Attacking and Defending Popular Election Systems

by

Curtis Menton

Submitted in Partial Fulfillment of the
Requirements for the Degree
Doctor of Philosophy

Supervised by Professor Lane A. Hemaspaandra

Department of Computer Science
Arts, Sciences, and Engineering
Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester
Rochester, NY
2013

To my parents and to my fiancée.

Biographical Sketch

Curtis Menton was born on April 14, 1986, in Ithaca, New York. He graduated from the Rochester Institute of Technology with a Bachelor of Science and a Master of Science in Computer Science in May 2010. In January of 2010 he began doctoral studies at the Department of Computer Science of the University of Rochester under the supervision of Lane A. Hemaspaandra. He was awarded a Master of Science in Computer Science by the University of Rochester in October 2011. He taught courses as an adjunct faculty member at the Rochester Institute of Technology during each of his three summers at the University of Rochester.

The following papers were the result of work conducted during doctoral study.

- C. Menton. **Normalized Range Voting Broadly Resists Control**, to appear in Theory of Computing Systems.
- C. Menton and P. Singh, **Manipulation Can Be Hard in Tractable Voting Systems Even for Constant-Sized Coalitions**, Computer Science Review, 6(2–3):71-87, 2012.
- L. Hemaspaandra, R. Lavaee, and C. Menton, **Schulze and Ranked-Pairs Voting Are Fixed-Parameter Tractable to Bribe, Manipulate, and Control**, to appear in *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, May 2013.
- E. Hemaspaandra, L. Hemaspaandra, and C. Menton, **Search Versus Decision for Election Manipulation Problems**, in *Proceedings of the 30th Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, February/March 2013.
- C. Menton and P. Singh, **Control Complexity of Schulze Voting**, to appear in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, August 2013.

Acknowledgements

I would like to thank my advisor Lane A. Hemaspaandra for his invaluable assistance in preparing this thesis and in guiding my academic career.

I would like to thank Edith Hemaspaandra for inspiring me towards academia and graduate study, and for her guidance and advice over the years, since her time as my master's thesis advisor at the Rochester Institute of Technology.

I thank Lane A. Hemaspaandra, Edith Hemaspaandra, Rahman Lavaee, and Preetjot Singh for their contributions as my collaborators on my work as a graduate student.

I thank the members of my dissertation committee: Lane A. Hemaspaandra, Edith Hemaspaandra, Joel Seiferas, Daniel Gildea, and Tasos Kalandrakis for volunteering their time and greatly improving this thesis with their helpful advice.

I am grateful to Jörg Rothe and the University of Düsseldorf for hosting me for a lovely and productive fall spent doing research in Düsseldorf.

I thank the Computer Science Department of the Rochester Institute of Technology for giving me the opportunity to teach during my time as a graduate student and giving me invaluable classroom experience.

I also thank Marzieh Bazrafshan, Rahman Lavaee, Andrew Lin, and Preetjot Singh for their assistance in editing this thesis and/or previous versions of the work presented in this thesis.

I have to thank my parents and my brother for all their support throughout my life; they truly made me who I am.

Finally, I owe everything to my fiancée Marzieh Bazrafshan for her love and support, and for bringing joy to my life.

Abstract

The thesis of this dissertation is that complexity and algorithms, used appropriately, are important factors in assessing the value and uses of election systems. The chapter on search versus decision points out the importance of that "appropriately"; it proves that unless integer factoring is easy, the standard definitions of manipulability do not capture what they were designed to capture. Other chapters use complexity and algorithms to analyze the complexity of various types of manipulative attacks on elections, as a way of understanding how computationally vulnerable election systems are. Among the contributions of those chapters are: showing that a type of range voting is the most control-attack resistant among all currently analyzed natural election systems; exploring for the first time the detailed control complexity of Schulze elections; and exploring the parameterized complexity of manipulative actions in Schulze and ranked-pairs elections. Such results will better allow choosers of election methods to match the protections of the systems they choose with the types of attack that are of greatest concern.

Contributors and Funding Sources

This work was supervised by a dissertation committee consisting of Professors Lane A. Hemaspaandra (advisor), Joel Seiferas, and Daniel Gildea of the Department of Computer Science of the University of Rochester, Tasos Kalandrakis of the Department of Political Science of the University of Rochester, and Edith Hemaspaandra of the Computer Science Department of the Rochester Institute of Technology. The work in Chapter 3 was conducted jointly with Preetjot Singh, with all presented proofs being the work of the author of this thesis. This work will be published in 2013 at the IJCAI conference [MS13]. The work in Chapter 4 was conducted jointly with Lane A. Hemaspaandra and Rahman Lavaee, with the proofs for control by deleting voters, control by partition of voters, and most results in the Weighted Case section and the Other Results section being the work of the author of this thesis. This work will be published in 2013 at the AAMAS conference [HLM13]. The work in Chapter 6 was conducted jointly with Edith Hemaspaandra and Lane A. Hemaspaandra, with the search-reduces-to-decision proofs being the work of the author of this thesis. This work was published in 2013 at the STACS conference [HHM13]. Chapter A was originally prepared as a paper jointly with Preetjot Singh, and was published in 2012 in Computer Science Review [MS12]. This material is based upon work supported by National Science Foundation grants CCF-0915792 and CCF-1101479. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the above named organization.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an introduction to the thesis, a summary of its results, and an overview of its organization.

1.1 Introduction

Elections have been a tool for collaborative decision-making for thousands of years, dating back at least as far as ancient Greece. The formal study of voting began in the 18th century and continues as a vibrant discipline drawing on mathematics, economics, and political science. The classic stream of voting theory work was concerned with evaluating election systems, often on the basis of various criteria. One of the earliest examples of this is the work of the Marquis de Condorcet, an 18th century French mathematician and philosopher. Condorcet's criterion states that an election system should select a candidate that is preferred pairwise to every other candidate by a majority of voters. This work initiated a rivalry with Jean Charles de Borda, whose voting system *Borda count* fails to satisfy this criterion. But in a foreshadowing of later key results, it turns out that no voting system can always satisfy Condorcet's goal, as such a candidate does not always exist, due to the possibility of cycles in pairwise preferences. This is termed the Condorcet Paradox, and systems that do select such a candidate if they do exist are considered to satisfy the Condorcet criterion.

Like the Condorcet Paradox, the key modern results of voting theory show that all election systems are imperfect. Arrow's Theorem shows that no election system can satisfy all of a set of reasonable and desirable criteria [Arr50]. According to Paul Samuelson [Pou08], "What Kenneth Arrow proved once and for all is that there cannot possibly be ... an ideal voting scheme." Another key flaw was identified by the Gibbard-Satterthwaite Theo-

rem [Gib73, Sat75] (and the later extension of this result by Duggan and Schwartz [DS00]): All reasonable voting systems are subject in some cases to manipulation (i.e., strategic voting). That is, there will be some cases where a voter will have incentive to vote contrary to their true preferences in order to achieve a better outcome. There had long been awareness of the problem of strategic voting, but it had never been adequately resolved. When confronted with the potential for strategic voting in his system, Jean Charles de Borda exclaimed "My scheme is only intended for honest men" [Bla58]. But much later the Gibbard-Satterthwaite Theorem proved that not just Borda's system but instead all reasonable systems have the same flaw.

Of course, even with these challenging results the study of voting continued to flourish, but with the understanding that no perfect system can exist, and one must instead find the best tradeoffs or compromises, and choose a system that satisfies the most important requirements for the task at hand. And with the increasing prevalence and importance of elections in our democratic world, solving the problems of voting is as important as ever.

The field of computational social choice offers a compromise solution to the problems of strategic voting and other manipulative attacks: Even if such actions are possible, it may be that in some systems planning such attacks is beyond the capabilities of a computationally limited attacker, which should provide significant protection against these attacks in practice. This field was kicked off by the work of Bartholdi, Tovey, and Trick, who introduced the computational study of manipulation [BTT89a] and control [BTT92]. Manipulation models the problem of strategic voting (the classic manipulation problem) and control models various manipulative actions taken by a dishonest election chair, e.g., deleting voters or adding candidates. Faliszewski, Hemaspaandra, and Hemaspaandra [FHH09] later introduced the bribery problem, a sort of extended version of manipulation where a briber first selects voters to bribe and then assigns their votes. All these problems are formalized as problems in the standard way in computer science and their difficulty is analyzed using the tools of complexity theory.

Computational issues in voting have become more significant and urgent with the increasing importance of voting in artificial intelligence and multiagent systems, with computational social choice sometimes being considered a subarea of multiagent systems. Some applications include recommender systems [GMHS99, PHG00], consensus mechanisms for planning [ER91] and search engine design [Lif00, DKNS01]. In such elections, electronically implemented, and with potentially large numbers of both voters and candidates, computational issues are more likely to come to the forefront.

Thus an explosion of work has followed in recent years, with a great variety of problems studied in many voting systems. There has been a great amount of work studying the standard worst-case complexity of manipulative actions in various voting systems.

In the case of manipulation, Bartholdi and Orlin closely followed the original paper of Bartholdi, Tovey, and Trick [BTT89a] with work showing manipulation to be hard in the popular system single transferable vote [BO91]. Conitzer, Sandholm, and Lang [CSL07] found exactly how many candidates are required to be present for manipulation to be NP-hard in a variety of voting systems. Xia, Zuckerman, Procaccia, Conitzer, and Rosenschein [XZP+09] found unweighted manipulation to be hard for maximin and ranked pairs. Work by Faliszewski, Hemaspaandra, and Schnoor studied the complexity of manipulation in several variants of the Copeland voting system [FHS08, FHS10]. The survey by Menton and Singh ([MS12], also included as Appendix A of this thesis) covers much of what is known about the basic unweighted manipulation problem.

In the case of bribery, Faliszewski, Hemaspaandra, Hemaspaandra, and Rothe studied the complexity of bribery in the Llull and Copeland systems [FHHR07]. Schlotter, Faliszewski, and Elkind studied the complexity of bribery in several approval-like voting systems [SFE11]. Elkind, Faliszewski, and Slinko studied the complexity of a finer-grained bribery variant termed *swap bribery* [EFS09].

In the case of control, Faliszewski, Hemaspaandra, Hemaspaandra, and Rothe found Llull and Copeland elections to resist every variant of constructive control [FHHR07]. Erdélyi, Nowak, and Rothe found the approval-like system SP-AV to be broadly resistant to control [ENR09]. Erdélyi, Piras, and Rothe found the system fallback voting to resist nearly every standard control case [EPR11], and Menton found normalized range voting to match fallback voting's control resistances ([Men12], also presented in part in Chapter 5 of this thesis).

More recently, a significant amount of work has branched out from standard worst-case results, including work on approximation algorithms [BFH+08, ZPR09, EF10a], phase transitions [Wal09], and parameterized complexity ([DS12, BU09, FHHR09], see also the survey [BBCN12]), and also including work on cases with restricted or alternative vote models, such as the irrational-vote model [FHS10] and single-peaked preferences [FHHR11].

The goal of this thesis is to study and classify the complexity of manipulative attacks on several popular election systems. The work here includes both classical worst-case complexity analysis and parameterized complexity analysis. The work in this thesis studies the systems normalized range voting [Smi00], Schulze voting [Sch11], and ranked-pairs voting [Tid87]. These systems are recently introduced, academically studied voting systems that also have garnered a great deal of interest, and especially in the case of Schulze voting, a great deal of real-world use. Any choice of a voting system involves tradeoffs, and it is important to have as much information as possible about the properties of various voting systems of interest so a system with the most important desirable properties can be chosen for a particular purpose. Thus we hope that this work can aid the choosers of election sys-

tems in cases when computational resistance to manipulative action is an election property of concern.

Beyond the primary focus of this thesis, the other main contribution is work comparing the complexity of search and decision for election manipulative actions. This work is targeted primarily at the stream of computational social choice research rather than at the organizers of elections, and it shows that for many of the standard set of election manipulative actions, the complexity of the search and decision version can differ. Thus the standard definition of computational vulnerability, which is based on the decision problem, is too broad, as it is possible for the decision version to be easy while the search version is hard. This work implies that decades of work in computational social choice has used the wrong definition for the key notion of system vulnerability, and the field should refocus to be more concerned about search problems.

1.2 Summary of Results

1.2.1 Control in Schulze Voting

Schulze voting is a recently introduced voting system that has gained a large amount of real-world use. Though it hails from academic literature [Sch11] and is more complex to describe than most other popular voting systems, it has become quite widespread as the voting system of choice for a large number of organizations, including political parties, open-source software projects, and MTV [Wik12]. According to Wikipedia, "currently the Schulze method is the most widespread Condorcet method" [Wik12].

Schulze voting determines the winners of an election in the following way: The system builds a graph representation of the election called the weighted majority graph (WMG), where vertices are candidates and edges represent with their weights the result of each pairwise contest between candidates. Then we find all the best paths in the graph from every vertex to every other vertex. The winners are then the candidates that are unbeaten by any other candidate in best-path strength. Though this is a much more algorithmically involved winner problem that most typical popular voting systems, it is still solvable in polynomial time, so Schulze is a tractable voting system.

Earlier work by Parkes and Xia [PX12] examined the manipulation, bribery, and control complexity of Schulze voting, but it left many open problems, including all control cases but a few. So here we present results resolving many of these open problems. We prove that Schulze voting is computationally resistant to at least 15 of the 22 standard control cases and we prove that it is vulnerable to several others. Additionally, in the course of proving our results we find that in fact a broad class of voting systems are vulnerable to several

of these cases, namely Condorcet voting systems that also possess a "weak" version of the Condorcet criterion, and the relevant control cases are all the possible variants of destructive control by partition of candidates. For the few control problems that we do not resolve, we connect their complexity to a graph cut variant called *path-preserving vertex cut*.

The work in this chapter is in the vein of classical worst-case complexity. We proved our resistance results using standard polynomial-time many-one reductions, in every case from 3SAT.

This work achieves valuable results expanding what is known about the popular system Schulze voting. Given Schulze voting's popularity and widespread use, it is important to understand its behavior under election manipulative actions so that its use can be evaluated in situations when computational resistance is a desirable property.

1.2.2 Parameterized Complexity in Schulze Voting and Ranked Pairs

The work of Parkes and Xia [PX12] and the work presented here found that Schulze voting and another system, ranked pairs, resist many of the standard manipulative actions. But a common worry is that worst-case hardness results will not mean that carrying out a manipulative action is hard in practice, as such actions may be relatively easy in common inputs with a small number of candidates. This work formally explores this notion by examining the parameterized complexity of manipulative actions in Schulze and ranked pairs.

Ranked pairs is a voting system with a highly sequential winner determination procedure. The winners are determined by sorting the WMG edges by decreasing weight, and building a new graph by including them in order, but discarding edges that create cycles (there are other intricacies having to do with tiebreaking that are described in Section 2.1.7). The winner is the candidate that has no in-edges in this final graph. Recall that the Condorcet Paradox is due to the possibility of cycles in the WMG. In ranked-pairs voting we build a new cycle-free graph, and from this graph it is easy to determine a clear Condorcet-like winner.

We find that every standard manipulative action problem is in FPT for both Schulze and ranked pairs when parameterized on the number of candidates, thus proving that the difficulty of any of these problems depends in an essential way on the number of candidates, and proving that they are relatively easy when the number of candidates is bounded. The candidate control cases were reasonably trivially solvable through brute force without special structures or algorithms, as when the number of candidates is bounded, the number of possible actions in these cases is bounded too and we can just brute force over them.

However, the manipulation, bribery, and voter control results required much more sophisticated techniques, to the extent that techniques used to achieve our results are as significant a contribution as the results themselves. The results were obtained through building a structure for each of these voting systems that models a particular strategy for manipulative action, and embedding the problem of using this strategy into an integer linear program. We can then brute-force over all possible strategies and solve these programs efficiently (with a fixed number of candidates) using Lenstra's algorithm [Len83]. Although other work has taken a similar approach of embedding manipulative action problems in linear programs (for instance, [FHHR09]), the complexity of the two systems studied here requires much more sophisticated structures for describing the strategies involved in winning such an election.

The structure for Schulze elections, the Schulze Winner-Set Certification Framework (SWCF), describes the winner set of the election and the particular strategy for victory that each winner took. For Schulze, this means the strong paths from each winning candidate to every other candidate. In the case of losing candidates, it specifies a path to that candidate that the loser cannot beat. Normally the number of such structures would be far to large to loop over, but when we consider a fixed number of candidates the number is fixed. We then embed each into an integer linear programming feasibility problem (ILPFP), with the constraints primarily modeling inequalities between WMG edge weights—We enforce the claims of the SWCF by claiming that every edge in every supposedly strong path is at least as strong as every edge in every path in the opposite direction. An assignment to the variables of this ILPFP corresponds to an attempt to satisfy the strategy of that particular SWCF, so if the ILPFP is feasible, we have success. Again, in normal circumstances the size of such an ILPFP would be simply enormous, but it remains polynomial in size with a fixed number of candidates, and furthermore the number of variables is fixed, enabling us to achieve good performance from Lenstra's algorithm [Len83].

The Ranked-Pairs Winner-Set Certification Framework is similar in concept, though it makes very different claims due to the unusually sequential nature of the voting system. The framework consists of a script describing the sequential operation of the system: what edges are considered in what order, and whether and how the edges are set in the election graph that is being built. The number of such structures is exponential in the number of candidates, but it is constant with a fixed number of candidates. And we can embed the problem of satisfying such a structure in an ILPFP, largely in terms of inequalities between edges as before. Thus we can reuse much of our work for Schulze for ranked-pairs given the frameworks we build for each.

Once we have our basic approach and the WCF structures for both Schulze and ranked pairs, achieving the results is relatively straightforward. For each manipulative action,

we describe how to implement the core types of constraints that we need for both voting systems (built in terms of inequalities between edge weights), and also additional constraints to ensure the validity of the manipulative action.

We pushed these techniques to their limits for these two voting systems, even handling some weighted cases of manipulative actions, and also achieved results in some other parameterizations, in some cases achieving W[2]-hardness results and showing that those problems are very unlikely to be in FPT. The techniques used here could lead to results in many similar voting systems.

1.2.3 Control in NRV

Range voting (RV) is a voting system using an alternative voter model that allows voters to score candidates individually in a range from 0 to some k, rather than having to rank the candidates in a linear order. Normalized range voting (NRV) is a range voting variant that maximizes the influence of each voter by normalizing their vote so that their scores span the entire range available to them. The complex, shifting behavior induced by the normalization step subtly but significantly changes the properties of the voting system. Normalized range voting does not possess some of the nice proprieties possessed by range voting (especially independence of irrelevant alternatives, one of Arrow's criteria), but instead it possesses a notable number of resistances to control.

Earlier work resolved the complexity of all but one case of control for normalized range voting [Men09]. The work presented here resolves the final open case, destructive control by partition of voters in the ties-eliminate model. This work was again working in the standard classical complexity model, and this proof was accomplished with a polynomial-time many-one reduction from the Exact Cover by Three-Sets problem.

Thus this work shows that normalized range voting possesses 20 of the 22 standard control cases, a number unbeaten by any natural voting system. Unnatural systems exist that are resistant to every single case [HHR09], though they would likely not be acceptable for practical use. Besides NRV, the system fallback voting is currently known to possess the same set of 20 control resistances [ER10, EF10b, EPR11].

Range voting and normalized range voting were already popular voting systems, with many enthusiastic advocates. This work shows that normalized range voting is even more attractive than thought when computational resistance to control is a significant factor of concern.

1.2.4 Reducing Search to Decision for Election Problems

The standard definitions of resistance and vulnerability of a voting system to a manipulative action are based on decision problems for the manipulative actions [BTT89a]. Studying problems in terms of decision problems has been the primary convention over decades of theoretical computer science research. However, in most cases the complexity of the associated search problem—the problem of finding a solution, not just finding whether one exists—is more practically relevant. We don't just want to know whether we can succeed in manipulation, we want to know how to do it! If the complexities of search and decision are always same, then there is no problem. But if the complexities can differ, then we may misassess the practical difficulty of a problem: We could say that a system is vulnerable to a particular manipulative action when actually finding the solution for that action is hard.

We prove that for many of the standard election manipulative actions, the search problem efficiently reduces to the decision problem, thereby tying the complexity of search and decision together. For these cases the conventions of the field present no problems and a claim of either vulnerability or resistance will correspond to the practical difficulty of the problem.

However, in the remaining cases, we show that there is a voting system for which the complexities of the search and decision versions of that problem are provably different, given the common complexity theoretic assumption that $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$. These proofs draw in part on a complexity-theoretic result known as the Borodin-Demers Theorem [BD76]. That result says that unless $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ there exists a polynomial-time recognizable set that is a subset of SAT, yet whose search version is not polynomial-time computable. For each manipulative action problem, we construct a voting system based on such a Borodin-Demers set where that manipulative action decision problem is in P, but for which the manipulative action search problem cannot have a polynomial-time function (since the Borodin-Demers set search problem reduces to this problem). We note that this is the first use of the Borodin-Demers Theorem in an applied domain that we are aware of. And we also note that it is highly surprising that it is possible to achieve such results for elections: the Borodin-Demers set is in the domain of SAT, but SAT is easily self-reducible and so the complexities of search and decision are tied together, but elections, despite being an entirely different domain, have so much potential complexity and so little structure that we are able successfully embed the Borodin-Demers set within them.

