipulator, is $2t + 2m/3 + 1$. Now we show equivalence of the two instances.

In the forward direction, suppose we have an exact set cover $\mathcal{T} \subset \mathcal{S}$. Let the vote of the manipulator $\mathfrak{v}$ be $c \succ D \succ$ others. We consider the following extension $\overline{\mathcal{P}}$ of $\mathcal{P}$.

$$\mathcal{W} \succ S_i \succ c \succ \mathcal{X} \succ (\mathcal{U} \setminus S_i) \succ \mathcal{D}$$

On the other hand, if $S_i \in \mathcal{S} \setminus \mathcal{T}$, then we have:

$$\mathcal{W} \succ \mathcal{X} \succ S_i \succ c \succ (\mathcal{U} \setminus S_i) \succ \mathcal{D}$$

We claim that c is the unique simplified Bucklin winner in the profile $(\overline{\mathcal{P}}, \mathcal{W}, \mathfrak{v})$. Notice that the simplified Bucklin score of c is $m + 1$ in this extension, since it appears in the top $m + 1$ positions in the $m/3$ votes corresponding to the set cover, t votes from the complete profile $\mathcal{Q}$ and one vote $\mathfrak{v}$ of the manipulator. For any other candidate $u_i \in \mathcal{U}$, u_i appears in the top $m + 1$ positions once in $\overline{\mathcal{P}}$ and $t + \frac{m}{3} - 1$ times in $\mathcal{Q}$. Thus, u_i does not get majority in top $m + 1$ positions making its simplified Bucklin score at least $m + 2$. Hence, c is the unique simplified Bucklin winner in the profile $(\overline{\mathcal{P}}, \mathcal{W}, \mathfrak{v})$. Similarly, the candidate w_1 appears only t times in the top $m + 1$ positions. The same can be argued for the remaining candidates in $\mathcal{D}, \mathcal{W}$, and w.

In the reverse direction, suppose the WEAK MANIPULATION is a YES instance. We may assume without loss of generality that the manipulator's vote $\mathfrak{v}$ is $c \succ \mathcal{D} \succ$ others, since the simplified Bucklin score of the candidates in $\mathcal{D}$ is at least $2m$. Let $\overline{\mathcal{P}}$ be the extension of $\mathcal{P}$ such that c is the unique winner in the profile $(\overline{\mathcal{P}}, \mathcal{Q}, \mathfrak{v})$. As w_1 is ranked within top $m + 2$ positions in $t + \frac{m}{3} + 1$ votes in $\mathcal{Q}$, for c to win, $c \succ w_{m-2}$ must hold in at least $\frac{m}{3}$ votes in $\overline{\mathcal{P}}$. In those votes, all the candidates in S_i are also within top $m + 2$ positions. Now if any candidate in $\mathcal{U}$ is within top $m + 1$ positions in $\overline{\mathcal{P}}$ more than once, then c will not be the unique winner. Hence, the S_i's corresponding to the votes where $c \succ w_{m-2}$ in $\overline{\mathcal{P}}$ form an exact set cover.

The reduction above also works for the Bucklin voting rule. Specifically, the argument for the forward direction is exactly the same as the simplified Bucklin above and the argument for the reverse direction is as follows. The candidate w_1 is ranked within top $m + 2$ positions in $t + \frac{m}{3} + 1$ votes in $\mathcal{Q}$ and c is never placed within top $m + 2$ positions in any vote in $\mathcal{Q}$. Hence, for c to win, $c \succ w_{m-2}$ must hold in at least $\frac{m}{3}$ votes in $\overline{\mathcal{P}}$. In those votes, all the candidates in S_i are also within top m positions. Notice that c never gets placed within top m positions in any vote in $(\overline{\mathcal{P}}, \mathcal{Q})$. Now if any candidate $x \in \mathcal{U}$ is within top m positions in $\overline{\mathcal{P}}$ more than once, then x gets majority within top m positions and thus c cannot win.

The result for the Fallback and simplified Fallback voting rules follow from the corresponding results for the Bucklin and simplified Bucklin voting rules respectively since every Bucklin and simplified Bucklin election is also a Fallback and simplified Fallback election respectively. $\qquad\square$

8.2.2 Strong Manipulation Problem

We know that the COALITIONAL MANIPULATION problem is NP-complete for the Borda, maximin, and Copeland$^\alpha$ voting rules for every rational $\alpha \in [0,1] \setminus \{0.5\}$, when we have two manipulators. Thus, it follows from Observation 8.2 that STRONG MANIPULATION is NP-hard for Borda, maximin, and Copeland$^\alpha$ voting rules for every rational $\alpha \in [0,1] \setminus \{0.5\}$ for at least two manipulators.

For the case of one manipulator, STRONG MANIPULATION turns out to be polynomial-time solvable for most other voting rules. For Copeland$^\alpha$, however, we show that the problem is co-NP-hard for every $\alpha \in [0,1]$ for a single manipulator, even when the number of undetermined pairs in each vote is bounded by a constant. This is achieved by a careful reduction from $\overline{\text{X3C}}$.

We have following intractability result for the STRONG MANIPULATION problem for the Copeland$^\alpha$ rule with one manipulator and at most 10 undetermined pairs per vote.

Theorem 8.4. STRONG MANIPULATION *is* co-NP-*hard for Copeland$^\alpha$ voting rule for every* $\alpha \in [0,1]$ *even when we have only one manipulator and the number of undetermined pairs in each vote is no more than* 10.

Proof. We reduce X3C to STRONG MANIPULATION for Copeland$^\alpha$ rule. Let $(\mathcal{U} = \{u_1, \ldots, u_m\}, \mathcal{S} = \{S_1, S_2, \ldots, S_t\})$ is an X3C instance. We assume, without loss of generality, t to be an even integer (if not, replicate any set from $\mathcal{S}$). We construct a corresponding WEAK MANIPULATION instance for Copeland$^\alpha$ as follows.

$$\text{Candidate set } \mathcal{C} = \mathcal{U} \cup \{c, w, z, d\}$$

Partial votes $\mathcal{P}$:

$$\forall i \leqslant t, (\mathcal{U} \setminus S_i) \succ c \succ z \succ d \succ S_i \succ w \setminus \{\{z, c\} \times (S_i \cup \{d, w\})\}$$

Notice that the number of undetermined pairs in every vote in $\mathcal{P}$ is 10. Now we add a set $\mathcal{Q}$ of complete votes with $|\mathcal{Q}|$ even and $|\mathcal{Q}| = \text{poly}(m, t)$ using Lemma 7.2 to achieve the following margin of victories in pairwise elections.

$\triangleright$ $D_\mathcal{Q}(d, z) = D_\mathcal{Q}(z, c) = D_\mathcal{Q}(c, d) = D_\mathcal{Q}(w, z) = 4t$

$\triangleright$ $D_\mathcal{Q}(u_i, d) = D_\mathcal{Q}(c, u_i) = 4t \; \forall u_i \in \mathcal{U}$

$\triangleright$ $D_\mathcal{Q}(z, u_i) = t \; \forall u_i \in \mathcal{U}$

▷ $D_Q(c, w) = t - \frac{2q}{3} - 2$

▷ $D_Q(u_i, u_{i+1 \pmod{*} q}) = 4t \; \forall u_i \in \mathcal{U}$

▷ $D_Q(a, b) = 0$ for every $a, b \in \mathcal{C}, a \neq b$, not mentioned above

We have only one manipulator who tries to make c winner. Notice that the number of votes in the STRONG MANIPULATION instance $(\mathcal{P} \cup \mathcal{Q}, 1, c)$ including the manipulator's vote is odd (since $|\mathcal{P}|$ and $|\mathcal{Q}|$ are even integers). Therefore, $D_{\mathcal{P}^* \cup \mathcal{Q} \cup \{v^*\}}(a, b)$ is never zero for every $a, b \in \mathcal{C}, a \neq b$ in every extension $\mathcal{P}^*$ of $\mathcal{P}$ and manipulators vote v^* and consequently the particular value of α does not play any role in this reduction. Hence, we assume, without loss of generality, α to be zero from here on and simply use the term Copeland instead of Copeland$^\alpha$.

Now we show that the X3C instance $(\mathcal{U}, \mathcal{S})$ is a YES instance if and only if the STRONG MANIPULATION instance $(\mathcal{P} \cup \mathcal{Q}, 1, c)$ is a NO instance (a STRONG MANIPULATION instance is a NO instance if there does not exist a vote of the manipulator which makes c the unique winner in every extension of the partial votes). We can assume without loss of generality that manipulator puts c at first position and z at last position in her vote v.

Assume that the X3C instance is a YES instance. Suppose (by renaming) that $S_1, \ldots, S_{\frac{m}{3}}$ forms an exact set cover. We claim that the following extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ makes both z and c Copeland co-winners.

Extension $\overline{\mathcal{P}}$ of $\mathcal{P}$:

$$i \leqslant \frac{m}{3}, (\mathcal{U} \setminus S_i) \succ c \succ z \succ d \succ S_i \succ w$$

$$i \geqslant \frac{m}{3} + 1, (\mathcal{U} \setminus S_i) \succ d \succ S_i \succ w \succ c \succ z$$

We have summarize the pairwise margins between z and c and the rest of the candidates from the profile $(\overline{\mathcal{P}} \cup \mathcal{Q} \cup v)$ in Table 8.2. The candidates z and c are the co-winners with Copeland score $(m + 1)$.

$\mathcal{C} \setminus \{z\}$	$D_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup v}(z, \cdot)$		$\mathcal{C} \setminus \{c\}$	$D_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup v}(c, \cdot)$
c	$\geqslant 3t$		$z, u_i \in \mathcal{U}$	$\geqslant 3t$
w, d	$\leqslant -3t$		w	-1
$u_i \in \mathcal{U}$	1		d	$\leqslant -3t$

Table 8.2: $D_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup v}(z, \cdot)$ and $D_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup v}(c, \cdot)$

For the other direction, notice that Copeland score of c is at least $m + 1$ since c defeats d and every candidate in $\mathcal{U}$ in every extension of $\mathcal{P}$. Also notice that the Copeland score of z can be at most $m + 1$ since z loses to w and d in every extension of $\mathcal{P}$. Hence the only way c cannot be the unique winner is that z defeats all candidates in $\mathcal{U}$ and w defeats c.

This requires $w \succ c$ in at least $t - \frac{m}{3}$ extensions of $\mathcal{P}$. We claim that the sets S_i in the remaining of the extensions where $c \succ w$ forms an exact set cover for $(\mathcal{U}, \mathcal{S})$. Indeed, otherwise some candidate $u_i \in \mathcal{U}$ is not covered. Then, notice that $u_i \succ z$ in all t votes, making $D(z, u_i) = -1$. □

8.2.3 Opportunistic Manipulation Problem

All our reductions for the co-NP-hardness for OPPORTUNISTIC MANIPULATION start from $\overline{\text{X3C}}$. We note that all our hardness results hold even when there is only one manipulator. Our overall approach is the following. We engineer a set of partial votes in such a way that the manipulator is forced to vote in a limited number of ways to have any hope of making her favorite candidate win. For each such vote, we demonstrate a viable extension where the vote fails to make the candidate a winner, leading to a NO instance of OPPORTUNISTIC MANIPULATION. These extensions rely on the existence of an exact cover. On the other hand, we show that if there is no exact set cover, then there is no viable extension, thereby leading to an instance that is vacuously a YES instance of OPPORTUNISTIC MANIPULATION.

8.2.3.1 Result for the k-Approval Voting Rule

Our first result on OPPORTUNISTIC MANIPULATION shows that the OPPORTUNISTIC MANIPULATION problem is co-NP-hard for the k-approval voting rule for constant $k \geqslant 3$ even when the number of manipulators is one and the number of undetermined pairs in each vote is no more than 15.

Theorem 8.5. *The* OPPORTUNISTIC MANIPULATION *problem is* co-NP-*hard for the* k-*approval voting rule for constant* $k \geqslant 3$ *even when the number of manipulators is one and the number of undetermined pairs in each vote is no more than* 15.

Proof. We reduce $\overline{\text{X3C}}$ to OPPORTUNISTIC MANIPULATION for k-approval rule. Let $(\mathcal{U} = \{u_1, \ldots, u_m\}, \mathcal{S} = \{S_1, S_2, \ldots, S_t\})$ is an $\overline{\text{X3C}}$ instance. We construct a corresponding OPPORTUNISTIC MANIPULATION instance for k-approval voting rule as follows. We begin by introducing a candidate for every element of the universe, along with $k - 3$ dummy candidates

(denoted by $\mathcal{W}$), and special candidates $\{c, z_1, z_2, d, x, y\}$. Formally, we have:

$$\text{Candidate set } \mathcal{C} = \mathcal{U} \cup \{c, z_1, z_2, d, x, y\} \cup \mathcal{W}.$$

Now, for every set S_i in the universe, we define the following total order on the candidate set, which we denote by $\mathcal{P}'_i$:

$$\mathcal{W} \succ S_i \succ y \succ z_1 \succ z_2 \succ x \succ (\mathcal{U} \setminus S_i) \succ c \succ d$$

Using $\mathcal{P}'_i$, we define the partial vote $\mathcal{P}_i$ as follows:

$$\mathcal{P}_i = \mathcal{P}'_i \setminus (\{\{y, x, z_1, z_2\} \times S_i\} \cup \{(z_1, z_2), (x, z_1), (x, z_2)\}).$$

We denote the set of partial votes $\{\mathcal{P}_i : i \in [t]\}$ by $\mathcal{P}$ and $\{\mathcal{P}'_i : i \in [t]\}$ by $\mathcal{P}'$. We remark that the number of undetermined pairs in each partial vote $\mathcal{P}_i$ is 15.

We now invoke Lemma 1 from [DMN16c], which allows to achieve any pre-defined scores on the candidates using only polynomially many additional votes. Using this, we add a set $\mathcal{Q}$ of complete votes with $|\mathcal{Q}| = \text{poly}(m, t)$ to ensure the following scores, where we denote the k-approval score of a candidate from a set of votes $\mathcal{V}$ by $s_{\mathcal{V}}(\cdot)$: $s_{\mathcal{Q}}(z_1) = s_{\mathcal{Q}}(z_2) = s_{\mathcal{Q}}(y) = s_{\mathcal{Q}}(c) - m/3; s_{\mathcal{Q}}(d), s_{\mathcal{Q}}(w) \leqslant s_{\mathcal{Q}}(c) - 2t \ \forall w \in \mathcal{W}; s_{\mathcal{Q}}(x) = s_{\mathcal{Q}}(c) - 1; s_{\mathcal{P}' \cup \mathcal{Q}}(u_j) = s_{\mathcal{Q}}(c) + 1 \ \forall j \in [m]$.

Our reduced instance is $(\mathcal{P} \cup \mathcal{Q}, 1, c)$. The reasoning for this score configuration will be apparent as we argue the equivalence. We first argue that if we had a YES instance of $\overline{\text{X3C}}$ (in other words, there is no exact cover), then we have a YES instance of OPPORTUNISTIC MANIPULATION. It turns out that this will follow from the fact that there are no viable extensions, because, as we will show next, a viable extension implies the existence of an exact set cover.

To this end, first observe that the partial votes are constructed in such a way that c gets no additional score from *any* extension. Assuming that the manipulator approves c (without loss of generality), the final score of c in any extension is going to be $s_{\mathcal{Q}}(c) + 1$. Now, in any viable extension, every candidate u_j has to be "pushed out" of the top k positions at least once. Observe that whenever this happens, y is forced into the top k positions. Since y is behind the score of c by only $m/3$ votes, S_i's can be pushed out of place in only $m/3$ votes. For every u_j to lose one point, these votes must correspond to an exact cover. Therefore, if there is no exact cover, then there is no viable extension, showing one direction of the reduction.

On the other hand, suppose we have a NO instance of $\overline{\text{X3C}}$ – that is, there is an exact cover. We will now use the exact cover to come up with two viable extensions, both of which require the manipulator to vote in different ways to make c win. Therefore, there is no single manipulative vote that accounts for both extensions, leading us to a NO instance of OPPORTUNISTIC MANIPULATION.

First, consider this completion of the partial votes:

$$i = 1, \mathcal{W} \succ y \succ x \succ z_1 \succ z_2 \succ S_i \succ (\mathcal{U} \setminus S_i) \succ c \succ d$$

$$2 \leqslant i \leqslant {}^m/_3, \mathcal{W} \succ y \succ z_1 \succ z_2 \succ x \succ S_i \succ (\mathcal{U} \setminus S_i) \succ c \succ d$$

$${}^m/_3 + 1 \leqslant i \leqslant t, \mathcal{W} \succ S_i \succ y \succ z_1 \succ z_2 \succ x \succ (\mathcal{U} \setminus S_i) \succ c \succ d$$

Notice that in this completion, once accounted for along with the votes in $\mathcal{Q}$, the score of c is tied with the scores of all u_j's, z_1, x and y, while the score of z_2 is one less than the score of c. Therefore, the only k candidates that the manipulator can afford to approve are $\mathcal{W}$, the candidates c, d and z_2. However, consider the extension that is identical to the above except with the first vote changed to:

$$\mathcal{W} \succ y \succ x \succ z_2 \succ z_1 \succ S_i \succ (\mathcal{U} \setminus S_i) \succ c \succ d$$

Here, on the other hand, the only way for c to be an unique winner is if the manipulator approves $\mathcal{W}, c, d$ and z_1. Therefore, it is clear that there is no way for the manipulator to provide a consolidated vote for both these profiles. Therefore, we have a NO instance of OPPORTUNISTIC MANIPULATION. $\qquad\square$

8.2.3.2 Result for the k-Veto Voting Rule

We next move on to the k-veto voting rule and show that the OPPORTUNISTIC MANIPULATION problem for the k-veto is co-NP-hard for every constant $k \geqslant 4$ even when the number of manipulators is one and the number of undetermined pairs in each vote is no more than 9.

Theorem 8.6. *The* OPPORTUNISTIC MANIPULATION *problem is* co-NP-*hard for the k-veto voting rule for every constant $k \geqslant 4$ even when the number of manipulators is one and the number of undetermined pairs in each vote is no more than 9.*

Proof. We reduce X3C to OPPORTUNISTIC MANIPULATION for k-veto rule. Let $(\mathcal{U} = \{u_1, \ldots, u_m\}, \mathcal{S} = \{S_1, S_2, \ldots, S_t\})$ is an X3C instance. We construct a corresponding OPPORTUNISTIC MANIPULATION instance for k-veto voting rule as follows.

Candidate set $\mathcal{C} = \mathcal{U} \cup \{c, z_1, z_2, d, x, y\} \cup \mathcal{A} \cup \mathcal{W}$, where $\mathcal{A} = \{a_1, a_2, a_3\}, |\mathcal{W}| = k - 4$

For every $i \in [t]$, we define $\mathcal{P}_i'$ as follows:

$$\forall i \leqslant t, c \succ \mathcal{A} \succ (\mathcal{U} \setminus S_i) \succ d \succ S_i \succ y \succ x \succ z_1 \succ z_2 \succ \mathcal{W}$$

Using $\mathcal{P}_i'$, we define partial vote $\mathcal{P}_i = \mathcal{P}_i' \setminus (\{\{y, x, z_1, z_2\} \times S_i\} \cup \{(z_1, z_2), (x, z_1), (x, z_2)\})$ for every $i \in [t]$. We denote the set of partial votes $\{\mathcal{P}_i : i \in [t]\}$ by $\mathcal{P}$ and $\{\mathcal{P}_i' : i \in [t]\}$ by $\mathcal{P}'$. We note that the number of undetermined pairs in each partial vote $\mathcal{P}_i$ is 9. Using Lemma 7.1, we add a set $\mathcal{Q}$ of complete votes with $|\mathcal{Q}| = \text{poly}(m, t)$ to ensure the following. We denote the k-veto score of a candidate from a set of votes $\mathcal{W}$ by $s_{\mathcal{W}}(\cdot)$.

▷ $s_{\mathcal{P}' \cup \mathcal{Q}}(z_1) = s_{\mathcal{P}' \cup \mathcal{Q}}(z_2) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) - m/3$

▷ $s_{\mathcal{P}' \cup \mathcal{Q}}(a_i) = s_{\mathcal{P}' \cup \mathcal{Q}}(u_j) = s_{\mathcal{P}' \cup \mathcal{Q}}(w) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) \ \forall a_i \in \mathcal{A}, u_j \in \mathcal{U}, w \in \mathcal{W}$

▷ $s_{\mathcal{P}' \cup \mathcal{Q}}(y) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) - m/3 - 1$

▷ $s_{\mathcal{P}' \cup \mathcal{Q}}(x) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) - 2$

We have only one manipulator who tries to make c winner. Now we show that the X3C instance $(\mathcal{U}, \mathcal{S})$ is a YES instance if and only if the OPPORTUNISTIC MANIPULATION instance $(\mathcal{P} \cup \mathcal{Q}, 1, c)$ is a NO instance.

In the forward direction, let us now assume that the X3C instance is a YES instance. Suppose (by renaming) that $S_1, \ldots, S_{m/3}$ forms an exact set cover. Let us assume that the manipulator's vote $\mathfrak{v}$ disapproves every candidate in $\mathcal{W} \cup \mathcal{A}$ since otherwise c can never win uniquely. We now show that if $\mathfrak{v}$ does not disapprove z_1 then, $\mathfrak{v}$ is not a c-optimal vote. Suppose $\mathfrak{v}$ does not disapprove z_1. Then we consider the following extension $\overline{\mathcal{P}}$ of $\mathcal{P}$.

$$i = 1, c \succ \mathcal{A} \succ (\mathcal{U} \setminus S_i) \succ d \succ y \succ z_1 \succ x \succ z_2 \succ S_i \succ \mathcal{W}$$

$$2 \leqslant i \leqslant m/3, c \succ \mathcal{A} \succ (\mathcal{U} \setminus S_i) \succ d \succ y \succ z_1 \succ z_2 \succ x \succ S_i \succ \mathcal{W}$$

$$m/3 + 1 \leqslant i \leqslant t, c \succ \mathcal{A} \succ (\mathcal{U} \setminus S_i) \succ d \succ S_i \succ y \succ x \succ z_1 \succ z_2 \succ \mathcal{W}$$

We have the following scores $s_{\overline{\mathcal{P}} \cup \mathcal{Q}}(c) = s_{\overline{\mathcal{P}} \cup \mathcal{Q}}(z_1) = s_{\overline{\mathcal{P}} \cup \mathcal{Q}}(z_2) + 1 = s_{\overline{\mathcal{P}} \cup \mathcal{Q}}(x) + 1 = s_{\overline{\mathcal{P}} \cup \mathcal{Q}}(u_j) + 1 \, \forall u_j \in \mathcal{U}$. Hence, both c and z_1 win for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}\}$. However, the vote $\mathfrak{v}'$ which disapproves a_1, a_2, a_3, z_1 makes c a unique winner for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}'\}$. Hence, $\mathfrak{v}$ is not a c-optimal vote. Similarly, we can show that if the manipulator's vote does not disapprove z_2 then, the vote is not c-optimal. Hence, there does not exist any c-optimal vote and the OPPORTUNISTIC MANIPULATION instance is a NO instance.

In the reverse direction, we show that if the X3C instance is a NO instance, then there does not exist a vote $\mathfrak{v}$ of the manipulator and an extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ such that c is the unique winner for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}'\}$ thereby proving that the OPPORTUNISTIC MANIPULATION instance is vacuously YES (and thus every vote is c-optimal). Notice that, there must be at least $m/3$ votes $\mathcal{P}_1$ in $\overline{\mathcal{P}}$ where the corresponding S_i gets pushed in bottom k positions since $s_{\mathcal{P}' \cup \mathcal{Q}}(u_j) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) \, \forall a_i \in \mathcal{A}, u_j \in \mathcal{U}$. However, in each vote in $\mathcal{P}_1$, y is placed within top $m - k$ many position and thus we have $|\mathcal{P}_1|$ is exactly $m/3$ since $s_{\mathcal{P}' \cup \mathcal{Q}}(y) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) - m/3 - 1$. Now notice that there must be at least one candidate $u \in \mathcal{U}$ which is not covered by the sets S_is corresponding to the votes $\mathcal{P}_1$ because the X3C instance is a NO instance. Hence, c cannot win the election uniquely irrespective of the manipulator's vote. Thus every vote is c-optimal and the OPPORTUNISTIC MANIPULATION instance is a YES instance. □

8.2.3.3 Result for the Borda Voting Rule

We show next similar intractability result for the Borda voting rule too with only at most 7 undetermined pairs per vote.

Theorem 8.7. *The* OPPORTUNISTIC MANIPULATION *problem is* co-NP-*hard for the Borda voting rule even when the number of manipulators is one and the number of undetermined pairs in every vote is no more than 7.*

Proof. We reduce X3C to OPPORTUNISTIC MANIPULATION for the Borda rule. Let $(\mathcal{U} = \{u_1, \ldots, u_m\}, \mathcal{S} = \{S_1, S_2, \ldots, S_t\})$ is an X3C instance. Without loss of generality we assume that m is not divisible by 6 (if not, then we add three new elements b_1, b_2, b_3 to $\mathcal{U}$ and a set $\{b_1, b_2, b_3\}$ to $\mathcal{S}$). We construct a corresponding OPPORTUNISTIC MANIPULATION instance for the Borda voting rule as follows.

$$\text{Candidate set } \mathcal{C} = \mathcal{U} \cup \{c, z_1, z_2, d, y\}$$

For every $i \in [t]$, we define $\mathcal{P}'_i$ as follows:

$$\forall i \leqslant t, y \succ S_i \succ z_1 \succ z_2 \succ (\mathcal{U} \setminus S_i) \succ d \succ c$$

Using $\mathcal{P}_i'$, we define partial vote $\mathcal{P}_i = \mathcal{P}_i' \setminus (\{(\{y\} \cup S_i) \times \{z_1, z_2\}\} \cup \{(z_1, z_2)\})$ for every $i \in [t]$. We denote the set of partial votes $\{\mathcal{P}_i : i \in [t]\}$ by $\mathcal{P}$ and $\{\mathcal{P}_i' : i \in [t]\}$ by $\mathcal{P}'$. We note that the number of undetermined pairs in each partial vote $\mathcal{P}_i$ is 7. Using Lemma 7.1, we add a set $\mathcal{Q}$ of complete votes with $|\mathcal{Q}| = \text{poly}(m, t)$ to ensure the following. We denote the Borda score of a candidate from a set of votes $\mathcal{W}$ by $s_{\mathcal{W}}(\cdot)$.

$\triangleright$ $s_{\mathcal{P}' \cup \mathcal{Q}}(y) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) + m + m/3 + 3$

$\triangleright$ $s_{\mathcal{P}' \cup \mathcal{Q}}(z_1) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) - 3 \lfloor m/6 \rfloor - 2$

$\triangleright$ $s_{\mathcal{P}' \cup \mathcal{Q}}(z_2) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) - 5 \lfloor m/6 \rfloor - 3$

$\triangleright$ $s_{\mathcal{P}' \cup \mathcal{Q}}(u_i) = s_{\mathcal{P}' \cup \mathcal{Q}}(c) + m + 5 - i \; \forall i \in [m]$

$\triangleright$ $s_{\mathcal{P}' \cup \mathcal{Q}}(d) \leqslant s_{\mathcal{P}' \cup \mathcal{Q}}(c) - 5m$

We have only one manipulator who tries to make c winner. Now we show that the X3C instance $(\mathcal{U}, \mathcal{S})$ is a YES instance if and only if the OPPORTUNISTIC MANIPULATION instance $(\mathcal{P} \cup \mathcal{Q}, 1, c)$ is a NO instance. Notice that we can assume without loss of generality that the manipulator places c at the first position, d at the second position, the candidate u_i at $(m + 5 - i)^{\text{th}}$ position for every $i \in [m]$, and y at the last position, since otherwise c can never win uniquely irrespective of the extension of $\mathcal{P}$ (that it, the manipulator's vote looks like $c \succ d \succ \{z_1, z_2\} \succ u_m \succ u_{m-1} \succ \cdots \succ u_1 \succ y$).