Assuming that $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$, there will be voting systems for which a group of manipulators will sometimes note that they are sure there is some way for them to succeed in manipulation, but they have no idea how to carry that plan out. It must be that $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$ for the problem of factoring large integers to be hard, so the usability of the RSA cryptosystem and the security of a large portion of secure communications depends on this

assumption. This work thus implies that of computational social choice should realign to be more concerned with the complexity of search problems, at least in the case of easiness results.

1.3 Organization of this Thesis

This thesis is organized as follows: Chapter 2 provides preliminaries and definitions. Chapter 3 describes work on the control complexity of Schulze voting. Chapter 4 describes work on the parameterized complexity of manipulative actions in Schulze as well as another system known as ranked pairs. Chapter 5 describes work completing the worst-case analysis of control in the system normalized range voting. Chapter 6 describes work comparing the complexity of the search and decision versions of election manipulative action problems. Additionally, a survey on tractable voting systems that are hard to manipulate with a constant number of manipulators is included as Chapter A.

Chapter 2

Preliminaries

2.1 Voting Theory

Formally, an election is a pair (C, V) where C is the set of candidates and V is the set of voters. Each voter has a unique name so they can be uniquely identified, and (except when otherwise specified) a strict linear preference ordering over the candidates. Often the voter names are disregarded and a voting system only uses the voter preferences to determine the winners. When we write out a preference ordering, we write the most-preferred candidate first, as $a > b > c$ for a vote preferring a over b over c.

A voting system (or an election system) is a function that takes as input an election (C, V) and produces a set of winners, a subset of the candidate set C. While in a political context we often want an election to result in a single winner, there are other situations where it is reasonable to select multiple winners or possibly none at all. For instance, several countries have allowed voters to select "none of the above" in elections otherwise operating as plurality elections (where voters select their top choice and the candidate with the most votes wins), with no candidate being selected if this choice is the most favored. And the voting process used to select entrants to the Baseball Hall of Fame can potentially select no entrants, and has done so on two separate occasions.

Also, many natural voting systems can potentially result in a two-or-more-way tie, and some tiebreaking procedure would have to be implemented to select a single winner. We choose not to require the incorporation of such a mechanism into our voting systems and instead allow them to output multiple winners.

2.1.1 The Condorcet Criterion

The key results in voting theory show that no voting system is perfect, that there are inherent problems with voting that no voting system can fix. The first example of such a result was the work of the Marquis de Condorcet, an 18th century French mathematician. He introduced a seemingly reasonable goal for a voting system to achieve: A voting system should select a candidate that when compared pairwise to each other candidate, is preferred to those candidates by a majority of voters. Such a candidate is known as a *Condorcet winner*. However, Condorcet discovered that such a candidate does not exist in every election. This is because there can be cycles in the pairwise preferences in an election.

For instance, consider an election over candidates $\{a, b, c\}$ where we have the following votes.

- $a > b > c$
- $b > c > a$
- $c > a > b$.

In this election, two voters prefer a over b, two prefer b over c, and two prefer c over a. Thus our overall preferences are irrational and cyclic and there is no Condorcet winner.

A voting system is said to satisfy the Condorcet criterion, or to be a Condorcet or Condorcet-consistent voting system, if it selects a Condorcet winner as the only winner whenever one does exist. Notably, many common voting systems do not satisfy this criterion. For instance, consider the following plurality election over candidates $\{a, b, c\}$.

# Voters	Preferences
5	$a > b > c$
4	$b > c > a$
2	$c > b > a$

In this election, using the plurality system, a will be the only winner, having the most first-place votes, but b in fact is the Condorcet winner, with six out of eleven votes ranking b over a and nine out of eleven votes ranking b over c.

2.1.2 Arrow's Theorem

The modern study of elections began with the work of Kenneth Arrow [Arr50]. Arrow formalized the standard model of a voting system, where votes are given as linear orderings of the candidates. This was in contrast to the previous economic model which aggregated

preferences from numerical utilities, representing the economic utility a voter would gain from the selection of each candidate.

Next, he described a set of reasonable criteria that one would want any voting system to satisfy, and showed that no voting system can satisfy all of them. The criteria are the following. One is *unrestricted domain*, meaning that no set of votes should be inadmissible as input to the voting system. Another is *nonimposition*, meaning that all election outcomes should be possible. Another is *monotonicity*, meaning that increasing the ranking for some candidate in a vote should never hurt the overall performance of that candidate. Another is *nondictatorship*, meaning that no one voter should decide the entire outcome of an election. The last and hardest to understand is *independence of irrelevant alternatives*. This criterion is satisfied if the overall relative ranking of two candidates is not affected by changes in the preferences of voters for other candidates.

Arrow showed that no voting system can satisfy all of these criteria, so we must always compromise and be willing to give up at least one of them. For instance, the voting system plurality fails to meet the criterion independence of irrelevant alternatives, and the system single transferable vote is not even monotonic, inspiring Doron and Kronick to refer to it as a "perverse social choice function" [DK77].

2.1.3 Gibbard-Satterthwaite Theorem

The Gibbard-Satterthwaite Theorem shows that any voting system that chooses a unique winner is susceptible to strategic voting, also known as manipulation [Gib73, Sat75]. That is, in any voting system there will be cases where a voter can achieve a result more in line with their preferences by voting in some way counter to their true preferences. The later Duggan-Schwartz Theorem extended this result to elections that can have multiple winners [DS00].

For instance, consider a plurality election over candidates $\{a, b, c\}$ where we have the following voters.

#	Preferences
5	$a > b > c$
4	$b > a > c$
2	$c > b > a$

In the initial election, a has the most first-place votes (five) so a will be the plurality winner. However, if both of the voters in the third class strategically vote by reversing c and b in their votes, b will be the clear winner of the election with six first-place votes. Thus by voting contrary to their true preferences, these voters achieve a better result for themselves,

the selection of their second favorite instead of their least favorite candidate.

2.1.4 Voting Systems

Though there are many voting systems, most common voting systems can be organized into a few categories. Many common voting systems can be formalized as *scoring protocols*. In a scoring protocol, we have a scoring vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ of nonnegative integers in an election with m candidates, where $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$. This vector is used to award points to candidates based on their position in each vote, with the first-ranked candidate in a vote earning α_1 points from that vote, the second-ranked candidate earning α_2 points from that vote, and so on, down to the least-preferred candidate earning α_m points. The winners for the election are the candidates with the most cumulative points over all votes.

The plurality voting system, for instance, can be considered a scoring protocol with a scoring vector $(1, 0, 0, \ldots, 0)$ of appropriate length, while the "opposite" voting system, *veto*, where each voter picks one candidate to reject, is represented with a vector matching $(1, 1, \ldots, 1, 0)$. Borda voting is a scoring protocol with the scoring vector $(m-1, m-2, \ldots, 1, 0)$.

Another class of voting systems are those where the winner is determined through pairwise comparisons of the candidates. These are often voting systems that meet the Condorcet criterion. One such system is Copeland voting, where candidates are awarded points from pairwise victories (and potentially some points from pairwise ties, depending on the variant) and the candidate with the most points wins. A variant of this system was introduced by the 13th century mystic Ramon Llull, with this work, some of the earliest known in the study of voting, lying undiscovered for hundreds of years.

For such voting systems we determine the winner through comparison of the relative preferences of voters between pairs of candidates rather than with candidate's specific rankings within votes. We will thus introduce some useful objects and notation. The *weighted majority graph* (WMG) is a graph representation of an election, with vertices corresponding to candidates and edges representing the relative performance of pairs of candidates. The weight of an edge from a to b, also termed the *net advantage* for a over b, is the number of votes that prefer a over b minus the number that prefer b over a.

It is obviously easily possible to construct the WMG given a collection of votes, but it is interesting to note that we can easily convert in the other direction as well. Given a WMG where either all the edge weights are even or all the edge weights are odd, we can construct an equivalent set of votes in time polynomial in the number of candidates and in the maximum magnitude of any edge weight. This method is due to McGarvey [McG53]. We will use this method later to more easily construct elections in the course of reductions,

in order to avoid having to explicitly specify the votes.

Other voting systems eschew the standard linear-order vote model and instead allow voters to rate candidates individually. Rather than ranking the candidates in order of preference, voters give some abstraction of their utility for the selection of each candidate. Such voting systems are thus known as utilitarian voting systems [Hil05]. This is essentially the model that Arrow rejected due to the incomparability of personal utilities. The best known system in this class is approval voting, where voters choose to approve or disapprove of each candidate (or score them as 0 or 1), and the most approved candidates win. A natural extension, range voting, allows voters to use a wider score range, ranging from 0 to k, with k being a parameter for the voting system, and the candidates with the highest sum score over all the votes win [Smi00]. There are also some voting systems that make use of both a standard preference ordering and an approval vector, such as sincere-strategy preference-based approval voting (SP-AV) [BS06] and fallback voting [ER10].

2.1.5 Range Voting and Normalized Range Voting

Range voting (RV) is a voting system that uses an alternative voter model [Smi00]. Range voting is essentially an extension of approval voting, where voters can simply score candidates within a larger range than 0–1. Given the votes, the winners are determined by adding up the scores for all the candidates and the highest scored candidates win. We say that a k-range election is one where the available scoring range is 0–k. Of course, when k is 1 the system is identical to approval voting. As this system does not use Arrow's model of a voting system, it is not constrained by Arrow's celebrated Impossibility Theorem [Smi00] (which is also true for approval voting).

Normalized range voting (NRV) is a range voting variant that maximizes the influence of each voter by normalizing their vote so that their scores span the entire range available to them. Voters specify their vote in the same way as in standard range voting, but as part of the vote aggregation process, each vote is normalized to span the entire rational range $[0, k]$ with the same relative preferences between candidates as given in the initial vote.

Formally, in a vote with maximum and minimum scores m and n with $m > n$, each score s is normalized to $\frac{k(s-n)}{m-n}$. This will give the voter's highest ranked candidate a score of k, their lowest ranked candidate a score of 0, and all candidates in between will be scored proportionally with the original vote. Any vote that gives the same score to each candidate will not be counted, as these unconcerned votes would not affect the election anyways. We make no effort to coerce an unconcerned vote into one distinguishing between candidates.

For instance, consider the following 10-normalized range election over candidates $\{a, b, c\}$.

# Voters	a score	b score	c score
3	10	5	0
3	4	10	0
1	10	9	8
score	52	54	8

In this election, b has the most points before the normalization step, but the last voter is not making full use of their influence. The normalization step will reevaluate the votes and arrive at the following.

# Voters	a score	b score	c score
3	10	5	0
3	4	10	0
1	10	5	0
score	52	50	0

Now, the final voter is making use of the full scoring range and exerting their full influence on the election. With the normalized scores, a receives the most points and is the only winner of the election.

This new behavior significantly alters the properties of the voting system. Notably, normalized range voting no longer possesses some of the nice properties of standard range voting. Namely, it no longer satisfies the criterion independence of irrelevant alternatives.

2.1.6 Schulze Voting

Schulze voting is a Condorcet voting system recently introduced by Marcus Schulze [Sch11]. It has a somewhat more complex winner procedure than other common voting systems, requiring the use of a graph best-path finding algorithm, but it is still solvable in polynomial time, rendering Schulze a tractable voting system.

Schulze voting is currently used by a number of organizations for their internal elections, including the Wikimedia foundation, several branches of the Pirate Party, a civil-liberties focused political party, and by MTV to rank music videos [Wik12]. This level of real-world usage is unusual for a new, academically proposed and studied voting system, but it makes Schulze voting more compelling to analyze.

The winners in a Schulze election are determined as follows. Generate the weighted majority graph (WMG) for the election. Determine the strongest paths in this graph between every pair of candidates by the following metric: The weight of a path is the lowest weight edge in the path. This is the "bottleneck" metric. We can adapt the Floyd-Warshall dynamic programming algorithm to find the weight of such paths in polynomial time [Sch11],

as shown in Algorithm 1.

Algorithm 1 Procedure to calculate Schulze best-path scores, adapted from the Floyd-Warshall algorithm.

```
Start
initialize paths to WMG edge weights
for k = 1 to m do
  for i = 1 to m do
    for j = 1 to m do
      if i == j then
        next
      end if
      newpath ← min(paths[i][k], paths[k][j])
      paths[i][j] ← max(newpath, paths[i][j])
    end for
  end for
end for
return paths
End
```

Once we have the best path weights, we build another graph with the same vertices where there is a directed edge from a to b if the best path from a to b is better than the best path from b to a. The winners of the election are the candidates with no in-edges in this final graph, and there will always be at least one winner.

Note that a Condorcet winner will have positive net advantage edges to every other candidate, and so they will easily have the better paths to every other candidate and be the only winner of the election.

Example Election

Consider a Schulze election over candidates $\{a, b, c, d\}$ with the following table expressing the net advantage function.

	a	b	c	d
a		4	-2	0
b	-4		4	-2
c	2	-4		6
d	0	2	-6	

We can also represent this election as a graph as in Figure 2.1.

Now we must find the best paths between pairs of candidates. Note that there are several Condorcet cycles and there is no clearly dominant candidate: Each of them lose to at least one other candidate, and each candidate has a path to every other candidate. The following

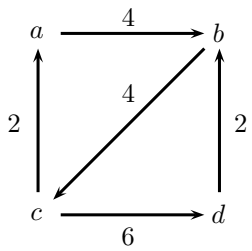

Figure 2.1: WMG for the example Schulze election. For simplicity we show edges in only one direction, and tied contests are represented with missing edges.

are the Schulze best path scores.

	a	b	c	d
a		4	4	4
b	2		4	4
c	2	2		6
d	2	2	2	

The only winner of the election will be a, as it has a stronger path to every other candidate than those candidates have back to a.

2.1.7 Ranked-Pairs Voting

Ranked pairs is a voting system that was introduced by Nicolaus Tideman in 1987 [Tid87], designed in part to be resistant to candidate cloning: In many voting systems, the inclusion of several similar candidates ends up diluting their support and lessening the influence of their supporters, and this system is less effected by these situations.

Ranked pairs is a Condorcet system with an unusually procedural winner determination method. It works by creating a directed acyclic graph derived from the WMG for an election and then selects the unbeaten candidate in this resulting graph as the winner. The algorithm requires us to iterate through all the pairs of candidates in a procedure highly sensitive to the exact ordering of the pairs, and so to rigorously define the system, we need a tie-breaking ordering over all the unordered pairs of candidates, and also an ordering over the candidates themselves.

The winners are determined as follows. Rank all the pairs of candidates by decreasing WMG edge weight (considering nonzero weight edges in the direction that they are of positive weight). Break ties between pairs of edges that have the same weight using the given ordering of pairs of candidates. Now we proceed down the list of pairs, considering them one at a time. We lock in a given edge if it will not create any cycles when added to

the edges we have already locked in. Edges that will create cycles are discarded. If a pair of candidates tie and have zero-weight edges between them, we determine the direction to set the edge in using the tiebreaking order over candidates. After considering all of the pairs, there will be a single source vertex with no in-edges, and this candidate will be the winner. This version of the election system will always select a single winner.

Example Election

We will give an example using the same election as for Schulze, with the WMG as given in Figure 2.1. We will use the $a > b > c > d$ as our tiebreaking ordering over candidates, and sort pairs first by the greater candidate in the pair (according to our candidate ordering) and next by the lesser candidate. Thus our overall schedule of pairs to consider is the following: $(c, d), (a, b), (b, c), (a, c), (b, d), (a, d)$. These pairs have edges with weights of $6, 4, 4, 2, 2, 0$, and (a, b) is ranked before (b, c) and (a, c) is ranked before (b, d) due to the tiebreaking ordering.

The first three pairs are locked in without incident, but the pairs (a, c) and (b, d) are discarded to avoid cycles. Finally we consider the tied pair (a, d), and a wins due to the candidate ordering. In the final graph, given in Figure 2.2, a is the only unbeaten candidate, and thus is the only winner.

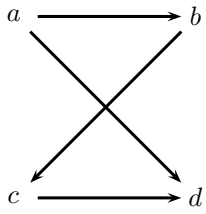

Figure 2.2: Final graph of locked-in edges for the example ranked-pairs election.

2.2 Computational Social Choice

The field of computational social choice analyzes computational issues relating to elections and other mechanisms of social choice. Traditional social choice theory comes from economics and political science and does not generally anticipate or investigate computational issues with the mechanisms it develops.

However, issues of computational feasibility can hamper the usefulness of otherwise attractive voting systems. For instance, determining who is a winner in Dodgson elections (a voting system that selects a candidate that is closest to being a Condorcet winner)

or in Kemeny elections (a voting system that produces an overall preference order of the candidates with minimized difference between this ordering and the voter preferences) is NP-hard [BTT89b].

Complexity can also protect voting systems from actions that subvert their democratic intent. We know from the Gibbard-Satterthwaite Theorem that all reasonable voting systems are susceptible to manipulation, but in some voting systems it is computationally infeasible for voters to find a way to successfully vote strategically. Thus one major track of research in computational social choice involves analyzing the complexity of various manipulative action problems that model ways election participants or organizers can subvert the election to change the result.

The primary classes of these problems are *manipulation*, where a voter or voters strategically vote to affect the result of an election [BTT89a], *bribery*, where an outside briber pays off voters to change their votes [FHH09], and *control*, where an election organizer alters the structure of an election to change the result [BTT92].

The original versions of these problems are the *constructive* cases, where the goal is to make a particular candidate win. The alternative class of problems are the *destructive* cases, where the goal is to make a particular candidate not win. These were introduced by Conitzer, Sandholm, and Lang [CSL07] in the context of manipulation and later by Hemaspaandra, Hemaspaandra, and Rothe in the context of control [HHR07]. Though the constructive goal may be more desirable, the destructive version of a problem may be easy where the constructive version is not, so investigating both is useful.

There are a few standard terms for the complexity of a given manipulative action problem in an election system.

Susceptibility A voting system is *susceptible* to a case of control if that action has potential to affect the result of an election. That is, for a constructive case of control, there is at least one positive instance where the distinguished candidate was not originally a winner of the election, or for a destructive case of control, there is at least one positive instance where the distinguished candidate was originally a winner.

Vulnerability A voting system is *vulnerable* to a case of control if it is susceptible to that case, and the associated decision problem can be solved in polynomial time (that is, the associated decision problem is in P). A voting system is vulnerable to a case of bribery or manipulation if the associated decision problem can be solved in polynomial time. This has a very good practical correspondence with real world efficiency of the problem, and thus the manipulative action is computationally easy.

Certifiable Vulnerability A voting system is *certifiably vulnerable* to a case of control if it is susceptible to that case, and the associated *search* problem can be solved in polynomial time. A voting system is certifiably vulnerable to a case of bribery or manipulation if the associated search problem can be solved in polynomial time. That is, we can find the specific action (e.g., the manipulative vote, the set of voters to add, the partition of the candidates) that will successfully carry out the manipulative action. The original formulation of this notion required that the algorithm find an "optimal" action [HHR07], but subsequent work (for instance, [FHH11b, FHHR11, HHM12]) has used the broader meaning of an algorithm providing any successful action.

Resistance A voting system is *resistant* to a case of control if it is susceptible to that case, and the associated decision problem is NP-hard. A voting system is resistant to a case of bribery or manipulation if the associated decision problem is NP-hard. The idea of NP-hardness has a long and storied history, but for the current purposes it suffices to say that such problems are very unlikely to have efficient general solutions barring a major shift in our understanding of computer science.

Immunity A voting system is *immune* to a case of control if that action cannot affect the result of the election. This is obviously a desirable notion but it is generally harder to come by than resistance.

Manipulation

The manipulation problem is the most basic of the election manipulative action problems. The manipulation problem models strategic voting, where a voter or set of voters attempt to vote in some way (not necessarily reflecting their true preferences) to sway the result of the election and cause their favorite candidate to win, or else to cause some hated candidate to lose. The computational complexity of this problem was first studied by Bartholdi, Tovey, and Trick [BTT89a]. We will now formally define the manipulation problem.

Manipulation

Given An election (C, V), a set of manipulators M, and a distinguished candidate p.

Question (Constructive) Is there a way to assign the votes of M such that p is a winner of the election $(C, V \cup M)$?

Question (Destructive) Is there a way to assign the votes of M such that p is not a winner of the election $(C, V \cup M)$?

There are other common variants of this problem as well, for instance the case where each of the voters and manipulators has a weight that determines their level of influence in the election.

Bribery

Election bribery, naturally, is the problem of a briber paying off voters to change their votes in order to change the result of an election. This problem was first studied by Faliszewski, Hemaspaandra, and Hemaspaandra [FHH09]. Bribery can be thought of as a more complex form of manipulation where the briber first has to select a set of voters to bribe and then assign their votes to achieve a desired result. Despite this natural characterization, there do exist voting systems where the bribery problem is easy (in P) while manipulation is NP-hard [FHH09]. We will now formally define the bribery problem.

Bribery

Given An election (C, V), a distinguished candidate $p \in C$, and an integer bribing limit $k \geq 0$.

Question (Constructive) Is there a way to change the preferences of no more than k voters in V such that p is a winner of the election with those votes thusly changed?

Question (Destructive) Is there a way to change the preferences of no more than k voters in V such that p is not a winner of the election with those votes thusly changed?

As with manipulation there is a weighted variant, and there is additionally a priced variant where voters have individual costs to be bribed rather than all having unit cost [FHH09], or even where some alterations to a vote are more expensive to make than others [Fal08].

Control

Control encompasses actions taken by an election chair to change the structure of an election to achieve a desired result. This can include adding or deleting candidates or voters, or partitioning either candidates or voters and performing initial subelections. The study of election control was initiated by Bartholdi, Tovey, and Trick [BTT92], and many subsequent papers have investigated the complexity of control in various voting systems [FHHR09, HHR09, ENR09, EFPR10].

The various control problems loosely model many real-world actions. The cases of adding and deleting voters correspond to voter registration drives and voter suppression efforts. The cases of adding and deleting candidates correspond to ballot-access procedures that effectively remove many candidates from elections. Cases of partitioning voters are similar to the

real-world practice of gerrymandering (though that has additional geographic constraints), and cases of partitioning candidates correspond to primary elections or runoffs.

The following definitions follow the nonunique winner model, where we claim success in our control action when the preferred candidate is one of possibly several winners in the election in the constructive case, or is not among the set of winners in the election in the destructive case. In contrast, the original paper exploring control [BTT92] used the unique winner model, where the goal is to cause a candidate to be the only winner of an election. We include both constructive and destructive variations, following [HHR07], and define the cases of adding candidates with an adding limit, symmetrically with the adding voters cases, and also include the original version of the adding candidates problem as *control by unlimited adding of candidates*, following [FHHR07], among others.

The unique winner versions of the problems can be easily obtained by changing the words "a winner" to "a unique winner" in each definition.

There is an additional concern in the various control by partitioning cases, where we may handle ties in the initial elections in different ways. In the ties-promote model, every candidate that is a winner in an initial election is promoted to the final election, while in the ties-eliminate model only a unique winner will be promoted to the final election, so no candidates will be promoted from a subelection where there is a tie for winner. We will now formally define the various cases of control.

Control by Adding Candidates

Given Disjoint candidate sets C and D, a voter set V with preferences over $C \cup D$, a distinguished candidate $p \in C$, and $k \in \mathbb{N}$.

Question (Constructive) Is it possible to make p a winner of an election $(C \cup D', V)$ with some $D' \subseteq D$ where $\|D'\| \leq k$?

Question (Destructive) Is it possible to make p not a winner of an election $(C \cup D', V)$ with some $D' \subseteq D$ where $\|D'\| \leq k$?

Control by Adding an Unlimited Number of Candidates

Given Disjoint candidate sets C and D, a voter set V with preferences over $C \cup D$, and a distinguished candidate $p \in C$.

Question (Constructive) Is it possible to make p a winner of an election $(C \cup D', V)$ with some $D' \subseteq D$?

Question (Destructive) Is it possible to make p not a winner of an election $(C \cup D', V)$ with some $D' \subseteq D$?

Control by Deleting Candidates

Given An election $E = (C, V)$, a distinguished candidate $p \in C$, and $k \in \mathbb{N}$.

Question (Constructive) Is it possible to make p a winner of an election $(C - C', V)$ with some $C' \subseteq C$ where $\|C'\| \leq k$?

Question (Destructive) Is it possible to make p not a winner of an election $(C - C', V)$ with some $C' \subseteq (C - \{p\})$ where $\|C'\| \leq k$?

Control by Adding Voters

Given A candidate set C, disjoint voter sets V and W, a distinguished candidate $p \in C$, and $k \in \mathbb{N}$.

Question (Constructive) Is it possible to make p a winner of an election $(C, V \cup W')$ for some $W' \subseteq W$ where $\|W'\| \leq k$?

Question (Destructive) Is it possible to make p not a winner of an election $(C, V \cup W')$ for some $W' \subseteq W$ where $\|W'\| \leq k$?

Control by Deleting Voters

Given An election $E = (C, V)$, a distinguished candidate $p \in C$, and $k \in \mathbb{N}$.

Question (Constructive) Is it possible to make p a winner of an election $(C, V - V')$ for some $V' \subseteq V$ where $\|V'\| \leq k$?

Question (Destructive) Is it possible to make p not a winner of an election $(C, V - V')$ for some $V' \subseteq V$ where $\|V'\| \leq k$?

Control by Partition of Candidates

Given An election $E = (C, V)$ and a distinguished candidate $p \in C$.

Question (Constructive) Is there a partition (C_1, C_2) of C such that p is a final winner of the election $(D \cup C_2, V)$, where D is the set of candidates surviving the initial subelection (C_1, V)?

Question (Destructive) Is there a partition C_1, C_2 of C such that p is not a final winner of the election $(D \cup C_2, V)$, where D is the set of candidates surviving the subelection (C_1, V)?

Control by Runoff Partition of Candidates

Given An election $E = (C, V)$ and a distinguished candidate $p \in C$.

Question (Constructive) Is there a partition C_1, C_2 of C such that p is a final winner of the election $(D_1 \cup D_2, V)$, where D_1 and D_2 are the sets of surviving candidates from the subelections (C_1, V) and (C_2, V)?

Question (Destructive) Is there a partition C_1, C_2 of C such that p is not a final winner of the election $(D_1 \cup D_2, V)$, where D_1 and D_2 are the sets of surviving candidates from the subelections (C_1, V) and (C_2, V)?

Control by Partition of Voters

Given An election $E = (C, V)$ and a distinguished candidate $p \in C$.

Question (Constructive) Is there a partition V_1, V_2 of V such that p is a final winner of the election $(D_1 \cup D_2, V)$ where D_1 and D_2 are the sets of surviving candidates from the subelections (C, V_1) and (C, V_2)?

Question (Destructive) Is there a partition V_1, V_2 of V such that p is not a final winner of the election $(D_1 \cup D_2, V)$ where D_1 and D_2 are the sets of surviving candidates from the subelections (C, V_1) and (C, V_2)?

2.3 Parameterized Complexity

Parameterized complexity is a field that takes a more fine-grained approach to complexity analysis. Rather than just analyzing the running time of a problem in terms of the size of the input, in parameterized complexity it is also mapped in terms of some other parameter to the problem. There are many options as to what can serve as a parameter, and the most natural parameters vary based on the problem domain. In the case of problems relating to voting, we feel the most natural parameter is the number of candidates in the input election.

We will describe a few of the key parameterized complexity classes.

Fixed Parameter Tractable A parameterized problem is fixed-parameter tractable (is in FPT) if there is an algorithm for that problem that runs in time $f(j) \cdot n^{O(1)}$, where n is the size of the input and j is the parameter [Nie06]. That is, for every fixed value for the parameter, the algorithm runs in uniform polynomial time, though the function f of the parameter can grow arbitrarily. This class is in effect the parameterized analog to P, and problems in this class are likely to be fairly efficiently solvable for small parameter values.

W Hierarchy The W hierarchy is a hierarchy of parameterized complexity classes based on a complex set of weighted circuit satisfiability problems [DF99]. The significance of these classes is that the first few levels, W[1] and W[2], contain natural hard and complete problems, and there are complexity-theoretic consequences if these classes collapse to FPT. Thus we can show a problem is unlikely to be in FPT by showing it is W[1] or W[2]-hard with a parameterized reduction from a known hard problem from one of these classes, such as dominating set [DF95]. A parameterized reduction is similar to a standard polynomial-time many-one reduction, except that the running time is not required to be polynomially bounded in terms of the parameter value, and the parameter in the output must be bounded by a function of the input parameter [DF99]. These classes are thus in a sense the parameterized analogs to NP: Hard problems for these classes are very unlikely to be in FPT.

Chapter 3

Control in Schulze Voting

3.1 Introduction

Here we study Schulze voting, a new and sophisticated voting system that has become fairly popular in recent years. We investigate its complexity under control, a class of manipulative action problems where the election chair changes the structure of the election to affect the result. We build on earlier work by Parkes and Xia [PX12] and further characterize the worst-case behavior of Schulze voting under control, resolving all but three control cases. We find that it possesses a good but not exceptional number of control resistances, fewer than the best known natural systems. Schulze voting currently is known to possess the same control resistances as Copeland$^{\alpha}$ with $\alpha \in (0, 1)$, which was at one point the most resistant system known that uses the standard linear vote model [FHHR09]. While it fails to stand among the most resistant systems, the popularity of the system increases the relevance of these results, perhaps calling into question the use of the system in contexts where these control actions are of concern.

3.2 Schulze Voting

Schulze voting is a Condorcet voting system recently introduced by Marcus Schulze [Sch11]. It was designed in part to effectively handle candidate *cloning*: In many voting systems, the inclusion of several similar candidates ends up diluting their support and lessening the influence of their supporters. It has a somewhat more complex winner procedure than other common voting systems, requiring the use of a graph best-path finding algorithm, but it is still solvable in polynomial time, rendering Schulze a tractable voting system.

Schulze voting has garnered a high level of real-world use, especially in the free software

and free culture communities. Users include several national branches of the Pirate Party, the Wikimedia foundation, many free-software projects such as Ubuntu, Debian, Gentoo, and KDE, and even MTV. This level of real-world usage is unusual for a new, academically proposed and studied voting system, but it makes Schulze voting more compelling to analyze. The winner determination method for Schulze voting is presented in Section 2.1.6.

It is obviously easily possible to construct the net advantage function given a collection of votes, but it is interesting to note that we can easily convert in the other direction as well. Given a net advantage function where either all the scores are even or all the scores are odd, we can construct an equivalent set of votes in time polynomial in the number of candidates and in the maximum magnitude of any score in the net advantage function. This method is due to McGarvey [McG53]. We will use this method later to more easily construct elections in the course of reductions, in order to avoid having to explicitly specify the votes.

3.3 Results

Parkes and Xia studied the complexity of the manipulative action problems for Schulze voting [PX12]. They proved resistance for Schulze voting for bribery and for some of the control cases, and showed that constructive manipulation for Schulze is easy with a single manipulator. Recently, Gaspers, Kalinowski, Narodytska, and Walsh showed that coalitional manipulation is easy as well [GKNW13]. However several control cases remained open. Our new results are the following.

- Schulze voting is resistant to constructive control by adding an unlimited number of candidates.
- Schulze voting is resistant to constructive control by deleting candidates.
- Schulze voting is resistant to constructive control by partition/runoff partition of candidates, ties promote or ties eliminate.
- Schulze voting is vulnerable to destructive control by partition/runoff partition of candidates, ties promote or ties eliminate.
- Schulze voting is resistant to constructive or destructive control by partition of voters, ties promote or ties eliminate.

Table 3.1 shows the behavior of Schulze voting, as well as several other voting systems under control. We include here proofs of a selection of our results.

Theorem 3.3.1. *Schulze voting is resistant to constructive control by unlimited adding of candidates.*

Control by	Tie Model	Plurality		Approval		Fallback		Schulze	
		C	D	C	D	C	D	C	D
Adding Candidates		R	R	I	V	R	R	R	S
Adding Candidates (unlimited)		R	R	I	V	R	R	**R**	S
Deleting Candidates		R	R	V	I	R	R	**R**	S
Partition of Candidates	TE	R	R	V	I	R	R	**R**	**V**
	TP	R	R	I	I	R	R	**R**	**V**
Run-off Partition of Candidates	TE	R	R	V	I	R	R	**R**	**V**
	TP	R	R	I	I	R	R	**R**	**V**
Adding Voters		V	V	R	V	R	V	R	R
Deleting Voters		V	V	R	V	R	V	R	R
Partition of Voters	TE	V	V	R	V	R	R	**R**	**R**
	TP	R	R	R	V	R	R	**R**	**R**

Table 3.1: Control behavior under Schulze voting and other voting systems for comparison. V, R, S, and I stand for vulnerable, resistant, susceptible, and immune. Results proved in this work in bold, other results from [ER10, EF10b, EPR11, HHR07, PX12].

Proof. We prove this case through a reduction from 3SAT, defined as follows [GJ79].

Given A set U of variables and a collection Cl of clauses over U such that each clause $c \in Cl$ has $|c| = 3$.

Question Is there a satisfying truth assignment for Cl?

Given a 3SAT instance (U, Cl), we will construct a control instance (C, D, p, k) as follows. The original candidate set C will be the following:

- The distinguished candidate p.
- An additional special candidate a.
- A candidate for each clause c_i.
- For each variable x_i, a candidate x_i'.
- For each variable x_i, a candidate x_i''.

The auxiliary candidate set D will consist of the following:

- The "literal" candidates: For each variable x_i, candidates x_i^+ and x_i^-.

The voter set V will be constructed such that we have the following relationships between the candidates (both those in C and in D):

- For every clause c_i, candidate c_i beats p by 2 votes.

- For every variable x_i, candidate x_i' beats p by 2 votes.
- For every variable x_i, p beats candidate x_i'' by 2 votes.
- For every variable x_i, candidates x_i^+ and x_i^- beat x_i' by 4 votes.
- For every variable x_i, x_i'' beats x_i^-, x_i^- beats x_i^+, and x_i^+ beats p, all by four votes.
- For every variable x_i, candidates x_i^+ beats every clause candidate c_i where the corresponding clause is satisfied by x_i assigned to true, by 4 votes.
- For every variable x_i, candidates x_i^- beats every clause candidate c where the corresponding clause is satisfied by x_i assigned to false, by 4 votes.
- p beats a by 4 votes.
- For every variable x_i, a beats candidates x_i^+ and x_i^- by 4 votes.

We will show that p can be made a winner through adding candidates if and only if there is a satisfying assignment for the 3SAT instance.

Given a satisfying assignment for the 3SAT instance, build the corresponding set of added candidates, where x_i^+ is included if x_i is set to true, and x_i^- is included otherwise. We can show that p will be a winner with this set of candidates added. Since these candidates correspond to a satisfying assignment, and every literal candidate x_i^+ or x_i^- beats by 4 every clause candidate they satisfy, and since p has a strong path to each literal candidate (through a), there will be a strong path from p to each clause candidate c_i, stronger than the strength-of-2 edge from c_i to p. Since we have added exactly one of x_i^+, x_i^- for each x_i, we create a strong path from p to the x_i' candidate, and we prevent there from being a strong path from x_i'' to p, as such a path would have to go through both of those candidates. Thus no candidate will have a better path to p than p has to that candidate, so p will be a winner.

We will show that if we map to a positive control instance, we must have mapped from a satisfiable 3SAT instance. To make p a winner of the election, we must ensure p's paths to other candidates are at least as strong as other candidates' paths to p. p immediately has strong paths to any of the added literal candidates. To beat each of the clause candidates we must add literal candidates that give p a path to each of them. To beat the x_i' candidates we must add at least one of x_i^+ or x_i^- for each x_i. To beat the x_i'' candidates we must not add both of x_i^+ and x_i^- for each x_i. Thus to have a successful control instance, we must set every variable to one of true or false in such a way that satisfies each clause. Thus if we have a positive control instance we must have a positive 3SAT instance. □

Theorem 3.3.2. *Schulze voting is resistant to constructive control by deleting candidates.*

Proof. We prove this case through a reduction from 3SAT. Given a 3SAT instance (U, Cl) we will construct a control instance $((C, V), p, k)$ as follows. Our deletion limit k will be equal to $||Cl||$. The candidate set will be the following:

- Our special distinguished candidate p.
- For each clause $c_i \in Cl$, $k+1$ candidates $c_i^1, \ldots, c_i^{k+1}$.
- A candidate for each literal x_i^j in each of the clauses (where x_i^j is the jth literal in the ith clause).
- For each pair of literals x_i^j, x_m^n where one is the negation of the other, $k+1$ candidates $n_{i,j,m,n}^1, \ldots, n_{i,j,m,n}^{k+1}$.
- An additional auxiliary candidate a.

The $k+1$ candidates for each clause and for each conflicting pair of literals are treated as copies of each other, and are included to prevent successful control by simply deleting the tough opponents rather than solving the tricky problem corresponding to the 3SAT instance.

The voters will be constructed according to the McGarvey method [McG53] such that we have the following relationships between candidates.

- For the three literal candidates x_i^1, x_i^2, x_i^3 in a clause c_i, c_i^j beats x_i^1 (for all j), x_i^1 beats x_i^2, x_i^2 beats x_i^3, and x_i^3 beats p, all by two votes.
- For a pair of literal candidates x_i^j and x_m^n that are negations of each other, each beats $n_{i,j,m,n}^l$ (for all l) by two votes.
- Every negation candidate $n_{i,j,m,n}^l$ beats p by two votes.
- p beats a by two votes.
- a beats every x_i^j by two votes.

This completes the specification of the reduction, which clearly can be performed in polynomial time. The intuition is as follows: Deleting a literal corresponds to assigning that literal to be true, and to make p win we must "assign" literals that satisfy every clause without ever "assigning" a variable to be both true and false, so a successful deletion corresponds to a valid satisfying assignment.

We will now show that if (U, Cl) is a positive 3SAT instance, p can be made a winner of this election by deleting k candidates. We will delete one literal candidate for each clause, selecting a literal that is satisfied by the satisfying assignment for Cl. This will require us to

delete $||Cl||$ candidates, which is equal to our deletion limit k. By deleting these candidates, we break all the paths from the clause candidates to p, so now p is tied with each clause candidate instead of being beaten by them. Also, since we deleted literals that were satisfied according to a satisfying assignment, we must not have deleted any pair of literals that were the negations of each other, so we will still have a path from p to each negation candidate. Thus p at least ties every other candidate in Schulze score, so they will be a winner.

If we map to a positive control instance, our 3SAT instance must be positive as well. First, the most serious rivals to p are the clause candidates, who each have a path of strength two to p while p only has a path of strength 0 to them. Since there are many duplicates of each of them, we cannot remove them directly but must instead remove other vertices along the paths to p to remove the threat. The deletion limit allows us to delete one candidate for every clause. However we must choose which ones to delete carefully, as if we delete a literal and a different one that is the negation of the first, we destroy our only paths to the corresponding negation candidate. Thus we must delete one literal for each clause, while avoiding deleting a variable and its negation. If there is a successful way to do this, there will be a satisfying assignment for the input instance that we can generate using our selection of deleted/satisfied literals (arbitrarily assigning variables that were not covered in this selection). □

Theorem 3.3.3. *Schulze voting is resistant to constructive control by partition and runoff partition of candidates, ties eliminate.*

Proof. We will reduce from 3SAT. Given a 3SAT instance (Cl, U), we construct a control instance $((C, V), p)$. The candidate set C will consist of the following.

- The distinguished candidate p.
- For every variable $x_i \in U$, candidates x_i^+, x_i^-, and x_i'.
- For every clause $c_i \in Cl$, a candidate c_i.
- Adversary candidates c' and u'.
- Helper candidates a and a'.

The relationships between candidates will be as follows.

- Every clause candidate c beats p by 4 votes.
- Every clause candidate c beats a by 4 votes.
- c' beats p by 2 votes.

- c' beats a by 4 votes.
- u' beats p by 4 votes.
- a beats a' by 4 votes.
- a' beats every x_i^+ and x_i^- by 4 votes.
- p beats every x_i^+ and x_i^- by 2 votes.
- c' beats u' by 2 votes.
- c' beats every c_i by 2 votes.
- u' beats every x_i' by 4 votes.
- For every x_i, x_i' beats x_i^-, x_i^- beats x_i^+, and x_i^+ beats a, all by 4 votes.
- a beats u' by 2 votes.
- For every variable x_i, candidates x_i^+ beats every clause candidate c_i where the corresponding clause is satisfied by x_i assigned to true, by 4 votes.
- For every variable x_i, candidates x_i^- beats every clause candidate c where the corresponding clause is satisfied by x_i assigned to false, by 4 votes.

We will show that p can be made a winner of (C, V) through control by partition (or runoff partition) of candidates if and only if (U, Cl) is a positive instance of 3SAT.

Assume that (U, Cl) is a satisfiable 3SAT instance. Then consider the partition of the candidates with the first partition (the one that undergoes the initial election in the non-runoff case) containing the x_i^+ and x_i^- candidates corresponding to the satisfying assignment, and also all other candidates except p. We will show that no candidates will be promoted from this first subelection, and p will win the final election. Since we have x_i^+ and x_i^- candidates corresponding to a satisfying assignment, there is a strong path from a to every clause candidate c_i. One of each of x_i^+ and x_i^- is not present for every x_i, breaking potential paths from x_i' candidates to a. The overall effect is that a will be a winner, along with c' and a number of other candidates, and since we are in the ties-eliminate model, no candidates will be promoted from this subelection. Depending on whether we are in the partition or runoff partition case, p either faces the remaining candidates (the unchosen variable assignment candidates) in an initial subelection or in the final election, and they will win.