In the forward direction, let us now assume that the X3C instance is a YES instance. Suppose (by renaming) that $S_1, \ldots, S_{m/3}$ forms an exact set cover. Let the manipulator's vote $\mathfrak{v}$ be $c \succ d \succ z_1 \succ z_2 \succ u_m \succ \cdots \succ u_1 \succ y$. We now argue that $\mathfrak{v}$ is not a c-optimal vote. The other case where the manipulator's vote $\mathfrak{v}'$ be $c \succ d \succ z_2 \succ z_1 \succ u_m \succ \cdots \succ u_1 \succ y$ can be argued similarly. We consider the following extension $\overline{\mathcal{P}}$ of $\mathcal{P}$.

$$1 \leqslant i \leqslant \lfloor m/6 \rfloor, z_2 \succ y \succ S_i \succ z_1 \succ (\mathcal{U} \setminus S_i) \succ d \succ c$$

$$\lceil m/6 \rceil \leqslant i \leqslant m/3, z_1 \succ y \succ S_i \succ z_2 \succ (\mathcal{U} \setminus S_i) \succ d \succ c$$

$$m/3 + 1 \leqslant i \leqslant t, y \succ S_i \succ z_1 \succ z_2 \succ (\mathcal{U} \setminus S_i) \succ d \succ c$$

We have the following Borda scores $s_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}\}}(c) = s_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}\}}(y) + 1 = s_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}\}}(z_2) + 6 = s_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}\}}(z_1) = s_{\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}\}}(u_i) + 1 \; \forall i \in [m]$. Hence, c does not win uniquely for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}\}$. However, c is the unique winner for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}'\}$. Hence, there does not exist any c-optimal vote and the OPPORTUNISTIC MANIPULATION instance is a NO instance.

In the reverse direction, we show that if the X3C instance is a NO instance, then there does not exist a vote $\mathfrak{v}$ of the manipulator and an extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ such that c is the unique winner for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}'\}$ thereby proving that the OPPORTUNISTIC MANIPULATION instance is vacuously YES (and thus every vote is c-optimal). Notice that the score of y must decrease by at least $m/3$ for c to win uniquely. However, in every vote v where the score of y decreases by at least one in any extension $\overline{\mathcal{P}}$ of $\mathcal{P}$, at least one of z_1 or z_2 must be placed at top position of the vote v. However, the candidates z_1 and z_2 can be placed at top positions of the votes in $\overline{\mathcal{P}}$ at most $m/3$ many times while ensuring c does not lose the election. Also, even after manipulator places the candidate u_i at $(m + 5 - i)^{\text{th}}$ position for every $i \in [m]$, for c to win uniquely, the score of every u_i must decrease by at least one. Hence, altogether, there will be exactly $m/3$ votes (denoted by the set $\mathcal{P}_1$) in any extension of $\mathcal{P}$ where y is placed at the second position. However, since the X3C instance is a NO instance, the S_is corresponding to the votes in $\mathcal{P}_1$ does not form a set cover. Let $u \in \mathcal{U}$ be an element not covered by the S_is corresponding to the votes in $\mathcal{P}_1$. Notice that the score of u does not decrease in the extension $\overline{\mathcal{P}}$ and thus c does not win uniquely irrespective of the manipulator's vote. Thus every vote is c-optimal and thus the OPPORTUNISTIC MANIPULATION instance is a YES instance. Thus every vote is c-optimal and the OPPORTUNISTIC MANIPULATION instance is a YES instance. □

8.2.3.4 Result for the Maximin Voting Rule

For the maximin voting rule, we show intractability of OPPORTUNISTIC MANIPULATION with one manipulator even when the number of undetermined pairs in every vote is at most 8.

Theorem 8.8. *The* OPPORTUNISTIC MANIPULATION *problem is* co-NP-*hard for the maximin voting rule even when the number of manipulators is one and the number of undetermined pairs in every vote is no more than* 8.

Proof. We reduce X3C to OPPORTUNISTIC MANIPULATION for the maximin rule. Let $(\mathcal{U} = \{u_1, \ldots, u_m\}, \mathcal{S} = \{S_1, S_2, \ldots, S_t\})$ is an X3C instance. We construct a corresponding OPPORTUNISTIC MANIPULATION instance for the maximin voting rule as follows.

$$\text{Candidate set } \mathcal{C} = \mathcal{U} \cup \{c, z_1, z_2, z_3, d, y\}$$

For every $i \in [t]$, we define $\mathcal{P}'_i$ as follows:

$$\forall i \leqslant t, S_i \succ x \succ d \succ y \succ (\mathcal{U} \setminus S_i) \succ z_1 \succ z_2 \succ z_3$$

Using $\mathcal{P}'_i$, we define partial vote $\mathcal{P}_i = \mathcal{P}'_i \setminus (\{(\{x\} \cup S_i) \times \{d, y\}\})$ for every $i \in [t]$. We denote the set of partial votes $\{\mathcal{P}_i : i \in [t]\}$ by $\mathcal{P}$ and $\{\mathcal{P}'_i : i \in [t]\}$ by $\mathcal{P}'$. We note that the number of undetermined pairs in each partial vote $\mathcal{P}_i$ is 8. We define another partial vote $\mathfrak{p}$ as follows.

$$\mathfrak{p} = (z_1 \succ z_2 \succ z_3 \succ \text{ others }) \setminus \{(z_1, z_2), (z_2, z_3), (z_1, z_3)\}$$

Using Lemma 7.2, we add a set $\mathcal{Q}$ of complete votes with $|\mathcal{Q}| = \text{poly}(m, t)$ to ensure the following pairwise margins (notice that the pairwise margins among z_1, z_2, and z_3 does not include the partial vote $\mathfrak{p}$).

▷ $D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(d, c) = 4t + 1$

▷ $D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(x, d) = 4t + 2m/3 + 1$

▷ $D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(y, x) = 4t - 2m/3 + 1$

▷ $D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(d, u_j) = 4t - 1 \ \forall u_j \in \mathcal{U}$

▷ $D_{\mathcal{P}' \cup \mathcal{Q}}(z_1, z_2) = D_{\mathcal{P}' \cup \mathcal{Q}}(z_2, z_3) = D_{\mathcal{P}' \cup \mathcal{Q}}(z_3, z_1) = 4t + 2$

▷ $|D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(a, b)| \leqslant 1$ for every $a, b \in \mathcal{C}$ not defined above.

We have only one manipulator who tries to make c winner. Now we show that the X3C instance $(\mathcal{U}, \mathcal{S})$ is a YES instance if and only if the OPPORTUNISTIC MANIPULATION instance $(\mathcal{P} \cup \mathcal{Q} \cup \{\mathfrak{p}\}, 1, c)$ is a No instance. Notice that we can assume without loss of generality that the manipulator's vote prefers c to every other candidate, y to x, x to d, and d to u_j for every $u_j \in \mathcal{U}$.

In the forward direction, let us now assume that the X3C instance is a YES instance. Suppose (by renaming) that $S_1, \ldots, S_{m/3}$ forms an exact set cover. Notice that the manipulator's vote must prefer either z_2 to z_1 or z_1 to z_3 or z_3 to z_2. We show that if the manipulator's vote $\mathfrak{v}$ prefers z_2 to z_1, then $\mathfrak{v}$ is not a c-optimal vote. The other two cases are symmetrical. Consider the following extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ and $\overline{\mathfrak{p}}$ of $\mathfrak{p}$.

$$1 \leqslant i \leqslant m/3, d \succ y \succ S_i \succ x \succ (\mathcal{U} \setminus S_i) \succ z_1 \succ z_2 \succ z_3$$

$$\text{m}/3 + 1 \leqslant i \leqslant t, S_i \succ x \succ d \succ y \succ (\mathcal{U} \setminus S_i) \succ z_1 \succ z_2 \succ z_3$$

$$\overline{\mathfrak{p}} = z_2 \succ z_3 \succ z_1 \succ \quad \text{others}$$

From the votes in $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}, \overline{\mathfrak{p}}\}$, the maximin score of c is $-4t$, of $d, x, u_j \ \forall u_j \in \mathcal{U}$ are $-4t - 2$, of z_1, z_3 are at most than $-4t - 2$, and of z_2 is $-4t$. Hence, c is not the unique maximn winner. However, the manipulator's vote $c \succ z_1 \succ z_2 \succ z_3 \succ$ other makes c the unique maximin winner. Hence, $\mathfrak{v}$ is not a c-optimal vote.

For the reverse direction, we show that if the X3C instance is a No instance, then there does not exist a vote $\mathfrak{v}$ of the manipulator and an extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ such that c is the unique winner for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}'\}$ thereby proving that the OPPORTUNISTIC MANIPULATION instance is vacuously YES (and thus every vote is c-optimal). Consider any extension $\overline{\mathcal{P}}$ of $\mathcal{P}$. Notice that, for c to win uniquely, $y \succ x$ must be at least $\text{m}/3$ of the votes in $\overline{\mathcal{P}}$; call these set of votes $\mathcal{P}_1$. However, $d \succ x$ in every vote in $\mathcal{P}_1$ and $d \succ x$ can be in at most $\text{m}/3$ votes in $\overline{\mathcal{P}}$ for c to win uniquely. Hence, we have $|\mathcal{P}_1| = \text{m}/3$. Also for c to win, each $d \succ u_j$ must be at least one vote of $\overline{\mathcal{P}}$ and $d \succ u_j$ is possible only in the votes in $\mathcal{P}_1$. However, the sets S_is corresponding to the votes in $\mathcal{P}_1$ does not form a set cover since the X3C instance is a No instance. Hence, there must exist a $u_j \in \mathcal{U}$ for which $u_j \succ d$ in every vote in $\overline{\mathcal{P}}$ and thus c cannot win uniquely irrespective of the vote of the manipulator. Thus every vote is c-optimal and the OPPORTUNISTIC MANIPULATION instance is a YES instance. $\qquad \square$

8.2.3.5 Result for the Copeland$^\alpha$ Voting Rule

Our next result proves that the OPPORTUNISTIC MANIPULATION problem is co-NP-hard for the Copeland$^\alpha$ voting rule too for every $\alpha \in [0, 1]$ even with one manipulator and at most 8 undetermined pairs per vote.

Theorem 8.9. *The* OPPORTUNISTIC MANIPULATION *problem is* co-NP-*hard for the Copeland$^\alpha$ voting rule for every $\alpha \in [0, 1]$ even when the number of manipulators is one and the number of undetermined pairs in each vote is no more than* 8.

Proof. We reduce X3C to OPPORTUNISTIC MANIPULATION for the Copeland$^\alpha$ voting rule. Let $(\mathcal{U} = \{u_1, \ldots, u_m\}, \mathcal{S} = \{S_1, S_2, \ldots, S_t\})$ is an X3C instance. We construct a corresponding OPPORTUNISTIC MANIPULATION instance for the Copeland$^\alpha$ voting rule as follows.

$$\text{Candidate set } \mathcal{C} = \mathcal{U} \cup \{c, z_1, z_2, z_3, d_1, d_2, d_3, x, y\}$$

For every $i \in [t]$, we define $\mathcal{P}'_i$ as follows:

$$\forall i \leqslant t, S_i \succ x \succ y \succ c \succ \text{ others}$$

Using $\mathcal{P}'_i$, we define partial vote $\mathcal{P}_i = \mathcal{P}'_i \setminus (\{(\{x\} \cup S_i) \times \{c, y\}\})$ for every $i \in [t]$. We denote the set of partial votes $\{\mathcal{P}_i : i \in [t]\}$ by $\mathcal{P}$ and $\{\mathcal{P}'_i : i \in [t]\}$ by $\mathcal{P}'$. We note that the number of undetermined pairs in each partial vote $\mathcal{P}_i$ is 8. We define another partial vote $\mathfrak{p}$ as follows.

$$\mathfrak{p} = (z_1 \succ z_2 \succ z_3 \succ \text{ others }) \setminus \{(z_1, z_2), (z_2, z_3), (z_1, z_3)\}$$

Using Lemma 7.2, we add a set $\mathcal{Q}$ of complete votes with $|\mathcal{Q}| = \text{poly}(m, t)$ to ensure the following pairwise margins (notice that the pairwise margins among z_1, z_2, and z_3 does not include the partial vote $\mathfrak{p}$).

▷ $D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(u_j, c) = 2 \ \forall u_j \in \mathcal{U}$

▷ $D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(x, y) = {}^{2m}/3$

▷ $D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(c, y) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(x, c) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(d_i, c) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(z_k, c) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(u_j, x) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(x, z_k) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(d_i, x) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(y, u_j) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(d_i, y) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(y, z_k) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(z_k, u_j) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(u_j, d_i) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(z_k, d_1) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(z_k, d_2) = D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(d_3, z_k) = 4t \ \forall i, k \in [3], j \in [m]$

▷ $D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(u_j, u_\ell) = -4t$ for at least ${}^m/3$ many $u_\ell \in \mathcal{U}$

▷ $D_{\mathcal{P}' \cup \mathcal{Q}}(z_1, z_2) = D_{\mathcal{P}' \cup \mathcal{Q}}(z_2, z_3) = D_{\mathcal{P}' \cup \mathcal{Q}}(z_3, z_1) = 1$

▷ $|D_{\mathcal{P}' \cup \mathcal{Q} \cup \{\mathfrak{p}\}}(a, b)| \leqslant 1$ for every $a, b \in \mathcal{C}$ not defined above.

We have only one manipulator who tries to make c winner. Now we show that the X3C instance $(\mathcal{U}, \mathcal{S})$ is a Yes instance if and only if the Opportunistic Manipulation instance $(\mathcal{P} \cup \mathcal{Q} \cup \{\mathfrak{p}\}, 1, c)$ is a No instance. Since the number of voters is odd, α does not play any role in the reduction and thus from here on we simply omit α. Notice that we can assume without loss of generality that the manipulator's vote prefers c to every other candidate and x to y.

In the forward direction, let us now assume that the X3C instance is a Yes instance. Suppose (by renaming) that $S_1, \ldots, S_{m/3}$ forms an exact set cover. Suppose the manipulator's vote $\mathfrak{v}$ order z_1, z_2, and z_3 as $z_1 \succ z_2 \succ z_3$. We will show that $\mathfrak{v}$ is not a c-optimal vote.

Symmetrically, we can show that the manipulator's vote ordering z_1, z_2, and z_3 in any other order is not c-optimal. Consider the following extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ and $\overline{\mathfrak{p}}$ of $\mathfrak{p}$.

$$1 \leqslant i \leqslant {}^m\!/_3, y \succ c \succ S_i \succ x \succ \text{others}$$

$$^m\!/_3 + 1 \leqslant i \leqslant t, S_i \succ x \succ y \succ c \succ \text{others}$$

$$\overline{\mathfrak{p}} = z_1 \succ z_2 \succ z_3 \succ \text{others}$$

From the votes in $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}, \overline{\mathfrak{p}}\}$, the Copeland score of c is $m + 4$ (defeating $y, z_k, u_j \ \forall k \in [3], j \in [m]$), of y is $m+3$ (defeating $z_k, u_j \ \forall k \in [3], j \in [m]$), of u_j is at most ${}^{2m}\!/_3 + 4$ (defeating $x, d_i \ \forall i \in [3]$ and at most ${}^{2m}\!/_3$ many $u_\ell \in \mathcal{U}$), of x is 5 (defeating $c, y, z_k \ \forall l \in [3]$), of d_1, d_2 is 2 (defeating y and c), of d_3 is 5 (defeating $y, c, z_k \ \forall k \in [3]$). of z_3 is $m + 3$ (defeating $d_i, u_j \forall i \in [3], j \in [m]$) for every $k \in [3]$, of z_3 is $m + 2$ (defeating $d_1, d_2, u_j i \in [3], j \in [m]$), z_2 is $m + 3$ (defeating $d_1, d_2, z_3, u_j i \in [3], j \in [m]$), z_1 is $m + 4$ (defeating $d_1, d_2, z_2, z_3, u_j i \in [3], j \in [m]$). Hence, c co-wins with z_1 with Copeland score $m+4$. However, the manipulator's vote $c \succ z_3 \succ z_2 \succ z_1$ makes c win uniquely. Hence, $\mathfrak{v}$ is not a c-optimal vote and thus the OPPORTUNISTIC MANIPULATION instance is a NO instance.

For the reverse direction, we show that if the X3C instance is a NO instance, then there does not exist a vote $\mathfrak{v}$ of the manipulator and an extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ such that c is the unique winner for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}'\}$ thereby proving that the OPPORTUNISTIC MANIPULATION instance is vacuously YES (and thus every vote is c-optimal). Consider any extension $\overline{\mathcal{P}}$ of $\mathcal{P}$. Notice that, for c to win uniquely, c must defeat each $u_j \in \mathcal{U}$ and thus c is preferred over u_j in at least one vote in $\overline{\mathcal{P}}$; we call these votes $\mathcal{P}_1$. However, in every vote in $\mathcal{P}_1$, y is preferred over x and thus $|\mathcal{P}_1| \leqslant {}^m\!/_3$ because x must defeat y for c to win uniquely. Since the X3C instance is a NO instance, there must be a candidate $u \in \mathcal{U}$ which is not covered by the sets corresponding to the votes in $\mathcal{P}_1$ and thus u is preferred over c in every vote in $\mathcal{P}$. Hence, c cannot win uniquely irrespective of the vote of the manipulator. Thus every vote is c-optimal and the OPPORTUNISTIC MANIPULATION instance is a YES instance. $\qquad\square$

8.2.3.6 Result for the Bucklin Voting Rule

For the Bucklin and simplified Bucklin voting rules, we show intractability of the OPPORTUNISTIC MANIPULATION problem with at most 15 undetermined pairs per vote and only one manipulator.

Theorem 8.10. *The* OPPORTUNISTIC MANIPULATION *problem is* co-NP-*hard for the Bucklin and simplified Bucklin voting rules even when the number of manipulators is one and the number of undetermined pairs in each vote is no more than* 15.

Proof. We reduce X3C to OPPORTUNISTIC MANIPULATION for the Bucklin and simplified Bucklin voting rules. Let $(\mathcal{U} = \{u_1, \ldots, u_m\}, \mathcal{S} = \{S_1, S_2, \ldots, S_t\})$ is an X3C instance. We assume without loss of generality that m is not divisible by 6 (if not, we introduce three elements in $\mathcal{U}$ and a set containing them in $\mathcal{S}$) and t is an even integer (if not, we duplicate any set in $\mathcal{S}$). We construct a corresponding OPPORTUNISTIC MANIPULATION instance for the Bucklin and simplified Bucklin voting rules as follows.

$$\text{Candidate set } \mathcal{C} = \mathcal{U} \cup \{c, z_1, z_2, x_1, x_2, d\} \cup W, \text{ where } |W| = m - 3$$

For every $i \in [t]$, we define $\mathcal{P}_i'$ as follows:

$$\forall i \leqslant t, (\mathcal{U} \setminus S_i) \succ S_i \succ d \succ x_1 \succ x_2 \succ z_1 \succ z_2 \succ \text{ others}$$

Using $\mathcal{P}_i'$, we define partial vote $\mathcal{P}_i = \mathcal{P}_i' \setminus (\{(\{d\} \cup S_i) \times \{x_1, x_2, z_1, z_2\}\} \cup \{(z_1, z_2)\})$ for every $i \in [t]$. We denote the set of partial votes $\{\mathcal{P}_i : i \in [t]\}$ by $\mathcal{P}$ and $\{\mathcal{P}_i' : i \in [t]\}$ by $\mathcal{P}'$. We note that the number of undetermined pairs in each partial vote $\mathcal{P}_i$ is 15. We introduce the following additional complete votes $\mathcal{Q}$:

▷ $t/2 - \lfloor m/6 \rfloor - 1$ copies of $W \succ z_1 \succ z_2 \succ x_1 \succ c \succ \text{ others}$

▷ $t/2 - \lfloor m/6 \rfloor - 1$ copies of $W \succ z_1 \succ z_2 \succ x_2 \succ c \succ \text{ others}$

▷ $2\lceil m/6 \rceil$ copies of $W \succ z_1 \succ z_2 \succ d \succ c \succ \text{ others}$

▷ $\lfloor m/6 \rfloor$ copies of $W \succ z_1 \succ d \succ x_1 \succ c \succ \text{ others}$

▷ $\lfloor m/6 \rfloor$ copies of $W \succ z_1 \succ d \succ x_2 \succ c \succ \text{ others}$

▷ $2\lceil m/6 \rceil - 1$ copies of $\mathcal{U} \succ x_1 \succ \text{ others}$

▷ One $\mathcal{U} \succ c \succ \text{ others}$

We have only one manipulator who tries to make c winner. Now we show that the X3C instance $(\mathcal{U}, \mathcal{S})$ is a YES instance if and only if the OPPORTUNISTIC MANIPULATION instance $(\mathcal{P} \cup \mathcal{Q}, 1, c)$ is a NO instance. The total number of voters in the OPPORTUNISTIC MANIPULATION

instance is $2t + 2m/3 + 1$. We notice that within top $m + 1$ positions of the votes in $\mathcal{P}' \cup \mathcal{Q}$, c appears $t + m/3$ times, z_1 and z_2 appear $t + \lfloor m/6 \rfloor$ times, x_1 appears $t/2 + m/3 - 1$ times, x_2 appears $t/2 - 1$ times, every candidate in W appears $t + m/3 - 1$ times, every candidate in $\mathcal{U}$ appears $t + m/3 + 1$ times. Also every candidate in $\mathcal{U}$ appears $t + m/3 + 1$ times within top m positions of the votes in $\mathcal{P} \cup \mathcal{Q}$. Hence, for both Bucklin and simplified Bucklin voting rules, we can assume without loss of generality that the manipulator puts c, every candidate in W, x_1, x_2, and exactly one of z_1 and z_2.

In the forward direction, let us now assume that the X3C instance is a YES instance. Suppose (by renaming) that $S_1, \ldots, S_{m/3}$ forms an exact set cover. Suppose the manipulator's vote $\mathfrak{v}$ puts c, every candidate in W, x_1, x_2, and z_1 within top $m + 1$ positions. We will show that $\mathfrak{v}$ is not c-optimal. The other case where the manipulator's vote $\mathfrak{v}'$ puts c, every candidate in W, x_1, x_2, and z_2 within top $m + 1$ positions is symmetrical. Consider the following extension $\overline{\mathcal{P}}$ of $\mathcal{P}$:

$$1 \leqslant i \leqslant \lfloor m/6 \rfloor, (\mathcal{U} \setminus S_i)d \succ x_1 \succ x_2 \succ z_2 \succ S_i \succ\succ z_1 \succ \text{ others}$$

$$\lceil m/6 \rceil \leqslant i \leqslant m/3, (\mathcal{U} \setminus S_i)d \succ x_1 \succ x_2 \succ z_1 \succ S_i \succ\succ z_2 \succ \text{ others}$$

$$m/3 + 1 \leqslant i \leqslant t, (\mathcal{U} \setminus S_i) \succ S_i \succ d \succ x_1 \succ x_2 \succ z_1 \succ z_2 \succ \text{ others}$$

For both Bucklin and simplified Bucklin voting rules, c co-wins with z_1 for the votes in $\mathcal{P} \cup \mathcal{Q} \cup \{\mathfrak{v}\}$. However, c wins uniquely for the votes in $\mathcal{P} \cup \mathcal{Q} \cup \{\mathfrak{v}'\}$. Hence, $\mathfrak{v}$ is not a c-optimal vote and thus the OPPORTUNISTIC MANIPULATION instance is a NO instance.

For the reverse direction, we show that if the X3C instance is a NO instance, then there does not exist a vote $\mathfrak{v}$ of the manipulator and an extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ such that c is the unique winner for the votes $\overline{\mathcal{P}} \cup \mathcal{Q} \cup \{\mathfrak{v}'\}$ thereby proving that the OPPORTUNISTIC MANIPULATION instance is vacuously YES (and thus every vote is c-optimal). Consider any extension $\overline{\mathcal{P}}$ of $\mathcal{P}$. Notice that, for c to win uniquely, every candidate must be pushed out of top $m + 1$ positions in at least one vote in $\mathcal{P}$; we call these set of votes $\mathcal{P}_1$. Notice that, $|\mathcal{P}_1| \geqslant m/3$. However, in every vote in $\mathcal{P}_1$, at least one of z_1 and z_2 appears within top $m + 1$ many positions. Since, the manipulator has to put at least one of z_1 and z_2 within its top $m + 1$ positions and z_1 and z_2 appear $t + \lfloor m/6 \rfloor$ times in the votes in $\mathcal{P}' \cup \mathcal{Q}$, we must have $|\mathcal{P}_1| \leqslant m/3$ and thus $|\mathcal{P}_1| = m/3$, for c to win uniquely. However, there exists a candidate $u \in \mathcal{U}$ not covered by the S_is corresponding to the votes in $\mathcal{P}_1$. Notice that u gets majority within top m positions of the votes and c can never get majority within top $m + 1$ positions of the votes. Hence, c

cannot win uniquely irrespective of the vote of the manipulator. Thus every vote is c-optimal and the OPPORTUNISTIC MANIPULATION instance is a YES instance. □

8.3 Polynomial Time Algorithms for WEAK MANIPULATION, STRONG MANIPULATION, and OPPORTUNISTIC MANIPULATION Problems

We now turn to the polynomial time cases depicted in Table 8.1. This section is organized in three parts, one for each problem considered.

8.3.1 Weak Manipulation Problem

Since the POSSIBLE WINNER problem is in P for the plurality and the veto voting rules [BD09], it follows from Observation 8.1 that the WEAK MANIPULATION problem is in P for the plurality and veto voting rules for any number of manipulators.

Proposition 8.1. *The* WEAK MANIPULATION *problem is in* P *for the plurality and veto voting rules for any number of manipulators.*

Proof. The POSSIBLE WINNER problem is in P for the plurality and the veto voting rules [BD09]. Hence, the result follows from Observation 8.1. □

8.3.2 Strong Manipulation Problem

We now discuss our algorithms for the STRONG MANIPULATION problem. The common flavor in all our algorithms is the following: we try to devise an extension that is as adversarial as possible for the favorite candidate c, and if we can make c win in such an extension, then roughly speaking, such a strategy should work for other extensions as well (where the situation only improves for c). However, it is challenging to come up with an extension that is globally dominant over all the others in the sense that we just described. So what we do instead is we consider every potential nemesis w who might win instead of c, and we build profiles that are "as good as possible" for w and "as bad as possible" for c. Each such profile leads us to constraints on how much the manipulators can afford to favor w (in terms of which positions among the manipulative votes are *safe* for w). We then typically show that

we can determine whether there exists a set of votes that respects these constraints, either by using a greedy strategy or by an appropriate reduction to a flow problem. We note that the overall spirit here is similar to the approaches commonly used for solving the NECESSARY WINNER problem, but as we will see, there are non-trivial differences in the details. We begin with the k-approval and k-veto voting rules.

Theorem 8.11. *The* STRONG MANIPULATION *problem is in* P *for the* k-*approval and* k-*veto voting rules, for any* k *and any number of manipulators.*

Proof. For the time being, we just concentrate on non-manipulators' votes. For each candidate $c' \in \mathcal{C} \setminus \{c\}$, calculate the maximum possible value of $s_{NM}^{max}(c, c') = s_{NM}(c') - s_{NM}(c)$ from non-manipulators' votes, where $s_{NM}(a)$ is the score that candidate a receives from the votes of the non-manipulators. This can be done by checking all possible $O(m^2)$ pairs of positions for c and c' in each vote v and choosing the one which maximizes $s_v(c') - s_v(c)$ from that vote. We now fix the position of c at the top position for the manipulators' votes and we check if it is possible to place other candidates in the manipulators' votes such that the final value of $s_{NM}^{max}(c, c') + s_M(c') - s_M(c)$ is negative which can be solved easily by reducing it to the max flow problem which is polynomial time solvable. □

We now prove that the STRONG MANIPULATION problem for scoring rules is in P for one manipulator.

Theorem 8.12. *The* STRONG MANIPULATION *problem is in* P *for any scoring rule when we have only one manipulator.*

Proof. For each candidate $c' \in \mathcal{C} \setminus \{c\}$, calculate $s_{NM}^{max}(c, c')$ using same technique described in the proof of Theorem 8.11. We now put c at the top position of the manipulator's vote. For each candidate $c' \in \mathcal{C} \setminus \{c\}$, c' can be placed at positions $i \in \{2, \ldots, m\}$ in the manipulator's vote which makes $s_{NM}^{max}(c, c') + \alpha_i - \alpha_1$ negative. Using this, construct a bipartite graph with $\mathcal{C} \setminus \{c\}$ on left and $\{2, \ldots, m\}$ on right and there is an edge between c' and i iff the candidate c' can be placed at i in the manipulator's vote according to the above criteria. Now solve the problem by finding existence of perfect matching in this graph. □

Our next result proves that the STRONG MANIPULATION problem for the Bucklin, simplified Bucklin, fallback, and simplified fallback voting rules are in P.

Theorem 8.13. *The* STRONG MANIPULATION *problem is in* P *for the Bucklin, simplified Bucklin, fallback, and simplified fallback voting rules, for any number of manipulators.*

Proof. Let $(\mathcal{C}, \mathcal{P}, M, c)$ be an instance of STRONG MANIPULATION for simplified Bucklin, and let m denote the total number of candidates in this instance. Recall that the manipulators have to cast their votes so as to ensure that the candidate c wins in every possible extension of $\mathcal{P}$. We use $\mathcal{Q}$ to denote the set of manipulating votes that we will construct. To begin with, without loss of generality, the manipulators place c in the top position of all their votes. We now have to organize the positioning of the remaining candidates across the votes of the manipulators to ensure that c is a necessary winner of the profile $(\mathcal{P}, \mathcal{Q})$.

To this end, we would like to develop a system of constraints indicating the overall number of times that we are free to place a candidate $w \in \mathcal{C} \setminus \{c\}$ among the top ℓ positions in the profile $\mathcal{Q}$. In particular, let us fix $w \in \mathcal{C} \setminus \{c\}$ and $2 \leqslant \ell \leqslant m$. Let $\eta_{w,\ell}$ be the maximum number of votes of $\mathcal{Q}$ in which w can appear in the top ℓ positions. Our first step is to compute necessary conditions for $\eta_{w,\ell}$.

We use $\overline{\mathcal{P}}_{w,\ell}$ to denote a set of complete votes that we will construct based on the given partial votes. Intuitively, these votes will represent the "worst" possible extensions from the point of view of c when pitted against w. These votes are engineered to ensure that the manipulators can make c win the elections $\overline{\mathcal{P}}_{w,\ell}$ for all $w \in \mathcal{C} \setminus \{c\}$ and $\ell \in \{2, \ldots, m\}$, if, and only if, they can strongly manipulate in favor of c. More formally, there exists a voting profile $\mathcal{Q}$ of the manipulators so that c wins the election $\overline{\mathcal{P}}_{w,\ell} \cup \mathcal{Q}$, for all $w \in \mathcal{C} \setminus \{c\}$ and $\ell \in \{2, \ldots, m\}$ if and only if c wins in every extension of the profile $\mathcal{P} \cup \mathcal{Q}$.

We now describe the profile $\overline{\mathcal{P}}_{w,\ell}$. The construction is based on the following case analysis, where our goal is to ensure that, to the extent possible, we position c out of the top $\ell - 1$ positions, and incorporate w among the top ℓ positions.

▷ Let $v \in \mathcal{P}$ be such that either c and w are incomparable or $w \succ c$. We add the complete vote v' to $\overline{\mathcal{P}}_{w,\ell}$, where v' is obtained from v by placing w at the highest possible position and c at the lowest possible position, and extending the remaining vote arbitrarily.

▷ Let $v \in \mathcal{P}$ be such that $c \succ w$, but there are at least ℓ candidates that are preferred over w in v. We add the complete vote v' to $\overline{\mathcal{P}}_{w,\ell}$, where v' is obtained from v by placing c at the lowest possible position, and extending the remaining vote arbitrarily.

▷ Let $v \in \mathcal{P}$ be such that c is forced to be within the top $\ell - 1$ positions, then we add the complete vote v' to $\overline{\mathcal{P}}_{w,\ell}$, where v' is obtained from v by first placing w at the highest possible position followed by placing c at the lowest possible position, and extending the remaining vote arbitrarily.

▷ In the remaining votes, notice that whenever w is in the top ℓ positions, c is also in the top $\ell - 1$ positions. Let $\mathcal{P}^*_{w,\ell}$ denote this set of votes, and let t be the number of votes in $\mathcal{P}^*_{w,\ell}$.

We now consider two cases. Let $d_\ell(c)$ be the number of times c is placed in the top $\ell - 1$ positions in the profile $\overline{\mathcal{P}}_{w,\ell} \cup \mathcal{Q}$, and let $d_\ell(w)$ be the number of times w is placed in the top ℓ positions in the profile $\overline{\mathcal{P}}_{w,\ell}$. Let us now formulate the requirement that in $\overline{\mathcal{P}}_{w,\ell} \cup \mathcal{Q}$, the candidate c does *not* have a majority in the top $\ell - 1$ positions and w *does* have a majority in the top ℓ positions. Note that if this requirement holds for any w and ℓ, then strong manipulation is not possible. Therefore, to strongly manipulate in favor of c, we must ensure that for every choice of w and ℓ, we are able to negate the conditions that we derive.

The first condition from above simply translates to $d_\ell(c) \leqslant n/2$. The second condition amounts to requiring first, that there are at least $n/2$ votes where w appears in the top ℓ positions, that is, $d_\ell(w) + \eta_{w,\ell} + t > n/2$. Further, note that the gap between $d_\ell(w) + \eta_{w,\ell}$ and majority will be filled by using votes from $\mathcal{P}^*_{w,\ell}$ to "push" w forward. However, these votes contribute equally to w and c being in the top ℓ and $\ell - 1$ positions, respectively. Therefore, the difference between $d_\ell(w) + \eta_{w,\ell}$ and $n/2$ must be less than the difference between $d_\ell(c)$ and $n/2$. Summarizing, the following conditions, which we collectively denote by $(\star)$, are sufficient to defeat c in some extension: $d_\ell(c) \leqslant n/2$, $d_\ell(w) + \eta_{w,\ell} + t > n/2$, $n/2 - d_\ell(w) + \eta_{w,\ell} < n/2 - d_\ell(c)$.

From the manipulator's point of view, the above provides a set of constraints to be satisfied as they place the remaining candidates across their votes. Whenever $d_\ell(c) > n/2$, the manipulators place any of the other candidates among the top ℓ positions freely, because c already has majority. On the other hand, if $d_\ell(c) \leqslant n/2$, then the manipulators must respect at least one of the following constraints: $\eta_{w,\ell} \leqslant n/2 - t - d_\ell(w)$ and $\eta_{w,\ell} \leqslant d_\ell(c) - d_\ell(w)$.

Extending the votes of the manipulator while respecting these constraints (or concluding that this is impossible to do) can be achieved by a natural greedy strategy — construct the manipulators' votes by moving positionally from left to right. For each position, consider each manipulator and populate her vote for that position with any available candidate. We output the profile if the process terminates by completing all the votes, otherwise, we say NO.

We now argue the proof of correctness. Suppose the algorithm returns NO. This implies that there exists a choice of $w \in \mathcal{C} \setminus \{c\}$ and $\ell \in \{2, \ldots, m\}$ such that for any voting profile $\mathcal{Q}$ of the manipulators, the conditions in $(\star)$ are satisfied. (Indeed, if there exists a voting profile that violated at least one of these conditions, then the greedy algorithm would have

discovered it.) Therefore, no matter how the manipulators cast their vote, there exists an extension where c is defeated. In particular, for the votes in $\mathcal{P} \setminus \mathcal{P}^*_{w,\ell}$, this extension is given by $\overline{\mathcal{P}}_{w,\ell}$. Further, we choose $n/2 - \eta_{w,\ell} - d_\ell(w)$ votes among the votes in $\mathcal{P}^*_{w,\ell}$ and extend them by placing w in the top ℓ positions (and extending the rest of the profile arbitrary). We extend the remaining votes in $\mathcal{P}^*_{w,\ell}$ by positioning w outside the top ℓ positions. Clearly, in this extension, c fails to achieve majority in the top $\ell - 1$ positions while w does achieve majority in the top ℓ positions.

On the other hand, if the algorithm returns YES, then consider the voting profile of the manipulators. We claim that c wins in every extension of $\mathcal{P} \cup \mathcal{Q}$. Suppose, to the contrary, that there exists an extension $\mathcal{R}$ and a candidate w such that the simplified Bucklin score of c is no more than the simplified Bucklin score of w in $\mathcal{R}$. In this extension, therefore, there exists $\ell \in \{2, \dots, m\}$ for which w attains majority in the top ℓ positions and c fails to attain majority in the top $\ell - 1$ positions. However, note that this is already impossible in any extension of the profile $\overline{\mathcal{P}}_{w,1} \cup \mathcal{P}^*_{w,\ell}$, because of the design of the constraints. By construction, the number of votes in which c appears in the top $\ell - 1$ positions in $\mathcal{R}$ is only greater than the number of times c appears in the top $\ell - 1$ positions in any extension of $\overline{\mathcal{P}}_{w,1} \cup \mathcal{P}^*_{w,\ell}$ (and similarly for w). This leads us to the desired contradiction.

For the Bucklin voting rule, we do the following modifications to the algorithm. If $d_\ell(c) > d_\ell(w)$ for some $w \in \mathcal{C} \setminus \{c\}$ and $\ell < m$, then we make $\eta_{w,\ell} = \infty$. The proof of correctness for the Bucklin voting rule is similar to the proof of correctness for the simplified Bucklin voting rule above.

For Fallback and simplified Fallback voting rules, we consider the number of candidates each voter approves while computing $\eta_{w,\ell}$. We output YES if and only if $\eta_{w,\ell} \geqslant 0$ for every $w \in \mathcal{C} \setminus \{c\}$ and every $\ell \leqslant m$, since we can assume, without loss of generality, that the manipulator approves the candidate c only. Again the proof of correctness is along similar lines to the proof of correctness for the simplified Bucklin voting rule. $\qquad \square$

We next show that the STRONG MANIPULATION problem for the maximin voting rule is polynomial-time solvable when we have only one manipulator.

Theorem 8.14. *The* STRONG MANIPULATION *problem for the maximin voting rules are in* P*, when we have only one manipulator.*

Proof. For the time being, just concentrate on non-manipulators' votes. Using the algorithm for NW for maximin in [XC11], we compute for all pairs $w, w' \in \mathcal{C}$, $N_{(w,w')}(w, d)$ and

$N_{(w,w')}(c, w')$ for all $d \in \mathcal{C} \setminus \{c\}$. This can be computed in polynomial time. Now we place c at the top position in the manipulator's vote and increase all $N_{(w,w')}(c, w')$ by one. Now we place a candidate w at the second position if for all $w' \in \mathcal{C}$, $N'_{(w,w')}(w, d) < N_{(w,w')}(c, w')$ for all $d \in \mathcal{C} \setminus \{c\}$, where $N'_{(w,w')}(w, d) = N_{(w,w')}(w, d)$ of the candidate d has already been assigned some position in the manipulator's vote, and $N'_{(w,w')}(w, d) = N_{(w,w')}(w, d) + 1$ else. The correctness argument is in the similar lines of the classical greedy manipulation algorithm of [BTT89]. $\qquad\square$

8.3.3 Opportunistic Manipulation Problem

For the plurality, fallback, and simplified fallback voting rules, it turns out that the voting profile where all the manipulators approve only c is a c-opportunistic voting profile, and therefore it is easy to devise a manipulative vote.

Observation 8.4. *The* OPPORTUNISTIC MANIPULATION *problem is in* P *for the plurality and fallback voting rules for a any number of manipulators.*

For the veto voting rule, however, a more intricate argument is needed, that requires building a system of constraints and a reduction to a suitable instance of the maximum flow problem in a network, to show polynomial time tractability of OPPORTUNISTIC MANIPULATION.

Theorem 8.15. *The* OPPORTUNISTIC MANIPULATION *problem is in* P *for the veto voting rule for a constant number of manipulators.*

Proof. Let $(\mathcal{P}, \ell, c)$ be an input instance of OPPORTUNISTIC MANIPULATION. We may assume without loss of generality that the manipulators approve c. We view the voting profile of the manipulators as a tuple $(n_a)_{a \in \mathcal{C} \setminus \{c\}} \in (\mathbb{N} \cup \{0\})^{m-1}$ with $\sum_{a \in \mathcal{C} \setminus \{c\}} n_a = \ell$, where the n_a many manipulators disapprove a. We denote the set of such tuples as $\mathcal{T}$ and we have $\mathcal{T} = O((2m)^\ell)$ which is polynomial in m since ℓ is a constant. A tuple $(n_a)_{a \in \mathcal{C} \setminus \{c\}} \in \mathcal{T}$ is not c-optimal if there exists another tuple $(n'_a)_{a \in \mathcal{C} \setminus \{c\}} \in \mathcal{T}$ and an extension $\overline{\mathcal{P}}$ of $\mathcal{P}$ with the following properties. We denote the veto score of a candidate from $\mathcal{P}$ by $s(\cdot)$. For every candidate $a \in \mathcal{C} \setminus \{c\}$, we define two quantities $w(a)$ and $d(a)$ as follows.

▷ $s(c) > s(a)$ for every $a \in \mathcal{C} \setminus \{c\}$ with $n_a = n'_a = 0$ and we define $w(a) = s(c) - 1, d(a) = 0$

▷ $s(c) > s(a) - n'_a$ for every $a \in \mathcal{C} \setminus \{c\}$ with $n_a \geqslant n'_a$ and we define $w(a) = s(c) - n'_a - 1, d(a) = 0$

▷ $s(a) - n_a \geqslant s(c) > s(a) - n'_a$ for every $a \in \mathcal{C} \setminus \{c\}$ with $n_a < n'_a$ and we define $w(a) = s(c) - n'_a, d(a) = s(a) - n_a$

We guess the value of $s(c)$. Given a value of $s(c)$, we check the above two conditions by reducing this to a max flow problem instance as follows. We have a source vertex s and a sink t. We have a vertex for every $a \in \mathcal{C}$ (call this set of vertices Y) and a vertex for every vote $v \in \mathcal{P}$ (call this set of vertices X). We add an edge from s to each in X of capacity one. We add an edge of capacity one from a vertex $x \in X$ to a vertex $y \in Y$ if the candidate corresponding to the vertex y can be placed at the last position in an extension of the partial vote corresponding to the vertex x. We add an edge from a vertex y to t of capacity $w(a)$, where a is the voter corresponding to the vertex y. We also set the demand of every vertex y $d(a)$ (that is the total amount of flow coming into vertex y must be at least $d(a)$), where a is the voter corresponding to the vertex y. Clearly, the above three conditions are met if and only if there is a feasible $|\mathcal{P}|$ amount of flow in the above flow graph. Since $s(c)$ can have only $|\mathcal{P}| + 1$ possible values (from 0 to $\mathcal{P}$) and $|\mathcal{T}| = O((2m)^\ell)$, we can iterate over all possible pairs of tuples in $\mathcal{T}$ and all possible values of $s(c)$ and find a c-optimal voting profile if there exists a one. □

8.4 Conclusion

We revisited many settings where the complexity barrier for manipulation was non-existent, and studied the problem under an incomplete information setting. Our results present a fresh perspective on the use of computational complexity as a barrier to manipulation, particularly in cases that were thought to be dead-ends (because the traditional manipulation problem was polynomially solvable). To resurrect the argument of computational hardness, we have to relax the model of complete information, but we propose that the incomplete information setting is more realistic, and many of our hardness results work even with very limited amount of incompleteness in information.

In the next chapter, we will see how possible instances of manipulation can be detected for various voting rules.

Chapter 9

Manipulation Detection

Manipulation is a problem of fundamental importance in voting theory in which the voters exercise their votes strategically instead of voting honestly to make an alternative that is more preferred to her, win the election. The classical Gibbard-Satterthwaite theorem shows that there is no strategy-proof voting rule that simultaneously satisfies certain combination of desirable properties. Researchers have attempted to get around the impossibility result in several ways such as domain restriction and computational hardness of manipulation. However, these approaches are known to have fundamental limitations. Since prevention of manipulation seems to be elusive even after substantial research effort, an interesting research direction therefore is detection of manipulation. Motivated by this, we initiate the study of detecting possible instances of manipulation in elections.

We formulate two pertinent computational problems in the context of manipulation detection - Coalitional Possible Manipulators (CPM) and Coalitional Possible Manipulators given Winner (CPMW), where a suspect group of voters is provided as input and we have to find whether they can be a potential coalition of manipulators. In the absence of any suspect group, we formulate two more computational problems namely Coalitional Possible Manipulators Search (CPMS) and Coalitional Possible Manipulators Search given Winner (CPMSW). We provide polynomial time algorithms for these problems, for several popular voting rules. For a few other voting rules, we show that these problems are

NP-complete. We observe that detecting possible instances of manipulation may be easy even when the actual manipulation problem is computationally intractable, as seen for example, in the case of the Borda voting rule.

9.1 Introduction

A basic problem with voting rules is that the voters may vote strategically instead of voting honestly, leading to the selection of a candidate which is not the *actual winner*. We call a candidate actual winner if it wins the election when every voter votes truthfully. This phenomenon of strategic voting is called *manipulation* in the context of voting. The Gibbard-Satterthwaite (G-S) theorem [Gib73, Sat75] proves that manipulation is unavoidable for any *unanimous* and *non-dictatorial* voting rule if we have at least three candidates. A voting rule is called *unanimous* if whenever any candidate is most preferred by all the voters, such a candidate is the winner. A voting rule is called *non-dictatorial* if there does not exist any voter whose most preferred candidate is always the winner irrespective of the votes of other voters. The problem of manipulation is particularly relevant for multiagent systems since agents have computational power to determine strategic votes. There have been several attempts to bypass the impossibility result of the G-S theorem.

Economists have proposed domain restriction as a way out of the impossibility implications of the G-S theorem. The G-S theorem assumes all possible preference profiles as the domain of voting rules. In a restricted domain, it has been shown that we can have voting rules that are not vulnerable to manipulation. A prominent restricted domain is the domain of single peaked preferences, in which the median voting rule provides a satisfactory solution [MCWG95]. To know more about other domain restrictions, we refer to [MCWG95, Gae01]. This approach of restricting the domain, however, suffers from the requirement that the social planner needs to know the domain of preference profiles of the voters, which is often impractical.

9.1.1 Related Work

Researchers in computational social choice theory have proposed invoking computational intractability of manipulation as a possible work around for the G-S theorem. Bartholdi et al. [BO91, BTT89] first proposed the idea of using computational hardness as a barrier

against manipulation. Bartholdi et al. defined and studied the computational problem called manipulation where a set of manipulators have to compute their votes that make their preferred candidate win the election. The manipulators know the votes of the truthful voters and the voting rule that will be used to compute the winner. Following this, a large body of research [NWX11, DKNW11, XZP+09, XCP10, CSL07, OEH11, EL05, FHH13, NW13, GKNW13, OE12, FHH10, ZLR11, DN14, FRRS14, EE12] shows that the manipulation problem is NP-complete for many voting rules. However, Procaccia et al. [PR06, PR07] showed average case easiness of manipulation assuming *junta* distribution over the voting profiles. Friedgut et al. [FKN08] showed that any *neutral* voting rule which is sufficiently far from being dictatorial is manipulable with non-negligible probability at any uniformly random preference profile by a uniformly random preference. The above result holds for elections with three candidates only. A voting rule is called *neutral* if the names of the candidates are immaterial. Isaksson et al. [IKM12] generalize the above result to any number of candidates which has been further generalized to all voting rules which may not be neutral by Mossel and Racz in [MR15]. Walsh [Wal10] empirically shows ease of manipulating an STV (single transferable vote) election – one of the very few voting rules where manipulation, even by one voter, is NP-complete [BO91]. In addition to the results mentioned above, there exist many other results in the literature that emphasize the weakness of considering computational complexity as a barrier against manipulation [CS06, XC08b, XC08a, FP10, Wal11a]. Hence, the barrier of computational hardness is ineffective against manipulation in many settings.

9.1.2 Motivation

In a situation where multiple attempts for prevention of manipulation fail to provide a fully satisfactory solution, detection of manipulation is a natural next step of research. There have been scenarios where establishing the occurrence of manipulation is straightforward, by observation or hindsight. For example, in sport, there have been occasions where the very structure of the rules of the game have encouraged teams to deliberately lose their matches. Observing such occurrences in, for example, football (the 1982 FIFA World Cup football match played between West Germany and Austria) and badminton (the quarter-final match between South Korea and China in the London 2012 Olympics), the relevant authorities have subsequently either changed the rules of the game (as with football) or disqualified the teams in question (as with the badminton example). The importance of detecting manipulation lies in the potential for implementing corrective measures in the future. For reasons that will be

evident soon, it is not easy to formally define the notion of manipulation detection. Assume that we have the votes from an election that has already happened. A voter is potentially a manipulator if there exists a preference $\succ$, different from the voter's reported preference, which is such that the voter had an "incentive to deviate" from the $\succ$. Specifically, suppose the candidate who wins with respect to this voter's reported preference is preferred (in $\succ$) over the candidate who wins with respect to $\succ$. In such a situation, $\succ$ could potentially be the voter's truthful preference, and the voter could be refraining from being truthful because an untruthful vote leads to a more favorable outcome with respect to $\succ$. Note that we do not (and indeed, cannot) conclusively suggest that a particular voter has manipulated an election. This is because the said voter can always claim that she voted truthfully; since her actual preference is only known to her, there is no way to prove or disprove such a claim. Therefore, we are inevitably limited to asking only whether or not a voter has *possibly* manipulated an election.

Despite this technical caveat, it is clear that efficient detection of manipulation, even if it is only possible manipulation, is potentially of interest in practice. We believe that, the information whether a certain group of voters have possibly manipulated an election or not would be useful to social planners. For example, the organizers of an event, say London 2012 Olympics, may be interested to have this information. Also, in settings where data from many past elections (roughly over a fixed set of voters) is readily available, it is conceivable that possible manipulation could serve as suggestive evidence of real manipulation. Aggregate data about possible manipulations, although formally inconclusive, could serve as an important evidence of real manipulation, especially in situations where the instances of possible manipulation turn out to be statistically significant. Thus, efficient detection of possible manipulation would provide an useful input to a social planner for future elections. We remark that having a rich history is typically not a problem, particularly for AI related applications, since the data generated from an election is normally kept for future requirements (for instance, for data mining or learning). For example, several past affirmatives for possible manipulation is one possible way of formalizing the notion of erratic past behavior. Also, applications where benefit of doubt may be important, for example, elections in judiciary systems, possible manipulation detection may be useful. Thus the computational problem of detecting possible manipulation is of definite interest in many settings.

9.1.3 Our Contribution

The novelty of this work is in initiating research on *detection of possible manipulators* in elections. We formulate four pertinent computational problems in this context:

▷ CPM: In the *coalitional possible manipulators* problem, we are interested in whether or not a given subset of voters is a possible coalition of manipulators [Definition 9.4].

▷ CPMW: The *coalitional possible manipulators given winner* is the CPM problem with the additional information about who the winner would have been if the possible manipulators had all voted truthfully [Definition 9.2].

▷ CPMS, CPMSW: In CPMS (*Coalitional Possible Manipulators Search*), we want to know, whether there exists any coalition of possible manipulators of a size at most k [Definition 9.6]. Similarly, we define CPMSW (*Coalitional Possible Manipulators Search given Winner*) [Definition 9.5].

Our specific findings are as follows.

▷ We show that all the four problems above, for scoring rules and the maximin voting rule, are in P when the coalition size is one [Theorem 9.1 and 9.4].

▷ We prove that all the four problems, *for any coalition size*, are in P for a wide class of scoring rules which include the Borda voting rule [Theorem 9.2, Theorem 9.3 and Corollary 9.1].

▷ We show that, for the Bucklin voting rule [Theorem 9.6], both the CPM and CPMW problems are in P, *for any coalition size*. The CPMS and CPMSW problems for the Bucklin voting rule are also in P, when we have maximum possible coalition size $k = O(1)$.

▷ We show that both the CPM and the CPMW problems are NP-complete for the STV voting rule [Theorem 9.7 and Corollary 9.2], *even for a coalition of size one*. We also prove that the CPMW problem is NP-complete for maximin voting rule [Theorem 9.5], *for a coalition of size two*.

We observe that all the four problems are computationally easy for many voting rules that we study in this work. This can be taken as a positive result. The results for the CPM and the CPMW problems are summarized in Table 9.1.

Voting Rule	CPM, $k = 1$	CPM	CPMW, $k = 1$	CPMW
Scoring Rules	P	?	P	?
Borda	P	P	P	P
k-approval	P	P	P	P
Maximin	P	?	P	NP-complete
Bucklin	P	P	P	P
STV	NP-complete	NP-complete	NP-complete	NP-complete

Table 9.1: Results for CPM and CPMW (k denotes coalition size). The '?' mark means that the problem is open.

9.2 Problem Formulation

Consider an election that has already happened in which all the votes are known and thus the winner $x \in \mathcal{C}$ is also known. We call the candidate x the *current winner* of the election. The authority may suspect that the voters belonging to a subset $M \subset \mathcal{V}$ of the set of voters have formed a coalition among themselves and manipulated the election by voting non-truthfully. The authority believes that other voters who do not belong to M, have voted truthfully. We denote $|M|$, the size of the coalition, by k. Suppose the authority has auxiliary information, may be from some other sources, which says that the *actual winner* should have been some candidate $y \in \mathcal{C}$ other than x. We call a candidate *actual winner* if it wins the election where all the voters vote truthfully. This means that the authority thinks that, had the voters in M voted truthfully, the candidate y would have been the winner. We remark that there are practical situations, for example, 1982 FIFA World cup or 2012 London Olympics, where the authority knows the actual winner. This situation is formalized below.

Definition 9.1. *Let r be a voting rule, and $(\succ_i)_{i \in \mathcal{V}}$ be a voting profile of a set $\mathcal{V}$ of n voters. Let x be the winning candidate with respect to r for this profile. For a candidate $y \neq x$, $M \subset \mathcal{V}$ is called a coalition of possible manipulators against y with respect to r if there exists a $|M|$-voters' profile $(\succ'_j)_{j \in M} \in \mathcal{L}(\mathcal{C})^{|M|}$ such that $x \succ'_j y, \forall j \in M$, and further, $r((\succ_j)_{j \in \mathcal{V} \setminus M}, (\succ'_i)_{i \in M}) = y$.*

Using the notion of coalition of possible manipulators, we formulate a computational problem called *Coalitional Possible Manipulators given Winner (CPMW)* as follows. Let r be any voting rule.

Definition 9.2. *(r–CPMW Problem)*
Given a preference profile $(\succ_i)_{i \in \mathcal{V}}$ of a set of voters $\mathcal{V}$ over a set of candidates $\mathcal{C}$, a subset of voters

$M \subset \mathcal{V}$, *and a candidate* y, *determine if* M *is a coalition of possible manipulators against* y *with respect to* r.

In the CPMW problem, the actual winner is given in the input. However, it may very well happen that the authority does not have any other information to guess the *actual winner* – the candidate who would have won the election had the voters in M voted truthfully. In this situation, the authority is interested in knowing whether there is a $|M|$-voter profile which along with the votes in $\mathcal{V} \setminus M$ makes some candidate $y \in \mathcal{C}$ the winner who is different from the current winner $x \in \mathcal{C}$ and all the preferences in the $|M|$-voters' profile prefer x to y. If such a $|M|$-voter profile exists for the subset of voters M, then we call M a *coalition of possible manipulators* and the corresponding computational problem is called *Coalitional Possible Manipulators (CPM)*. These notions are formalized below.

Definition 9.3. *Let* r *be a voting rule, and* $(\succ_i)_{i \in \mathcal{V}}$ *be a voting profile of a set* $\mathcal{V}$ *of* n *voters. A subset of voters* $M \subset \mathcal{V}$ *is called a coalition of possible manipulators with respect to* r *if* M *is a coalition of possible manipulators against some candidate* y *with respect to* r.