Assume that we map to an accepting instance of the control problem. We will show that there must be a satisfying assignment for the 3SAT instance. p only beats the variable assignment candidates and a, and only narrowly. p can't survive an election that contains

any of the other candidates, so we must eliminate them in an initial subelection. c' has a path to every candidate of strength 2, and no other candidate has a path to them of greater than strength 2, so they will invariably be a winner of any subelection of which they are a part. So the only way to eliminate them is to force a tie in an initial subelection. a has potential to tie c', but to do so we have to give them a strong path to every clause candidate, and ensure there are no paths from any of the x' candidates. Thus we need to include variable assignment candidates satisfying each clause, but not include both the x^+ and x^- candidates for any x. So if control is possible, there must be a valid satisfying assignment for the 3SAT instance. □

Theorem 3.3.4. *Schulze voting is resistant to constructive control by partition and runoff partition of candidates, ties promote.*

Proof. In the ties-promote case we can use a similar reduction to the previous case. In this version, we alter the candidate set and net advantage scores so that the helper candidate a can be made a unique winner of their subelection if and only if there is a satisfying assignment, and p can only win the final election if a is a unique winner of a subelection.

The candidate set is as follows. Compared to the previous case, we have eliminated the adversary candidates c' and u'. They were important previously to ensure that there was a unique winner other than a in any subelection not corresponding to a satisfying assignment, but in this case they are not necessary.

- The distinguished candidate p.
- For every variable $x_i \in U$, candidates x_i^+, x_i^-, and x_i'.
- For every clause $c_i \in Cl$, a candidate c_i.
- Helper candidates a and a'.

The new scores will be as follows.

- p beats a by 2 votes.
- Every clause candidate c beats p by 4 votes.
- Every clause candidate c beats a by 4 votes.
- a beats a' by 6 votes.
- a' beats every x_i^+ and x_i^- by 6 votes.
- a' beats every x_i' by 2 votes.

- p beats every x_i^+ and x_i^- by 2 votes.
- For every x_i, x_i' beats x_i^-, x_i^- beats x_i^+, and x_i^+ beats a, all by 4 votes.
- For every variable x_i, candidates x_i^+ beats every clause candidate c_i where the corresponding clause is satisfied by x_i assigned to true, by 6 votes.
- For every variable x_i, candidates x_i^- beats every clause candidate c where the corresponding clause is satisfied by x_i assigned to false, by 6 votes.

If we map from a positive 3SAT instance we must have a positive control instance. The partition can be as follows: In the first partition include a, a', variable assignment candidates corresponding to a satisfying assignment, and every c_i and x' candidate. The other partition will contain just p and the remaining variable assignment candidates. In the first subelection, a will be the only winner, as they will have strong paths to all the clause candidates and no candidate has a strong path to them. p will win the final election with a and possibly also with some variable assignment candidates (in the nonrunoff case)

If we map to a positive control instance we must have a satisfiable 3SAT instance. Since we are working in the TP model, at least one candidate will survive each subelection. The only candidates that p beat are a and the variable assignment candidates (by only a narrow margin), so to win we must use a to eliminate any candidates that beat p in a subelection. Thus we have to make sure that a is a winner of their subelection, and no other candidate that beats p is a winner. For this to happen a must have a strong path to every clause candidate and none of the x' candidates can have strong paths back to a. We must include a valid satisfying assignment of variable assignment candidates for this to be the case, so we must have a satisfiable formula. □

Theorem 3.3.5. *Schulze voting is resistant to constructive control by partition of voters, ties eliminate.*

Proof. We will reduce from a 3SAT instance (U, Cl) and output a control instance $((C, V), p)$. The candidate set C will be as follows:

- A distinguished candidate p.
- A candidate c_i for every clause $c_i \in Cl$.
- A candidate x_i for every variable $x_i \in U$.
- An auxiliary candidate a.

The voter set V will be as follows.

- For each variable x_i, where D is the set of clauses satisfied by x_i assigned to true, a voter ranking p over c_i for $c_i \in D$ and c_j over p for $c_j \notin D$, ranking x_i over p, but ranking p over x_j for $x_j \in U - \{x_i\}$, and ranking p over a.
- For each variable x_i, where D is the set of clauses satisfied by x_i assigned to false, a voter ranking p over c_i for $c_i \in D$ and c_j over p for $c_j \notin D$, ranking x_i over p, but ranking p over x_j for $x_j \in U - \{x_i\}$, and ranking p over a.
- $||U|| - 3$ voters preferring p over every c_i, but preferring every x_i and a over p.
- 1 voter preferring p over every c_i and over a, but preferring every x_i over p.
- 2 voters preferring p over every c_i and x_i but with a preferred over p.

Unspecified portions of the votes (the relative rankings of the clause candidates and of the variable candidates) will be set according to the McGarvey method [McG53] to equalize as much as possible the pairwise contests between these candidates.

If we are mapping from a positive 3SAT instance, it will be possible to make p win the final election through this type of control. The partition will be as follows: The first partition will contain all of the third, fourth, and fifth groups of voters, and will contain voters from the first two groups corresponding to a satisfying assignment. The result of adding the voters from the third, fourth, and fifth groups will be to give p $||U||$ votes over each c_i, each x_i $||U|| - 4$ votes over p, and a $||U|| - 2$ votes over p. By adding the variable assignment voters corresponding to the satisfying assignment, we decrease every p-c_i edge by no more than $||U|| - 2$, increase every p-x_i edge by $||U|| - 2$, and increase the p-a edge by $||U||$. The result is that we boost p over a and over the variable candidates while preserving the paths from p to the clause candidates. Thus p will beat every candidate and win their subelection. In the other subelection, with the remaining variable assignment candidates, p will tie with some of the clause candidates (unless the inverse assignment is also satisfiable) and so no candidates will be promoted. p will thus be alone in the final election and will win.

If we map to a positive control instance, the mapped-from 3SAT instance will be satisfiable. No voter prefers p outright, so we must balance their votes against the other voters to have a chance. The voters in the first two groups prefer p over a and they mostly prefer p over the variable candidates, but they mostly prefer the clause candidates over p. The voters in the third, fourth, and fifth groups can boost p over the clause candidates, but they give a and the variable candidates an advantage over p. To make p the only winner of a subelection, we have to carefully select the voters from the first two groups to keep p ahead of both the variable and clause candidates. To do so we must ensure that at least one voter prefers p over each c_i, forcing the corresponding assignment to satisfy every clause, and

that only one voter prefers each x_i over p, forcing the corresponding assignment to assign each variable as either "true" or "false" but not both. These voters thus correspond to a satisfying assignment, so the 3SAT instance must be satisfiable. □

Theorem 3.3.6. *Schulze voting is resistant to constructive control by partition of voters, ties promote.*

Proof. This case can be shown through a reduction from 3SAT very similar to the previous case. We will map from a 3SAT instance (U, Cl) to a control instance $((C, V), p)$. We will have the following candidate set C:

- A distinguished candidate p.
- A candidate c_i for every clause $c_i \in Cl$.
- A candidate x_i for every variable $x \in U$.
- An auxiliary candidate a.

The voter set V will be the following:

- For each variable x_i, where D is the set of clauses satisfied by x_i assigned to true, a voter ranking p over c_i for $c_i \in D$ and c_j over p for $c_j \notin D$, ranking x_i over p, but ranking p over x_j for $x_j \in U - \{x_i\}$, and ranking p over a.
- For each variable x_i, where D is the set of clauses satisfied by x_i assigned to false, a voter ranking p over c_i for $c_i \in D$ and c_j over p for $c_j \notin D$, ranking x_i over p, but ranking p over x_j for $x_j \in U - \{x_i\}$, and ranking p over a.
- $||U|| - 1$ voters preferring p over every c_i, but preferring every x_i and a over p.
- 1 voter preferring p over every c_i but with a and every x_i preferred over p.
- $||U|| + 2$ voters preferring a over the variable assignment candidates over the clause candidates over p.

Relative to the last case we have just changed the later groups of voters and left the variable assignment voters the same. Now, with just the third and fourth groups of voters present a has $||U||$ votes over every c_i and $||U|| - 2$ votes over every x_i, and a has $||U|| - 2$ votes over p. This proof can proceed similarly to the previous case, but noting that now we are in the TP model, multiple candidates can be promoted to the final election, so we must ensure that no candidates that can beat p with the entire voter set will make it. Thus to make p a winner we must cause them to at least tie the x_i and a candidates, and cause

them to beat the c_i candidates in the first round. For this to happen we must include in a partition voters corresponding to a satisfying assignment, as well as the third and fourth groups of voters. In the other partition we must prevent any of the clause candidates from winning, and this will happen with the remaining variable assignment voters and the fifth group of voters: a will be the only winner. Thus p will end up being a winner of the final election.

If we have a positive control instance, p must be a winner of at least one initial subelection, and so this partition must have balanced the points p gets over the clause candidates from the third and fourth groups with the points p gets over the variable candidates and a from the first two groups, and the voters from the first two groups must have been carefully chosen so at least one of them has p over each clause candidate, and no more than one has each variable candidate over p. Thus there must be a satisfying assignment for the 3SAT instance. □

Theorem 3.3.7. *Schulze voting is resistant to destructive control by partition of voters, ties eliminate.*

Proof. We will reduce from a 3SAT instance (U, Cl) and construct a control instance $((C, V), p)$. The candidate set C will be as follows:

- The distinguished candidate p.
- The adversary candidates a and b.
- Candidates c_i and c_i' for each clause $c_i \in Cl$.
- Candidates x_i and x_i' for each variable $x_i \in U$.

The voter set V will be as follows:

- For each variable x_i, where D is the set of clauses satisfied by x_i assigned to true, a voter ranking c_i' over c_i for $c_i \in D$ and c_j over c_j' for $c_j \in Cl - D$, ranking x_i' over x_i, but ranking x_j over x_j' for $x_j \in U - \{x_i\}$, ranking p over a and b, the clause candidates over a and the variable candidates over b.
- For each variable x_i, where D is the set of clauses satisfied by x_i assigned to false, a voter ranking c_i' over c_i for $c_i \in D$ and c_j over c_j' for $c_j \in Cl - D$, ranking x_i' over x_i, but ranking x_j over x_j' for $x_j \in U - \{x\}$, ranking p over a and b, the clause candidates over a and the variable candidates over b.
- $2||U|| - 2$ voters ranking $a > p > b$, with the other candidates evenly distributed.
- $2||U|| - 2$ voters ranking $b > p > a$, with the other candidates evenly distributed.

The unspecified parts of the votes should be set to make the groups of candidates as even as possible, except for the c'_i candidates, among whom there should be strong paths from each candidate to each candidate.

If the mapped-from 3SAT instance is satisfiable, we will be able to defeat p through partition of voters. The partition we use will be the following. The first part will consist of voters from the first two groups corresponding to a satisfying assignment, and the voters from the third group. The second part will consist of the remaining voters from the first two groups, and the voters from the fourth group. a will defeat p in the first partition. By selecting the voters to correspond to a satisfying assignment we weaken all of the paths from p to a going through the clause candidates to be $||U|| - 2$ in strength. The third group of voters provide support for a over p, bringing the $a \to p$ edge to be $||U|| - 2$ in strength. Thus a ties p, and no candidate will be promoted from this subelection. In the other subelection, b will tie p for similar reasons, as they will have an edge of $||U|| - 2$ to p while that will also be the strength of the best path from p to b going through any of the variable candidates. Thus no candidate makes it to the final election and control is successful.

If the mapped-to control instance is positive, we must have a positive 3SAT instance. p will win the final election against any other candidate if they make it there, so to defeat them we must make them lose both initial subelections. No voters rank p below both a and b, the strongest other candidates, so we must defeat p with a in one subelection and defeat p with b in the other. To make b beat p, we have to include the fourth group of voters, the only voters that rank b over p. b would defeat p if we include just these voters, but then p would win the other subelection with the support of all the variable assignment voters. Thus we have to include half the variable assignment voters to give a a chance as well, and we must pick voters to weaken the paths from p to b. This will require us to pick only one voter corresponding to each variable. The other subelection will proceed similarly, but we will have to pick at least one voter satisfying each clause to weaken the p-a paths. Thus there must be a valid satisfying assignment for the 3SAT instance if control is possible. □

Theorem 3.3.8. *Schulze voting is resistant to destructive control by partition of voters, ties promote.*

Proof. This case can proceed very similarly to the previous case in the TE model, except that we just have to slightly change some of the numbers so that the adversary candidates each will fully defeat rather than tie with p in order to eliminate p from both initial subelections. The new voters will be as follows, and everything else can proceed in the same way. We will just give the new voter set specification here.

- For each variable x_i, where D is the set of clauses satisfied by x_i assigned to true, a voter ranking c'_i over c_i for $c_i \in D$ and c_j over c'_j for $c_j \in Cl - D$, ranking x'_i over x_i,

but ranking x_j over x'_j for $x_j \in U - \{x_i\}$, ranking p over a and b, the clause candidates over a and the variable candidates over b.

- For each variable x_i, where D is the set of clauses satisfied by x_i assigned to false, a voter ranking c'_i over c_i for $c_i \in D$ and c_j over c'_j for $c_j \in Cl - D$, ranking x'_i over x_i, but ranking x_j over x'_j for $x_j \in U - \{x_i\}$, ranking p over a and b, the clause candidates over a and the variable candidates over b.
- $2||U||$ voters ranking $a > p > b$, with the other candidates evenly distributed.
- $2||U||$ voters ranking $b > p > a$, with the other candidates evenly distributed.

□

Theorem 3.3.9. *Schulze voting is vulnerable to destructive control by partition of candidates and destructive control by runoff-partition of candidates in either the ties-promote or ties-eliminate model.*

Schulze voting is vulnerable to each of the variants of destructive control by partition of candidates. That is, to destructive control by partition of candidates, ties promote; destructive control by runoff partition of candidates, ties promote; destructive control by partition of candidates, ties eliminate; and destructive control by runoff partition of candidates, ties eliminate. We will show this, and additionally show that these problems are easy for a broad class of Condorcet voting systems.

In the nonunique-winner model that we are following here, both of the pairs destructive control by partition of candidates, ties eliminate and destructive control by runoff partition of candidates, ties eliminate; and destructive control by partition of candidates, ties promote and destructive control by runoff partition of candidates, ties promote are in fact the same problems, that is, the same sets. This fact was discovered by Hemaspaandra, Hemaspaandra, and Menton ([HHM12], also presented in Chapter 6 of this thesis) by noting shared alternative characterizations of these problems. The nominal difference is that in the "partition of candidates" case, the candidate set is partitioned and only one part undergoes a culling subelection while the other part gets a bye, while in the "runoff partition of candidates" case, both parts of the partition first face a subelection. In truth they are much simpler problems, and identical (within the same tie-handling model). Namely, the sets DC-PC-TE and DC-RPC-TE are equivalent to the set of instances $((C, V), p)$ where there is some set $C' \subseteq C$ with $p \in C'$ such that p is not a winner of the election (C', V). The sets DC-PC-TP and DC-RPC-TP are equivalent to the set of instances $((C, V), p)$ where there is some set $C' \subseteq C$ with $p \in C'$ such that p is not a unique winner of the election (C', V). We will build algorithms for these cases aided by these characterizations.

These algorithms are optimal in a class of voting systems that are Condorcet voting systems that also possess a weaker version of the Condorcet criteria—where if there are any candidates that do no worse than tying in pairwise contests with other candidates (known as weak condorcet winners), they will be winners (possibly, but not necessarily unique) of the election. There can potentially be one or more than one such candidates. Schulze voting is such a voting system: If a candidate is a weak Condorcet winner, doing no worse than tying against any other candidate, no candidate can have a path to that candidate of strength greater than 0, and since that candidate at least ties every other candidate, he or she has a path of strength at least 0 to every other candidate. Thus he or she is unbeaten in Schulze score, and will be a winner.

Other Condorcet voting systems that possess this property include Copeland[1], minimax, and ranked pairs.

First we will handle the DC-PC-TP/DC-RPC-TP case.

Proof. Recall our alternate characterization for these problems, that the set of positive instances is the same as the set of instances $((C, V), p)$ where there is some set $C' \subseteq C$ with $p \in C'$ such that p is not a winner of the election (C', V). Thus, finding if we have a positive instance is very similar to the problem of finding if we have a positive instance in the destructive control by deleting candidates problem, except that there is no longer a limit on the number of candidates we can delete. Thus we do not have to carefully limit the number of deleted candidates, but we can freely delete as many candidates as necessary to put p in a losing scenario.

Given a control instance $((C, V), p)$, all we must do is see if there is any candidate $a \in C$, $a \neq p$, such that $netadv(a, p) > 0$. If such a candidate exists, we let our first partition be $\{a, p\}$ and let the second be $C - \{a, p\}$. p will lose their initial election to a, and be eliminated from the election. If there is no such candidate, p is a weak Condorcet winner, and thus they will always be a winner of the election among any subset of the candidates, as they will be a weak Condorcet winner (at least, or possible a Condorcet winner) among any subset of the candidates. Thus our algorithm will indicate failure. We can perform this check through a simple examination of the net advantage scores which can be generated from an election in polynomial time, and so this algorithm runs in polynomial time. Thus Schulze voting (and any other system meeting the aforementioned criteria) is vulnerable and constructively vulnerable to DC-PC-TP/DC-RPC-TP. □

Now we will handle the DC-PC-TE/DC-RPC-TE case.

Proof. These cases have the alternate characterization that the set of positive instances is equivalent to the set of instances $((C, V), p)$ where there is some set $C' \subseteq C$ with $p \in C'$

such that p is not a unique winner of the election (C', V), that is, there are multiple winners, or no winners, or there is a single winner that is not p. This case thus differs only slightly from the DC-PC-TP/DC-RPC-TP case and we can create a very similar simple algorithm.

Given a control instance $((C, V), p)$, we can find a successful action if one exists by simply checking if there is any candidate $a \in C$, $a \neq p$, such that $netadv(a, p) \geq 0$. If so, we let our first partition be $\{a, p\}$ and let the second be $C - \{a, p\}$. This results in p either not being a winner of the subelection at all or being a winner along with a if the net advantage score is 0. Either way, p will not be promoted to the final election and thus they will not be a winner of the final election. If there is no such candidate, p is a Condorcet winner, and they will be a Condorcet winner among any subset of the candidates, so this type of control will never be possible, and our algorithm will indicate that. As before, generating the net advantage function is easily possible in polynomial time, and the simple check will only take polynomial time, so Schulze voting (and any other system meeting the aforementioned criteria) is vulnerable and constructively vulnerable to DC-PC-TE/DC-RPC-TE. □

For the final cases of control, we do not resolve their complexity, but we do reduce the problems to their core: They are no harder than a simpler problem called path-preserving vertex cut.

Proposition 3.3.10. *Each of destructive control by adding candidates, destructive control by unlimited adding of candidates, and destructive control by deleting candidates in Schulze voting polynomial-time Turing reduces to* path-preserving vertex cut*.*

Proof. We define path-preserving vertex cut as follows.

Given A directed graph $G = (V, E)$, distinct vertices $s, t \in V$, and a deleting limit $k \in \mathbb{N}$.

Question Is there a set of vertices V', $||V'|| \leq k$, such that the induced graph on G with V' removed contains a path from t to s but not any path from s to t?

We note that the standard vertex cut problem is well-known to be solvable in polynomial time, but to the best of our knowledge the complexity of this variant is unknown (other than that it is clearly in NP). Given this result, Schulze voting will be vulnerable to each of these control cases if this problem is found to be in P.

We will describe how these control cases can reduce to this problem, first handling the DC-DC case. Our input will be a DC-DC instance $((C, V), p, k)$. Since this destructive case of control, we only have to make the distinguished candidate p not a winner rather than making any particular candidate positively a winner. To do this we must alter the election so that some other candidate has a better path to p than p has to that candidate. Since this is a case of control by deleting candidates, we can only eliminate paths in the election

graph, not create them. So we have to try to succeed by breaking the paths from p to some other candidate.

We can first handle the cases where p is a Condorcet winner (control will be impossible) or where p is already beaten by some candidate (control is trivial). Otherwise, we will loop through the candidates and see if we can make any of them beat p.

For each candidate a, we will first find the subgraph containing all maximum-strength paths to p (using a modified Floyd-Warshall algorithm), and also the subgraph containing all paths from p to a at least as strong as the strongest a-p path. We will then take the union of these graphs, disregard the edge weights, and apply a subroutine for path-preserving vertex cut. This will tell us if there is any way to cut all of the strong paths from p to a while preserving one of the strongest ones from a to p. If we ever get a positive result from this call, we can indicate success. Now, it may be that we can succeed by preserving a path other than the strongest path from a to p. So if we failed, we repeat the process except considering paths from a to p and from p to a that are a little less strong, going down as far as the strength of the p-a edge (which is not cuttable). This loop is polynomially bounded in the number of edges, as there can only be as many path strengths as there are edges. We indicate failure if we never achieve success with any vertex or path strength.