Definition 9.4. *(r–CPM Problem)*
Given a preference profile $(\succ_i)_{i \in \mathcal{V}}$ *of a set of voters* $\mathcal{V}$ *over a set of candidates* $\mathcal{C}$, *and a subset of voters* $M \subset \mathcal{V}$, *determine if* M *is a coalition of possible manipulators with respect to* r.

In both the CPMW and CPM problems, a subset of voters which the authority suspect to be a coalition of manipulators, is given in the input. However, there can be situations where there is no specific subset of voters to suspect. In those scenarios, it may still be useful to know, what are the possible coalition of manipulators of size less than some number k. Towards that end, we extend the CPMW and CPM problems to search for a coalition of potential possible manipulators and call them *Coalitional Possible Manipulators Search given Winner (CPMSW)* and *Coalitional Possible Manipulators Search (CPMS)* respective.

Definition 9.5. *(r–CPMSW Problem)*
Given a preference profile $(\succ_i)_{i \in \mathcal{V}}$ *of a set of voters* $\mathcal{V}$ *over a set of candidates* $\mathcal{C}$, *a candidate* y, *and an integer* k, *determine whether there exists any* $M \subset \mathcal{V}$ *with* $|M| \leqslant k$ *such that* M *is a coalition of possible manipulators against* y.

Definition 9.6. *(r–CPMS Problem)*
Given a preference profile $(\succ_i)_{i \in \mathcal{V}}$ *of a set of voters* $\mathcal{V}$ *over a set of candidates* $\mathcal{C}$, *and an integer* k, *determine whether there exists any* $M \subset \mathcal{V}$ *with* $|M| \leqslant k$ *such that* M *is a coalition of possible manipulators.*

9.2.1 Discussion

The CPMW problem may look very similar to the manipulation problem [BTT89, CSL07] – in both the problems a set of voters try to make a candidate winner. However, in the CPMW problem, the actual winner must be less preferred to the current winner in every manipulator's vote. Although it may look like a subtle difference, it changes the nature and complexity theoretic behavior of the problem completely. For example, we show that all the four problems have an efficient algorithm for a large class of voting rules that includes the Borda voting rule, for any coalition size. However, the manipulation problem for the Borda voting rule is NP-complete, even when we have at least two manipulators [DKNW11, BNW11]. Another important difference is that the manipulation problem, in contrast to the problems studied in this work, does not take care of manipulators' preferences. We believe that there does not exist any formal reduction between the CPMW problem and the manipulation problem.

On the other hand, the CPMS problem is similar to the margin of victory problem defined by Xia [Xia12], where also we are looking for changing the current winner by changing at most some k number of votes, which in turn identical to the destructive bribery problem [FHH09]. Whereas, in the CPMS problem, the vote changes can occur in a restricted fashion. An important difference between the two problems is that the margin of victory problem has the hereditary property which the CPMS problem does not possess (there is no coalition of possible manipulators of size n in any election for all the common voting rules). These two problems do not seem to have any obvious complexity theoretic implications.

Now we explore the connection among the four problems that we study here. Notice that, a polynomial time algorithm for the CPM and the CPMW problems gives us a polynomial time algorithm for the CPMS and the CPMSW problems for any maximum possible coalition size $k = O(1)$. Also, observe that, a polynomial time algorithm for the CPMW (respectively CPMSW) problem implies a polynomial time algorithm for the CPM (respectively CPMS) problem. Hence, we have the following observations.

Observation 9.1. *For every voting rule, if the maximum possible coalition size $k = O(1)$, then,*

$$\text{CPMW} \in P \Rightarrow \text{CPM, CPMSW, CPMS} \in P$$

Observation 9.2. *For every voting rule,*

$$\text{CPMSW} \in P \Rightarrow \text{CPMS} \in P$$

9.3 Results for the CPMW, CPM, CPMSW, and CPMS Problems

In this section, we present our algorithmic results for the CPMW, CPM, CPMSW, and CPMS problems for various voting rules.

9.3.1 Scoring Rules

Below we have certain lemmas which form a crucial ingredient of our algorithms. To begin with, we define the notion of a *manipulated preference*. Let r be a scoring rule and $\succ := (\succ_i, \succ_{-i})$ be a voting profile of n voters. Let $\succ_i'$ be a preference such that

$$r(\succ) >_i' r(\succ_i', \succ_{-i})$$

Then we say that $\succ_i'$ is a $(\succ, i)$-manipulated preference with respect to r. We omit the reference to r if it is clear from the context.

Lemma 9.1. *Let r be a scoring rule and $\succ := (\succ_i, \succ_{-i})$ be a voting profile of n voters. Let a and b be two candidates such that $\mathrm{score}_{\succ_{-i}}(a) > \mathrm{score}_{\succ_{-i}}(b)$, and let $\succ_i'$ be $(\succ, i)$-manipulated preference where a precedes b:*

$$\succ_i' := \cdots > a > \cdots > b > \cdots$$

If a and b are not winners with respect to either $(\succ_i', \succ_{-i})$ or $\succ$, then the preference $\succ_i''$ obtained from $\succ_i'$ by interchanging a and b is also $(\succ, i)$-manipulated.

Proof. Let $x := r(\succ_i', \succ_{-i})$. If suffices to show that x continues to win in the proposed profile $(\succ_i'', \succ_{-i})$. To this end, it is enough to argue the scores of a and b with respect to x. First, consider the score of b in the new profile:

$$
\begin{aligned}
\mathrm{score}_{(\succ_i'', \succ_{-i})}(b) &= \mathrm{score}_{\succ_i''}(b) + \mathrm{score}_{\succ_{-i}}(b) \\
&< \mathrm{score}_{\succ_i'}(a) + \mathrm{score}_{\succ_{-i}}(a) \\
&= \mathrm{score}_{(\succ_i', \succ_{-i})}(a) \\
&\leqslant \mathrm{score}_{(\succ_i', \succ_{-i})}(x) \\
&= \mathrm{score}_{(\succ_i'', \succ_{-i})}(x)
\end{aligned}
$$

The second line uses the fact that $\text{score}_{\succ_i''}(b) = \text{score}_{\succ_i'}(a)$ and $\text{score}_{\succ_{-i}}(b) < \text{score}_{\succ_{-i}}(a)$. The fourth line comes from the fact that x is the winner and the last line follows from the fact that the position of x is same in both profiles. Similarly, we have the following argument for the score of a in the new profile (the second line below simply follows from the definition of scoring rules).

$$
\begin{aligned}
\text{score}_{(\succ_i'',\succ_{-i})}(a) &= \text{score}_{\succ_i''}(a) + \text{score}_{\succ_{-i}}(a) \\
&\leqslant \text{score}_{\succ_i'}(a) + \text{score}_{\succ_{-i}}(a) \\
&= \text{score}_{(\succ_i',\succ_{-i})}(a) \\
&\leqslant \text{score}_{(\succ_i',\succ_{-i})}(x) \\
&= \text{score}_{(\succ_i'',\succ_{-i})}(x)
\end{aligned}
$$

Since the tie breaking rule is according to some predefined fixed order $\succ_t \in \mathcal{L}(\mathcal{C})$ and the candidates tied with winner in $(\succ_i'', \succ_{-i})$ also tied with winner in $(\succ_i', \succ_{-i})$, we have the following,

$$
r(\succ) >_i'' r(\succ_i'', \succ_{-i})
$$

$\square$

We now show that, if there is some $(\succ, i)$-manipulated preference with respect to a scoring rule r, then there exists a $(\succ, i)$-manipulated preference with a specific structure.

Lemma 9.2. *Let r be a scoring rule and $\succ := (\succ_i, \succ_{-i})$ be a voting profile of n voters. If there is some $(\succ, i)$-manipulated preference with respect to r, then there also exists a $(\succ, i)$-manipulated preference $\succ_i'$ where the actual winner y immediately follows the current winner x:*

$$
\succ_i' := \cdots > x > y > \cdots
$$

and the remaining candidates are in nondecreasing ordered of their scores from $\succ_{-i}$.

Proof. Let $\succ''$ be a $(\succ, i)$-manipulated preference with respect to r. Let $x := r(\succ), y := r(\succ'', \succ_{-i})$. From Lemma 9.1, without loss of generality, we may assume that, all candidates except x, y are in nondecreasing order of $\text{score}_{\succ_{-i}}(.)$ in the preference $\succ''$. If $\succ_i'' := \cdots \succ x \succ \cdots \succ y \succ \cdots \succ \cdots$, we define $\succ_i' := \cdots \succ x \succ y \succ \cdots \succ \cdots$ from $\succ_i''$ where y is moved to the position following x and the position of the candidates in between x and y in $\succ_i''$ is

deteriorated by one position each. The position of the rest of the candidates remain same in both $\succ_i''$ and $\succ_i'$. Now we have following,

$$
\begin{aligned}
\text{score}_{(\succ_i', \succ_{-i})}(y) &= \text{score}_{\succ_i'}(y) + \text{score}_{\succ_{-i}}(y) \\
&\geqslant \text{score}_{\succ_i''}(y) + \text{score}_{\succ_{-i}}(y) \\
&= \text{score}_{(\succ_i'', \succ_{-i})}(y)
\end{aligned}
$$

We also have,

$$
\text{score}_{(\succ_i', \succ_{-i})}(a) \leqslant \text{score}_{(\succ_i'', \succ_{-i})}(a), \forall a \in \mathcal{C} \setminus \{y\}
$$

Since the tie breaking rule is according to some predefined order $\succ_t \in \mathcal{L}(\mathcal{C})$, we have the following,

$$
r(\succ) >_i' r(\succ', \succ_{-i})
$$

$\square$

Using Lemma 9.1 and 9.2, we now present our results for the scoring rules.

Theorem 9.1. *The CPMW, CPM, CPMSW, and CPMS problems for scoring rules are in* P *for a coalition of size* 1 *(that is, the coalition size* $k = 1$*).*

Proof. From Observation 9.1, it is enough to give a polynomial time algorithm for the CPMW problem. So consider the CPMW problem. We are given the actual winner y and we compute the current winner x with respect to r. Let $\succ_{[j]}$ be a preference where x and y are in positions j and $(j + 1)$ respectively, and the rest of the candidates are in nondecreasing order of the score that they receive from $\succ_{-i}$. For $j \in \{1, 2, \ldots, m - 1\}$, we check if y wins with the profile $(\succ_{-i}, \succ_{[j]})$. If we are successful with at least one j we report YES, otherwise we say NO. The correctness follows from Lemma 9.2. Thus we have a polynomial time algorithm for CPMW when $k = 1$. $\square$

Now we present our results for the CPMW and the CPM problems when $k > 1$. If $m = O(1)$, then both the CPMW and the CPM problems for any anonymous and efficient voting rule r can be solved in polynomial time by iterating over all possible $\binom{m!+k-1}{m!}$ ways the manipulators can have actual preferences. A voting rule is called efficient if winner determination under it is in P.

Theorem 9.2. *For scoring rules with $\alpha_1 - \alpha_2 \leqslant \alpha_i - \alpha_{i+1}, \forall i$, the CPMW and the CPM problems are in* P*, for any coalition size.*

Proof. We provide a polynomial time algorithm for the CPMW problem in this setting. Let x be the current winner and y be the given actual winner. Let M be the given subset of voters. Let $((\succ_i)_{i \in M}, (\succ_j)_{j \in V \setminus M})$ be the reported preference profile. Without loss of generality, we assume that x is the most preferred candidate in every $\succ_i, i \in M$. Let us define $\succ'_i, i \in M$, by moving y to the second position in the preference $\succ_i$. In the profile $((\succ'_i)_{i \in M}, (\succ_j)_{j \in V \setminus M})$, the winner is either x or y since only y's score has increased. We claim that M is a coalition of possible manipulators with respect to y *if and only if* y is the winner in preference profile $((\succ'_i)_{i \in M}, (\succ_j)_{j \in V \setminus M})$. This can be seen as follows. Suppose there exist preferences $\succ''_i$, with $x \succ''_i y, i \in M$, for which y wins in the profile $((\succ''_i)_{i \in M}, (\succ_j)_{j \in V \setminus M})$. Now without loss of generality, we can assume that y immediately follows x in all $\succ''_i, i \in M$, and $\alpha_1 - \alpha_2 \leqslant \alpha_i - \alpha_{i+1}, \forall i$ implies that we can also assume that x and y are in the first and second positions respectively in all $\succ''_i, i \in M$. Now in both the profiles, $((\succ'_i)_{i \in M}, (\succ_j)_{j \in V \setminus M})$ and $((\succ''_i)_{i \in M}, (\succ_j)_{j \in V \setminus M})$, the score of x and y are same. But in the first profile x wins and in the second profile y wins, which is a contradiction. □

We now prove a similar result for the CPMSW and CPMS problems.

Theorem 9.3. *For scoring rules with $\alpha_1 - \alpha_2 \leqslant \alpha_i - \alpha_{i+1}, \forall i$, the CPMSW and the CPMS problems are in* P*, for any coalition size.*

Proof. From Observation 9.2, it is enough to prove that CPMSW $\in \mathcal{P}$. Let x be the current winner, y be the given actual winner and $s(x)$ and $s(y)$ be their current respective scores. For each vote $v \in \mathcal{V}$, we compute a number $\Delta(v) = \alpha_2 - \alpha_j - \alpha_1 + \alpha_i$, where x and y are receiving scores α_i and α_j respectively from the vote v. Now, we output yes iff there are k votes $v_i, 1 \leqslant i \leqslant k$ such that, $\sum_{i=1}^{k} \Delta(v_i) \geqslant s(x) - s(y)$, which can be checked easily by sorting the $\Delta(v)$'s in nonincreasing order and checking the condition for the first k $\Delta(v)$'s, where k is the maximum possible coalition size specified in the input. The proof of correctness follows by exactly in the same line of argument as the proof of Theorem 9.2. □

For the k-approval voting rule, we can solve all the problems easily using max flow. Hence, from Theorem 9.2 and Theorem 9.3, we have the following result.

Corollary 9.1. *The CPMW, CPM, CPMSW, and CPMS problems for the Borda and k-approval voting rules are in* P*, for any coalition size.*

9.3.2 Maximin Voting Rule

For the maximin voting rule, we show that all the four problems are in P, when we have a coalition of size one.

Theorem 9.4. *The CPMW, CPM, CPMSW, and CPMS problems for maximin voting rule are in* P *for any coalition size* $k = 1$ *(for CPMW and CPM) or maximum possible coalition size* $k = 1$ *(for CMPWS and CPMS).*

Proof. Given a n-voters' profile $\succ \in \mathcal{L}(\mathcal{C})^n$ and a voter v_i, let the *current winner* be $x := r(\succ)$ and the given *actual winner* be y. We will construct $\succ' = (\succ_i', \succ_{-i})$, if it exists, such that $r(\succ) >_i' r(\succ') = y$, thus deciding whether v_i is a possible manipulator or not. Now, the maximin score of x and y in the profile $\succ'$ can take one of values from the set $\{score_{\succ_{-i}}(x) \pm 1\}$ and $\{score_{\succ_{-i}}(y) \pm 1\}$. The algorithm is as follows. We first guess the maximin score of x and y in the profile $\succ'$. There are only four possible guesses. Suppose, we guessed that x's score will decrease by one and y's score will decrease by one assuming that this guess makes y win. Now notice that, without loss of generality, we can assume that y immediately follows x in the preference $\succ_i'$ since y is the winner in the profile $\succ'$. This implies that there are only $O(m)$ many possible positions for x and y in $\succ_i'$. We guess the position of x and thus the position of y in $\succ_i'$. Let $B(x)$ and $B(y)$ be the sets of candidates with whom x and respectively y performs worst. Now since, x's score will decrease and y's score will decrease, we have the following constraint on $\succ_i'$. There must be a candidate each from $B(y)$ and $B(x)$ that will precede x. We do not know a-priori if there is one candidate that will serve as a witness for both $B(x)$ and $B(y)$, or if there separate witnesses. In the latter situation, we also do not know what order they appear in. Therefore we guess if there is a common candidate, and if not, we guess the relative ordering of the distinct candidates from $B(x)$ and $B(y)$. Now we place any candidate at the top position of $\succ_i'$ if this action does not make y lose the election. If there are many choices, we prioritize in favor of candidates from $B(x)$ and $B(y)$ — in particular, we focus on the candidates common to $B(x)$ and $B(y)$ if we expect to have a common witness, otherwise, we favor a candidate from one of the sets according to the guess we start with. If still there are multiple choices, we pick arbitrarily. After that we move on to the next position, and do the same thing (except we stop prioritizing explicitly for $B(x)$ and $B(y)$ once we have at least one witness from each set). The other situations can be handled similarly with minor modifications. In this way, if it is able to get a complete preference, then it checks whether v_i is a possible manipulator or not using this preference. If *yes*, then it returns YES. Otherwise, it

tries other positions for x and y and other possible scores of x and y. After trying all possible guesses, if it cannot find the desired preference, then it outputs NO. Since there are only polynomial many possible guesses, this algorithm runs in a polynomial amount of time. The proof of correctness follows from the proof of Theorem 1 in [BTT89]. □

We now show that the CPMW problem for the maximin voting rule is NP-complete when we have $k > 1$. Towards that, we use the fact that the *unweighted coalitional manipulation (UCM)* problem for the maximin voting rule is NP-complete [XZP$^+$09], when we have $k > 1$. The UCM problem is as follows. Let r be any voting rule.

Definition 9.7. *(r–UCM Problem)*
Given a set of manipulators $M \subset \mathcal{V}$, a profile of non-manipulators' vote $(\succ_i)_{i \in \mathcal{V} \setminus M}$, and a candidate $z \in \mathcal{C}$, we are asked whether there exists a profile of manipulators' votes $(\succ'_j)_{j \in M}$ such that $r((\succ_i)_{i \in \mathcal{V} \setminus M}, (\succ'_j)_{j \in M}) = z$. Assume that ties are broken in favor of z.

We define a restricted version of the UCM problem called R-UCM as follows.

Definition 9.8. *(r–R-UCM Problem)*
This problem is the same as the UCM problem with a given guarantee - let $k := |M|$. The candidate z loses pairwise election with every other candidate by $4k$ votes. For any two candidates $a, b \in \mathcal{C}$, either a and b ties or one wins pairwise election against the other one by margin of either $2k + 2$ or of $4k$ or of $8k$. We denote the margin by which a candidate a defeats b, by $d(a, b)$.

The R-UCM problem for the maximin voting rule is NP-complete [XZP$^+$09], when we have $k > 1$.

Theorem 9.5. *The CPMW problem for the maximin voting rule is* NP-*complete, for a coalition of size at least* 2.

Proof. Clearly the CPMW problem for maximin voting rule is NP. We provide a many-one reduction from the R-UCM problem for the maximin voting rule to it. Given a R-UCM problem instance, we define a CPMW problem instance $\Gamma = (\mathcal{C}', (\succ'_i)_{i \in \mathcal{V}}, M')$ as follows.

$$\mathcal{C}' := \mathcal{C} \cup \{w, d_1, d_2, d_3\}$$

We define $\mathcal{V}'$ such that $d(a, b)$ is the same as the R-UCM instance, for all $a, b \in \mathcal{C}$ and $d(d_1, w) = 2k + 2, d(d_1, d_2) = 8k, d(d_2, d_3) = 8k, d(d_3, d_1) = 8k$. The existence of such

219

a $\mathcal{V}'$ is guaranteed from Lemma [7.2](7.2). Moreover, Lemma [7.2](7.2) also ensures that $|\mathcal{V}'|$ is $O(mc)$. The votes of the voters in M is $w \succ \ldots$. Thus the *current winner* is w. The *actual winner* is defined to be z. The tie breaking rule is $\succ_t = w \succ z \succ \ldots$, where z is the candidate whom the manipulators in M want to make winner in the R-UCM problem instance. Clearly this reduction takes polynomial amount of time. Now we show that, M is a coalition of possible manipulators *iff* z can be made a winner.

The *if* part is as follows. Let $\succ_i, i \in M$ be the votes that make z win. We can assume that z is the most preferred candidate in all the preferences $\succ_i, i \in M$. Now consider the preferences for the voters in M is follows.

$$\succ_i' := d_1 \succ d_2 \succ d_3 \succ w \succ_i, i \in M$$

The score of every candidate in $\mathcal{C}$ is not more than z. The score of z is $-3k$. The score of w is $-3k-2$ and the scores of $d_1, d_2,$ and d_3 are less than $-3k$. Hence, M is a coalition of possible manipulators with the actual preferences $\succ_i' := d_1 \succ d_2 \succ d_3 \succ w \succ_i, i \in M$.

The *only if* part is as follows. Suppose M is a coalition of possible manipulators with actual preferences $\succ_i', i \in M$. Consider the preferences $\succ_i', i \in M$, but restricted to the set $\mathcal{C}$ only. Call them $\succ_i, i \in M$. We claim that $\succ_i, i \in M$ with the votes from $\mathcal{V}$ makes z win the election. If not then, there exists a candidate, say $a \in \mathcal{C}$, whose score is strictly more than the score of z - this is so because the tie breaking rule is in favor of z. But this contradicts the fact that z wins the election when the voters in M vote $\succ_i', i \in M$ along with the votes from $\mathcal{V}'$. $\qquad\square$

9.3.3 Bucklin Voting Rule

In this subsection, we design polynomial time algorithms for both the CPMW and the CPM problems for the Bucklin voting rule. Again, we begin by showing that if there are profiles witnessing manipulation, then there exist profiles that do so with some additional structure, which we will exploit subsequently in our algorithm.

Lemma 9.3. *Consider a preference profile $(\succ_i)_{i \in \mathcal{V}}$, where x is the winner with respect to the Bucklin voting rule. Suppose a subset of voters $M \subset \mathcal{V}$ forms a coalition of possible manipulators. Let y be the actual winner. Then there exist preferences $(\succ_i')_{i \in M}$ such that y is a Bucklin winner in $((\succ_i)_{i \in \mathcal{V} \setminus M}, (\succ_i')_{i \in M})$, and further:*

1. y *immediately follows* x *in each* $\succ'_i$.

2. *The rank of* x *in each* $\succ'_i$ *is in one of the following - first,* $b(y) - 1$, $b(y)$, $b(y) + 1$, *where* $b(y)$ *be the Bucklin score of* y *in* $((\succ_i)_{i \in \mathcal{V} \setminus \mathcal{M}}, (\succ'_i)_{i \in \mathcal{M}})$.

Proof. From Definition 9.3, y's rank must be worse than x's rank in each $\succ'_i$. We now exchange the position of y with the candidate which immediately follows x in $\succ'_i$. This process does not decrease Bucklin score of any candidate except possibly y's, and x's score does not increase. Hence y will continue to win and thus $\succ'_i$ satisfies the first condition.

Now to begin with, we assume that $\succ'_i$ satisfies the first condition. If the position of x in $\succ'_i$ is $b(y) - 1$ or $b(y)$, we do not change it. If x is above $b(y) - 1$ in $\succ'_i$, then move x and y at the first and second positions respectively. Similarly if x is below $b(y) + 1$ in $\succ'_i$, then move x and y at the $b(y) + 1$ and $b(y) + 2$ positions respectively. This process does not decrease score of any candidate except y because the Bucklin score of x is at least $b(y)$. The transformation cannot increase the score y since its position has only been improved. Hence y continues to win and thus $\succ'_i$ satisfies the second condition. □

Lemma 9.3 leads us to the following theorem.

Theorem 9.6. *The CPMW problem and the CPM problems for Bucklin voting rule are in* P *for any coalition of size. Therefore, by Observation 9.1, the CPMSW and the CPMS problems are in* P *when the maximum coalition size* $k = O(1)$.

Proof. Observation 9.1 says that it is enough to prove that the CPMW problem is in P. Let x be the *current winner* and y be the given *actual winner*. For any final Bucklin score $b(y)$ of y, there are polynomially many possibilities for the positions of x and y in the profile of $\succ_i, i \in \mathcal{M}$, since Bucklin voting rule is anonymous. Once the positions of x and y is fixed, we try to fill the top $b(y)$ positions of each $\succ'_i$ - place a candidate in an empty position above $b(y)$ in any $\succ'_i$ if doing so does not make y lose the election. If we are able to successfully fill the top $b(y)$ positions of all $\succ'_i$ for all $i \in \mathcal{M}$, then $\mathcal{M}$ is a coalition of possible manipulators. If the above process fails for all possible above mentioned positions of x and y and all possible guesses of $b(y)$, then $\mathcal{M}$ is not a coalition of possible manipulators. Clearly the above algorithm runs in *poly(m,n)* time.

The proof of correctness is as follows. If the algorithm outputs that $\mathcal{M}$ is a coalition of possible manipulators, then it actually has constructed $\succ'_i$ for all $i \in \mathcal{M}$ with respect to which they form a coalition of possible manipulators. On the other hand, if they form a coalition of

possible manipulators, then Lemma 9.3 ensures that our algorithm explores all the sufficient positions of x and y in $\succ_i'$ for all $i \in M$. Now if M is a possible coalition of manipulators, then the corresponding positions for x and y have also been searched. Our greedy algorithm must find it since permuting the candidates except x and z which are ranked above $b(y)$ in $\succ_i'$ cannot stop y to win the election since the Bucklin score of other candidates except y is at least $b(y)$. $\qquad\square$

9.3.4 STV Voting Rule

Next we prove that the CPMW and the CPM problems for STV rule is NP-complete. To this end, we reduce from the Exact Cover by 3-Sets Problem (X3C), which is known to be NP-complete [GJ79]. The X3C problem is as follows.

Definition 9.9. *(X3C Problem)*
Given a set S of cardinality n and m subsets $S_1, S_2, \ldots, S_m \subset S$ with $|S_i| = 3, \forall i = 1, \ldots, m$, does there exist an index set $I \subseteq \{1, \ldots, m\}$ with $|I| = \frac{|S|}{3}$ such that $\cup_{i \in I} S_i = S$.

Theorem 9.7. *The CPM problem for STV rule is NP-complete even for a coalition of size 1.*

Proof. Clearly the problem is NP. To show NP hardness, we show a many-one reduction from the X3C problem to it. The reduction is analogous to the reduction given in [BO91]. Given an X3C instance, we construct an election as follows. The unspecified positions can be filled in any arbitrary way. The candidate set is as follows.

$$
\begin{aligned}
\mathcal{C} = \{x, y\} \ &\cup \ \{a_1, \ldots, a_m\} \cup \{\overline{a}_1, \ldots, \overline{a}_m\} \\
&\cup \ \{b_1, \ldots, b_m\} \cup \{\overline{b}_1, \ldots, \overline{b}_m\} \\
&\cup \ \{d_0, \ldots, d_n\} \cup \{g_1, \ldots, g_m\}
\end{aligned}
$$

The votes are as follows.

▷ $12m$ votes for $y \succ x \succ \ldots$

▷ $12m - 1$ votes for $x \succ y \succ \ldots$

▷ $10m + \frac{2n}{3}$ votes for $d_0 \succ x \succ y \succ \ldots$

▷ $12m - 2$ votes for $d_i \succ x \succ y \succ \ldots, \forall i \in [n]$

▷ $12m$ votes for $g_i \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ $6m + 4i - 5$ votes for $b_i \succ \overline{b}_i \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ 2 votes for $b_i \succ d_j \succ x \succ y \succ \ldots, \forall i \in [m], \forall j \in S_i$

▷ $6m + 4i - 1$ votes for $\overline{b}_i \succ b_i \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ 2 votes for $\overline{b}_i \succ d_0 \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ $6m + 4i - 3$ votes for $a_i \succ g_i \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ 1 vote for $a_i \succ b_i \succ g_i \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ 2 votes for $a_i \succ \overline{a}_i \succ g_i \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ $6m + 4i - 3$ votes for $\overline{a}_i \succ g_i \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ 1 vote for $\overline{a}_i \succ \overline{b}_i \succ g_i \succ x \succ y \succ \ldots, \forall i \in [m]$

▷ 2 votes for $\overline{a}_i \succ a_i \succ g_i \succ x \succ y \succ \ldots, \forall i \in [m]$

The tie breaking rule is $\succ_t = \cdots \succ x$. The vote of v is $x \succ \cdots$. We claim that v is a possible manipulator *iff* the X3C is a *yes* instance. Notice that, of the first $3m$ candidates to be eliminated, $2m$ of them are $a_1, \ldots, a_m$ and $\overline{a}_1, \ldots, \overline{a}_m$. Also exactly one of b_i and $\overline{b}_i$ will be eliminated among the first $3m$ candidates to be eliminated because if one of b_i, $\overline{b}_i$ then the other's score exceeds $12m$. We show that the winner is either x or y irrespective of the vote of one more candidate. Let $J := \{j : b_j \text{ is eliminated before } \overline{b}_j\}$. If J is an index of set cover then the winner is y. This can be seen as follows. Consider the situation after the first $3m$ eliminations. Let $i \in S_j$ for some $j \in J$. Then b_j has been eliminated and thus the score of d_i is at least $12m$. Since J is an index of a set cover, every d_i's score is at least $12m$. Notice that $\overline{b}_j$ has been eliminated for all $j \notin J$. Thus the revised score of d_0 is at least $12m$. After the first $3m$ eliminations, the remaining candidates are $x, y, \{d_i : i \in [n]\}, \{g_i : i \in [m]\}, \{b_j : j \notin J\}, \{\overline{b}_j : j \in J\}$. All the remaining candidates except x has score at least $12m$ and x's score is $12m - 1$. Hence x will be eliminated next which makes y's score at least $24m - 1$. Next d_i's will get eliminated which will in turn make y's score $(12n + 36)m - 1$. At this point g_i's score is at most $32m$. Also all the remaining b_i and $\overline{b}_i$'s score is at most $32m$. Since each of the remaining candidate's scores gets transferred to y once they are eliminated, y is the winner.

Now we show that, if J is not an index of set cover then the winner is x. This can be seen as follows. If $|J| > \frac{n}{3}$, then the number of $\overline{b}_j$ that gets eliminated in the first $3m$ iterations

is less than $m - \frac{n}{3}$. This makes the score of d_0 at most $12m - 2$. Hence d_0 gets eliminated before x and all its scores gets transferred to x. This makes the elimination of x impossible before y and makes x the winner of the election.

If $|J| \leqslant \frac{n}{3}$ and there exists an $i \in S$ that is not covered by the corresponding set cover, then d_i gets eliminated before x with a score of $12m - 2$ and its score gets transferred to x. This makes x win the election.

Hence y can win *iff* X3C is a *yes* instance. Also notice that if y can win the election, then it can do so with the voter v voting a preference like $\cdots \succ x \succ y \succ \cdots$. $\qquad \square$

From the proof of the above theorem, we have the following corollary by specifying y as the *actual winner* for the CPMW problem.

Corollary 9.2. *The CPMW problem for STV rule is* NP-*complete even for a coalition of size* 1.

9.4 Conclusion

In this work, we have initiated a promising research direction for detecting possible instances of manipulation in elections. We have proposed the notion of *possible* manipulation and explored several concrete computational problems, which we believe to be important in the context of voting theory. These problems involve identifying if a given set of voters are possible manipulators (with or without a specified candidate winner). We have also studied the search versions of these problems, where the goal is to simply detect the presence of possible manipulation with the maximum coalition size. We believe there is theoretical as well as practical interest in studying the proposed problems. We have provided algorithms and hardness results for these problems for many common voting rules. It is our conviction that both the problems that we have studied here have initiated an interesting research direction with significant promise and potential for future work.

In the next chapter of the thesis, we study another interesting form of election control called bribery.

Chapter 10

Frugal Bribery

Bribery in elections is an important problem in computational social choice theory. We introduce and study two important special cases of the classical $BRIBERY problem, namely, FRUGAL-BRIBERY and FRUGAL-$BRIBERY where the briber is frugal in nature. By this, we mean that the briber is only able to influence voters who benefit from the suggestion of the briber. More formally, a voter is *vulnerable* if the outcome of the election improves according to her own preference when she accepts the suggestion of the briber. In the FRUGAL-BRIBERY problem, the goal is to make a certain candidate win the election by changing *only* the vulnerable votes. In the FRUGAL-$BRIBERY problem, the vulnerable votes have prices and the goal is to make a certain candidate win the election by changing only the vulnerable votes, subject to a budget constraint. We further formulate two natural variants of the FRUGAL-$BRIBERY problem namely UNIFORM-FRUGAL-$BRIBERY and NONUNIFORM-FRUGAL-$BRIBERY where the prices of the vulnerable votes are, respectively, all the same or different.

We observe that, even if we have only a small number of candidates, the problems are intractable for all voting rules studied here for weighted elections, with the sole exception of the FRUGAL-BRIBERY problem for the plurality voting rule. In contrast, we have polynomial time algorithms for the FRUGAL-BRIBERY problem for plurality, veto, k-approval, k-veto, and plurality with runoff voting rules for unweighted elections. However, the FRUGAL-$BRIBERY problem is intractable for all the voting rules studied here barring the

plurality and the veto voting rules for unweighted elections. These intractability results demonstrate that bribery is a hard computational problem, in the sense that several special cases of this problem continue to be computationally intractable. This strengthens the view that bribery, although a possible attack on an election in principle, may be infeasible in practice.

10.1   Introduction

Activities that try to influence voter opinions, in favor of specific candidates, are very common during the time that an election is in progress. For example, in a political election, candidates often conduct elaborate campaigns to promote themselves among a general or targeted audience. Similarly, it is not uncommon for people to protest against, or rally for, a national committee or court that is in the process of approving a particular policy. An extreme illustration of this phenomenon is *bribery* — here, the candidates may create financial incentives to sway the voters. Of course, the process of influencing voters may involve costs even without the bribery aspect; for instance, a typical political campaign or rally entails considerable expenditure.

All situations involving a systematic attempt to influence voters usually have the following aspects: an external agent, a candidate that the agent would like to be the winner, a budget constraint, a cost model for a change of vote, and knowledge of the existing election. The formal computational problem that arises from these inputs is the following: is it possible to make a distinguished candidate win the election in question by incurring a cost that is within the budget? This question, with origins in [FHH06, FHH09, FHHR09], has been subsequently studied intensely in computational social choice literature. In particular, bribery has been studied under various cost models, for example, uniform price per vote which is known as \$BRIBERY [FHH06], nonuniform price per vote [Fal08], nonuniform price per shift of the distinguished candidate per vote which is called SHIFT BRIBERY, nonuniform price per swap of candidates per vote which is called SWAP BRIBERY [EFS09]. A closely related problem known as campaigning has been studied for various vote models, for example, truncated ballots [BFLR12], soft constraints [PRV13], CP-nets [DK16], combinatorial domains [MPVR12] and probabilistic lobbying [BEF$^+$14]. The bribery problem has also been studied under voting rule uncertainty [EHH14]. Faliszewski et al. [FRRS14] study the complexity of bribery in Bucklin and Fallback voting rules. Xia [Xia12] studies destructive bribery, where the goal

of the briber is to change the winner by changing minimum number of votes. Dorn et al. [DS12] studies the parameterized complexity of the SWAP BRIBERY problem and Bredereck et al. [BCF+14b] explores the parameterized complexity of the SHIFT BRIBERY problem for a wide range of parameters. We recall again that the costs and the budgets involved in all the bribery problems above need not necessarily correspond to actual money traded between voters and candidates. They may correspond to any cost in general, for example, the amount of effort or time that the briber needs to spend for each voter.

10.1.1 Motivation

In this work, we propose an effective cost model for the bribery problem. Even the most general cost models that have been studied in the literature fix absolute costs per voter-candidate combination, with no specific consideration to the voters' opinions about the current winner and the distinguished candidate whom the briber wants to be the winner. In our proposed model, a change of vote is relatively easier to effect if the change causes an outcome that the voter would find desirable. Indeed, if the currently winning candidate is, say, a, and a voter is (truthfully) promised that by changing her vote from $c \succ d \succ b \succ a$ to $d \succ b \succ c \succ a$, the winner of the election would change from a to d, then this is a change that the voter is likely to be happy to make. While the change does not make her most favorite candidate win the election, it does improve the result from her point of view. Thus, given the circumstances (namely that of her least favorite candidate winning the election), the altered vote serves the voter better than the original one.

We believe this perspective of voter influence is an important one to study. The cost of a change of vote is proportional to the nature of the outcome that the change promises — the cost is low or nil if the change results in a better outcome with respect to the voter's original ranking, and high or infinity otherwise. A frugal agent only approaches voters of the former category, thus being able to effectively bribe with minimal or no cost. Indeed the behavior of agents in real life is often frugal. For example, consider campaigners in favor of a relatively smaller party in a political election. They may actually target only vulnerable voters due to lack of human and other resources they have at their disposal.

More formally, let c be the winner of an election and p (other than c) the candidate whom the briber wishes to make the winner of the election. Now the voters who prefer c to p will be reluctant to change their votes, and we call these votes *non-vulnerable with respect to* p — we do not allow these votes to be changed by the briber, which justifies the *frugal* nature of

the briber. On the other hand, if a voter prefers p to c, then it may be very easy to convince her to change her vote if doing so makes p win the election. We name these votes *vulnerable with respect to* p. When the candidate p is clear from the context, we simply call these votes non-vulnerable and vulnerable, respectively.

The computational problem is to determine whether there is a way to make a candidate p win the election by changing *only* those votes that are vulnerable with respect to p. We call this problem FRUGAL-BRIBERY. Note that there is no cost involved in the FRUGAL-BRIBERY problem — the briber does not incur any cost to change the votes of the vulnerable votes. We also extend this basic model to a more general setting where each vulnerable vote has a certain nonnegative integer price which may correspond to the effort involved in approaching these voters and convincing them to change their votes. We also allow for the specification of a budget constraint, which can be used to enforce auxiliary constraints. This leads us to define the FRUGAL-$BRIBERY problem, where we are required to find a subset of vulnerable votes with a total cost that is within a given budget, such that these votes can be changed in some way to make the candidate p win the election. Note that the FRUGAL-$BRIBERY problem can be either uniform or nonuniform depending on whether the prices of the vulnerable votes are all identical or different. If not mentioned otherwise, the prices of the vulnerable votes will be assumed to be nonuniform. We remind that the briber is not allowed to change the non-vulnerable votes in both the FRUGAL-BRIBERY and the FRUGAL-$BRIBERY problems.

10.1.2   Our Contribution

Our primary contribution in this work is to formulate and study two important and natural models of bribery which turn out to be special cases of the well studied $BRIBERY problem in elections. Our results show that both the FRUGAL-BRIBERY and the FRUGAL-$BRIBERY problems are intractable for many commonly used voting rules for weighted as well as unweighted elections, barring a few exceptions. These intractability results can be interpreted as an evidence that the bribery in elections is a hard computational problem in the sense that even many of its important and natural special cases continue to be intractable. Thus bribery, although a possible attack on elections in principle, may be practically not viable. From a more theoretical perspective, our intractability results strengthen the existing hardness results for the $BRIBERY problem. On the other hand, our polynomial time algorithms exhibit interesting tractable special cases of the $BRIBERY problem.

Our Results for Unweighted Elections

We have the following results for unweighted elections.

▷ The FRUGAL-BRIBERY problem is in P for the k-approval, Bucklin, and plurality with runoff voting rules. Also, the FRUGAL-\$BRIBERY problem is in P for the plurality and veto voting rules. In contrast, the FRUGAL-\$BRIBERY problem is NP-complete for the Borda, maximin, Copeland, and STV voting rules [Observation 10.3].

▷ The FRUGAL-BRIBERY problem is NP-complete for the Borda voting rule [Theorem 10.1]. The FRUGAL-\$BRIBERY is NP-complete for the k-approval for any constant $k \geqslant 5$ [Theorem 10.2], k-veto for any constant $k \geqslant 3$ [Theorem 10.3], and a wide class of scoring rules [Theorem 10.5] even if the price of every vulnerable vote is either 1 or ∞. Moreover, the UNIFORM-FRUGAL-\$BRIBERY is NP-complete for the Borda voting rule even if all the vulnerable votes have a uniform price of 1 and the budget is 2 [Theorem 10.6].

▷ The FRUGAL-\$BRIBERY problem is in P for the k-approval, Bucklin, and plurality with runoff voting rules when the budget is a constant [Theorem 10.4].

Our Results for Weighted Elections

We have the following results for weighted elections.

▷ The FRUGAL-BRIBERY problem is in P for the maximin and Copeland voting rules when we have only three candidates [Observation 10.4], and for the plurality voting rule for any number of candidates [Theorem 10.7].

▷ The FRUGAL-BRIBERY problem is NP-complete for the STV [Theorem 10.10], plurality with runoff [Corollary 10.1], and every scoring rule except the plurality voting rule [Observation 10.5] for three candidates. The FRUGAL-\$BRIBERY problem is NP-complete for the plurality voting rule for three candidates [Theorem 10.8].

▷ When we have only four candidates, the FRUGAL-BRIBERY problem is NP-complete for the maximin [Theorem 10.9], Bucklin [Theorem 10.12], and Copeland [Theorem 10.13] rules.

We summarize the results in the Table 10.1.

Voting Rules	Unweighted		Weighted	
	FRUGAL-BRIBERY	FRUGAL-$BRIBERY	FRUGAL-BRIBERY	FRUGAL-$BRIBERY
Plurality	P [Observation 10.1]	P [Observation 10.2]	P [Theorem 10.7]	NP-complete [Theorem 10.8]
Veto	P [Observation 10.1]	P [Observation 10.2]	NP-complete [Observation 10.5]	NP-complete [Observation 10.5]
k-approval	P [Observation 10.1]	NP-complete* [Theorem 10.2]	NP-complete$^\diamond$ [Observation 10.5]	NP-complete [Observation 10.5]
k-veto	P [Observation 10.1]	NP-complete$^\bullet$ [Theorem 10.3]	NP-complete$^\diamond$ [Observation 10.5]	NP-complete [Observation 10.5]
Borda	NP-complete [Theorem 10.1]	NP-complete† [Theorem 10.5]	NP-complete [Observation 10.5]	NP-complete [Observation 10.5]
Runoff	P [Observation 10.1]	?	NP-complete [Corollary 10.1]	NP-complete [Corollary 10.1]
Maximin	?	NP-complete [Observation 10.3]	NP-complete [Theorem 10.9]	NP-complete [Theorem 10.9]
Copeland	?	NP-complete [Observation 10.3]	NP-complete [Theorem 10.13]	NP-complete [Theorem 10.13]
STV	?	NP-complete [Observation 10.3]	NP-complete [Theorem 10.10]	NP-complete [Theorem 10.10]

Table 10.1: $\star$- The result holds for $k \geqslant 5$.
•- The result holds for $k \geqslant 3$.
†- The result holds for a much wider class of scoring rules.
$\diamond$- The results do not hold for the plurality voting rule.
?- The problem is open.

10.1.3   Related Work

The pioneering work of [FHH06] defined and studied the $BRIBERY problem wherein, the input is a set of votes with prices for each vote and the goal is to make some distinguished candidate win the election, subject to a budget constraint of the briber. The FRUGAL-$BRIBERY problem is the $BRIBERY problem with the restriction that the price of every non-vulnerable vote is infinite. Also, the FRUGAL-BRIBERY problem is a special case of the FRUGAL-$BRIBERY problem. Hence, whenever the $BRIBERY problem is computationally easy in a setting, both the FRUGAL-BRIBERY and the FRUGAL-$BRIBERY problems are also computationally easy (see Proposition 10.1 for a more formal proof). However, the $BRIBERY problem is computationally intractable in most of the settings. This makes the study of important special cases such as FRUGAL-BRIBERY and FRUGAL-$BRIBERY, interesting. We note that a notion similar to vulnerable votes has been studied in the context of dominating manipulation by [CWX11]. Hazon et al. [HLK13] introduced and studied PERSUASION and k-PERSUASION problems where an external agent suggests votes to vulnerable voters which are beneficial for the vulnerable voters as well as the external agent. It turns out that the PERSUASION and the k-PERSUASION problems Turing reduce to the FRUGAL-BRIBERY and the FRUGAL-$BRIBERY problems respectively (see Proposition 10.3). Therefore, the polynomial time algorithms we propose in this work imply polynomial time algorithms for the persuasion analog. On the other hand, since the reduction in Proposition 10.3 from PERSUASION to FRUGAL-BRIBERY is a Turing reduction, the existing NP-completeness results for the persuasion problems do not imply NP-completeness results for the corresponding frugal bribery variants. We refer to [RR67] for Turing reductions.

10.2   Problem Definition

In all the definitions below, r is a fixed voting rule. We define the notion of vulnerable votes as follows. Intuitively, the vulnerable votes are those votes whose voters can easily be persuaded to change their votes since doing so will result in an outcome that those voters prefer over the current one.

Definition 10.1. *(Vulnerable votes)*
Given a voting rule r, a set of candidates $\mathcal{C}$, a profile of votes $\succ = (\succ_1, \ldots, \succ_n)$, and a distinguished candidate p, we say a vote $\succ_i$ is p-vulnerable if $p \succ_i r(\succ)$.

Recall that, whenever the distinguished candidate is clear from the context, we drop it from the notation. With the above definition of vulnerable votes, we formally define the FRUGAL-BRIBERY problem as follows. Intuitively, the problem is to determine whether a particular candidate can be made winner by changing only the vulnerable votes.

Definition 10.2. *(r-FRUGAL-BRIBERY)*
Given a preference profile $\succ = (\succ_1, \ldots, \succ_n)$ over a candidate set $\mathcal{C}$, and a candidate p, determine if there is a way to make p win the election by changing only the vulnerable votes.

Next we generalize the FRUGAL-BRIBERY problem to the FRUGAL-\$BRIBERY problem which involves prices for the vulnerable votes and a budget for the briber. Intuitively, the price of a vulnerable vote v is the cost the briber incurs to change the vote v.

Definition 10.3. *(r-FRUGAL-\$BRIBERY)*
Let $\succ = (\succ_1, \ldots, \succ_n)$ be a preference profile over a candidate set $\mathcal{C}$. We are given a candidate p, a finite budget $b \in \mathbb{N}$, and a price function $c : [n] \longrightarrow \mathbb{N} \cup \{\infty\}$ such that $c(i) = \infty$ if $\succ_i$ is not a p-vulnerable vote. The goal is to determine if there exist p vulnerable votes $\succ_{i_1}, \ldots, \succ_{i_\ell} \in \succ$ and votes $\succ'_{i_1}, \ldots, \succ'_{i_\ell} \in \mathcal{L}(\mathcal{C})$ such that:

(a) the total cost of the chosen votes is within the budget, that is, $\sum_{j=1}^{\ell} c(i_j) \leqslant b$, and

(b) the new votes make the desired candidate win, that is, $r(\succ_{[n] \setminus \{i_1, \ldots, i_\ell\}}, \succ'_{i_1}, \ldots, \succ'_{i_\ell}) = p$.

The special case of the problem when the prices of all the vulnerable votes are the same is called UNIFORM-FRUGAL-\$BRIBERY. *We refer to the general version as* NONUNIFORM-FRUGAL-\$BRIBERY. *If not specified,* FRUGAL-\$BRIBERY *refers to the nonuniform version.*

The above problems are important special cases of the well studied \$BRIBERY problem. Also, the COALITIONAL-MANIPULATION problem [BTT89, CSL07], one of the classic problems in computational social choice theory, turns out to be a special case of the FRUGAL-\$BRIBERY problem [see Proposition 10.1].

For the sake of completeness, we include the definitions of these problems here.

Definition 10.4. *(r-\$BRIBERY) [FHH09]*
Given a preference profile $\succ = (\succ_1, \ldots, \succ_n)$ over a set of candidates $\mathcal{C}$, a distinguished candidate p, a price function $c : [n] \longrightarrow \mathbb{N} \cup \{\infty\}$, and a budget $b \in \mathbb{N}$, determine if there a way to make p win the election.

Definition 10.5. (COALITIONAL-MANIPULATION) *[BTT89, CSL07]*
Given a preference profile $\succ^t = (\succ_1, \ldots, \succ_n)$ of truthful voters over a set of candidates $\mathcal{C}$, an integer ℓ, and a distinguished candidate p, determine if there exists a ℓ voter preference profile $\succ^\ell$ such that the candidate p wins uniquely (does not tie with any other candidate) in the profile $(\succ^t, \succ^\ell)$.

The following proposition shows the relationship among the above problems. Proposition 10.1 to 10.3 below hold for both weighted and unweighted elections.

Proposition 10.1. *For every voting rule,* FRUGAL-BRIBERY $\leqslant_P$ UNIFORM-FRUGAL-\$BRIBERY $\leqslant_P$ NONUNIFORM-FRUGAL-\$BRIBERY $\leqslant_P$ \$BRIBERY. *Also,* COALITIONAL-MANIPULATION $\leqslant_P$ NONUNIFORM-FRUGAL-\$BRIBERY.

Proof. In the reductions below, let us assume that the election to start with is a weighted election. Since we do not change the weights of any vote in the reduction and since there is a natural one to one correspondence between the votes of the original instance and the reduced instance, the proof also works for unweighted elections.

Given a FRUGAL-BRIBERY instance, we construct a UNIFORM-FRUGAL-\$BRIBERY instance by defining the price of every vulnerable vote to be zero and the budget to be zero. Clearly, the two instances are equivalent. Hence, FRUGAL-BRIBERY $\leqslant_P$ UNIFORM-FRUGAL-\$BRIBERY.

UNIFORM-FRUGAL-\$BRIBERY $\leqslant_P$ NONUNIFORM-FRUGAL-\$BRIBERY $\leqslant_P$ \$BRIBERY follows from the fact that UNIFORM-FRUGAL-\$BRIBERY is a special case of NONUNIFORM-FRUGAL-\$BRIBERY which in turn is a special case of \$BRIBERY.

Given a COALITIONAL-MANIPULATION instance, we construct a NONUNIFORM-FRUGAL-\$BRIBERY instance as follows. Let p be the distinguished candidate of the manipulators and $\succ_f = p \succ \text{others}$ be any arbitrary but fixed ordering of the candidates given in the COALITIONAL-MANIPULATION instance. Without loss of generality, we can assume that p does not win if all the manipulators vote $\succ_f$ (Since, this is a polynomially checkable case of COALITIONAL-MANIPULATION). We define the vote of the manipulators to be $\succ_f$, the distinguished candidate of the campaigner to be p, the budget of the campaigner to be zero, the price of the manipulators to be zero (notice that all the manipulators' votes are p-vulnerable), and the price of the rest of the vulnerable votes to be one. Clearly, the two instances are equivalent. Hence, COALITIONAL-MANIPULATION $\leqslant_P$ NONUNIFORM-FRUGAL-\$BRIBERY. $\qquad\square$

Also, the FRUGAL-BRIBERY problem reduces to the COALITIONAL-MANIPULATION problem by simply making all vulnerable votes to be manipulators.

Proposition 10.2. *For every voting rule,* FRUGAL-BRIBERY $\leqslant_P$ COALITIONAL-MANIPULATION.

We can also establish the following relationship between the PERSUASION (respectively k-PERSUASION) problem and the FRUGAL-BRIBERY (respectively FRUGAL-$BRIBERY) problem. The persuasions differ from the corresponding frugal bribery variants in that the briber has her own preference order, and desires to improve the outcome of the election with respect to her preference order. The following proposition is immediate from the definitions of the problems.

Proposition 10.3. *For every voting rule, there is a Turing reduction from* PERSUASION *(respectively* k-PERSUASION*) to* FRUGAL-BRIBERY *(respectively* FRUGAL-$BRIBERY*).*

Proof. Given an algorithm for the FRUGAL-BRIBERY problem, we iterate over all possible distinguished candidates to have an algorithm for the persuasion problem.

Given an algorithm for the FRUGAL-$BRIBERY problem, we iterate over all possible distinguished candidates and fix the price of the corresponding vulnerables to be one to have an algorithm for the k-persuasion problem. □

10.3 Results for Unweighted Elections

Now we present the results for unweighted elections. We begin with some easy observations that follow from known results.

Observation 10.1. *The* FRUGAL-BRIBERY *problem is in* P *for the* k*-approval voting rule for any* k*, Bucklin, and plurality with runoff voting rules.*

Proof. The COALITIONAL-MANIPULATION problem is in P for these voting rules [XZP⁺09]. Hence, the result follows from Proposition 10.2. □

Observation 10.2. *The* FRUGAL-$BRIBERY *problem is in* P *for the plurality and veto voting rules.*

Proof. The $BRIBERY problem is in P for the plurality [FHH06] and veto [Fal08] voting rules. Hence, the result follows from Proposition 10.1. □

Observation 10.3. *The* FRUGAL-$BRIBERY *problem is* NP*-complete for Borda, maximin, Copeland, and STV voting rules.*

Proof. The COALITIONAL-MANIPULATION problem is NP-complete for the above voting rules. Hence, the result follows from Proposition 10.1. □

We now present our main results. We begin with showing that the FRUGAL-BRIBERY problem for the Borda voting rule and the FRUGAL-$BRIBERY problem for various scoring rules are NP-complete. To this end, we reduce from the PERMUTATION SUM problem, which is known to be NP-complete [YHL04]. The PERMUTATION SUM problem is defined as follows.

PERMUTATION SUM

Input: n integers $X_i, i \in [n]$ with $1 \leqslant X_i \leqslant 2n$ for every $i \in [n]$ and $\sum_{i=1}^{n} X_i = n(n+1)$.
Question: Do there exist two permutations π and σ of $[n]$ such that $\pi(i) + \sigma(i) = X_i$ for every $i \in [n]$?

We now prove that the FRUGAL-BRIBERY problem is NP-complete for the Borda voting rule, by a reduction from PERMUTATION SUM. Our reduction is inspired by the reduction used by Davies et al. [DKNW11] and Betzler et al. [BNW11] to prove NP-completeness of the COALITIONAL-MANIPULATION problem for the Borda voting rule.

Theorem 10.1. *The* FRUGAL-BRIBERY *problem is* NP-*complete for the Borda voting rule.*

Proof. The problem is clearly in NP. To show NP-hardness, we reduce an arbitrary instance of the PERMUTATION SUM problem to the FRUGAL-BRIBERY problem for the Borda voting rule. Let $(X_1, \ldots, X_n)$ be an instance of the PERMUTATION SUM problem. Without loss of generality, let us assume that n is an odd integer – if n is an even integer, then we consider the instance $(X_1, \ldots, X_n, X_{n+1} = 2(n+1))$ which is clearly equivalent to the instance $(X_1, \ldots, X_n)$.

We define a FRUGAL-BRIBERY instance $(\mathcal{C}, \mathcal{P}, p)$ as follows. The candidate set is:

$$\mathcal{C} = \mathcal{X} \uplus D \uplus \{p, c\}, \text{ where } \mathcal{X} = \{\mathfrak{x}_i : i \in [n]\} \text{ and } |D| = 3n - 1$$

Note that the total number of candidates is $4n + 1$, and therefore the Borda score of a candidate placed at the top position is $4n$.

Before describing the votes, we give an informal overview of how the reduction will proceed. The election that we define will consist of exactly two vulnerable votes. Note that when placed at the top position in these two votes, the distinguished candidate p gets a score of $8n$ ($4n$ from each vulnerable vote). We will then add non-vulnerable votes, which will be

designed to ensure that, among them, the score of $\mathfrak{x}_i$ is $8n - X_i$ more than the score of the candidate p. Using the "dummy candidates", we will also be able to ensure that the candidates $\mathfrak{x}_i$ receive (without loss of generality) scores between 1 and n from the modified vulnerable votes.