The other cases can be handled similarly. The difference is that they are adding-candidates problems instead of deleting-candidates problems, but we can reduce them to modified deleting-candidates problems by including all of the addable vertices and then solving them as deleting-candidates problems where we can only delete the vertices in the added sets. Additionally, in the non-unlimited case, we would be restricted as to how many of the vertices are added, so we would have to only look at paths from a to p that use a limited number of the added vertices. □

3.4 Conclusions

We found that Schulze voting resists every case of constructive control and several cases of destructive control, but it is vulnerable to several cases as well. These results may be of interest to some of the system's many users, especially those concerned with candidate control. There are still several open questions relating the complexity of manipulative actions in Schulze voting. Our unresolved control cases are of course of interest, and they may be resolved through development of a polynomial-time algorithm for the path-preserving vertex cut problem.

Chapter 4

Parameterized Complexity in Schulze Voting and Ranked Pairs

4.1 Introduction

In this chapter we study the parameterized complexity of manipulative actions in Schulze and ranked-pairs voting. We find that every standard manipulative action in those voting systems, even those previously found to be NP-hard, is fixed-parameter tractable (FPT) when parameterized on the number of candidates. Parkes and Xia [PX12], followed by Menton and Singh ([MS13], also presented in Chapter 3 of this thesis) studied the classical worst case complexity of manipulative actions in these voting systems, finding many problems to be NP-hard. This work implies that such problems are not as hard as previously thought when we have a small number of candidates, and that the difficulty of these problems depends in an essential way on the number of candidates.

The structure of our proofs is that we define a structure that allows us to brute-force search over possible strategies of action, and then for every possible strategy, we embed the problem of trying to achieve it in an integer linear programming feasibility problem (ILPFP). We can then solve each of these problems using Lenstra's algorithm, which achieves polynomial-time performance with a fixed number of variables [Len83].

The end result is an algorithm that runs in polynomial time, with a fixed polynomial, for every fixed number of candidates, though with large and growing constants for increasing numbers of candidates. This approach may not construct practically efficient algorithms,

but it does achieve the goal of placing these problems in FPT. Our proofs of this type are given in Section 4.3.

Additionally, we push our approach further by handling weighted cases of manipulation and bribery in Section 4.5. And we show our results obtained through other techniques in Section 4.4.

4.2 Presentation of Key Idea

We will now describe the key idea of this work and the key tool with which we achieve our results. What we need is a structure that can express a claim about the winner set of a Schulze or ranked-pairs election, and the strategy that went towards achieving that winner set, but where the number of such structures is bounded in terms of the number of candidates. And additionally we must be able to embed such a structure and the task of trying to achieve it in an integer linear program with a fixed number of variables for a fixed number of candidates.

Both Schulze and ranked pairs are Condorcet voting systems where the winners are determined using information from the weighted majority graph. So naturally we will try to express who are the winners using information about the WMG. But several natural approaches will not work. Expressing a claim about the winner set using information about what candidate wins each pairwise election, as was done by Faliszewski, Hemaspaandra, Hemaspaandra, and Rothe in Copeland voting [FHHR09], is not workable as that information is not enough to determine the winners in Schulze or ranked-pairs elections. Expressing a claim about the winner set using information about the specific weight of each edge in the WMG is not workable, as even with a bounded number of candidates there are an unbounded number of possible edge weights, so the number of such structures would not be appropriately bounded.

Our structures fall somewhere in between these approaches. We express our claims using information about the relative strength of edges in the WMG. This gives us enough information to determine the winners in either Schulze or ranked-pairs elections, and also the number of structures we must create is bounded in both systems.

We will now say more about what actually goes into these structures, first in the case of Schulze (defined in Section 2.1.6). A Schulze Winner-Set Certification Framework (SWCF) is a structure that makes a claim about who the winners of a given election are, and also claims what specific path to victory those candidates took. Since the winners in a Schulze election are determined through analyzing paths in the WMG, an SWCF will make claims about some of those paths. Namely, for every candidate a in the winner set, an SWCF will describe a best path from a to every other candidate, and for every candidate b not in the

winner set, it will describe a path that causes some other candidate to strictly b. To build our algorithms we embed these SWCFs into ILPFPs that model the task of trying to achieve them. The SWCF gives a story of how some particular set of candidates could become the winners, and then the ILPFP is given the task of solving the manipulative action problem in a particular way in line with the SWCF.

This structure meets all our requirements. First, the information in this structure is enough to claim what candidates are the winners in the election. Second, the number of such structures is large but bounded with a fixed number of candidates, as the number of simple paths through the WMG is bounded in the number of candidates. And third, we can embed this structure in an ILPFP. The key observation that allows the embedding is that we can assert that a particular path is a strong path through a series of inequalities between the weights of edges. An ILPFP can be built in terms of *Bigger* and *StrictlyBigger* relations between pairs of edges, stating non-strict and strict inequalities between those edges. Namely, to claim that a particular path from a to b allows a to at least tie b (as must be the case for every b if a is a winner), we claim that every edge in that path is at least as strong as some edge in every path from b to a. To claim that a particular path from a to b allows a to strictly beat b (and thus b is not a winner), we claim that every edge in that path is strictly stronger than some edge in every path from b to a. Thus we can implement an SWCF in terms of strict and non-strict inequalities between edges, which can be handily implemented in a linear program. We will describe the details of this in the next section.

A Ranked-Pairs Winner-Set Certification Framework (RPWCF) is the analogous structure for ranked-pairs elections (defined in Section 2.1.7). Ranked-pairs elections, unlike Schulze elections, are highly sequential in nature. Such a system is less natural to embed in a system of constraints, but the great flexibility of linear programming and the amount of work we can do with a fixed number of candidates allows us to succeed.

The RPWCF will in effect "tell a story" about the evaluation of the procedural winner determination method. It will consist of the schedule of locked-in edges through the operation of the ranked-pairs election, both the order in which the edges are considered, and the direction in which they are set (or whether they are discarded due to transitivity). The total number of such schedules is large but again bounded in terms of the number of candidates, and through the locked-in edges we can determine what candidate is selected as the winner (note that in our formulation of ranked pairs there is always exactly one winner). To build these structures we need both our input election as well as the ranked pairs tiebreaking orderings over the candidates and over the pairs of candidates (described in Section 2.1.7)

Like with Schulze, we can implement an ILPFP enforcing an RPWCF in terms of inequalities between edges. The RPWCF makes claims about the order edges are considered in and how they are set, and we can enforce these claims with inequalities. Namely, if an

edge (a, b) is considered at some point in time, it must be at least as strong as all other edges that follow it in the ordering, and it must be strictly stronger than edges that follow it that are ahead of (a, b) in the edge tiebreaking ordering. So like with Schulze we can enforce these constraints in terms of *Bigger* and *StrictlyBigger* terms. And then if the RP-WCF claims that the edge is set in the direction $a \to b$, we must enforce either that the edge (a, b) has positive weight, or that (a, b) has weight 0 and a beats b in the candidate tiebreaking ordering. So we also need to enforce a three-way constraint about the direction of each edge, whether it points in one direction or the other or whether it has zero weight. These constraints can easily be built in terms of inequalities as the other constraints. Thus as with Schulze we can embed the framework into an ILPFP, as long as we can implement inequalities between WMG edge weights in the ILPFP. We will demonstrate how this is accomplished for the various manipulative actions in the following sections.

4.3 Results by Looping over Frameworks

We now present our results that are established by our looping-over-frameworks idea. We will handle in separate sections manipulation, bribery, and control, showing how to achieve FPT results for each. The first section, on manipulation, will be given in more detail so as to ease the reader into the proof structure that the remainder of these proofs will follow.

4.3.1 Manipulation Results

Theorem 4.3.1. *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, manipulation is in FPT with respect to the number of candidates, in both the succinct and nonsuccinct input models, for both constructive and destructive manipulation, and in both the nonunique-winner model and the unique-winner model.*

Proof. We will assume without loss of generality that the candidates in the manipulation problem are $1, \ldots, j$ with the distinguished candidate being 1 (thus j is also the number of candidates). This does not affect the workings of either of the voting systems we are concerned with here, and it simplifies the construction of the ILPFP. The rest of the input to the manipulation problem will be the voter set V, either given as a list of votes in the standard nonsuccinct version, or as a list of which types of votes occur at least once, along with the multiplicities of each in the succinct version, and the manipulator set M. Algorithm 2 specifies the top-level loop for this algorithm, looping over all possible WCFs and considering those that satisfy the particular goal we are interested in.

So now we need to specify how to implement the WCF-enforcing ILPFP for either Schulze or ranked-pairs voting. Although the structure of the WCF for those two systems is fairly

Algorithm 2 Top level loop for manipulation

Start
for each j-WCF, (call it) K **do**
  if candidate 1 (is/is not) a (unique/nonunique) winner according to K and K is an internally consistent, well-formed j-WCF **then**
    (1) build an ILPFP that checks whether there is an assignment to the votes of M such that K's winner-set certification framework is realized by the set of votes $V \cup M$
    (2) run that ILPFP and if it can be satisfied then halt and accept (note: the satisfying settings will even let us output the precise manipulation that succeeds)
  end if
end for
declare that the given goal cannot be reached with $\|M\|$ manipulators
End

different, they are alike in being built from the same primitive objects: inequalities between weights of edges of the WMG, the *Bigger* and *StrictlyBigger* terms. And additionally for ranked pairs, we have to create inequalities stating that a particular edge is either positive, negative, or zero in weight. All of these terms can easily be written in terms of the WMG edge weight, so we just have to be able to express as a linear function of our ILPFP constants and variables the weight of an edge after manipulation. And then the ILPFP constants will define the parameters of the problem, with values derived from the input instance, and an assignment to the ILPFP variables defines a particular attempt to satisfy the current WCF for that iteration and to solve the problem.

We will now give the details of the ILPFP. First, we will number the $j!$ possible vote types over the j candidates as 1 through $j!$. We will then have a constant n_i, $1 \leq i \leq j!$, denoting how many votes of type i are in the nonmanipulative voter set V. We will have a variable m_i, $1 \leq i \leq j!$, denoting how many of the manipulative voters are chosen to cast a vote of type i. So we can see that we have $j!$ variables total, which is quite large but constant in terms of the number of candidates j, thus enabling us to get the desired performance from Lenstra's algorithm.

There are essentially two types of constraints that we must build: those enforcing the WCF, which we will build in terms of the WMG edge weight after manipulation, and those enforcing the additional constraints of the manipulation problem. We will handle the first type first. Let $D(a, b)$ describe the weight of a WMG edge (a, b) after manipulation. Given that we can express $D(a, b)$ in our linear program, we can build all of the necessary terms for enforcing the WCF. The term $\mathit{Bigger}(a, b, c, d)$, stating that the edge from a to b is at least as strong as the edge from b to a can be expressed as the following:

$$D(a, b) \geq D(c, d).$$

And $\mathit{StrictlyBigger}(a, b, c, d)$ can be expressed as the following:

$$D(a, b) \geq D(c, d) + 1.$$

And then the three-way constraints for ranked pairs can be expressed as one of the following:

$$\begin{array}{rcl} D(a, b) & \geq & 0 \\ D(a, b) & \leq & 0 \\ D(a, b) & = & 0 \end{array}.$$

So now we just have to express $D(a, b)$ in our linear program. We will define some more notation for simplification. Let $\mathit{pref}(a, b)$ be the set of vote types (chosen from $\{1, \ldots, j!\}$) in which candidate a is preferred to candidate b. Now we can implement $D(a, b)$ as follows:

$$\sum_{i \in \mathit{pref}(a,b)} (n_i + m_i) - \sum_{i \in \mathit{pref}(b,a)} (n_i + m_i).$$

In short, this is the number of voters from both the original voter set and the manipulators that prefer a over b, minus the number of voters from both the original set and the manipulator set that prefer b over a, and it gives us exactly $D(a, b)$, the weight of the edge (a, b) in the WMG.

Now we just have to implement the remaining constraints to ensure that the manipulation chosen is a valid one. In this case we only need a few additional constraints to ensure that we make a valid selection of manipulator votes when assigning the variables. We will have a the following constraint to ensure that the appropriate number of manipulator votes is assigned:

$$\sum_{1 \leq i \leq j!} m_i = ||M||.$$

And finally we have the following constraint for every i, $1 \leq i \leq j!$: $m_i \geq 0$, to prohibit a negative number of manipulators of any type.

This completes the specification of the algorithm. Thus we have algorithms to solve the manipulation problem for Schulze and ranked pairs that will run in a fixed polynomial time for any constant number of candidates, putting this problem in FPT. □

4.3.2 Bribery Results

Theorem 4.3.2. *For Schulze and (with any feasible tie-breaking functions) ranked-pairs elections, bribery is in FPT with respect to the number of candidates, in both the succinct*

and nonsuccinct input models, for both constructive and destructive bribery, in both the nonunique-winner model and the unique-winner model.

Proof. Again we assume the candidates are $1, \ldots, j$ and the distinguished candidate is candidate 1. The remainder of the input is the set of initial votes V (or for the succinct version, a list describing which types of votes occur at least once in the set, along with the multiplicities of each), and the bribing limit k. Algorithm 3 specifies the top-level loop.

Algorithm 3 Top level loop for bribery

Start
for each j-SWCF, (call it) K **do**
  if candidate 1 is a unique winner according to K and K is an internally consistent, well-formed j-SWCF **then**
    (1) build an ILPFP that checks whether there is a way of bribing at most k of the voters such that K's winner-set certification framework is realized by that bribe
    (2) run that ILPFP and if it can be satisfied then halt and accept (note: the satisfying settings will even let us output the precise bribe that succeeds)
  end if
end for
declare that the given goal cannot be reached by k bribes
End

So now we just have to specify how we build the ILPFP inside the loop for a particular WCF. Again we will number the vote types from 1 to $j!$. The constants of the ILPFP will include a constant n_i, $1 \leq i \leq j!$ denoting how many voters there are of vote type i in V. The variables will include a variable $m_{i,\ell}$, $1 \leq i, \ell \leq j!$, denoting how many voters of vote type i are bribed away to vote type ℓ. Note that we do not restrict that $i \neq \ell$, and although it would not be useful to bribe a voter to vote exactly the same way he or she already did, there is nothing in the model that prevents it either.

Now we will specify how to build the constraints of the ILPFP in terms of these constants and variables. As before, the main WCF-enforcing constraints can all be implemented in terms of $D(a, b)$, the weight of a WMG edge after manipulation. So we just have to be able to specify this term in order to be able to build all the necessary constraints. In the case of bribery, we can specify $D(a, b)$ as follows:

$$\left(\sum_{i \in pref(a,b)} n_i + \left(\sum_{1 \leq \ell \leq j!} m_{\ell,i} \right) - \left(\sum_{1 \leq \ell \leq j!} m_{i,\ell} \right) \right) - \left(\sum_{i \in pref(b,a)} n_i + \left(\sum_{1 \leq \ell \leq j!} m_{\ell,i} \right) - \left(\sum_{1 \leq \ell \leq j!} m_{i,\ell} \right) \right)$$

In other words, we add up the number of voters of each vote type preferring a over b, adding in the voters that are bribed into this vote type and subtracting out the voters that are bribed away from that vote type, and then subtract the same expression except with the vote types preferring b over a. With this expression we can build all the necessary constraints to enforce the WCF.

Additionally we must include constraints to enforce that we have a valid selection of voters to bribe. For every i, $1 \leq i \leq j!$, we must have that $\sum_{1 \leq \ell \leq j!} m_{i,\ell} \leq n_i$, enforcing that we do not bribe away more voters of a certain type than existed in the first place. For every i, ℓ, $1 \leq i, \ell \leq j!$, we have a constraint $m_{i,\ell} \geq 0$, enforcing that all our counts of bribed voters are nonnegative. And finally we have a constraint $\sum_{1 \leq i, \ell \leq j!} m_{i,\ell} \leq k$, enforcing that the total number of bribes is bounded by the limit.

This completes our description of the WCF-enforcing ILPFP and our the algorithm for these cases. This gives us an algorithm that will run uniformly in polynomial time for every fixed number of candidates, putting this problem in FPT.

□

4.3.3 Control Results

In this section, we prove our results for control gained through our looping-over-frameworks approach. Our proofs for these case will closely follow our general looping-over-frameworks structure, and we will just have to appropriately build the constraints of our ILPFPs to handle the details of the control problems.

Theorem 4.3.3. *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control by adding voters is in FPT with respect to the number of candidates, in both the succinct and nonsuccinct input models, for both constructive and destructive control, and in both the nonunique-winner model and the unique-winner model.*

Proof. As before, we assume the candidates are $1, \ldots, j$ and the distinguished candidate is candidate 1. The remainder of the input is the set of initial votes V, the set of additional votes W (or for the succinct version, one list for each of these two sets describing which types of votes occur at least once in that set, along with the multiplicities of each), and an adding limit k. Algorithm 4 specifies the top-level loop.

Now we will specify how we build the WCF-enforcing ILPFP inside the loop. The constants of the ILPFP will include a constant n_i, $1 \leq i \leq j!$, representing how many votes of each vote type i are in the initial election and a constant h_i, $1 \leq i \leq j!$, representing how many votes of each vote type i are in the additional set W. The variables of the ILPFP will include a variable v_i, $1 \leq i \leq j!$, representing how many voters of each type are added to the election.

Algorithm 4 Top level loop for control by adding voters

Start
for each j-WCF, (call it) K **do**
  if candidate 1 (is/is not) a (unique/nonunique) winner according to K and K is an internally consistent, well-formed j-WCF **then**
    (1) build an ILPFP that checks whether there is a set of votes $W' \subseteq W$, with $\|W'\| \leq k$, such that K's winner-set certification framework is realized by the set of votes $V \cup W'$
    (2) run that ILPFP and if it can be satisfied then halt and accept (note: the satisfying settings will even let us output the precise added set that succeeds)
  end if
end for
declare that the given goal cannot be reached by adding at most k voters
End

To finish the specification, we just have to describe how to implement $D(a, b)$ as a linear expression of these constants and variables so we can build the *Bigger* and *StrictlyBigger* terms and the additional constraints for ranked pairs that comprise our WCFs, and additionally how to implement the other constraints to enforce the validity of the manipulative action. We can implement $D(a, b)$ in this case as follows:

$$\sum_{i \in pref(a,b)} (n_i + v_i) - \sum_{i \in pref(b,a)} (n_i + v_i).$$

There are a few necessary constraints to enforce the validity of the manipulative action. We include a constraint $v_i \geq 0$ for every i, $1 \leq i \leq j!$, enforcing that we cannot add a negative number of voters of any type. We include constraints $h_i \geq v_i$ for every i, $1 \leq i \leq j!$, enforcing that we cannot add more voters of any type than we have available in W. And we add a constraint $\sum_{1 \leq i \leq j!} v_i \leq k$, bounding the number of additions by the bound k.

This suffices to describe how we can build a WCF-enforcing ILPFP for the control by adding voters problem in a way appropriate for our looping-over-frameworks technique. Thus we have an algorithm that will run in polynomial time for every fixed parameter value, putting this problem in FPT. □

Theorem 4.3.4. *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control by deleting voters is in FPT with respect to the number of candidates, in both the succinct and nonsuccinct input models, for both constructive and destructive control, and in both the nonunique-winner model and the unique-winner model.*

Proof. The input for this problem is a control instance with a set of votes V over j candidates $1, \ldots, j$, with candidate 1 being the distinguished candidate, and with a bound k on the number of votes to be deleted. Algorithm 5 specifies the top-level loop.

Algorithm 5 Top level loop for control by deleting voters

Start
for each j-WCF, (call it) K **do**
if candidate 1 (is/is not) a (unique/nonunique) winner according to K and K is an internally consistent, well-formed j-WCF **then**
(1) build an ILPFP that checks whether there is a subset of the voters V', with $\|V'\| \leq k$, such that K's winner-set certification framework is realized by the set of votes $V - V'$
(2) run that ILPFP and if it can be satisfied then halt and accept (note: the satisfying settings will even let us output the precise deleted set that succeeds)
end if
end for
declare that the given goal cannot be reached by deleting at most k voters
End

The ILPFP we build inside the loop will include a constant n_i, $1 \leq i \leq j!$, representing how many votes of each vote type i are in the initial election. We will include in the ILPFP variables v_i, $1 \leq i \leq j!$, representing the number of votes of type i that are deleted. With this new variable, we will formulate $D(a, b)$ in the ILPFP as follows:

$$\sum_{i \in pref(a,b)} (n_i - v_i) - \sum_{i' \in pref(b,a)} (n_{i'} - v_{i'}).$$

With this we can easily build the *Bigger* and *StrictlyBigger* predicates we use to enforce the WCF, as well as the additional constraints necessary for an RPWCF.