Now suppose these two vulnerable votes can be modified to make p win the election. Let s_1 and s_2 be the scores that $\mathfrak{x}_i$ obtains from these altered vulnerable votes. It is clear that for p to emerge as a winner, $s_1 + s_2$ must be at most X_i. Since the Borda scores for the candidates in $\mathcal{X}$ range from 1 to n in the altered vulnerable votes, the total Borda score that all the candidates in $\mathcal{X}$ can accumulate from two altered vulnerable votes is $n(n+1)$. On the other hand, since the sum of the X_i's is also $n(n + 1)$, it turns out that $s_1 + s_2$ must in fact be equal to X_i for the candidate p to win. From this point, it is straightforward to see how the permutations σ and π can be inferred from the modified vulnerable votes: $\sigma(i)$ is given by the score of the candidate $\mathfrak{x}_i$ from the first vote, while $\pi(i)$ is the score of the candidate $\mathfrak{x}_i$ from the second vote. These functions turn out to be permutations because these n candidates receive n distinct scores from these votes.

We are now ready to describe the construction formally. We remark that instead of $8n - X_i$, as described above, we will maintain a score difference of either $8n - X_i$ or $8n - X_i - 1$ depending on whether X_i is even or odd respectively — this is a minor technicality that comes from the manner in which the votes are constructed and does not affect the overall spirit of the reduction.

Let us fix any arbitrary order $\succ_f$ among the candidates in $\mathcal{X} \uplus D$. For any subset $A \subset \mathcal{X} \uplus D$, let $\overrightarrow{A}$ be the ordering among the candidates in A as defined in $\succ_f$ and $\overleftarrow{A}$ the reverse order of $\overrightarrow{A}$. For each $i \in [n]$, we add two votes v_i^j and $v_i^{j'}$ as follows for every $j \in [4]$. Let ℓ denote $|D| = 3n - 1$. Also, for $d \in D$, let $D_i, D_{\ell/2} \subset D \setminus \{d\}$ be such that:

$$|D_i| = \ell/2 + n + 1 - \lceil X_i/2 \rceil \text{ and } |D_{\ell/2}| = \ell/2.$$

$$v_i^j : \begin{cases} c \succ p \succ d \succ \overrightarrow{\mathcal{C} \setminus (\{d, c, p, \mathfrak{x}_i\} \uplus D_i)} \succ \mathfrak{x}_i \succ \overrightarrow{D_i} & \text{for } 1 \leqslant j \leqslant 2 \\ \mathfrak{x}_i \succ \overleftarrow{D_i} \succ \overleftarrow{\mathcal{C} \setminus (\{d, c, p, \mathfrak{x}_i\} \uplus D_i)} \succ c \succ p \succ d & \text{for } 3 \leqslant j \leqslant 4 \end{cases}$$

$$v_i^{j'} : \begin{cases} c \succ p \succ d \succ \overrightarrow{\mathcal{C} \setminus (\{d, c, p, \mathfrak{x}_i\} \uplus D_{\ell/2})} \succ \mathfrak{x}_i \succ \overrightarrow{D_{\ell/2}} & \text{for } 1 \leqslant j' \leqslant 2 \\ \mathfrak{x}_i \succ \overleftarrow{D_{\ell/2}} \succ \overleftarrow{\mathcal{C} \setminus (\{d, c, p, \mathfrak{x}_i\} \uplus D_{\ell/2})} \succ c \succ p \succ d & \text{for } 3 \leqslant j' \leqslant 4 \end{cases}$$

It is convenient to view the votes corresponding to $j = 3, 4$ as a near-reversal of the votes in $j = 1, 2$ (except for candidates c, d and $\mathfrak{r}_i$). Let $\mathcal{P}_1 = \{v_i^j, v_i^{j'} : i \in [n], j \in [4]\}$. Since there are $8n$ votes in all, and c always appears immediately before p, it follows that the score of c is exactly $8n$ more than the score of the candidate p in $\mathcal{P}_1$.

We also observe that the score of the candidate $\mathfrak{r}_i$ is exactly $2(\ell + n + 1) - X_i = 8n - X_i$ more than the score of the candidate p in $\mathcal{P}_1$ for every $i \in [n]$ such that X_i is an even integer. On the other hand, the score of the candidate $\mathfrak{r}_i$ is exactly $2(\ell + n + 1) - X_i - 1 = 8n - X_i - 1$ more than the score of the candidate p in $\mathcal{P}_1$ for every $i \in [n]$ such that X_i is an odd integer. Note that for $i' \in [n] \setminus \{i\}$, p and $\mathfrak{r}_i$ receive the same Borda score from the votes $v_{i'}^j$ and $v_{i'}^{j'}$ (where $j, j' \in [4]$).

We now add the following two votes μ_1 and μ_2.

$$\mu_1 : p \succ c \succ \text{others}$$

$$\mu_2 : p \succ c \succ \text{others}$$

Let $\mathcal{P} = \mathcal{P}_1 \uplus \{\mu_1, \mu_2\}, \mathcal{X}^o = \{\mathfrak{r}_i : i \in [n], X_i \text{ is odd}\}$, and $\mathcal{X}^e = \mathcal{X} \setminus \mathcal{X}^o$. We recall that the distinguished candidate is p. The tie-breaking rule is according to the order $\mathcal{X}^o \succ p \succ \text{others}$. We claim that the FRUGAL-BRIBERY instance $(\mathcal{C}, \mathcal{P}, p)$ is equivalent to the PERMUTATION SUM instance $(X_1, \ldots, X_n)$.

In the forward direction, suppose there exist two permutations π and σ of $[n]$ such that $\pi(i) + \sigma(i) = X_i$ for every $i \in [n]$. We replace the votes μ_1 and μ_2 with respectively μ_1' and μ_2' as follows.

$$\mu_1' : p \succ D \succ \mathfrak{r}_{\pi^{-1}(n)} \succ \mathfrak{r}_{\pi^{-1}(n-1)} \succ \cdots \succ \mathfrak{r}_{\pi^{-1}(1)} \succ c$$

$$\mu_2' : p \succ D \succ \mathfrak{r}_{\sigma^{-1}(n)} \succ \mathfrak{r}_{\sigma^{-1}(n-1)} \succ \cdots \succ \mathfrak{r}_{\sigma^{-1}(1)} \succ c$$

We observe that, the candidates c and every $\mathfrak{r} \in \mathcal{X}^e$ receive same score as p, every candidate $\mathfrak{r}' \in \mathcal{X}^o$ receives 1 score less than p, and every candidate in D receives less score than p in $\mathcal{P}_1 \uplus \{\mu_1', \mu_2'\}$. Hence p wins in $\mathcal{P}_1 \uplus \{\mu_1', \mu_2'\}$ due to the tie-breaking rule. Thus $(\mathcal{C}, \mathcal{P}, p)$ is a YES instance of FRUGAL-BRIBERY.

To prove the other direction, suppose the FRUGAL-BRIBERY instance is a YES instance. Notice that the only vulnerable votes are μ_1 and μ_2. Let μ_1' and μ_2' be two votes such that the candidate p wins in the profile $\mathcal{P}_1 \uplus \{\mu_1', \mu_2'\}$. We assume, without loss of generality, that

candidate p is placed at the first position in both μ'_1 and μ'_2. Since c receives $8n$ scores more than p in $\mathcal{P}_1$, c must be placed at the last position in both μ'_1 and μ'_2 since otherwise p cannot win in $\mathcal{P}_1 \uplus \{\mu'_1, \mu'_2\}$. We also assume, without loss of generality, that every candidate in D is preferred over every candidate in $\mathcal{X}$ since otherwise, if $\mathfrak{x} \succ d$ in either μ'_1 or μ'_2 for some $\mathfrak{x} \in \mathcal{X}$ and $d \in D$, then we can exchange the positions of $\mathfrak{x}$ and d and p continues to win since no candidate in D receives more score than p in $\mathcal{P}_1$. Hence, every $\mathfrak{x} \in \mathcal{X}$ receives some score between 1 and n in both the μ'_1 and μ'_2. Let us define two permutations π and σ of $[n]$ as follows. For every $i \in [n]$, we define $\pi(i)$ and $\sigma(i)$ to be the scores the candidate $\mathfrak{x}_i$ receives in μ'_1 and μ'_2 respectively. The fact that π and σ, as defined above, is indeed a permutation of $[n]$ follows from the structure of the votes μ'_1, μ'_2 and the Borda score vector. Since p wins in $\mathcal{P}_1 \uplus \{\mu'_1, \mu'_2\}$, we have $\pi(i) + \sigma(i) \leqslant X_i$. We now have the following.

$$n(n+1) = \sum_{i=1}^{n} (\pi(i) + \sigma(i)) \leqslant \sum_{i=1}^{n} X_i = n(n+1)$$

Hence, we have $\pi(i) + \sigma(i) = X_i$ for every $i \in [n]$ and thus $(X_1, \ldots, X_n)$ is a YES instance of PERMUTATION SUM. □

We will use Lemma 7.1 in subsequent proofs.

Theorem 10.2. *The* FRUGAL-\$BRIBERY *problem is* NP-*complete for the* k-*approval voting rule for any constant* $k \geqslant 5$, *even if the price of every vulnerable vote is either* 1 *or* ∞.

Proof. The problem is clearly in NP. To show NP-hardness, we reduce an arbitrary instance of X3C to FRUGAL-\$BRIBERY. Let $(U, \{S_1, \ldots, S_t\})$ be an instance of X3C. We define a FRUGAL-\$BRIBERY instance as follows. The candidate set is $\mathcal{C} = U \uplus D \uplus \{p, q\}$, where $|D| = k - 1$. For each $S_i, 1 \leqslant i \leqslant t$, we add a vote ν_i as follows.

$$\nu_i : \underbrace{p \succ q \succ S_i}_{\text{5 candidates}} \succ D \succ \text{others}$$

By Lemma 7.1, we can add $\text{poly}(|U|)$ many additional votes to ensure the following scores (denoted by $s(\cdot)$).

$$s(q) = s(p) + {}^{|U|}\!/\!3, s(x) = s(p) + 1, \forall x \in U,$$

$$s(d) < s(p) - {}^{|U|}\!/\!3, \forall d \in D$$

The tie-breaking rule is "$p \succ$ others". The winner is q. The distinguished candidate is p and thus all the votes in $\{v_i : 1 \leqslant i \leqslant t\}$ are vulnerable. The price of every v_i is 1 and the price of every other vulnerable vote is ∞. The budget is $|U|/3$. We claim that the two instances are equivalent. Suppose there exists an index set $I \subseteq [t]$ with $|I| = |U|/3$ such that $\biguplus_{i \in I} S_i = U$. We replace the votes v_i with $v_i', i \in I$, which are defined as follows.

$$v_i' : \underbrace{p \succ D}_{k \text{ candidates}} \succ \text{others}$$

This makes the score of p not less than the score of any other candidate and thus p wins.

To prove the result in the other direction, suppose the FRUGAL-\$BRIBERY instance is a YES instance. Then notice that there will be $|U|/3$ votes in $\{v_i : 1 \leqslant i \leqslant t\}$ where the candidate q should not be placed within the top k positions since $s(p) = s(q) - |U|/3$ and the budget is $|U|/3$. We claim that the S_i's corresponding to the v_i's that have been changed must form an exact set cover. Indeed, otherwise, there will be a candidate $x \in U$, whose score never decreases which contradicts the fact that p wins the election since $s(p) = s(x) - 1$. $\qquad\square$

We next present a similar result for the k-veto voting rule.

Theorem 10.3. *The* FRUGAL-\$BRIBERY *problem is* NP-*complete for the* k-*veto voting rule for any constant* $k \geqslant 3$, *even if the price of every vulnerable vote is either* 1 *or* ∞.

Proof. The problem is clearly in NP. To show NP-hardness, we reduce an arbitrary instance of X3C to FRUGAL-\$BRIBERY. Let $(U, \{S_1, S_2, \ldots, S_t\})$ be any instance of X3C. We define a FRUGAL-\$BRIBERY instance as follows. The candidate set is $\mathcal{C} = U \uplus Q \uplus \{p, a_1, a_2, a_3, d\}$, where $|Q| = k - 3$. For each $S_i, 1 \leqslant i \leqslant t$, we add a vote v_i as follows.

$$v_i : p \succ \text{others} \succ \underbrace{S_i \succ Q}_{k \text{ candidates}}$$

By Lemma 7.1, we can add $\text{poly}(|U|)$ many additional votes to ensure following scores (denoted by $s(\cdot)$).

$$s(p) > s(d), s(p) = s(x) + 2, \forall x \in U, s(p) = s(q) + 1, \forall q \in Q,$$

$$s(p) = s(a_i) - |U|/3 + 1, \forall i = 1, 2, 3$$

The tie-breaking rule is "$a_1 \succ \cdots \succ p$". The winner is a_1. The distinguished candidate is p and thus all the votes in $\{v_i : 1 \leqslant i \leqslant t\}$ are vulnerable. The price of every v_i is one and the price of any other vote is ∞. The budget is $|U|/3$.

We claim that the two instances are equivalent. In the forward direction, suppose there exists an index set $I \subseteq \{1, \ldots, t\}$ with $|I| = |U|/3$ such that $\uplus_{i \in I} S_i = U$. We replace the votes v_i with v_i', $i \in I$, which are defined as follows.

$$v_i' : \text{others} \succ \underbrace{a_1 \succ a_2 \succ a_3 \succ Q}_{k \text{ candidates}}$$

The score of each a_i decreases by $|U|/3$ and their final scores are $s(p) - 1$, since the score of p is not affected by this change. Also the final score of each $x \in U$ is $s(p) - 1$ since I forms an exact set cover. This makes p win the election.

To prove the result in the other direction, suppose the FRUGAL-$BRIBERY instance is a YES instance. Then, notice that there will be exactly $|U|/3$ votes in $v_i, 1 \leqslant i \leqslant t$, where every $a_j, j = 1, 2, 3$, should come in the last k positions since $s(p) = s(a_j) - |U|/3 + 1$ and the budget is $|U|/3$. Notice that candidates in Q must not be placed within top $m - k$ positions since $s(p) = s(q) + 1$, for every $q \in Q$. Hence, in the votes that have been changed, a_1, a_2, a_3 and all the candidates in Q must occupy the last k positions. We claim that the S_i's corresponding to the v_i's that have been changed must form an exact set cover. If not, then, there must exist a candidate $x \in U$ and two votes v_i and v_j such that, both v_i and v_j have been replaced by $v_i' \neq v_i$ and $v_j' \neq v_j$ and the candidate x was present within the last k positions in both v_i and v_j. This makes the score of x at least the score of p which contradicts the fact that p wins. $\square$

However, we show the existence of a polynomial time algorithm for the FRUGAL-$BRIBERY problem for the k-approval, Bucklin, and plurality with runoff voting rules, when the budget is a constant. The result below follows from the existence of a polynomial time algorithm for the COALITIONAL-MANIPULATION problem for these voting rules for a constant number of manipulators [XZP$^+$09].

Theorem 10.4. *The* FRUGAL-$BRIBERY *problem is in* P *for the* k-*approval, Bucklin, and plurality with runoff voting rules, if the budget is a constant.*

Proof. Let the budget b be a constant. Then, at most b many vulnerable votes whose price is not zero can be changed since the prices are assumed to be in $\mathbb{N}$. Notice that we may assume, without loss of generality, that all the vulnerable votes whose price is zero will be

changed. We iterate over all the $O(n^b)$ many possible vulnerable vote changes and we can solve each one in polynomial time since the COALITIONAL-MANIPULATION problem is in P for these voting rules [XZP$^+$09]. □

We show that the FRUGAL-\$BRIBERY problem is NP-complete for a wide class of scoring rules as characterized in the following result. Our next result shows that, the FRUGAL-\$BRIBERY problem is NP-complete for a wide class of scoring rules that includes the Borda voting rule. Theorem 10.5 can be proved by a reduction from the X3C problem.

Theorem 10.5. *For any positional scoring rule* r *with score vectors* $\{\overrightarrow{s_i} : i \in \mathbb{N}\}$*, if there exists a polynomial function* $f : \mathbb{N} \longrightarrow \mathbb{N}$ *such that, for every* $m \in \mathbb{N}, f(m) \geqslant 2m$ *and in the score vector* $(\alpha_1, \ldots, \alpha_{f(m)})$*, there exists a* $1 \leqslant \ell \leqslant f(m) - 5$ *satisfying the following condition:*

$$\alpha_i - \alpha_{i+1} = \alpha_{i+1} - \alpha_{i+2} > 0, \forall \ell \leqslant i \leqslant \ell + 3$$

then the FRUGAL-\$BRIBERY *problem is* NP-*complete for* r *even if the price of every vulnerable vote is either* 1 *or* ∞*.*

Proof. Since the scoring rules remain unchanged if we multiply every α_i by any constant $\lambda > 0$ and/or add any constant μ, we can assume the following without loss of generality.

$$\alpha_i - \alpha_{i+1} = \alpha_{i+1} - \alpha_{i+2} = 1, \forall \ell \leqslant i \leqslant \ell + 3$$

The problem is clearly in NP. To show NP-hardness, we reduce an arbitrary instance of X3C to FRUGAL-\$BRIBERY. Let $(U, \{S_1, \ldots, S_t\})$ be an instance of X3C. We define a FRUGAL-\$BRIBERY instance as follows. The candidate set is $\mathcal{C} = U \uplus Q \uplus \{p, a, d\}$, where $|Q| = f(|U|) - \ell - 4$. For each $S_i = \{x, y, z\}, 1 \leqslant i \leqslant t$, we add a vote ν_i as follows.

$$\nu_i : p \succ d \succ \text{others} \succ \underbrace{a \succ x \succ y \succ z \succ Q}_{l \text{ candidates}}$$

By Lemma 7.1, we can add $\text{poly}(|U|)$ many additional votes to ensure the following scores (denoted by $s(\cdot)$). Note that the proof of Lemma 7.1 in [BRR11] also works for the normalization of α defined in the beginning of the proof.

$$s(d) < s(p), s(x) = s(p) - 2, \forall x \in U,$$

$$s(a) = s(p) + |u|/3 - 1, s(q) = s(p) + 1$$

The tie-breaking rule is "$\cdots \succ p$". The distinguished candidate is p. The price of every v_i is 1 and the price of every other vulnerable vote is ∞. The budget is $|u|/3$.

We claim that the two instances are equivalent. In the forward direction, there exists an index set $I \subseteq [t], |I| = |u|/3$, such that $\uplus_{i \in I} S_i = U$. We replace the votes v_i with $v_i', i \in I$, which are defined as follows.

$$v_i' : p \succ d \succ \text{others} \succ x \succ y \succ z \succ a \succ Q$$

This makes the score of p at least one more than the score of every other candidate and thus p wins.

To prove the result in the other direction, suppose there is a way to make p win the election. Notice that the candidates in Q cannot change their positions in the changed votes and must occupy the last positions due to their score difference with p. Now we claim that there will be exactly $|u|/3$ votes where the candidate a must be placed at the $(l+4)^{th}$ position since $s(p) = s(a) - |u|/3 + 1$ and the budget is $|u|/3$. We claim that the S_i's corresponding to the changed votes must form an exact set cover. If not, then there must exist a candidate $x \in U$ whose score has increased by at least two contradicting the fact that p wins the election. $\square$

For the sake of concreteness, an example of a function f, stated in Theorem 10.5, that works for the Borda voting rule is $f(m) = 2m$. Theorem 10.5 shows that the FRUGAL-$BRIBERY problem is intractable for the Borda voting rule. However, the following theorem shows the intractability of the UNIFORM-FRUGAL-$BRIBERY problem for the Borda voting rule, even in a very restricted setting. Theorem 10.6 can be proved by a reduction from the COALITION MANIPULATION problem for the Borda voting rule for two manipulators which is known to be NP-complete [BNW11, DKNW11].

Theorem 10.6. *The* UNIFORM-FRUGAL-$BRIBERY *problem is* NP-*complete for the Borda voting rule, even when every vulnerable vote has a price of* 1 *and the budget is* 2.

Proof. The problem is clearly in NP. To show NP-hardness, we reduce an arbitrary instance of the COALITIONAL-MANIPULATION problem for the Borda voting rule with two manipulators to an instance of the UNIFORM-FRUGAL-$BRIBERY problem for the Borda voting rule. Let $(C, \succ^t, 2, p)$ be an arbitrary instance of the COALITIONAL-MANIPULATION problem for the Borda voting rule. The corresponding FRUGAL-$BRIBERY instance is as follows. The candidate set is

$C' = C \uplus \{d, q\}$. For each vote $v_i \in \succ^t$, we add a vote $v'_i : v_i \succ d \succ q$. Corresponding to the two manipulators', we add two more votes $v_1, v_2 : \overrightarrow{C \setminus \{p\}} \succ d \succ p \succ q$, where $\overrightarrow{C \setminus \{p\}}$ is an arbitrary but fixed order of the candidates in $C \setminus \{p\}$. We add more votes to ensure following score differences ($s(\cdot)$ and $s'(\cdot)$ are the score functions for the COALITIONAL-MANIPULATION and the UNIFORM-FRUGAL-\$BRIBERY instances respectively).

$$s'(p) = \lambda + s(p) - 2, s'(x) = \lambda + s(x) \text{ for every } x \in C,$$

$$s'(q) = s'(p) - 2m + 1, s'(p) > s'(d) + 2m \text{ for some } \lambda \in \mathbb{Z}$$

This will be achieved as follows. For any two arbitrary candidates a and b, the following two votes increase the score of a by one more than the rest of the candidates except b whose score increases by one less. This construction has been used before [XCP10, DKNW11].

$$a \succ b \succ \overrightarrow{C \setminus \{a, b\}}$$

$$\overleftarrow{C \setminus \{a, b\}} \succ a \succ b$$

Also, we can ensure that candidate p is always in $(^{m-1}/_2, {}^{m+1}/_2)$ positions and the candidate q never immediately follows p in these new votes. The tie-breaking rule is "others $\succ p$". The distinguished candidate is p. The price of every vulnerable vote is one and the budget is two. We claim that the two instances are equivalent.

In the forward direction, suppose the COALITIONAL-MANIPULATION instance is a YES instance. Let u_1, u_2 be the manipulators' votes that make p win. In the FRUGAL-\$BRIBERY instance, we replace v_i by $v'_i : p \succ d \succ u_i \succ q$ for $i = 1, 2$. This makes p win the election. In the reverse direction, recall that in all the vulnerable votes except v_1 and v_2, the candidate q never immediately follows candidate p. Therefore, changing any of these votes can never make p win the election since $s'(q) = s'(p) - 2m + 1$ and the budget is two. Hence, the only way p can win the election, if at all possible, is by changing the votes v_1 and v_2. Let a vote v'_i replaces v_i for $i = 1, 2$. We can assume, without loss of generality, that p and d are at the first and the second positions respectively in both v'_1 and v'_2. Let u_i be the order v'_i restricted only to the candidates in C. This makes p winner of the COALITIONAL-MANIPULATION instance since, $s'(p) = \lambda + s(p) - 2, s'(x) = \lambda + s(x)$ for every $x \in C$. $\qquad \square$

10.4 Results for Weighted Elections

Now we turn our attention to weighted elections. As before, we begin with some easy observations that follow from known results.

Observation 10.4. *The* FRUGAL-BRIBERY *problem is in* P *for the maximin and the Copeland voting rules for three candidates.*

Proof. The MANIPULATION problem is in P for the maximin, Copeland voting rules for three candidates [CSL07]. Hence, the result follows from Proposition 10.2. □

Observation 10.5. *The* FRUGAL-BRIBERY *problem is* NP-*complete for any scoring rule except plurality for three candidates.*

Proof. The same proof for Theorem 6 of [CSL07] would work here. □

Theorem 10.7. *The* FRUGAL-BRIBERY *problem is in* P *for the plurality voting rule.*

Proof. Let p be the distinguished candidate of the campaigner. The greedy strategy of just replacing every vulnerable vote by $p \succ others$ solves the problem due to the monotonicity property of the plurality voting rule. □

Our hardness results in this section are based on the PARTITION problem, which is known to be NP-complete [GJ79], and is defined as follows.

Definition 10.6. *(*PARTITION *Problem)*
Given a finite multi-set W of positive integers with $\sum_{w \in W} w = 2K$, does there exist a subset $W' \subset W$ such that $\sum_{w \in W'} w = K$? An arbitrary instance of PARTITION *is denoted by* $(W, 2K)$.

We define another problem which we call $\frac{1}{4}$-PARTITION as below. We prove that $\frac{1}{4}$-PARTITION is also NP-complete. We will use this fact in the proof of Theorem 10.10.

Definition 10.7. *(The* $\frac{1}{4}$-PARTITION *Problem)*
Given a finite multi-set W of positive integers with $\sum_{w \in W} w = 4K$, does there exist a subset $W' \subset W$ such that $\sum_{w \in W'} w = K$? An arbitrary instance of $\frac{1}{4}$-PARTITION *is denoted by* $(W, 4K)$.

Lemma 10.1. $\frac{1}{4}$-PARTITION *problem is* NP-*complete.*

Proof. The problem is clearly in NP. To show NP-hardness, we reduce the PARTITION problem to it. Let $(W, 2K)$ be an arbitrary instance of the PARTITION problem. We can assume, without loss of generality, that $2K \notin W$, since otherwise the instance is trivially a *no* instance. The corresponding $\frac{1}{4}$-PARTITION problem instance is defined by $(W_1, 4K)$, where $W_1 = W \cup \{2K\}$. We claim that the two instances are equivalent. Suppose the PARTITION instance is a YES instance and thus there exists a set $W' \subset W$ such that $\sum_{w \in W'} w = K$. This W' gives a solution to the $\frac{1}{4}$-PARTITION instance. To prove the result in the other direction, suppose there is a set $W' \subset W_1$ such that $\sum_{w \in W'} w = K$. This W' gives a solution to the PARTITION problem instance since $2K \notin W'$. $\qquad\square$

In the rest of this section, we present the hardness results in weighted elections for the following voting rules: plurality, maximin, STV, Copeland$^\alpha$, and Bucklin. For plurality, recall that the FRUGAL-BRIBERY problem is in P, and we will show that FRUGAL-\$BRIBERY is NP-complete. For all the other rules, we will establish that even FRUGAL-BRIBERY is NP-complete.

Theorem 10.8. *The* FRUGAL-\$BRIBERY *problem is* NP-*complete for the plurality voting rule for three candidates.*

Proof. The problem is clearly in NP. We reduce an arbitrary instance of PARTITION to an instance of FRUGAL-\$BRIBERY for the plurality voting rule. Let $(W, 2K)$, with $W = \{w_1, \ldots, w_n\}$ and $\sum_{i=1}^{n} w_i = 2K$, be an arbitrary instance of the PARTITION problem. The candidates are $p, a,$ and b. We will add votes in such a way that makes b win the election. The distinguished candidate is p. For every $i \in [n]$, there is a vote $a \succ p \succ b$ of both weight and price w_i. There are two votes $b \succ p \succ a$ of weight $3K$ (we do not need to define the price of this vote since it is non-vulnerable) and $p \succ a \succ b$ of both weight and price $2K + 1$. The tie-breaking rule is "$a \succ b \succ p$". We define the budget to be K.

We claim that the two instances are equivalent. Suppose there exists a $W' \subset W$ such that $\sum_{w \in W'} w = K$. We change the votes corresponding to the weights in W' to $p \succ a \succ b$. This makes p win the election with a score of $3K + 1$. To prove the other direction, for p to win, its score must increase by at least K. Also, the prices ensure that p's score can increase by at most K. Hence, p's score must increase by exactly by K and the only way to achieve this is to increase its score by changing the votes corresponding to the weights in W. Thus, p can win only if there exists a $W' \subset W$ such that $\sum_{w \in W'} w = K$. $\qquad\square$

Next we show the hardness result for the maximin voting rule.

Theorem 10.9. *The* FRUGAL-BRIBERY *problem is* NP-*complete for the maximin voting rule for four candidates.*

Proof. The problem is clearly in NP. We reduce an arbitrary instance of PARTITION to an instance of FRUGAL-BRIBERY for the maximin voting rule. Let $(W, 2K)$, with $W = \{w_1, \ldots, w_n\}$ and $\sum_{i=1}^{n} w_i = 2K$, be an arbitrary instance of the PARTITION problem. The candidates are p, a, b, and c. For every $i \in [n]$, there is a vote $p \succ a \succ b \succ c$ of weight w_i. There is one vote $c \succ a \succ b \succ p$, one $b \succ c \succ a \succ p$, and one $a \succ c \succ b \succ p$ each of weight K. The tie-breaking rule is "$p \succ a \succ b \succ c$". The distinguished candidate is p. Let T denotes the set of votes corresponding to the weights in W and the rest of the votes S. Notice that only the votes in T are vulnerable. We claim that the two instances are equivalent.