Additionally, we need to include constraints that ensure that the control action chosen through the assignment to the variables is a legal one. We include a constraint $v_i \geq 0$ for every i, $1 \leq i \leq j!$, enforcing that we cannot delete a negative number of voters of any type. We include constraints $n_i \geq v_i$ for every i, $1 \leq i \leq j!$, enforcing that we cannot delete more voters of any type than there were in the first place. And finally we add a constraint $k \geq \sum_{1 \leq i \leq j!} v_i$, bounding the total number of deletions by the bound k.

These modifications to our basic approach embed the problem of deciding what subset of voters to delete to satisfy a given WCF into an ILPFP, and there will only be a constant number of ILPFPs to test for each number of candidates. Thus the algorithm specified by the outer loop above is uniformly polynomial for every fixed parameter value, putting this problem in FPT. □

Theorem 4.3.5. *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control by partition of voters is in FPT with respect to the number of candidates, in both the ties-eliminate and ties-promote models, in both the succinct and nonsuccinct input models, for both constructive and destructive control, and in both the nonunique-winner*

model and the unique-winner model.

Proof. These cases can again be handled with our looping-over-frameworks approach, and the constraints and modifications necessary to properly implement the WCF-enforcing ILPFP are similar to the previous cases, though we have to loop over not just single WCFs but rather pairs of them, one for each part of the partition. As before we will assume the candidate names are $1, \ldots, j$ with 1 as the distinguished candidate. We specify the outer loop for these cases in Algorithm 6.

Algorithm 6 Top level loop for control by partition of voters

Start
for each j-WCF, (call it) K_1 **do**
  for each j-WCF, (call it) K_2 **do**
    if K_1 and K_2 are internally consistent, well-formed j-WCFs **then**
      (1) build an ILPFP that checks whether there is a partition of the voters V into V_1, V_2, such that K_1's winner-set certification framework is realized by the set of votes V_1 and K_2's winner-set certification framework is realized by the set of votes V_2
      (2) run that ILPFP and if it can be satisfied, and if candidate 1 (is/is not) a (unique/nonunique) winner in the election with voters V and the surviving candidates from K_1 and K_2 (according to the appropriate tie-handling rule), halt and accept (note: the satisfying settings will even let us output the precise partition that succeeds)
    end if
  end for
end for
declare that the given goal cannot be reached by any partition of voters
End

Now we must specify the new details of the partition-handling ILPFPs. For every i, $1 \leq i \leq j!$, we will include constants n_i representing how many votes of each vote type i are in the initial election and variables v_i representing how many votes of type i are placed in V_1.

Though we must use two WCFs, we can implement each as before, and in parallel in a single ILPFP. Let us use the predicates $Bigger_1$ and $StrictlyBigger_1$ ($Bigger_2$ and $StrictlyBigger_2$) to handle the constraints of WCF K_1 (K_2) over the weight of its WMG edges which we will denote by D_1 (D_2).

$Bigger_1$ and $StrictlyBigger_1$ can be formulated appropriately in terms of D_1, while $Bigger_2$ and $StrictlyBigger_2$ can be formulated in terms of D_2. The extra constraints for ranked pairs will be implemented in terms of D_1 and D_2 as appropriate as well. We will now show how to handle D_1 and D_2. $D_1(a, b)$ denotes how many voters in V_1 prefer candidate a to candidate b and we can formulate it as follows:

$$\sum_{i \in pref(a,b)} (v_i) - \sum_{i \in pref(b,a)} (v_i).$$

$D_2(a, b)$ denotes how many voters in V_2 prefer a to b, and we formulate it as follows:

$$\sum_{i \in pref(a,b)} (n_i - v_i) - \sum_{i \in pref(b,a)} (n_i - v_i).$$

Finally, we must add constraints to ensure the chosen partition is a legal one. We include a constraint $v_i \geq 0$ for every i, $1 \leq i \leq j!$, enforcing that a nonnegative number of voters of each type are chosen for the first partition. We include a constraint $n_i \geq v_i$ for every i, $1 \leq i \leq j!$, enforcing that we do not take more voters than exist of each type for the first partition.

These modifications to our basic approach embed the problem of deciding what partition of voters to use to satisfy a pair of WCFs into an ILPFP, and there will only be a constant number of ILPFPs to test for each number of candidates. Thus the algorithm specified by the outer loop above is uniformly polynomial for every fixed parameter value, putting this problem in FPT. □

4.4 Other Results

4.4.1 Candidate Control Parameterized on the Number of Candidates

Under our primary parameterization of interest, parameterizing on the number of candidates, and when considering manipulation, bribery, and voter control, we achieved FPT results through great effort using our looping-over-frameworks technique. In contrast, when considering candidate control problems parameterized on the number of candidates, we need not use such a powerful technique. Instead it is sufficient to brute-force search over all possible control solutions to see if any of them are successful. At every possible value for the parameter, there are only a constant number of possible solutions to any of the candidate control problems, and checking the success of the possible solution will only require a simple polynomial-time task. Thus the full procedures at every fixed parameter value will run in time of a large constant times a small uniform polynomial, putting the problems in FPT.

Theorem 4.4.1. *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control is in FPT with respect to the number of candidates, in both the succinct and nonsuccinct input models, for both constructive and destructive control, for all standard types of candidate control (adding/unlimited adding/deleting candidates, and, in both the*

ties-eliminate and ties-promote first-round promotion models, partition and runoff partition of candidates), in both the nonunique-winner model and the unique-winner model.

Proof. In the case of adding candidates, at most all the $2^{||D||}$ possible subsets of the auxiliary candidate set need be considered. In the case of deleting candidates, we must consider at most the $2^{||C||-1}$ possible subsets of the candidate set to delete. In the case of runoff partition, there are $2^{||C||-1}$ distinct partitions, while in the partition case there are $2^{||C||}$. The difference between the partition cases is because the runoff partition the two parts are handled symmetrically, while they are not in the partition case. For any of these cases, we have to at most run three iterations of the voting system function, so all these cases will be in FPT for any voting system with a polynomial-time winner problem. □

4.4.2 Voter Control Parameterized on the Number of Voters

Although we feel that the number of candidates is the most natural parameterization for manipulative action problems, it is natural to ask about parameterizing on the number of voters. We do not exhaustively handle this case for all manipulative action problems, but we note that voter control problems parameterized on the number of voters can be shown to be in FPT through simple brute-force. Again, as in the case of candidate control problems parameterized on the number of candidates, we note that the number of possible solutions to these problems is bounded by a constant for each parameter value, and checking each solution is easily done in polynomial time, giving us an FPT algorithm for each of the voter control problems.

Theorem 4.4.2. *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control is in FPT with respect to the number of voters, in both the succinct and nonsuccinct input models, for both constructive and destructive control, for all standard types of voter control (adding/deleting voters, and, in both the ties-eliminate and ties-promote first-round promotion models, partition of voters), in both the nonunique-winner model and the unique-winner model.*

Proof. In the case of adding voters we will have to try at most all of the $2^{||W||}$ possible subsets of the auxiliary voter set. In the case of deleting voters we will have to consider at most all the $2^{||V||}$ subsets of the voter set. And in the partition of voters cases we will have to consider the $2^{||V||-1}$ distinct partitions of the voter set. In all of these cases the large exponential term of the complexity will be constant with fixed parameter values. And beyond that term, we will just have to perform a few iterations of the voting system's winner function along with a few other simple checks, putting all these cases in FPT for any voting system with a polynomial-time winner problem. □

4.4.3 WMG Edge Bound Parameterization

The technique used here of looping over frameworks was necessitated by the fact that the natural structure for representing information about Schulze or ranked-pairs elections, the WMG, is too rich a structure for our purposes, and there are not a bounded number of possible WMGs for a fixed number of candidates. The number of WMGs with bounded edge weight is bounded in terms of the number of candidates, but working in this setting would give us results of narrow importance, applying only to cases where all pairwise contests are close to tied, even as the number of candidates grows arbitrarily large.

Even though that setting is questionably natural, one might assume that bounding the WMG edge weights would simplify many hard election problems. But we find this to be often untrue. As a corollary to the work of Menton and Singh ([MS13], also presented in Chapter 3 of this thesis), we observe that several of the control problems in Schulze voting are NP-complete even if the maximum WMG edge weight is bounded by a constant, or even if the set of possible edge weights is restricted to a few options.

Corollary 4.4.3. (Corollary to the proofs of [MS13].) *Even when restricted to elections having all pairwise contests so equal that each WMG edge has absolute value at most 2 (or alternatively that the total cardinality of the set of absolute values of WMG edges is at most 2; or alternatively that the total cardinality of the set of values of WMG edges is at most 3), Schulze elections are NP-complete (in the nonunique-winner model) for constructive control by deleting candidates. The same holds for constructive control by adding candidates, unlimited adding of candidates, partition of candidates in the ties-eliminate model, and run-off partition of candidates in the ties-eliminate model, except with the 2/2/3 above replaced by 4/3/5. The same holds for constructive control by partition of candidates in the ties-promote model and run-off partition of candidates in the ties-promote model, except with the 2/2/3 above replaced by 6/4/7.*

4.4.4 Adding/Deleting Bound Parameterization

For those control problems having as part of their inputs a limit on how many candidates or voters can be added/deleted, it is natural to consider parameterizing on that limit. This parameterization has been studied in some voting systems and the relevant problems have often been found to be W[1]-hard or W[2]-hard, and thus very unlikely to be fixed-parameter tractable. For example, under this parameterization, Betzler and Uhlmann [BU09] showed, for what are known as Copeland$^{\alpha}$ elections, that constructive control by adding candidates and constructive control by deleting candidates are W[2]-complete, Liu and Zhu [LZ10] proved, for maximin elections, that constructive control by adding candidates is W[2]-hard, and Liu and Zhu also achieved W[1]-hardness results for the relevant voter control problems.

For additional control results parameterized on the problem's internal addition/deletion limit, see Table 8 of [BBCN12].

We observe that, with respect to parameterizing on the internal addition/deletion bound, for Schulze elections, constructive control by adding voters and constructive control by deleting voters are both W[2]-hard. This can be seen through simple modifications of the NP-hardness proofs for these cases given by Menton and Singh [MS13]. Namely, we appropriately modify the proofs to reduce from the W[2]-complete problem hitting set instead of from exact cover by three-sets. Additionally, constructive control by adding candidates can be seen to be W[2]-hard for Schulze elections through a natural reduction from hitting set.

Theorem 4.4.4. *Constructive control by adding candidates parameterized on the adding bound is W[2]-hard for Schulze voting.*

Proof. We will reduce from the W[2]-hard problem hitting set.

Definition 4.4.5 (Hitting Set)**.** *Given a set of elements U, a collection $\mathcal{F}$ of subsets of U, and a positive integer k, does there exist $H \subseteq U$, with $\|H\| \leq k$, such that for every $S \in \mathcal{F}$, we have $S \cap H \neq \emptyset$ (i.e. H hits every set in $\mathcal{F}$)?*

Given a hitting set instance $(U, \mathcal{F}, k)$ as described in the definition, we will construct a control instance (C, D, V, p, k) where C is the set of original candidates, D is the set of auxiliary candidates, V is the set of voters, p is the distinguished candidate, and k is the adding bound. The original candidate set C will contain the following:

- The distinguished candidate p.
- A candidate S for every $S \in \mathcal{F}$.

The auxiliary candidate set D will contain the following:

- A candidate u for every $u \in U$.

The voter set V will be as follows. We will not explicitly construct the entire voter set, but rather we will specify the weight of the WMG edges between the candidates and let the voter set be as constructed by McGarvey's method [McG53].

- For every $S \in \mathcal{F}$, $D(S, p) = 2$.
- For every $u \in U$, $D(p, u) = 2$.
- For every $u \in U$, and for every $S \in \mathcal{F}$ such that $u \in S$, $D(u, S) = 2$.
- All other WMG edges will be of weight 0.

The distinguished candidate will be p and the adding limit will be the same limit k as in the hitting set instance. This completes the specification of the reduction.

If we map from a positive hitting set instance, we will have a positive instance of the control problem. Let $H \subseteq U$, $\|H\| \leq k$, be a solution to the hitting set instance. We will show that the set of candidates D' corresponding to the elements from H will be a solution to the control instance. First, $\|D'\| \leq k$, so we are within the adding bound. Also, since the hitting set solution includes members of every set in $\mathcal{F}$, there will be a path from p to every candidate corresponding to those sets, as including a candidate $u \in U$ creates paths from p to every set S hit by u. So p will have paths to every other candidate just as strong as they have back to p, and so p will be a winner.

If we map to a successful control instance, we must have a positive hitting set instance. Succeeding in control will require us to create paths to every candidate that has a path to the distinguished candidate, so we must add a set of candidates that give us all the necessary paths. But we are limited to adding at most k of the candidates from the auxiliary set. It is then easy to see that a solution to our control problem, creating paths to every adversary candidate with a bounded number of additions, corresponds to a solution to the hitting set instance.

This proof is an FPT reduction. It clearly runs in polynomial time in the size of the entire input (exceeding the requirements for an FPT reduction), and the parameter in the mapped-to instance will always be bounded by (in fact, identical to) the parameter in the mapped-from instance. So this proof shows that constructive control by adding candidates parameterized on the adding bound is W[2]-hard for Schulze voting. □

4.5 The Weighted Case

So far we have just been concerned with unweighted problem variants. However we can extend our methods to weighted manipulation and to weighted and/or priced bribery. To achieve FPT performance in these cases we must parameterize on the numbers of weights and prices, as present, in addition to the number of candidates. We can handle these cases as a straightforward extension of our previous proofs by bounding the maximum weight and/or price in an instance. But we can go further and achieve stronger results parameterizing not on the maximum weight/price but rather on the cardinality of the weight/price sets. We will first handle the case of bribery.

Theorem 4.5.1. *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, weighted, priced, and weighted and priced bribery is in FPT with respect to the combined parameter "number of candidates" and "cardinality of the set of voter weights" and "cardinality of the set of voter prices" in both the succinct and nonsuccinct input models,*

for both constructive and destructive manipulation, and in both the nonunique-winner model and the unique-winner model.

Proof. We will describe how to handle the weighted and priced case, and note that this is a generalization of either the weighted and unpriced case or the unweighted and priced case, so this algorithm will also work for either of these. The top-level loop will be as in the standard bribery proof in Section 4.3.2. We will specify the extensions to the ILPFP construction that are necessary to handle the weights and prices.

Let $1, \ldots, j$ be the candidates, let $w = \{w_1, \ldots, w_y\}$ be the bounded-cardinality set of weights of the voters, and let $p = \{p_1, \ldots, p_z\}$ be the bounded-cardinality set of prices. We will have a constant $n_i^{\alpha,\beta}$ denoting how many voters there are of type i with weight w_α and with price p_β for every i, $1 \leq i \leq j!$, α, $1 \leq \alpha \leq y$, and β, $1 \leq \beta \leq z$. We will have a variable $m_{i,\ell}^{\alpha,\beta}$, for every i, $1 \leq i \leq j!$, ℓ, $1 \leq \ell \leq j!$, α, $1 \leq \alpha \leq y$, and β, $1 \leq \beta \leq z$. $m_{i,\ell}^{\alpha,\beta}$ describes the number of voters with weight w_α and price p_β that are bribed from vote type i to vote type ℓ. Our total number of variables is larger than in the unweighted unpriced case, but it is still bounded in our parameters j, y, and z.

Now we will describe how to build the constraints in our ILPFP. As before, we can implement the predicates *Bigger* and *StrictlyBigger* and the additional constraints to handle ranked pairs in terms of $D(a, b)$, the weight of the edge from a to b in the WMG, so we need only specify how to formulate $D(a, b)$. We need to take into account both voters that are bribed away from preferring a over b and voters that are bribed towards preferring a over b. And we need to keep track of voters of every type, weight, and price. Still, what we need is very much in line with our simpler implementation of this function in the standard bribery case. So our new formulation of $D(a, b)$ will be the following:

$$\sum_{1 \leq \alpha \leq y} \sum_{1 \leq \beta \leq z} \sum_{i \in pref(a,b)} w_\alpha \left(n_i^{\alpha,\beta} - \sum_{1 \leq \ell \leq j!} \left(m_{i,\ell}^{\alpha,\beta} \right) + \sum_{1 \leq \ell \leq j!} \left(m_{\ell,i}^{\alpha,\beta} \right) \right)$$
$$- \sum_{1 \leq \alpha \leq y} \sum_{1 \leq \beta \leq z} \sum_{i \in pref(b,a)} w_\alpha \left(n_i^{\alpha,\beta} - \sum_{1 \leq \ell \leq j!} \left(m_{i,\ell}^{\alpha,\beta} \right) + \sum_{1 \leq \ell \leq j!} \left(m_{\ell,i}^{\alpha,\beta} \right) \right).$$

Additionally we need to implement constraints to make sure that the bribery action we select is a legal one. Therefore, besides the natural constraints $m_{i,\ell}^{\alpha,\beta} \geq 0$, ensuring that we do not bribe a negative number of voters of a certain type, we have to enforce that we do not bribe more voters than exist of every type. Thus for every i, $1 \leq i \leq j!$, α, $1 \leq \alpha \leq y$, and β, $1 \leq \beta \leq z$, we have the following constraint:

$$n_i^{\alpha,\beta} \geq \sum_{1 \leq \ell \leq j!} m_{i,\ell}^{\alpha,\beta}.$$

And also we must restrict the total cost of all our bribes to the bribing limit k. Thus we need the following constraint:

$$k \geq \sum_{1 \leq \alpha \leq y} \sum_{1 \leq \beta \leq z} \sum_{1 \leq i \leq j!} \sum_{1 \leq \ell \leq j!} p_\beta m_{i,\ell}^{\alpha,\beta}.$$

These additional constraints will enforce a legal selection of voters to bribe. These, together with our specification of $D(a, b)$, complete the specification of the WCF-enforcing ILPFP. Overall we have an structure that will have a constant number of constraints in terms of the parameters, and that will have a solution if and only if there is a successful bribery. Also there will only be a constant number of such objects for every fixed set of parameter values (in fact it will be constant in terms of just the number of candidates, and there will be no more WCFs than in the unweighted unpriced case).

Thus the weighted, priced, and weighted and priced variants of bribery are all in FPT when parameterized on the number of candidates as well as on the cardinalities of the weight and/or price sets, as present. □

For the case of manipulation, we can handle the case where only the cardinality of the set of weights of the *manipulative* voters is bounded by a parameter, and the number of weights among the nonmanipulative voters is unbounded.

Theorem 4.5.2. *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, coalitional weighted manipulation is in FPT with respect to the combined parameter "number of candidates" and "cardinality of the manipulators' weight set" in both the succinct and nonsuccinct input models, for both constructive and destructive manipulation, and in both the nonunique-winner model and the unique-winner model.*

Proof. Again let the candidates be $1, \ldots, j$ with candidate 1 being the distinguished candidate. The input specifies the nonmanipulator votes as a set of votes along with their weights, and it specifies the set of manipulator votes as a vector of pairs $((w_1, t_1), \ldots, (w_s, t_s))$ with there being t_i manipulators having weight w_i. Our top level loop will be essentially as in the unweighted manipulation case in Section 4.3.1.

Handling this case requires some surgery on our basic approach. Since we don't want to bound the number of weights of the nonmanipulative voters, we can't have constants representing how many voters there are of every weight and of every type. But we don't need that information to handle this case. All the "decision making" in the problem happens regarding the manipulative voters, so we only need a detailed representation of the manipulative voters. A summary of the preferences of the nonmanipulative voters will suffice. Namely, we have a constant $n_{a,b}$ describing the total weight of the nonmanipulative voters casting votes that prefer candidate a to candidate b for every pair of candidates a, b. The

values $((w_1, t_1), \ldots, (w_s, t_s))$ describing the weights and how many manipulators there are of each weight will be constants of the ILPFP as well. The variables of the ILPFP describe how many manipulators of each weight get manipulated to each particular vote type: m_i^α for every i, $1 \leq i \leq j!$ and α, $1 \leq \alpha \leq s$, describes how many manipulators of weight w_α are assigned to vote type i.

The WCF can be implemented in terms of *Bigger* and *StrictlyBigger* predicates as before, with these implemented in terms of $D(a, b)$. In this case we will express $D(a, b)$ with the following:

$$n_{a,b} + \sum_{1 \leq \alpha \leq s} \sum_{i \in pref(a,b)} (w_\alpha m_i^\alpha) - n_{b,a} - \sum_{1 \leq \alpha \leq s} \sum_{i \in pref(b,a)} (w_\alpha m_i^\alpha) .$$

Besides this we just need a few extra constraints to ensure that the manipulation chosen is a reasonable one. We include a constraint $m_i^\alpha \geq 0$ for every i, $1 \leq i \leq j!$, and α, $1 \leq \alpha \leq s$, enforcing that we do not include a negative number of voters of any type. And we include a constraint $\sum_{1 \leq i \leq j!} m_i^\alpha = t_\alpha$ for every α, $1 \leq \alpha \leq s$, ensuring that we use the same number of manipulators of every weight as we have available.