Suppose there exists a $W' \subset W$ such that $\sum_{w \in W'} w = K$. We change the votes corresponding to the weights in W' to $p \succ a \succ b \succ c$. We change the rest of the votes in T to $p \succ b \succ c \succ a$. The maximin score of every candidate is $-K$ and thus due to the tie-breaking rule, p wins the election.

On the other hand, suppose there is a way to change the vulnerable votes, that is the votes in T, that makes p win the election. Without loss of generality, we can assume that all the votes in T place p at top position. First notice that the only way p could win is that the vertices a, b, and c must form a cycle in the weighted majority graph. Otherwise, one of a, b, and c will be a Condorcet winner and thus the winner of the election. Now we show that the candidate b must defeat the candidate c. If not, then c must defeat b by a margin of K since the maximin score of p is fixed at $-K$. Also, a must defeat c by a margin of K, otherwise the maximin score of c will be more than $-K$. This implies that all the votes in T must be $p \succ a \succ c \succ b$ which makes a defeat b. This is a contradiction since the vertices a, b, and c must form a cycle in the weighted majority graph. Hence b must defeat c by a margin of K. This forces every vote in T to prefer b over c. Hence, without loss of generality, we assume that all the votes in T are either $p \succ a \succ b \succ c$ or $p \succ b \succ c \succ a$, since whenever c is right after a, we can swap a and c and this will only reduce the score of a without affecting the score of any other candidates. If the total weight of the votes $p \succ a \succ b \succ c$ in T is more than K, then $D_E(c, a) < K$, thereby making the maximin score of a more than the maximin score of p. If the total weight of the votes $p \succ a \succ b \succ c$ in T is less than K, then $D_E(a, b) < K$, thereby making the maximin score of b more than the maximin score of p. Thus the total weight of the votes $p \succ a \succ b \succ c$ in T should be exactly K which corresponds to a partition of W. $\qquad\square$

We now prove the hardness result for the STV voting rule.

Theorem 10.10. *The* FRUGAL-BRIBERY *problem is* NP-*complete for the STV voting rule for three candidates.*

Proof. The problem is clearly in NP. We reduce an arbitrary instance of $\frac{1}{4}$-PARTITION to an instance of FRUGAL-BRIBERY for the STV voting rule. Let $(W, 4K)$, with $W = \{w_1, \ldots, w_n\}$ and $\sum_{i=1}^{n} w_i = 4K$, be an arbitrary instance of the $\frac{1}{4}$-PARTITION problem. The candidates are p, a, and b. For every $i \in [n]$, there is a vote $p \succ a \succ b$ of weight w_i. There is a vote $a \succ p \succ b$ of weight $3K - 1$ and a vote $b \succ a \succ p$ of weight $2K$. The tie-breaking rule is "$a \succ b \succ p$". The distinguished candidate is p. Let T denotes the set of votes corresponding to the weights in W and the rest of the votes be S. Notice that only the votes in T are vulnerable. We claim that the two instances are equivalent.

Suppose there exists a $W' \subset W$ such that $\sum_{w \in W'} w = K$. We change the votes corresponding to the weights in W' to $b \succ p \succ a$. We do not change the rest of the votes in T. This makes p win the election.

For the other direction, suppose there is a way to change the votes in T that makes p win the election. First Notice that p can win only if b qualifies for the second round. Hence, the total weight of the votes in T that put b at the first position must be at least K. On the other hand, if the total weight of the votes in T that put b at the first position is strictly more than K, then p does not qualify for the second round and thus cannot win the election. Hence the total weight of the votes in T that put b at the first position must be exactly equal to K which constitutes a $\frac{1}{4}$-partition of W. □

For three candidates, the STV voting rule is the same as the plurality with runoff voting rule. Hence, we have the following corollary.

Corollary 10.1. *The* FRUGAL-BRIBERY *problem is* NP-*complete for the plurality with runoff voting rule for three candidates.*

We turn our attention to the Copeland$^\alpha$ voting rule next.

Theorem 10.11. *The* FRUGAL-BRIBERY *problem is* NP-*complete for the Copeland$^\alpha$ voting rule for four candidates, whenever $\alpha \in [0, 1)$.*

Proof. The problem is clearly in NP. We reduce an arbitrary instance of PARTITION to an instance of FRUGAL-BRIBERY for the Copeland$^\alpha$ voting rule. Let $(W, 2K)$, with $W = \{w_1, \ldots, w_n\}$

and $\sum_{i=1}^{n} w_i = 2K$, be an arbitrary instance of the PARTITION problem. The candidates are $p, a, b,$ and c. For every $i \in [n]$, there is a vote $p \succ a \succ b \succ c$ of weight w_i. There are two votes $a \succ p \succ b \succ c$ and $c \succ b \succ a \succ p$ each of weight $K + 1$. The tie-breaking rule is "$a \succ b \succ c \succ p$". The distinguished candidate is p. Let T denotes the set of votes corresponding to the weights in W and the rest of the votes be S. Notice that only the votes in T are vulnerable. We claim that the two instances are equivalent.

Suppose there exists a $W' \subset W$ such that $\sum_{w \in W'} w = K$. We change the votes corresponding to the weights in W' to $p \succ c \succ b \succ a$. We change the rest of the votes in T to $p \succ b \succ c \succ a$. This makes p win the election with a Copeland$^{\alpha}$ score of two.

On the other hand, suppose there is a way to change the votes in T that makes p win the election. Without loss of generality, we can assume that all the votes in T place p at top position. We will show that one of the three pairwise elections among $a, b,$ and c must be a tie. Suppose not, then a must lose to both b and c, otherwise a wins the election due to the tie-breaking rule. Now consider the pairwise election between b and c. If b defeats c, then b wins the election due to the tie-breaking rule. If c defeats b, then c wins the election again due to the tie-breaking rule. Hence, one of the pairwise elections among $a, b,$ and c must be a tie. Without loss of generality suppose a and b ties. Then the total weight of the votes that prefer a to b in T must be K which constitutes a partition of W. $\qquad \square$

Finally, we show that the FRUGAL-BRIBERY problem for the Bucklin voting rule is NP-complete.

Theorem 10.12. *The* FRUGAL-BRIBERY *problem is* NP-*complete for the Bucklin voting rule for four candidates.*

Proof. The problem is clearly in NP. We reduce an arbitrary instance of PARTITION to an instance of FRUGAL-BRIBERY for the Bucklin voting rule. Let $(W, 2K)$, with $W = \{w_1, \ldots, w_n\}$ and $\sum_{i=1}^{n} w_i = 2K$, be an arbitrary instance of the PARTITION problem. The candidates are $p, a, b,$ and c. For every $i \in [n]$, there is a vote $p \succ a \succ b \succ c$ of weight w_i. There are two votes $a \succ b \succ p \succ c$ and $c \succ b \succ a \succ p$ each of weight K. The tie-breaking rule is "$p \succ a \succ b \succ c$". The distinguished candidate is p. Let T denote the set of votes corresponding to the weights in W and the rest of the votes be S. Notice that only the votes in T are vulnerable. We claim that the two instances are equivalent.

Suppose there exists a $W' \subset W$ such that $\sum_{w \in W'} w = K$. We change the votes corresponding to the weights in W' to $p \succ c \succ b \succ a$. This makes p win the election with a Bucklin score of three.

To prove the result in the other direction, suppose there is a way to change the votes in T that makes p win the election. Without loss of generality, we can assume that all the votes in T place p at the first position. First Notice that the Bucklin score of p is already fixed at three. In the votes in T, the candidate b can never be placed at the second position since that will make the Bucklin score of b to be two. Also the total weight of the votes in T that place a in their second position can be at most K. The same holds for c. Hence, the total weight of the votes that place a in their second position will be exactly equal to K which constitutes a partition of W. $\qquad\square$

We also have the following results for the Copeland$^\alpha$ and Bucklin voting rules by reducing from PARTITION.

Theorem 10.13. *The* FRUGAL-BRIBERY *problem is* NP-*complete for the Copeland$^\alpha$ and Bucklin voting rules for four candidates, whenever $\alpha \in [0, 1)$.*

From Proposition 10.1, Observation 10.5, Theorem 10.8 to 10.10, 10.12 and 10.13, and Corollary 10.1, we get the following corollary.

Corollary 10.2. *The* UNIFORM-FRUGAL-\$BRIBERY *and the* NONUNIFORM-FRUGAL-\$BRIBERY *problems are* NP-*complete for the scoring rules except plurality, STV, and the plurality with runoff voting rules for three candidates and for the maximin, Copeland, and Bucklin voting rules for four candidates.*

10.5 Conclusion

We proposed and studied two important special cases of the \$BRIBERY problem where the briber is frugal. Our results show that even for these special cases, the bribery problem continues to be intractable, thus subsuming known hardness results in the literature. Our results reinforce that bribery is a rather hard computational problem, because of the hardness of several important special cases. This also strengthens the view that bribery, although a possible attack on an election in principle, may be infeasible in practice.

Chapter 11

Summary and Future Directions

In this chapter, we summarize our work in this thesis and provide interesting future directions of research.

In this thesis, we studied computational complexity of three fundamental aspects of voting. We began with studying efficient strategies for eliciting preferences of voters in the first part of the thesis. We then moved on to study the problem of finding a winner under various interesting circumstances in the second part of the thesis. Finally, we showed interesting complexity theoretic results for the computational problem of controlling an election system in various forms. We now summarize our work in this thesis.

11.1   Summary of Contributions

11.1.1   Part I: Preference Elicitation

Chapter 3: Preference Elicitation for Single Peaked Profiles on Trees

In this work, we presented algorithms for eliciting the preferences of a set of voters when the preferences are single peaked on a tree. Moreover, our algorithms ask minimum number of comparison queries up to constant factors. We also presented algorithms for finding a weak Condorcet winner from a set of preferences which are single peaked on a tree by asking minimum number of comparison queries up to constant factors. We observed that, the query complexity of finding a weak Condorcet is much less than the query complexity for preference elicitation.

Chapter 4: Preference Elicitation for Single Crossing Profiles

In this work, we presented preference elicitation algorithm for single crossing preference profiles. We studied this problem when an ordering of the voters with respect to which the profile is single crossing is *known* and when it is *unknown*. We also considered different access models: when the votes can be accessed at random, as opposed to when they are coming in a predefined sequence. In the sequential access model, we distinguished two cases when the ordering is known: the first is that sequence in which the votes appear is also a single-crossing order, versus when it is not. Our algorithms ask for minimum number of comparison queries up to constant factors for all the above situations except one when we have a large number of voters.

11.1.2   Part II: Winner Determination

Chapter 5: Winner Prediction and Margin of Victory Estimation

We presented efficient algorithms for predicting the winner of an election as well as estimating the margin of victory of an election. We also showed interesting lower bounds for sample complexity of these problems which establish that our algorithms are often optimal up to constant factors.

Chapter 6: Streaming Algorithms for Winner Determination

In this work, we studied the space complexity for determining approximate winners in the setting where votes are inserted continually into a data stream. We showed that allowing randomization and approximation indeed allows for much more space-efficient algorithms. Moreover, our bounds are tight in certain parameter ranges.

11.1.3   Part III: Election Control

Chapter 7: Kernelization for Possible Winner and Coalitional Manipulation

In this work, we proved that the possible winner problem does not admit any efficient pre-processing rules, more formally any kernelization algorithms for many common voting rules including scoring rules, maximin, Copeland, ranked pairs, and Bucklin when parameterized

by the number of candidates. However, we showed that the coalitional manipulation problem which is an important special case of the possible winner problem does admit polynomial time kernelization algorithms parameterized by the number of candidates.

Chapter 8: Manipulation with Partial Votes

In this chapter, we pursued a comprehensive study of manipulation with incomplete votes. We proposed three natural extension of manipulation in the usual complete information setting to the incomplete information setting namely, weak manipulation, opportunistic manipulation, and strong manipulation. We completely resolved computational complexity of all the three problems for many common voting rules including plurality, veto, k-approval, k-veto, Borda, maximin, Copeland, Bucklin, and Fallback voting rules.

Chapter 9: Manipulation Detection

In this work, we initiated a promising direction of research namely detecting instances of manipulation in elections. We showed that detecting possible instances of manipulation can often be a much easier computational problem than the corresponding problem of manipulating election itself as seen for the case of the Borda voting rule.

Chapter 10: Frugal Bribery

In this work, we studied the classical problem of bribery under a weak notion of briber namely when the briber if frugal in nature. We proved that the bribery problems remain intractable even with this weak briber thereby strengthening the intractability results from the literature on bribery. Hence, although theoretically possible, bribery may not be easy to do in practice.

11.2 Future Directions of Research

We next discuss some of the interesting directions of research from this thesis.

Chapter 3: Preference Elicitation for Single Peaked Profiles on Trees

▷ One can generalize the notion of single peaked profiles on trees for preferences that are not necessarily a complete order. Indeed, voters often are indifferent between two or

more candidates and there are interesting domains in this setting [EL15]. Eliciting such incomplete preference is an interesting direction of research to pursue.

▷ The domain of single peaked profiles on trees can further be generalized to single peaked profiles on forests. It would be interesting to study this both as a domain itself and from the point of view of existence of efficient preference elicitation strategies.

▷ Can we reduce the query complexity for preference elicitation further by assuming more on their preferences, for example, a social network structure on the voters?

▷ In this work, we assume complete knowledge of single peaked tree. How the query complexity would change if we only assume the knowledge of the structure of the single peaked tree without the identity of the candidate associated with each node of the tree?

Chapter 4: Preference Elicitation for Single Crossing Profiles

▷ An immediate direction of research is to close the gap between upper and lower bound on query complexity for preference elicitation when we know a single crossing order and voters are allowed to be queried randomly.

▷ Another interesting direction is to study preference elicitation assuming a social network structure among the voters where neighbors tend to have similar preferences.

Chapter 5: Winner Prediction and Margin of Victory Estimation

▷ Is there an axiomatic characterization of the voting rules for which the sample complexity is independent of m and n? We note that a similar problem in graph property testing was the subject of intense study [AFNS06, BCL$^+$06].

▷ Specifically for scoring rules, is the sample complexity determined by some natural property of the score vector, such as its sparsity?

▷ Is it worthwhile for the algorithm to elicit only part of the vote from each sampled voter instead of the full vote? As mentioned in the Introduction, vote elicitation is a well-trodden area, but as far as we know, it has not been studied how assuming a margin of victory can change the number of queries.

▷ How can knowledge of a social network on the voters be used to minimize the number of samples made? Some initial progress in this direction has been made by Dhamal and Narahari [DN13] and by Agrawal and Devanur (private communication).

Chapter 6: Streaming Algorithms for Winner Determination

▷ An immediate future direction of research is to find optimal algorithms for heavy hitters variants for other voting rule.

▷ It may be interesting to implement these streaming algorithms for use in practice (say, for participatory democracy experiments or for online social networks) and investigate how they perform.

▷ Finally, instead of having the algorithms which passive observes a few random votes, could we improve performance of the algorithm by actively querying voters as they appear in the stream?

Chapter 7: Kernelization for Possible Winner and Coalitional Manipulation

▷ There are other interesting parameterizations of these problems for which fixed parameter tractable algorithms are known but the corresponding kernelization questions are still open. One such parameter is the total number of pairs s in all the votes for which an ordering has not been specified. With this parameter, a simple $O(2^s. \text{poly}(m, n))$ algorithm is known [BHN09]. However, the corresponding kernelization question is still open.

▷ Another interesting problem in the context of incomplete votes is the necessary winner problem which asks for a candidate which wins in every extension of the partial votes. Necessary winner is known to be intractable for Copeland, ranked pairs, voting trees, and STV voting rules [XC11]. Studying parameterized complexity of the necessary winner problem is also another interesting direction of research to pursue in future.

Chapter 8: Manipulation with Partial Votes

▷ We leave open the problem of completely establishing the complexity of strong, opportunistic, and weak manipulations for all the scoring rules. It would be interesting to resolve it.

▷ Other fundamental forms of manipulation and control do exist in voting, such as destructive manipulation and control by adding candidates. It would be interesting to investigate the complexity of these problems in a partial information setting.

▷ Another exciting direction is the study of average case complexity, as opposed to the worst case results that we have pursued. These studies have already been carried out in the setting of complete information [PR06, FP10, Wal10]. Studying the problems that we propose in the average-case model would reveal further insights on the robustness of the incomplete information setting as captured by our model involving partial orders.

Chapter 9: Manipulation Detection

▷ In this work, we considered elections with unweighted voters only. An immediate future research direction is to study the complexity of these problems in weighted elections.

▷ Verifying the number of false manipulators that this model catches in a real or synthetic data set, where, we already have some knowledge about the manipulators, would be interesting.

Chapter 10: Frugal Bribery

▷ A potential and natural direction for future work is to study these problems under various other settings. One obvious setting is to restrict the campaigner's knowledge about the votes and/or the candidates who will actually turn up. The uncertainty can also arise from the voting rule that will eventually be used among a set of voting rules.

▷ Studying these bribery problems when the pricing model for vulnerable votes is similar to swap bribery would be another interesting future direction.

References

[ABW03]   Arvind Arasu, Shivnath Babu, and Jennifer Widom. CQL: A language for continuous queries over streams and relations. In *Proc. 9th International Workshop on Database Programming Languages DBPL*, pages 1–19, 2003.

[AFNS06] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. In *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 251–260, 2006.

[AM01]   Javed A Aslam and Mark Montague. Models for metasearch. In *Proc. 24th Annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 276–284. ACM, 2001.

[Arr50]   Kenneth J Arrow. A difficulty in the concept of social welfare. *J. Polit. Econ.*, pages 328–346, 1950.

[AS94]   Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.

[AT05]   Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[BBCN12] Nadja Betzler, Robert Bredereck, Jiehua Chen, and Rolf Niedermeier. The multivariate algorithmic revolution and beyond. pages 318–363, Berlin, Heidelberg, 2012. Springer-Verlag.

[BBCV09]   Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. Voting in social networks. In *Proc. 18th ACM Conference on Information and Knowledge Mmanagement*, pages 777–786. ACM, 2009.

[BBF10]   Yoram Bachrach, Nadja Betzler, and Piotr Faliszewski. Probabilistic possible winner determination. In *International Conference on Artificial Intelligence (AAAI)*, volume 10, pages 697–702, 2010.

[BBHH15]   Felix Brandt, Markus Brill, Edith Hemaspaandra, and Lane A Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. *J. Artif. Intell. Res.*, pages 439–496, 2015.

[BBN10]   Nadja Betzler, Robert Bredereck, and Rolf Niedermeier. Partial kernelization for rank aggregation: theory and experiments. In *International Symposium on Parameterized and Exact Computation (IPEC)*, pages 26–37. Springer, 2010.

[BBN14]   Nadja Betzler, Robert Bredereck, and Rolf Niedermeier. Theoretical and empirical evaluation of data reduction for exact kemeny rank aggregation. *Auton. Agent Multi Agent Syst.*, 28(5):721–748, 2014.

[BCE+15]   Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel Procaccia. Handbook of computational social choice, 2015.

[BCF+14a]   Robert Bredereck, Jiehua Chen, Piotr Faliszewski, Jiong Guo, Rolf Niedermeier, and Gerhard J. Woeginger. Parameterized algorithmics for computational social choice: Nine research challenges. In *Tsinghua Science and Technology*, volume 19, pages 358–373. IEEE, 2014.

[BCF+14b]   Robert Bredereck, Jiehua Chen, Piotr Faliszewski, André Nichterlein, and Rolf Niedermeier. Prices matter for the parameterized complexity of shift bribery. In *Proc. 28th International Conference on Artificial Intelligence (AAAI)*, pages 1398–1404, 2014.

[BCH+14]   Robert Bredereck, Jiehua Chen, Sepp Hartung, Stefan Kratsch, Rolf Niedermeier, Ondrej Such, and Gerhard J. Woeginger. A multivariate complexity analysis of lobbying in multiple referenda. *J. Artif. Intell. Res. (JAIR)*, 50:409–446, 2014.

[BCIW16] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, and David P. Woodruff. Beating countsketch for heavy hitters in insertion streams. In *Proc. 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 740–753, 2016.

[BCL⁺06] Christian Borgs, Jennifer T. Chayes, László Lovász, Vera T. Sós, Balázs Szegedy, and Katalin Vesztergombi. Graph limits and parameter testing. In *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 261–270, 2006.

[BD09] Nadja Betzler and Britta Dorn. Towards a dichotomy of finding possible winners in elections based on scoring rules. In *Mathematical Foundations of Computer Science (MFCS)*, pages 124–136. Springer, 2009.

[BDW16] Arnab Bhattacharyya, Palash Dey, and David P. Woodruff. An optimal algorithm for l1-heavy hitters in insertion streams and related problems. In *Proc. 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '16, pages 385–400, San Francisco, California, USA, 2016.

[BEF⁺14] Daniel Binkele-Raible, Gábor Erdélyi, Henning Fernau, Judy Goldsmith, Nicholas Mattei, and Jörg Rothe. The complexity of probabilistic lobbying. In *Algorithmic Decision Theory*, volume 11, pages 1–21. Discrete Optimization, 2014.

[Bet10] Nadja Betzler. On problem kernels for possible winner determination under the k-approval protocol. In *Mathematical Foundations of Computer Science (MFCS)*, pages 114–125. Springer, 2010.

[BF12] Markus Brill and Felix A. Fischer. The price of neutrality for the ranked pairs method. In *Proc. Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[BFG⁺09] Nadja Betzler, Michael R Fellows, Jiong Guo, Rolf Niedermeier, and Frances A Rosamond. Fixed-parameter algorithms for kemeny rankings. *THEOR COMPUT SCI*, 410(45):4554–4570, 2009.

[BFLR12] Dorothea Baumeister, Piotr Faliszewski, Jérôme Lang, and Jörg Rothe. Campaigns for lazy voters: truncated ballots. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, pages 577–584, 2012.

[BGS01] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In *Proc. 2nd International Conference on Mobile Data Management, MDM*, pages 3–14, 2001.

[BHN09] Nadja Betzler, Susanne Hemmann, and Rolf Niedermeier. A Multivariate Complexity Analysis of Determining Possible Winners given Incomplete Votes. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, volume 9, pages 53–58, 2009.

[BICS10] Radu Berinde, Piotr Indyk, Graham Cormode, and Martin J. Strauss. Space-optimal heavy hitters with strong error bounds. *ACM Trans. Database Syst.*, 35(4):26:1–26:28, October 2010.

[BITT89] John Bartholdi III, Craig A Tovey, and Michael A Trick. Voting schemes for which it can be difficult to tell who won the election. *Soc. Choice Welf.*, 6(2):157–165, 1989.

[Bla48] Duncan Black. On the rationale of group decision-making. *J. Polit. Econ.*, pages 23–34, 1948.

[BLOP14] Craig Boutilier, Jérôme Lang, Joel Oren, and Héctor Palacios. Robust winners and winner determination policies under candidate uncertainty. In *Proc. Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1391–1397. AAAI Press, 2014.

[BNM⁺58] Duncan Black, Robert Albert Newing, Iain McLean, Alistair McMillan, and Burt L Monroe. *The theory of committees and elections*. Springer, 1958.

[BNW11] Nadja Betzler, Rolf Niedermeier, and Gerhard J Woeginger. Unweighted coalitional manipulation under the borda rule is NP-hard. In *IJCAI*, volume 11, pages 55–60, 2011.

[BO91] J.J. Bartholdi and J.B. Orlin. Single transferable vote resists strategic voting. *Soc. Choice Welf.*, 8(4):341–354, 1991.

[bor] Borda count. https://en.wikipedia.org/wiki/Borda_count.

[BR99] Kevin S. Beyer and Raghu Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 359–370, 1999.

[BR12] Dorothea Baumeister and Jörg Rothe. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. *Inf. Process. Lett.*, 112(5):186–190, 2012.

[BRR11] Dorothea Baumeister, Magnus Roos, and Jörg Rothe. Computational complexity of two variants of the possible winner problem. In *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 853–860, 2011.

[BRR+12] Dorothea Baumeister, Magnus Roos, Jörg Rothe, Lena Schend, and Lirong Xia. The possible winner problem with uncertain weights. In *ECAI*, pages 133–138, 2012.

[BS09] Steven J Brams and M Remzi Sanver. Voting systems that combine approval and preference. In *The mathematics of preference, choice and order*, pages 215–237. Springer, 2009.

[BTT89] J.J. Bartholdi, C.A. Tovey, and M.A. Trick. The computational difficulty of manipulating an election. *Soc. Choice Welf.*, 6(3):227–241, 1989.

[BTY09] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel Bounds for Disjoint Cycles and Disjoint Paths. In Amos Fiat and Peter Sanders, editors, *Proc. 17th Annual European Symposium,on Algorithms (ESA 2009), Copenhagen, Denmark, September 7-9, 2009.*, volume 5757 of *Lecture Notes in Computer Science*, pages 635–646. Springer, 2009.

[BY03] Ziv Bar-Yossef. Sampling lower bounds via information theory. In *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 335–344. ACM, 2003.

[BYJKS02] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. In *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 209–218. IEEE, 2002.

[BYKS01] Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. Sampling algorithms: lower bounds and applications. In *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 266–275. ACM, 2001.

[Car11] David Cary. Estimating the margin of victory for instant-runoff voting. In *Proceedings of the 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'11, pages 3–3, Berkeley, CA, USA, 2011. USENIX Association.

[CCFC04] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters (IPL)*, 53(1):17–25, 1995.

[CFM09] Yousra Chabchoub, Christine Fricker, and Hanene Mohamed. Analysis of a bloom filter algorithm via the supermarket model. In *Proc. 21st International Teletraffic Congress, ITC*, pages 1–8, 2009.

[CH08] Graham Cormode and Marios Hadjieleftheriou. Finding frequent items in data streams. *Proc. VLDB Endow.*, 1(2):1530–1541, 2008.

[CKMS08] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Finding hierarchical heavy hitters in streaming data. *ACM Trans. Knowl. Discov. Data*, 1(4):2:1–2:48, February 2008.

[CLMM10] Yann Chevaleyre, Jérôme Lang, Nicolas Maudet, and Jérôme Monnot. Possible winners when new candidates are added: The case of scoring rules. In *Proc. International Conference on Artificial Intelligence (AAAI)*, pages 762–767, 2010.

[CM05] Graham Cormode and S Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[Con09] Vincent Conitzer. Eliciting single-peaked preferences using comparison queries. *J. Artif. Intell. Res.*, 35:161–191, 2009.

[Cor09] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[Cor12] Graham Cormode. Sketch techniques for massive data. In Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine, editors, *Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches*, volume 4 of *Foundations and Trends in Databases*, pages 1–294. Now Publishers Inc., Hanover, MA, USA, January 2012.

[CP06] Ben Carterette and Desislava Petkova. Learning a ranking from pairwise preferences. In *Proc. 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 629–630. ACM, 2006.

[CP11] Ioannis Caragiannis and Ariel D. Procaccia. Voting almost maximizes social welfare despite limited communication. *Artif. Intell.*, 175(9–10):1655 – 1671, 2011.

[CPS14] Ioannis Caragiannis, Ariel D Procaccia, and Nisarg Shah. Modal ranking: A uniquely robust voting rule. In *Proc. 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 616–622, 2014.

[CRT06] E. J. Candes, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Commun. Pur. Appl. Math.*, 59:1207–1223, 2006.

[CRX09] Vincent Conitzer, Matthew Rognlie, and Lirong Xia. Preference functions that score rankings and maximum likelihood estimation. In *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI)*, volume 9, pages 109–115, 2009.

[CS02] Vincent Conitzer and Tuomas Sandholm. Vote elicitation: Complexity and strategy-proofness. In *Eighteenth National Conference on Artificial Intelligence (AAAI)*, pages 392–397, 2002.