With these additions, we can build an ILPFP to encode our WCF that will be polynomially bounded in size with fixed values for the two parameters, and there will only be a constant number of such WCFs with fixed parameter values. Thus our algorithm specified above will run in uniformly polynomial time with fixed parameter values and this problem is in FPT. □

This still is all a valid framework for our many-uses-of-Lenstra-based approach. Note that for the case of the cardinality of the set of weights being 1, that gives the case of weighted noncoalitional manipulation mentioned as an aside by Dorn and Schlotter [DS12], though here we're handling even any fixed-constant number of manipulators (since any fixed-constant number have at most a fixed-constant cardinality of their weight set), and indeed, even a number of manipulators whose cardinality isn't bounded but who among them in total have a fixed-constant cardinality of occurring weights.

Chapter 5

Control in NRV: Closing the Final Case

5.1 Introduction

The ranked-vote model became the dominant one in voting theory with Arrow's seminal work, though not all voting systems follow this convention. The primary alternative vote model is for a vote to consist of a vector with scores independently awarded to each candidate. Voting systems in this class are sometimes known as "utilitarian" voting systems [Hil05], as they allow voters to vote by giving their utilities for each of the candidates. Approval voting, where each candidate is given a score of either 1 or 0 and the most approved candidates win, is the best known voting system in this model.

Range voting (RV) is a voting system using this alternative voter model that allows voters to score candidates individually with integer values in a specified range 0–k, and the winners of the election are those candidates with the highest sum scores [Smi00]. Normalized range voting (NRV) is a variant that normalizes the voter's scores to fill the entire range so as the maximize the influence of each voter. See Section 2.1.5 for a complete description of range voting and normalized range voting.

5.2 Behavior of RV and NRV Under Control

In previous work, I showed that normalized range voting resists a large number of cases of control, but one case, destructive control by partition of voters in the ties-eliminate model, remained open [Men09]. This last proof completed this analysis of the complexity of control in normalized range voting, showing that it is resistant to every standard case of

control other than destructive control by adding or deleting voters. This work, following the seminal paper on control [BTT92], was done in the unique winner model of control. Table 5.1 summarizes the behavior of range voting and normalized range voting under control, and compares them to the related system approval voting and to Bucklin and fallback voting, which are the other systems with the largest known numbers of control resistances.

Control by	Tie Model	Approval		Bucklin		Fallback		RV		NRV	
		C	D	C	D	C	D	C	D	C	D
Adding Candidates		I	V	R	R	R	R	**I**	**V**	**R**	**R**
Adding Candidates (unlimited)		I	V	R	R	R	R	**I**	**V**	**R**	**R**
Deleting Candidates		V	I	R	R	R	R	**V**	**I**	**R**	**R**
Partition of Candidates	TE	V	I	R	R	R	R	**V**	**I**	**R**	**R**
	TP	I	I	R	R	R	R	**I**	**I**	**R**	**R**
Run-off Partition of Candidates	TE	V	I	R	R	R	R	**V**	**I**	**R**	**R**
	TP	I	I	R	R	R	R	**I**	**I**	**R**	**R**
Adding Voters		R	V	R	V	R	V	**R**	**V**	**R**	**V**
Deleting Voters		R	V	R	V	R	V	**R**	**V**	**R**	**V**
Partition of Voters	TE	R	V	R	R	R	R	**R**	**V**	**R**	**R**
	TP	R	V	R	S	R	R	**R**	**V**	**R**	**R**

Table 5.1: Control results for approval [HHR07], Bucklin voting [EF10b, EPR11], fallback voting [ER10, EF10b, EPR11], range voting, normalized range voting.

Theorem 5.2.1. *4-NRV is resistant to destructive control by partition of voters in the ties-eliminate model.*

We will reduce from the Exact Cover by Three-Sets problem (X3C), defined as follows.

Given: A set $B = \{b_1, \ldots, b_{3j}\}$ and a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of sets of size three of elements from B.

Question: Is it possible to select j sets from $\mathcal{S}$ such that their union is exactly B?

Proof. Given an X3C instance $(B, \mathcal{S})$ we will construct a 2-range election (C, V) as follows. The candidate set will be $B \cup \{c, w\}$, where c will be the distinguished candidate. The voters set V will consist of the following:

- For every $S_i \in \mathcal{S}$, one voter with a score of 4 for every candidate in $B - S_i$, a score of 2 for c, and a score of 0 for every other candidate;
- $2n$ voters with a score of 4 for every candidate in B, a score of 2 for c, and a score of 0 for w;
- $j - 1$ voters with a score of 4 for w, a score of 2 for c, and a score of 0 for all other candidates;

- For every $b \in B$, 1 voter with a score of 4 for b, a score of 1 for every candidate in $B - \{b\}$, a score of 1 for c, and a score of 0 for w;
- $2j + 3n + 1$ voters with a score of 4 for w and a score of 0 for every other candidate.

We will assume that $1 \leq j \leq n$ and that each element in B is in at least one set from $\mathcal{S}$.

If there is an exact cover over B, then w can be made to lose the election through partition of voters in the ties-eliminate model. Consider the partition of the voter set into V_1, V_2, where V_1 consists of the voters from the first group corresponding to the elements of the set cover together with the third group of voters, and where $V_2 = V - V_1$.

The candidate w will win the subelection (C, V_2) and the distinguished candidate c will easily win the subelection (C, V_1). c will receive $2j$ points from the cover voters and $2j - 2$ points from the other voters for a total of $4j - 2$ points. For the B candidates, each will receive 4 points from each of the first-group voters except the one that corresponds to the set that covers them, coming to $4j - 4$ points in total. w will gain 4 points from each of the $j - 1$ voters that favors them, and so he or she will gain $4j - 4$ votes in total. Therefore c will win this subelection and will go on to face w in the final election.

In the final election, with almost all of the candidates eliminated, vote normalization will occur benefiting c to the point that he or she gains the advantage. All of the votes in the first, second, and fourth groups will now give four points to c, giving them $12n + 14j - 2$ points in total. This is at least as many as the $12n + 12j$ points w was given, and so w will no longer be the unique winner.

If there is no exact cover, w cannot be made to lose the election through partition of voters in the ties-eliminate model. This is due to several facts: w will win at least one of the two subelections, c cannot win either subelection, and w will win head to head against all candidates that it may face in the final election.

c cannot win an initial subelection except as previously described. No voter prefers them outright, so the only way to make them win is to balance the points he or she gains from the first group of voters and from the third group of voters and to give him or her an advantage over each B candidate by covering that candidate with a voter that prefers c. None of the other voters will help c win a subelection, as no others help c gain points relative to the B candidates. If a set of first group voters smaller than j that is not a cover is chosen, then at least one B candidate will gain points for every one of these voters. We will then not be able to boost c over the B candidates without giving too many points to w. If a cover larger than j of voters is chosen, then there will not be enough third group voters to include to boost c over the B voters and c will not be able to win.

The candidate w must win at least one of the initial subelections and make it to the final election. With the original candidate set and unnormalized scores, w has a considerable

advantage in points over all other candidates, and since it is largely the same voters that support all of the other candidates, there is no way to partition the voters to make w lose both subelections.

Against all candidates but c, w will win a head to head contest in the final election. In a head-to-head contest with w the other candidates gain about as many points as c through normalization, but they will still have fewer points total as each B candidate loses out on 4 points for every subset S_i they are a part of. The B candidate will thus have no more than $12n + 12j - 4$ votes, while w will have $12n + 12j$ votes and will be the winner.

Therefore w will win the final election for all partitions in the case that there is not an exact cover over the set B. □

Chapter 6

Reducing Search to Decision for Election Problems

6.1 Introduction

The dominant convention in the computational social choice literature has been to analyze the complexity of the decision versions of manipulative action problems such as manipulation, bribery, and control. Decision problems are simpler to define and analyze and tools of complexity theory such as reductions can be used to determine their complexity. However, in the case of, for instance, the manipulation problem, a manipulator would really be more concerned with specifically how to set their vote to make their favorite candidate win, rather than just whether it is possible for them to do so. So it may be more appropriate to examine the complexity of the search versions of these problems. And it is very important to note that the difficulty of these two versions of a problem can differ.

To be precise, by a decision version of a problem we mean the formally defined problems given in Section 2 phrased as yes/no questions. These problems in complexity theory convention can be interchangeably considered as sets, where the elements of the set are the positive instances of the problem. These problems we can properly consider to be in such decision problem classes as P or NP-hard. By the search version of an election manipulation problem, we mean the task of actually generating a successful action for a given manipulation problem instance if one exists, or else indicating failure. We say that a search problem is easy if there is a polynomial-time algorithm that does just that.

We can easily show that the decision version of any problem is no harder than the search version, as we can use the search problem to solve the decision problem. That is, decision always reduces to search. This can be done by simply querying the search version of the

problem, and if it returns a solution we know we have a positive instance. Therefore the decision version of a problem is never harder than the search version of a problem. A NP-hardness proof for the decision version of a problem implies that the search version is at least as hard, and so the search problem is also extremely unlikely to be solvable in polynomial time. However, a proof of a polynomial-time upper bound for the decision version of a problem does not by itself show that the search version is also easy.

The question then for any given problem is whether search reduces to decision. We will present here proofs for a number of the control cases showing that search does reduce to decision for any voting system, through giving polynomial-time Turing reductions.

In the remaining cases (manipulation, bribery, and the remaining cases of control), search provably does not reduce to decision given a common complexity-theoretic assumption. The Borodin-Demers Theorem states that $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$ implies the existence of a set $B \in \mathrm{P}$ where $B \subseteq \mathrm{SAT}$, but for which there is no polynomial-time algorithm that can find satisfying assignments for all the elements of B [BD76]. For these other cases, we build voting systems where the relevant manipulative action decision problem is in P, but where the problem of finding satisfying assignments for a Borodin-Demers set B polynomial-time reduces to the search problem for this manipulative action. Thus the search version cannot have a polynomial-time algorithm, despite the decision problem being in P, as for it to do so would contradict the properties of the Borodin-Demers set B and the assumption that $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$.

The nonunique winner model was used throughout this work. This is in contrast to the previously discussed study of control in normalized range voting.

6.2 Cases Where Search Reduces to Decision

Theorem 6.2.1. *For each manipulative action $\mathcal{A}$ belonging to the following list:*

1. *constructive control by adding voters,*
2. *destructive control by adding voters,*
3. *constructive control by deleting voters,*
4. *destructive control by deleting voters,*
5. *constructive control by adding candidates,*
6. *destructive control by adding candidates,*
7. *constructive control by deleting candidates,*

8. *destructive control by deleting candidates,*

9. *destructive control by partition of candidates, ties promote,*

10. *destructive control by partition of candidates, ties eliminate,*

11. *destructive control by run-off partition of candidates, ties promote,*

12. *destructive control by run-off partition of candidates, ties eliminate,*

13. *constructive control by unlimited adding of candidates, and*

14. *destructive control by unlimited adding of candidates,*

and for each election system $\mathcal{E}$, the $\mathcal{A}$ search problem for $\mathcal{E}$ polynomial-time Turing reduces to the $\mathcal{A}$ decision problem for $\mathcal{E}$.

6.2.1 Theorem 6.2.1, Control by Adding Voters Cases

We will show that the search versions of constructive and destructive control by adding voters polynomial-time Turing reduce to the decision versions by providing appropriate reductions. We will first consider the constructive case, as the destructive version of the algorithm can be easily adapted from it. The algorithm will proceed iteratively, making use of the self-reducibility of this problem. That is, an instance of control by adding voters can be successful if and only if it can be successful either with a particular voter added, or without that voter added.

Proof. Theorem 6.2.1, control by adding voters cases.

The algorithm will proceed as follows. The input will be a candidate set C, sets of voters V and W, a distinguished candidate $p \in C$, and an adding limit K (a nonnegative integer).

First, we will query to our CC-AV (constructive control by adding voters) subroutine the input instance. If the answer is "No," we will return a fixed value denoting failure. Otherwise, we know a successful control action exists and we will proceed to find it.

We will initialize our added voter set S to $\emptyset$, and then proceed by iterating as long as W is nonempty and $K > 0$. Let w be some arbitrary voter from W. We will make a call to the decision problem for the instance $(C, V \cup \{w\}, W - \{w\}, p, K - 1)$. That is, we are asking if we can make p the winner while adding w and no more than $K - 1$ other voters. If so, we remove w from W, add w to both V and to S, subtract one from K, and move on to the next iteration. If not, we instead just remove w from W and move on to the next iteration.] If we do not have a failed instance (and note that failure will not occur if it does not occur upon the initial query) we proceed until we either have exhausted all our voters from W or the parameter K is equal to zero. Note that if we have a positive instance of this case of

control (in the constructive case) with either an empty W or with $K = 0$, then p must be a winner of the election (C, V), as there is no more possibility to add voters. We can see that we can easily reduce from the winner problem for an election system to the decision version of this problem.

So at this point, our set S will contain all the added voters and p will win the election $(C, V \cup S)$ (using the original value for V). Thus S is a set of voters such that $||S|| \leq K$ and p is a winner of $(C, V \cup S)$. Thus we can return S as our successful search action.

This algorithm, making use of a unit-cost subroutine for the decision version of CC-AV, will easily run in polynomial time, will output a successful manipulative action when one exists, and will otherwise signify failure. Thus this algorithm provides a polynomial-time Turing reduction from the decision version of CC-AV to the search version of that problem, for any voting system.

The algorithm for the constructive case can be reused almost exactly for the destructive case, with the single change that we must make a call to a subroutine for the destructive version of this problem rather than the constructive version. □

6.2.2 Theorem 6.2.1, Control by Deleting Voters Cases

For these cases, again we will exploit a sort of self-reducibility of the control by deleting voters problem to build a simple iterative algorithm that uses the decision version of the problem to extract a successful control action if one exists. In particular, an instance of control by deleting voters can be successful if and only if it can be successful either when deleting a particular voter or when not deleting that particular voter. As before, we will handle the constructive case first and then describe how to adapt the reduction we give to the destructive case.

Proof. Theorem 6.2.1, control by deleting voters cases.

We will now give the reduction. The input to the algorithm will be an election (C, V), a distinguished candidate $p \in C$, and a nonnegative integer K. We will first query the input instance to our CC-DV (constructive control by deleting voters) decision problem subroutine. If the answer is "No," we will return a fixed value denoting failure. Otherwise, we know a successful control action exists and we will continue.

We will initialize a set S to $\emptyset$ in order to keep track of the set of voters to delete. We will then proceed to iterate over the set of voters V and build our set of deleted voters. For each voter $v \in V$, and as long as $K > 0$, we query to our CC-DV decision problem the instance $((C, V - \{v\}), p, K - 1)$. That is, we ask if we can make p a winner by deleting v and up to $K - 1$ other voters. If this is the case, we remove v from V, add it to S, subtract

one from K, and move on to the next iteration. If not, there must instead be a successful control action without v deleted, so we just move on to the next iteration.

We proceed in this way until we go through the entire voter set, or until K is zero. If we did not return failure at the start of the algorithm, we must then produce an S such that $||S|| \leq K$ and p is a winner of $(C, V - S)$ (with our original value for V). Thus we return a successful control action if one exists, and this algorithm will clearly run in polynomial time. Thus we have shown that search polynomial-time Turing reduces to decision for CC-DV.

As with the adding voters cases, we can provide an algorithm for the destructive version of this case almost exactly as for the constructive version of this control problem with the single change that we call a subroutine for the destructive version of the problem instead. □

6.2.3 Theorem 6.2.1, Control by Adding Candidates Cases

This case is almost the exact mirror of the cases for control by adding voters and we base the algorithm on a very similar self-reduction, except that we are now adding to the candidate set instead of the voter set. Again we will consider the constructive version first.

Proof. Theorem 6.2.1, control by adding candidates cases.

We will now give the reduction from the search version of CC-AC (constructive control by adding candidates) to the decision version of that problem. The input to our algorithm will be disjoint candidate sets C and A, a voter set V with preferences over C and A, a distinguished candidate $p \in C$, and a nonnegative integer K, our adding limit.

Initially we will query the input to our subroutine for the decision version of CC-AC. If the answer is "No," we will return a fixed value denoting failure. Otherwise, we know a successful control action exists and we will go on to find it.

First we must initialize our added set S to be $\emptyset$. We will then iterate as long as A is not empty and $K > 0$. First let a be some element of A. We will use the decision problem to determine whether we should add a or not. We will query to the decision problem the instance $(C \cup \{a\}, A - \{a\}, V, p, K - 1)$. That is, we are asking if we can make p the winner by adding a and up to $K - 1$ other candidates. If this comes back positive, we add a to S and to C, and remove it from A. We also subtract one from K. If it comes back negative, we remove a from A and move on to the next iteration, as in this case it must be possible to make p win while not adding a.

We continue until either $K = 0$ or $A = \emptyset$. At this point, there is no more allowance for adding candidates and changing the election, so if we did not hit failure, p will clearly win the election $(C \cup S, V)$ (with the original value for C). So we will return S, and S will be a successful control action. In the cases where control is not possible we will indicate failure after the first query, and this algorithm clearly runs in polynomial time. Thus we have

shown that search polynomial-time Turing-reduces to decision for CC-AC for any voting system.

As with the previously handled voter control cases, for the destructive version, we can exploit the same self-reducing structure as with the constructive version of this control problem and use essentially the same algorithm, but just substitute the appropriate destructive version for the subroutine call. □

6.2.4 Theorem 6.2.1, Control by Deleting Candidates Cases

This case is closely parallel to the proof for control by deleting voters and we will use a similar self-reductive structure to build this algorithm, except of course modifying the candidate set throughout instead of the voter set. As before we will first deal with the constructive case.

Proof. Theorem 6.2.1, control by deleting candidates cases.

The algorithm will proceed as follows. We will take as input an election (C, V), a distinguished candidate $p \in C$, and a nonnegative integer K, our deletion limit. We will first query our subroutine for the decision version of CC-DC (constructive control by deleting candidates) with the input instance. If the answer is "No," we will return a fixed value denoting failure. Otherwise, we know a successful control action exists and we will proceed to find it.

We again initialize a set S to $\emptyset$ to keep track of our deleted candidates. We will iterate as long as $K > 0$ through each $c \in C$, except for p, and do the following. We will query to the decision problem the instance $((C - \{c\}, V), p, K - 1)$. That is, we are asking if p can be made a winner with c deleted and with up to $K - 1$ other candidates from C deleted. If so, we remove c from C, add c to S, and subtract one from K. If not, we know that there must be some successful control action without c deleted, so we just move on to the next iteration.

We continue until either we have iterated through the entire set C (except for p) or $K = 0$. At this point p will be a winner of the election $(C - S, V)$ with S such that $||S|| \leq K$ (and with our original C). We thus return S as the successful control action. If there is no successful control action, we would have indicated failure after the initial query. Also, this algorithm clearly runs in polynomial time, so we have that search polynomial-time Turing-reduces to decision for CC-DC for any voting system.

Again we can reuse this algorithm for the destructive version of this problem, only changing the subroutine call to use the destructive version of the decision problem. □

6.2.5 Theorem 6.2.1, Control by Unlimited Adding of Candidates Cases

The case differs slightly from the others in that we do not have a parameter limiting the number of candidates that can be added, and so we will always have to iterate through the entire additional candidate set before generating the control action, but otherwise it follows essentially the same self-reducing structure as the previous proofs. Again, we will first handle the constructive case.

Proof. Theorem 6.2.1, control by unlimited adding of candidates cases.

Our algorithm will proceed as follows. We are given as input disjoint sets of candidates C and A, a voter set V with preferences over C and A, and a preferred candidate $p \in C$. We will first query the input instance to our CC-ACU (constructive control by unlimited adding of candidates) decision problem subroutine. If the answer is "No," we will return a fixed value denoting failure. Otherwise, we know a successful control action exists and we will continue.

As before we will use a set S, initialized to $\emptyset$, to keep track of the candidates we add. We iterate as long as A is not empty. Let a be some element of A. We query to the decision problem the instance $(C \cup \{a\}, A - \{a\}, V, p)$. That is, we are asking if we can make p the winner by adding a and possibly some other subset of the candidates from A to the election. If so, we add a to C and to S, remove it from A, and go on to the next iteration. If not, we remove a from A and go on to the next iteration, as there must be some way to make p win when not adding a.

We continue until $A = \emptyset$ and then return S. At this point, if we did not indicate failure (which again can only happen following the initial query), p will be a winner of the election $(C \cup S, V)$ (using the original value for C) and S will give us the successful control action we sought. Otherwise we quickly detect and indicate when control cannot succeed, and the whole algorithm clearly runs in polynomial time. Thus we have shown that search polynomial-time Turing-reduces to decision for CC-ACU for any voting system.