[CS05] Vincent Conitzer and Tuomas Sandholm. Communication complexity of common voting rules. In *Proc. 6th ACM conference on Electronic Commerce (EC)*, pages 78–87. ACM, 2005.

[CS06] Vincent Conitzer and Tuomas Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *International Conference on Artificial Intelligence (AAAI)*, volume 6, pages 627–634, 2006.

[CSL07] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *J. ACM*, 54(3):14, 2007.

[CSS99] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. Artif. Int. Res.*, 10(1):243–270, May 1999.

[CT12] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[CWX11] Vincent Conitzer, Toby Walsh, and Lirong Xia. Dominating manipulations in voting with partial information. In *International Conference on Artificial Intelligence (AAAI)*, volume 11, pages 638–643, 2011.

[DB15] Palash Dey and Arnab Bhattacharyya. Sample complexity for winner prediction in elections. In *Proc. 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1421–1430, 2015.

[Dem82] Gabrielle Demange. Single-peaked orders on a tree. *Math. Soc. Sci*, 3(4):389–396, 1982.

[Dey15] Palash Dey. Computational complexity of fundamental problems in social choice theory. In *Proc. 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1973–1974. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[DF99] Rod G Downey and Michael Ralph Fellows. *Parameterized Complexity*, volume 3. springer Heidelberg, 1999.

[DK16] Britta Dorn and Dominikus Krüger. On the hardness of bribery variants in voting with cp-nets. *Ann. Math. Artif. Intell.*, 77(3-4):251–279, 2016.

[DKNS01] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proc. 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.

[DKNW11] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for borda manipulation. In *Proc. International Conference on Artificial Intelligence (AAAI)*, pages 657–662, 2011.

[DKW56] A Dvoretzky, J Kiefer, and J Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *Ann. Math. Stat.*, 27(3):642 – 669, 1956.

[DL13a] Ning Ding and Fangzhen Lin. Voting with partial information: what questions to ask? In *Proc. 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1237–1238. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[DL13b] Ning Ding and Fangzhen Lin. Voting with partial information: What questions to ask? In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1237–1238, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[DLC14] John A Doucette, Kate Larson, and Robin Cohen. Approximate winner selection in social choice with partial preferences. In *Proc. 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[DLOM02] Erik D Demaine, Alejandro López-Ortiz, and J Ian Munro. Frequency estimation of internet packet streams with limited space. In *Proc. 10th Annual European Symposium on Algorithms*, pages 348–360. Springer, 2002.

[DLS09] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *Automata, Languages and Programming (ICALP), 36th International Colloquium, Part I*, volume 5555 of *LNCS*, pages 378–389, 2009.

[DM16a] Palash Dey and Neeldhara Misra. Elicitation for preferences single peaked on trees. In *Proc. Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 215–221, 2016.

[DM16b] Palash Dey and Neeldhara Misra. Preference elicitation for single crossing domain. In *Proc. Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 222–228, 2016.

[DMN15a] Palash Dey, Neeldhara Misra, and Y. Narahari. Detecting possible manipulators in elections. In *Proc. 2015 International Conference on Autonomous Agents and*

Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015, pages 1441–1450, 2015.

[DMN15b] Palash Dey, Neeldhara Misra, and Y. Narahari. Kernelization complexity of possible winner and coalitional manipulation problems in voting. In *Proc. 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 87–96, 2015.

[DMN16a] Palash Dey, Neeldhara Misra, and Y. Narahari. Complexity of manipulation with partial information in voting. In *Proc. Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 229–235, 2016.

[DMN16b] Palash Dey, Neeldhara Misra, and Y. Narahari. Frugal bribery in voting. In *Proc. Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2466–2472, 2016.

[DMN16c] Palash Dey, Neeldhara Misra, and Y. Narahari. Kernelization complexity of possible winner and coalitional manipulation problems in voting. *Theor. Comput. Sci.*, 616:111–125, 2016.

[DN13] Swapnil Dhamal and Y Narahari. Scalable preference aggregation in social networks. In *First AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, pages 42–50, 2013.

[DN14] Palash Dey and Y Narahari. Asymptotic collusion-proofness of voting rules: the case of large number of candidates. In *Proc. 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1419–1420. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[DN15a] Palash Dey and Y Narahari. Asymptotic collusion-proofness of voting rules: The case of large number of candidates. *Studies in Microeconomics*, 3(2):120–139, 2015.

[DN15b] Palash Dey and Y. Narahari. Estimating the margin of victory of an election using sampling. In *Proc. Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1120–1126, 2015.

[Dod76] Charles Lutwidge Dodgson. *A method of taking votes on more than two issues*. 1876.

[DS12] Britta Dorn and Ildikó Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 64(1):126–151, 2012.

[Dur10] Rick Durrett. *Probability: theory and examples*. Cambridge university press, 2010.

[EE12] Edith Elkind and Gábor Erdélyi. Manipulation under voting rule uncertainty. In *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 627–634. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[EFS09] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. Swap bribery. In *Proc. 2nd International Symposium on Algorithmic Game Theory (SAGT 2009)*, pages 299–310. Springer, 2009.

[EHH14] Gabor Erdelyi, Edith Hemaspaandra, and Lane A Hemaspaandra. Bribery and voter control under voting-rule uncertainty. In *Proc. 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 61–68. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[EJS10] Funda Ergün, Hossein Jowhari, and Mert Sağlam. Periodicity in streams. In *Proc. 14th Workshop on Randomization and Computation -(RANDOM 2010)*, pages 545–559. Springer, 2010.

[EL05] Edith Elkind and Helger Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Algorithms and Computation*, pages 206–215. Springer, 2005.

[EL14] U Endriss and J Leite. The margin of victory in schulze, cup, and copeland elections: Complexity of the regular and exact variants. In *STAIRS 2014: Proc. 7th European Starting AI Researcher Symposium*, volume 264, pages 250–259. IOS Press, 2014.

[EL15] Edith Elkind and Martin Lackner. Structure in dichotomous preferences. In *Proc. Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2019–2025, 2015.

[ER91] E. Ephrati and J.S. Rosenschein. The Clarke tax as a consensus mechanism among automated agents. In *Proc. Ninth International Conference on Artificial Intelligence (AAAI)*, pages 173–178, 1991.

[ER93] Eithan Ephrati and Jeffrey S. Rosenschein. Multi-agent planning as a dynamic search for social consensus. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence*, pages 423–429, Chambery, France, August 1993.

[EV03] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *Theor. Comput. Syst.*, 21(3):270–313, 2003.

[Fal08] Piotr Faliszewski. Nonuniform bribery. In *Proce. 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1569–1572. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume 3. Springer, 2006.

[FHH06] Piotr Faliszewski, Edith Hemaspaandra, and Lane A Hemaspaandra. The complexity of bribery in elections. In *Proc. 21st International Conference on Artificial Intelligence (AAAI)*, volume 6, pages 641–646, 2006.

[FHH09] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. How hard is bribery in elections? *J. Artif. Int. Res.*, 35(1):485–532, July 2009.

[FHH10] Piotr Faliszewski, Edith Hemaspaandra, and Lane A Hemaspaandra. Using complexity to protect elections. *Commun ACM*, 53(11):74–82, 2010.

[FHH13] Piotr Faliszewski, Edith Hemaspaandra, and Lane A Hemaspaandra. Weighted electoral control. In *Proc. 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 367–374. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[FHHR09] Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Llull and copeland voting computationally resist bribery and constructive control. *J. Artif. Intell. Res.*, 35:275–341, 2009.

[FHS08] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: Ties matter. In *Proc. 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 983–990. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[FHS10] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Manipulation of copeland elections. In *Proc. 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 367–374. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[FKN08] E. Friedgut, G. Kalai, and N. Nisan. Elections can be manipulated often. In *IEEE 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 243–249. IEEE, 2008.

[FKS03] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proc. 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 301–312, New York, NY, USA, 2003. ACM.

[Fla85] Philippe Flajolet. Approximate counting: a detailed analysis. *BIT Numer. Math.*, 25(1):113–134, 1985.

[FMT12] Dvir Falik, Reshef Meir, and Moshe Tennenholtz. On coalitions and stable winners in plurality. In *8th International Workshop on Internet and Network Economics (WINE)*, pages 256–269. Springer, 2012.

[FP10] Piotr Faliszewski and Ariel D Procaccia. Ai's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[FRRS14] Piotr Faliszewski, Yannick Reisch, Jörg Rothe, and Lena Schend. Complexity of manipulation, bribery, and campaign management in bucklin and fallback voting. In *Proc. 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1357–1358. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[FSG+98] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D. Ullman. Computing iceberg queries efficiently. In *Proc. 24rd International Conference on Very Large Data Bases*, pages 299–310, 1998.

[FVBNS13] Vincent Froese, René Van Bevern, Rolf Niedermeier, and Manuel Sorge. A parameterized complexity analysis of combinatorial feature selection problems. In *Mathematical Foundations of Computer Science (MFCS)*, pages 445–456. Springer, 2013.

[Gae01] Wulf Gaertner. *Domain Conditions in Social Choice Theory*. Cambridge University Press, 2001.

[Gib73] A. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, pages 587–601, 1973.

[GJ79] Michael R Garey and David S Johnson. *Computers and Intractability*, volume 174. freeman New York, 1979.

[GKNW13] Serge Gaspers, Thomas Kalinowski, Nina Narodytska, and Toby Walsh. Coalitional manipulation for schulze's rule. In *Proc. 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 431–438. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[GN07] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.

[GNNW14] Serge Gaspers, Victor Naroditskiy, Nina Narodytska, and Toby Walsh. Possible and necessary winner problem in social polls. In *Proc. 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 613–620. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[GSTV07] Anna C. Gilbert, Martin J. Strauss, Joel A. Tropp, and Roman Vershynin. One sketch for all: fast algorithms for compressed sensing. In *Proc. 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 237–246, 2007.

[HAKW08] Noam Hazon, Yonatan Aumann, Sarit Kraus, and Michael Wooldridge. Evaluation of election outcomes under uncertainty. In *Proc. 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 959–966. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

270

[Hei93] Katherine Heinrich. Path decomposition. *Le matematiche*, 47(2):241–258, 1993.

[HHR97] Edith Hemaspaandra, Lane A Hemaspaandra, and Jörg Rothe. Exact analysis of dodgson elections: Lewis carroll's 1876 voting system is complete for parallel access to np. *Journal of the ACM (JACM)*, 44(6):806–825, 1997.

[HHS94] Tin Kam Ho, Jonathan J Hull, and Sargur N Srihari. Decision combination in multiple classifier systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(1):66–75, 1994.

[Hid99] Christian Hidber. Online association rule mining. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 145–156, 1999.

[HKTR04] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[HLK13] Noam Hazon, Raz Lin, and Sarit Kraus. How to change a groups collective decision? In *Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 198–205, 2013.

[HM97] Mel vin J Hinich and Michael C Munger. *Analytical politics*. Cambridge University Press, 1997.

[HPDW01] Jiawei Han, Jian Pei, Guozhu Dong, and Ke Wang. Efficient computation of iceberg cubes with complex measures. In *Proc. 2001 ACM SIGMOD International Conference on Management of Data,*, pages 1–12, 2001.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2000.

[HSST05] John Hershberger, Nisheeth Shrivastava, Subhash Suri, and Csaba D. Tóth. Space complexity of hierarchical heavy hitters in multi-dimensional data streams. In *Proc. Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 338–347, 2005.

[HSV05] Edith Hemaspaandra, Holger Spakowski, and Jörg Vogel. The complexity of kemeny elections. *THEOR COMPUT SCI*, 349(3):382–391, 2005.

[HUA83] John E Hopcroft, Jeffrey David Ullman, and Alfred Vaino Aho. *Data structures and algorithms*, volume 175. Addison-Wesley Boston, MA, USA:, 1983.

[IKM12] M. Isaksson, G. Kindler, and E. Mossel. The geometry of manipulation - a quantitative proof of the gibbard-satterthwaite theorem. *Combinatorica*, 32(2):221–250, 2012.

[JMP⁺14] Albert Jiang, Leandro Soriano Marcolino, Ariel D Procaccia, Tuomas Sandholm, Nisarg Shah, and Milind Tambe. Diverse randomized agents vote to win. In *Proc. Annual Conference on Neural Information Processing Systems*, pages 2573–2581, 2014.

[JSA08] Benjamin G. Jackson, Patrick S. Schnable, and Srinivas Aluru. Consensus genetic maps as median orders from inconsistent sources. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 5(2):161–171, 2008.

[Kem59] John G Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, April 2007.

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *Ann. Math. Stat.*, pages 79–86, 1951.

[KL05] Kathrin Konczak and Jérôme Lang. Voting procedures with incomplete preferences. In *Proc. International Joint Conference on Artificial Intelligence-05 Multidisciplinary Workshop on Advances in Preference Handling*, volume 20, 2005.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

[KNR99] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Comput. Complex.*, 8(1):21–49, 1999.

[KSP03] Richard M Karp, Scott Shenker, and Christos H Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28(1):51–55, 2003.

[KTW11] Peter Kellner, Joe Twyman, and Anthony Wells. Polling voting intentions. In *Political Communication in Britain*, pages 94–108. Springer, 2011.

[KX06] Abhishek Kumar and Jun (Jim) Xu. Sketch guided sampling - using on-line estimates of flow size for adaptive data collection. In *Proc. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, 2006.

[LB11a] Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proc. 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, volume 22, pages 287–293, 2011.

[LB11b] Tyler Lu and Craig Boutilier. Vote elicitation with probabilistic preference models: Empirical estimation and cost tradeoffs. In *Algorithmic Decision Theory*, pages 135–149. Springer, 2011.

[LB13] Tyler Lu and Craig Boutilier. Multi-winner social choice with incomplete preferences. In *Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 263–270. AAAI Press, 2013.

[Li14] Hang Li. Learning to rank for information retrieval and natural language processing. In *Synthesis Lectures on Human Language Technologies*, volume 7, pages 1–121. Morgan & Claypool Publishers, 2014.

[Lin91] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theory*, 37(1):145–151, 1991.

[LN06] Alessandra Lumini and Loris Nanni. Detector of image orientation based on borda count. *Pattern Recogn. Lett.*, 27(3):180–186, 2006.

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy Le Nguyen, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. arXiv:1604.01357, April 2016.

[LPR⁺07] Jérôme Lang, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Winner determination in sequential majority voting. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 1372–1377, 2007.

[LPR+12] Jérôme Lang, Maria Silvia Pini, Francesca Rossi, Domenico Salvagnin, Kristen Brent Venable, and Toby Walsh. Winner determination in voting trees with incomplete preferences and weighted votes. *Auton. Agent Multi Agent Syst.*, 25(1):130–157, 2012.

[MAE05] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proc. 10th International Conference on Database Theory*, ICDT'05, pages 398–412, Berlin, Heidelberg, 2005. Springer-Verlag.

[MBC+16] Hervé Moulin, Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[MBG04] Amélie Marian, Nicolas Bruno, and Luis Gravano. Evaluating top-k queries over web-accessible databases. *ACM Trans. Database Syst.*, 29(2):319–362, 2004.

[McG53] David C McGarvey. A theorem on the construction of voting paradoxes. *Econometrica*, pages 608–610, 1953.

[MCWG95] Andreu Mas-Collel, Michael D Whinston, and Jerry Green. Microeconomic theory, 1995.

[MG82] Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982.

[MG09] Md Maruf Monwar and Marina L Gavrilova. Multimodal biometric system using rank-level fusion approach. *IEEE Trans. Syst. Man. Cybern. B, Cybern.*, 39(4):867–878, 2009.

[Mir71] James A Mirrlees. An exploration in the theory of optimum income taxation. *Rev. Econ. Stud.*, pages 175–208, 1971.

[ML15] Vijay Menon and Kate Larson. Complexity of manipulation in elections with partial votes. *CoRR*, abs/1505.05900, 2015.

[MM02] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proc. 28th International Conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002.

[MNSW98] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998.

[Moo81] J. Strother Moore. J. Algorithm, June 1981. p. 208–209.

[MOO05] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low in.uences invariance and optimality. In *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 21–30, 2005.

[Mor78] Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 1978.

[Mou91] Hervi Moulin. *Axioms of cooperative decision making*. Number 15. Cambridge University Press, 1991.

[MPC12] Andrew Mao, Ariel D. Procaccia, and Yiling Chen. Social Choice for Human Computation. In *Proc. Fourth Workshop on Human Computation (HCOMP-12)*, 2012.

[MPC13] Andrew Mao, Ariel D. Procaccia, and Yiling Chen. Better Human Computation Through Principled Voting. In *Proc. 27th Conference on Artificial Intelligence (AAAI'13)*, AAAI'13, pages 1142–1148. AAAI Press, 2013.

[MPVR12] Nicholas Mattei, Maria Silvia Pini, K Brent Venable, and Francesca Rossi. Bribery in voting over combinatorial domains is easy. In *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1407–1408. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[MR81] Allan Meltzer and Scott F Richard. A rational theory of the size of government. *J Public Econ*, 89(5):914–27, 1981.

[MR91] D.J. Mullen and B. Roth. *Decision making: Its logic and practice*. Savage, MD: Rowman and Littlefield Publishers, Inc., 1991.

[MR15] Elchanan Mossel and Miklós Z Rácz. A quantitative gibbard-satterthwaite theorem without neutrality. *Combinatorica*, 35(3):317–387, 2015.

[MRSW11] Thomas R Magrino, Ronald L Rivest, Emily Shen, and David Wagner. Computing the margin of victory in IRV elections. In *Proceedings of the 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'11, pages 4–4, Berkeley, CA, USA, 2011. USENIX Association.

[Mut05] Shanmugavelayutham Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[MYCC07] Nikos Mamoulis, Man Lung Yiu, Kit Hung Cheng, and David W. Cheung. Efficient top-k aggregation of ranked inputs. *ACM Trans. Database Syst*, 32(3):19–64, 2007.

[NC06] Rabia Nuray and Fazli Can. Automatic ranking of information retrieval systems using data fusion. *Inf. Process Manag.*, 42(3):595–614, 2006.

[Nel12] Jelani Nelson. Sketching and streaming algorithms for processing massive data. *XRDS: Crossroads, The ACM Magazine for Students*, 19(1):14–19, 2012.

[Nie02] Rolf Niedermeier. Invitation to fixed-parameter algorithms. *Habilitationschrift, University of Tübingen*, 2002.

[NL07] Lawrence Norden and Samuelson Law. *Post-election audits: Restoring trust in elections*. University of California, Berkeley School of Law Boalt Hall, 2007.

[NW13] Nina Narodytska and Toby Walsh. Manipulating two stage voting rules. In *Proc. 12th International Conference on Autonomous Agents and Multiagent Systems (AA-MAS)*, pages 423–430. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[NW14] Nina Narodytska and Toby Walsh. The computational impact of partial votes on strategic voting. In *Proc. 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 657–662, 2014.

[NWX11] N. Narodytska, T. Walsh, and L. Xia. Manipulation of nanson's and baldwin's rules. In *Proc. International Conference on Artificial Intelligence (AAAI)*, pages 713–718, 2011.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, NY, USA, 2014.

[OE12] Svetlana Obraztsova and Edith Elkind. Optimal manipulation of voting rules. In *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 619–626. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[OEH11] Svetlana Obraztsova, Edith Elkind, and Noam Hazon. Ties matter: Complexity of voting manipulation revisited. In *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 71–78. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[OFB13] Joel Oren, Yuval Filmus, and Craig Boutilier. Efficient vote elicitation under candidate uncertainty. In *Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 309–316. AAAI Press, 2013.

[PE16] Dominik Peters and Edith Elkind. Preferences single-peaked on nice trees. In *Proc. 30th International Conference on Artificial Intelligence (AAAI)*, pages 594–600, 2016.

[PHG00] David M. Pennock, Eric Horvitz, and C. Lee Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *Proc. Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA.*, pages 729–734, 2000.

[plu] Plurality voting system. https://en.wikipedia.org/wiki/Plurality_voting_system.

[PPR15] Adarsh Prasad, Harsh Pareek, and Pradeep Ravikumar. Distributional rank aggregation, and an axiomatic analysis. In *Proc. 32nd International Conference on Machine Learning (ICML-15)*, pages 2104–2112, 2015.

[PR06] A.D. Procaccia and J.S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. In *Proc. Fifth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 497–504. ACM, 2006.

[PR07] Ariel D. Procaccia and Jeffrey S. Rosenschein. Average-case tractability of manipulation in voting via the fraction of manipulators. In *Proc. 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007*, pages 718–720, 2007.

[PRV13] Maria Silvia Pini, Francesca Rossi, and Kristen Brent Venable. Bribery in voting with soft constraints. In *Proc. International Conference on Artificial Intelligence (AAAI)*, AAAI'13, pages 803–809. AAAI Press, 2013.

[PRVW07] Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Incompleteness and incomparability in preference aggregation. In *Proc. 20nd International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 1464–1469, 2007.

[RGMT06] Michel Regenwetter, Bernard Grofman, Anthony Marley, and Ilia Tsetlin. Behavioral social choice. *Cambridge University Press*, 13:58–68, 2006.

[Rob77] Kevin Roberts. Voting over income tax schedules. *J Public Econ*, 8(3):329–340, 1977.

[Ron01] Dana Ron. Property testing. *Combinatorial Optimization-Dordrecht*, 9(2):597–643, 2001.

[RR67] Hartley Rogers and H Rogers. *Theory of recursive functions and effective computability*, volume 126. McGraw-Hill New York, 1967.

[RSW+07] Marko A Rodriguez, Daniel J Steinbock, Jennifer H Watkins, Carlos Gershenson, Johan Bollen, Victor Grey, and Brad Degraf. Smartocracy: Social networks for collective decision making. In *In 40th Annual Hawaii International Conference on Systems Science (HICSS'07). Waikoloa*, pages 90–99. IEEE, 2007.

[RV97] Paul Resnick and Hal R Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[Sat75] M.A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *J. Econ. Theory*, 10(2):187–217, 1975.

[SBAS04] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash
Suri. Medians and beyond: new aggregation techniques for sensor networks. In
Proc. 2nd International Conference on Embedded Networked Sensor Systems, pages
239–249, 2004.

[SCS11] Anand Sarwate, Stephen Checkoway, and Hovav Shacham. Risk-limiting audits
for nonplurality elections. Technical report, DTIC Document, 2011.

[Smi88] D. Smirnov. Shannon's information methods for lower bounds for probabilistic
communication complexity. Master's thesis, Moscow University, 1988.

[SON95] Ashok Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient
algorithm for mining association rules in large databases. In *Proc. 21th International Conference on Very Large Data Bases*, pages 432–444, 1995.

[Sta08a] Philip B Stark. Conservative statistical post-election audits. *Ann. Appl. Stat.*,
pages 550–581, 2008.

[Sta08b] Philip B Stark. A sharper discrepancy measure for post-election audits. *Ann.
Appl. Stat.*, pages 982–985, 2008.

[Sta09] Philip B Stark. Efficient post-election audits of multiple contests: 2009 california
tests. In *CELS 2009 4th Annual Conference on Empirical Legal Studies Paper*,
2009.

[SW15] Xiaoming Sun and David P. Woodruff. Tight bounds for graph problems in insertion streams. In *Proc. 18th. International Workshop on Approximation Algorithms
for Combinatorial Optimization Problems (APPROX 2015)*, pages 435–448, 2015.

[SYE13] Dmitry Shiryaev, Lan Yu, and Edith Elkind. On elections with robust winners.
In *Proc. 12th International Conference on Autonomous Agents and Multiagent
Systems (AAMAS)*, pages 415–422. International Foundation for Autonomous
Agents and Multiagent Systems, 2013.

[Toi96] Hannu Toivonen. Sampling large databases for association rules. In *Proc. 22th
International Conference on Very Large Data Bases*, pages 134–145, 1996.

[Tri89] Michael A Trick. Recognizing single-peaked preferences on a tree. *Math. Soc.
Sci*, 17(3):329–334, 1989.

[VWWZ15] Dirk Van Gucht, Ryan Williams, David P Woodruff, and Qin Zhang. The communication complexity of distributed set-joins with applications to matrix multiplication. In *Proc. 34th ACM Symposium on Principles of Database Systems*, pages 199–212. ACM, 2015.

[VZ14] Maksims N Volkovs and Richard S Zemel. New learning methods for supervised and unsupervised preference aggregation. *J. Mach. Learn. Res.*, 15(1):1135–1176, 2014.

[Wag15] Bo Waggoner. Lp testing and learning of discrete distributions. In *Proc. 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 347–356, New York, NY, USA, 2015. ACM.

[Wal07] Toby Walsh. Uncertainty in preference elicitation and aggregation. In *Proc. International Conference on Artificial Intelligence (AAAI)*, volume 22, pages 3–8, 2007.

[Wal10] T. Walsh. An empirical study of the manipulability of single transferable voting. In *Proc. 19th European Conference on Artificial Intelligence (ECAI)*, pages 257–262, 2010.

[Wal11a] Toby Walsh. Is computational complexity a barrier to manipulation? *Ann Math Artif Intell*, 62(1-2):7–26, 2011.

[Wal11b] Toby Walsh. Where are the hard manipulation problems? *J. Artif. Intell. Res.*, pages 1–29, 2011.

[WL04] Gang Wang and Frederick H Lochovsky. Feature selection with conditional mutual information maximin in text categorization. In *Proc. 13th ACM International Conference on Information and Knowledge Management*, pages 342–349. ACM, 2004.

[WWH+10] Scott Wolchok, Eric Wustrow, J Alex Halderman, Hari K Prasad, Arun Kankipati, Sai Krishna Sakhamuri, Vasavya Yagati, and Rop Gonggrijp. Security analysis of india's electronic voting machines. In *Proc. 17th ACM Conference on Computer and Communications Security*, pages 1–14. ACM, 2010.

[XC08a] Lirong Xia and Vincent Conitzer. Generalized scoring rules and the frequency of coalitional manipulability. In *Proc. 9th ACM conference on Electronic Commerce (EC)*, pages 109–118. ACM, 2008.

[XC08b] Lirong Xia and Vincent Conitzer. A sufficient condition for voting rules to be frequently manipulable. In *Proc. 9th ACM conference on Electronic Commerce (EC)*, pages 99–108. ACM, 2008.

[XC11] Lirong Xia and Vincent Conitzer. Determining possible and necessary winners under common voting rules given partial orders. volume 41, pages 25–67. AI Access Foundation, 2011.

[XCP10] Lirong Xia, Vincent Conitzer, and Ariel D Procaccia. A scheduling approach to coalitional manipulation. In *Proc. 11th ACM conference on Electronic Commerce (EC)*, pages 275–284. ACM, 2010.

[Xia12] Lirong Xia. Computing the margin of victory for various voting rules. In *Proc. 13th ACM Conference on Electronic Commerce (EC)*, pages 982–999. ACM, 2012.

[XZP⁺09] Lirong Xia, Michael Zuckerman, Ariel D Procaccia, Vincent Conitzer, and Jeffrey S Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI)*, volume 9, pages 348–352, 2009.

[Yan14] Yongjie Yang. Election attacks with few candidates. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 1131–1132, 2014.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proc. eleventh annual ACM symposium on Theory of computing*, pages 209–213. ACM, 1979.

[YCE13] Lan Yu, Hau Chan, and Edith Elkind. Multiwinner elections under preferences that are single-peaked on a tree. In *Proc. Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 425–431. AAAI Press, 2013.

[YG13] Yongjie Yang and Jiong Guo. Exact algorithms for weighted and unweighted borda manipulation problems. In *Proc. 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1327–1328, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[YHL04] Wenci Yu, Han Hoogeveen, and Jan Karel Lenstra. Minimizing makespan in a two-machine flow shop with delays and unit-time operations is np-hard. *J. of Scheduling*, 7(5):333–348, September 2004.

[You77] H Peyton Young. Extending condorcet's rule. *J. Econ. Theory*, 16(2):335–353, 1977.

[ZLR11] Michael Zuckerman, Omer Lev, and Jeffrey S Rosenschein. An algorithm for the coalitional manipulation problem under maximin. In *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 845–852. International Foundation for Autonomous Agents and Multiagent Systems, 2011.