Again, we only must change the subroutine call from the previous proof to use the destructive version the decision problem to obtain a polynomial-time function for the destructive version of this problem. □

6.2.6 Theorem 6.2.1, Destructive Control by Partition of Candidates Cases

The control by partition of candidates cases are obviously different in feel compared to the previously handled cases of control by adding or deleting candidates or voters. The key

insight behind these self reductions is that to successfully achieve destructive control by partition of candidates, we have to make sure that the distinguished candidate loses either an initial election if they take part in one, or the final election if they make it that far. But in fact if there is any subset of the candidates among which the distinguished candidate will not win, we can simply put that set of candidates together with p in an initial election, thus eliminating p. There is never a need to allow the distinguished candidate to make it to the final election, and in the ties-eliminate cases, the first-round elections are more selective than the final election, with only a unique winner staying in the race, so it is easier to eliminate the distinguished candidate there. Consequently in each of the ties-eliminate and ties-promote models, the destructive control by partition of candidates and the destructive control by run-off partition of candidates problems are in fact the same sets. DC-PC-TP and DC-RPC-TP turn out to be equivalent to the problem "Is there any subset of the candidates among which p is not a winner?" and DC-PC-TE and DC-RPC-TE are equivalent to the problem "Is there any subset of the candidates among which p is not a unique winner?". Thus we only need two proofs to handle these four cases, and these two proofs are essentially the same as well.

Proof. Theorem 6.2.1, DC-PC-TP, DC-PC-TE, DC-RPC-TP, and DC-RPC-TE cases.

We will first handle DC-PC-TP and then describe the modifications necessary to cover the other cases. The algorithm will proceed as follows. Given our instance $((C, V), p)$, we first query our subroutine for the decision problem with the original instance. If the answer is "No," we return a fixed value denoting failure. Otherwise we proceed.

We will initialize a set S to $\{p\}$ to keep track of the first partition. We will then do the following for every candidate $c \in C - \{p\}$. We query to our decision problem subroutine the instance $((C - \{c\}, V), p)$. Keeping in mind our previous alternate characterization of this problem, we are asking if p will lose an election among any subset of $C - \{c\}$. If this instance is positive, we know that c is not an essential participant in the subset of candidates that will beat p, and so we will remove them from C. Otherwise, we know that c must in fact be in the subset that will beat p, and so we keep them in C and add them to S. After iterating through all elements of $C - \{p\}$, S will contain p along with a set of candidates that will cause p to not be a winner. Thus we can return $(S, C - S)$ as the partition that will cause p to lose.

We do not need to make any significant modifications to handle the other similar cases other than querying the appropriate subroutines. For all of these cases, we can use a unit-cost subroutine for the decision version of the problem to build a polynomial-time algorithm for the search version, and so for all of the cases search polynomial-time Turing reduces to decision for any voting system. □

6.3 Cases Where Search Separates From Decision

Theorem 6.3.1. *If* $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$*, then for each manipulative action* $\mathcal{A}$ *belonging to the following list:*

1. *constructive manipulation,*
2. *destructive manipulation,*
3. *constructive bribery,*
4. *destructive bribery,*
5. *constructive control by partition of voters, ties promote,*
6. *destructive control by partition of voters, ties promote,*
7. *constructive control by partition of voters, ties eliminate,*
8. *destructive control by partition of voters, ties eliminate,*
9. *constructive control by partition of candidates, ties promote,*
10. *constructive control by partition of candidates, ties eliminate,*
11. *constructive control by run-off partition of candidates, ties promote, and*
12. *constructive control by run-off partition of candidates, ties eliminate,*

there exists an election system $\mathcal{E}$ *(which may differ based on* $\mathcal{A}$*), whose winner problem is in polynomial time, such that the* $\mathcal{A}$*-decision problem for* $\mathcal{E}$ *is in* P *but the* $\mathcal{A}$*-search problem for* $\mathcal{E}$ *is not polynomial-time computable.*

Corollary 6.3.2. *If* $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$*, then for each of the manipulative actions* $\mathcal{A}$ *listed in Theorem 6.3.1,* $\mathcal{A}$*-search for* $\mathcal{E}$ *does not polynomial-time Turing reduce to* $\mathcal{A}$*-decision for* $\mathcal{E}$*.*

We will now formally give the Borodin-Demers Theorem, which is central to our results.

Theorem 6.3.3 (The Borodin-Demers Theorem [BD76])**.** *If* $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$ *then there is a set* B *so*

1. $B \in \mathrm{P}$*,*
2. $B \subseteq \mathrm{SAT}$*, and*
3. *no* P *machine can find solutions for all formulas in* B*. That is, for* no *polynomial-time computable function* g *do we have* $(\forall f)[f \in B \Rightarrow g(f)$ *is a satisfying assignment of* $f]$*.*

All of these proofs will have a similar structure. We start by assuming that P $\neq$ NP $\cap$ coNP. Then we fix a set B that fulfills the conditions of the Borodin-Demers Theorem, i.e., B is a set that is in P, is a subset of SAT, and for which there is no polynomial-time function that for every formula in B finds a satisfying assignment for that formula. We will then define a voting system using B, show that its winner problem is in P, show that the decision problem for the relevant manipulative action is in P, and then show that there can be no polynomial-time function for the search problem for the manipulative action. This last task will be accomplished through showing that we can in polynomial time encode a formula from our Borodin-Demers set B into the search problem of the relevant manipulative action, in such a way that a solution to this search problem gives a satisfying assignment for the formula.

6.3.1 Theorem 6.3.1, Constructive Manipulation Case

This proof is likely the simplest of this subsection and closely follows the outline above. We construct an election system where there are no winners unless the candidates in the election specify a formula from our Borodin-Demers set B and a voter specifies a satisfying assignment for this formula. This makes the decision version of the manipulation problem trivial, but it makes finding a successful manipulation equivalent to finding a satisfying assignment for a formula from B.

Proof. Theorem 6.3.1, constructive manipulation case.

Assume that P $\neq$ NP$\cap$coNP and let B be a set that fulfills the conditions of the Borodin-Demers Theorem. We assume that our encoding of SAT is such that for each string $x \in$ SAT, the number of distinct variables in formula x is at most $\max(0, \lfloor \frac{|x|}{2} \rfloor - 1)$.

In our voting system, the candidate names specify a string, puzzle(C), as follows: Sort the candidate names lexicographically and then in order take the low-order bit from each name other than ϵ and concatenate those. Each voter will also specify a string, which we will use as an assignment to the variables of puzzle(C), as follows. Recall that each vote is given as a tie-free linear ordering of the candidates. Sort the candidates of the election in lexicographic order. Each candidate with odd rank in this sorting will be assigned bit value 1, and each candidate with even rank will be assigned bit value 0. So for example if C = {"Alice", "Bob", "Carol", "David"}, "Alice" and "Carol" are 1's and "Bob" and "David" are 0's. Let $\text{bit}_C(c)$ be the bit value of candidate c among the set of candidates C. In a vote $c_1 < c_2 < \cdots < c_k$, the bitstring we associate with the vote is $\text{bit}_C(c_1) \cdot \text{bit}_C(c_2) \cdot \cdots \cdot \text{bit}_C(c_{\lfloor \frac{k}{2} \rfloor - 1})$. Note that this allows us to have a voter put his or her favorite candidate as his or her top choice, regardless of the bitstring the voter is going to represent and whether that candidate is a 0 or a 1, since we have enough 0's and 1's to form any string

in $\{0,1\}^{\max(0,\lfloor \frac{k}{2} \rfloor -1)}$ and place any candidate out of the encoding region in first place.

Now we will specify the actual election system, which we will call $\mathcal{E}_1$. On any input (C,V) we first compute puzzle(C). If puzzle$(C) \notin B$, then every candidate loses. If puzzle$(C) \in B$, we examine every voter $v_i \in V$ and compute the bitstring b_i of that voter as specified. Note that this bitstring has $\max(0, \lfloor \frac{||C||}{2} \rfloor -1)$ bits. By our assumption about the encoding of SAT (and noting that $|\text{puzzle}(C)| \leq ||C||$), this will give us enough bits to specify an assignment to the variables of puzzle(C). If the first d bits (where puzzle(C) has d distinct variables) of b_i form a satisfying assignment for puzzle(C), then the top choice of v_i will be a winner.[1]

We do this for all v_i, so the complete winner set is the set of all candidates who are the top choice of at least one voter that has a solution to the puzzle. This completes our specification of $\mathcal{E}_1$.

Now we have several things to show about this voting system to complete our proof.

First of all, the winner problem for $\mathcal{E}_1$ is in P. Why? $B \in$ P and puzzle(C) is easy to build from C, so detecting whether puzzle(C) is in B is easy. In the case that puzzle$(C) \notin B$, no one wins, and we are done. In the case that puzzle$(C) \in B$, we have to compute and check the solution attempt of each voter. However, this is easy as well since computing each b_i is easy and testing whether an assignment satisfies a formula can easily be done in polynomial time (we can even do it in LOGSPACE, and with the right encoding even in ALOGTIME [Bus87]). Then we just need to toss the top choice of each successful vote into the winner set and we are done.

Next, the manipulation decision problem for $\mathcal{E}_1$ is in P. Why? If puzzle$(C) \notin B$, then no one ever wins so manipulation cannot succeed. If puzzle$(C) \in B$ and there is at least one manipulator whose vote we can change, there will exist a successful manipulation to make any candidate p a winner. Namely, set that manipulator to a vote that gives a correct answer to the puzzle in its lower-order bits and has p as its top choice. If puzzle$(C) \in B$ but we cannot change any votes, the only possible manipulation is the empty one. Thus p can be made a winner if and only if it is already a winner of the election, and we can determine this as the winner problem is in P. So the problem of determining whether manipulation is possible is easy.

Finally, the manipulation search problem for $\mathcal{E}_1$ is not in FP. Why? If it were in FP B would violate its conditions. In particular, given a string $F \in B$, we could build a candidate set $\widehat{C}$ encoding F as its puzzle. We will have just one voter, who will be manipulative. Let p be any arbitrary candidate in C. An algorithm for the manipulation search problem would give us a vote that specifies a satisfying assignment for F. Thus we could use a polynomial-time function for the manipulation search problem to build a polynomial-time

[1] We could also have defined every candidate to be a winner in this case, though this gives a less natural election system.

function that for every formula in B finds a satisfying assignment for that formula. Such a function cannot exist by the third item of the Borodin-Demers Theorem, so the constructive manipulation search problem must not be in FP for $\mathcal{E}_1$. □

6.3.2 Theorem 6.3.1, Destructive Manipulation Case

This case will heavily draw from our previous proof for the constructive manipulation case, but we will need to construct a slightly altered voting system such that voters must "solve the puzzle" in order to prevent a candidate from winning, rather than to cause a candidate to win.

Proof. Theorem 6.3.1, destructive manipulation case.

Again we assume that P $\neq$ NP$\cap$coNP, we let B be a set that fulfills the conditions of the Borodin-Demers Theorem, and we assume that our encoding of SAT is such that for each string $x \in$ SAT, the number of distinct variables in formula x is at most $\max(0, \lfloor \frac{|x|}{2} \rfloor - 1)$. We will now describe the voting system $\mathcal{E}_2$ that will show the separation of search and decision for destructive manipulation.

In $\mathcal{E}_2$ we will interpret the set of candidates as specifying a "puzzle" in the same way as with $\mathcal{E}_1$, and we will use the same technique for extracting a bitstring from each vote as well. $\mathcal{E}_2$ will then operate as follows. First we must compute puzzle(C) from the candidate set C. If puzzle(C) $\notin B$, then no candidate wins. In the case that puzzle(C) $\in B$, we extract the bitstring b_i from each voter v_i and check if any of them give a satisfying assignment for puzzle(C). If so, then there are no winners, but if not, every candidate is a winner. This ends our specification of $\mathcal{E}_2$.

Again we have several things to show about $\mathcal{E}_2$. First, we show that the winner problem is in P. As before, computing the puzzle and checking for membership in B are both in P, so in the case that puzzle(C) $\notin B$ we find the (empty) set of winners easily. If puzzle(C) $\in B$, we also have to extract the bitstring from each vote and check particular assignments to puzzle(C), but again this is easily doable in polynomial time. Thus the winner problem is easily in P.

We also need that the destructive manipulation decision problem is in P. In any case where puzzle(C) $\notin B$, there are no winners, so we have a positive instance. If puzzle(C) $\in B$ and the instance allows us to change the vote of a manipulator, then we have a positive instance, as setting the manipulator vote to specify a satisfying assignment for puzzle(C) will cause no candidate to win the election, including the hated candidate. If we do not have the ability to assign a manipulator vote, we can determine if p will win using the winner problem for $\mathcal{E}_2$, which we already saw was in P. We can easily perform any of these steps,

including computing the puzzle, checking for membership in B, and checking assignments to a boolean formula, so the destructive manipulation decision problem is in P for $\mathcal{E}_2$.

Finally we need that the destructive manipulation search problem is not in FP for $\mathcal{E}_2$. We will again show that the existence of a polynomial-time search algorithm for destructive manipulation leads to the existence of a polynomial-time algorithm that for every formula in B finds a satisfying assignment for that formula. This can proceed exactly as with constructive manipulation. Let F be a string in B. We will then build a candidate set $\widehat{C}$ encoding F, we will have just one voter, who will be manipulative, and we let p be any arbitrary candidate in $\widehat{C}$. Finding a successful manipulation to make p not win will give us a satisfying assignment for F. So we have a polynomial-time function that for every formula in B finds a satisfying assignment for that formula. But this cannot exist by the third item of the Borodin-Demers Theorem, so the destructive manipulation search problem must not be in FP for $\mathcal{E}_2$. □

6.3.3 Theorem 6.3.1, Constructive Bribery Case

For this case we can reuse the voting system $\mathcal{E}_1$ from the constructive manipulation case and only slightly change the reduction from the Borodin-Demers set search problem so as to correspond with bribery rather than manipulation. Since we are using the same voting system we do not have to again show that the winner problem is in P but we will pick up from there.

Proof. Theorem 6.3.1, constructive bribery case.

We are dealing with a different manipulative action, however, bribery instead of manipulation, so we will have to show that the constructive bribery decision problem is in P. We can easily compute puzzle(C) and check for membership in B, and if puzzle(C) $\notin B$, then there can be no winners and bribery will not succeed. On the other hand, if puzzle(C) $\in B$, if there is at least one vote, and if we can bribe at least one voter, then this will be a positive instance, as bribing that voter to give a vote encoding a satisfying assignment together with the preferred candidate p at the top of the voter's list will cause p to be a winner. If there are not any voters or if we cannot bribe any voters, then we cannot alter the election, so we can check whether this is a positive instance simply with the winner problem which we have already seen is in P.

Now we will show that a polynomial-time function for the search version of this problem would lead to a polynomial-time function for finding satisfying assignments for the elements of B. Let F be an element of B. We will build a candidate set $\widehat{C}$ encoding F as its puzzle. We will then let the voter set V contain a single voter with an arbitrary initial preference over the candidates and allow the bribery of a single voter. We run our hypothetical polynomial-

time algorithm for the bribery search problem on this instance to find a vote that makes some candidate $p \in \widehat{C}$ win (if no bribery is needed, the initial preference will work). This vote must specify a satisfying assignment for F, and so we have a polynomial-time algorithm that for every formula in B finds a satisfying assignment for that formula. Such a function cannot exist by the third item of the Borodin-Demers Condition, so the constructive bribery search problem must not be in FP for $\mathcal{E}_1$. □

6.3.4 Theorem 6.3.1, Destructive Bribery Case

Here we can easily adapt our proof for destructive manipulation and use the same voting system $\mathcal{E}_2$ but just change the reduction from the Borodin-Demers set search problem to correspond with bribery, as in the previous proof. Again we have already shown that the winner problem for $\mathcal{E}_2$ is in P.

Proof. Theorem 6.3.1, destructive bribery case.

We will show now that the decision problem destructive bribery is in P. As before we can easily compute the puzzle from the candidate set and test membership in B. If puzzle$(C) \notin B$, there can be no winners so the destructive bribery instance will always succeed. If puzzle$(C) \in B$, there is at least one voter, and we can bribe at least one voter, then we could make the hated candidate lose by changing a vote to specify a satisfying assignment for B. If there are not any voters or if we cannot bribe any voters, then we can not alter the election, so we can check whether this is a positive instance simply with the winner problem which we have already seen is in P.

We will now show that there cannot be a polynomial-time function for the destructive bribery search problem. This can proceed just as the constructive bribery case. Let F be an element of B. We will build a candidate set $\widehat{C}$ encoding F as its puzzle, include a single voter with an arbitrary initial preference over the candidates, and allow bribery of a single voter. The hated candidate p can be any arbitrary candidate in the election. By running a hypothetical polynomial-time function for the destructive bribery search problem, we would acquire a changed vote for the single voter that would specify a satisfying assignment for F (if no bribery is needed, the initial preference will work) and cause the election to have no winners. Therefore this could also be used to develop a polynomial-time algorithm that for every formula in B finds a satisfying assignment for that formula. By the third item in the Borodin-Demers Condition, this cannot exist, so the destructive bribery search problem must not be in FP for $\mathcal{E}_2$. □

6.3.5 Theorem 6.3.1, Constructive Control by Partition of Voters (CC-PV) Case

Now we move into the control by partition cases, which are arguably harder and have a much different feel compared to the previous cases of manipulation and bribery. Now we need to build a voting system where the partition of the voters into two separate first-round subelections corresponds to an attempted solution to the "puzzle" of a Borodin-Demers formula. As in the previous cases, the candidate names specify the puzzle. Unlike the previous cases, we will use the set of voters to specify the assignment.

Proof. Theorem 6.3.1, constructive control by partition of voters case.

We assume that P $\neq$ NP $\cap$ coNP, we let B be a set that fulfills the conditions of the Borodin-Demers Theorem, and we assume without loss of generality that every element of B contains at least two distinct variables. We assume that our encoding of SAT is such that for each string $x \in$ SAT, the number of distinct variables in x is at most $\lfloor \frac{|x|}{2} \rfloor$.

For this case, we use the same scheme as in $\mathcal{E}_1$ for computing puzzle(C) from the candidate set. Suppose puzzle(C) is a boolean formula with d variables. Let $c_1, c_2, \ldots, c_{2d}$ be the first $2d$ candidates in the lexicographic order. This many candidates exist by our assumption on the encoding of SAT. Let V be a set of voters. Consider the set of candidates that are the top choice of some voter in V. If that set is of the form $\{c_{1+\alpha_1}, c_{3+\alpha_2} \ldots, c_{2d-1+\alpha_d}\}$ with each $\alpha_i \in \{0, 1\}$, we will interpret $\alpha_1 \cdot \alpha_2 \cdot \cdots \cdot \alpha_d$ as an assignment to the variables of puzzle(C).

We will now describe our election system $\mathcal{E}_3$. Given an election (C, V):

- If $||C|| \leq 2$, the lexicographically last candidate is the only winner.
- Else if $V = \emptyset$ or there exist two distinct voters in V that have the same candidate as their top choice, the lexicographically first candidate is the only winner.
- Else if puzzle(C) $\in B$ and V specifies an assignment for puzzle(C) as described above and this assignment is either a satisfying assignment or the complement of a satisfying assignment (that is, it is a satisfying assignment if we flip each 0 to a 1 and each 1 to a 0) for puzzle(C), then the lexicographically first candidate is the only winner.
- Else the lexicographically last candidate is the only winner.

This completes our description of $\mathcal{E}_3$.

Since this voting system always has exactly one winner in a given election, it will have the exact same behavior for control by partition of voters in the ties promote and in the ties eliminate models. Thus we can prove our desired result for both of these cases simultaneously.

Now we again have several things to show about $\mathcal{E}_3$. First we need to show that its winner problem is in P. This is clear from the description of the voting system. Only in the third case are more than trivial checks performed, and still we must only compute the puzzle from the candidates and the variable assignment from the voters, check for membership in B, and check at most two assignments for a boolean formula, all of which can easily be performed in polynomial time.

We next need to show that the CC-PV decision problem is in P for $\mathcal{E}_3$. If there are at most two candidates, the lexicographically last candidate will be the only winner of both subelections (regardless of the voter partition) and then will be the only winner in the final election. So control in this case is possible if and only if the designated candidate is the lexicographically last. We claim that constructive control by partition of voters on input $((C, V), p)$ for $||C|| > 2$ is possible if and only if:

- p is the lexicographically last candidate and $V \neq \emptyset$, or
- p is the lexicographically first candidate and we are in one of the following, easily identifiable, four cases:
  - $V = \emptyset$, or
  - there exist two distinct voters in V that have the same candidate as their top choice, or
  - puzzle$(C) \in B$ and V specifies a satisfying assignment for puzzle(C) or the complement of a satisfying assignment for puzzle(C), or
  - puzzle$(C) \in B$ and the set of candidates that are the top choice of some voter in V is the set of the first $2d$ candidates in the lexicographic order, where d is the number of variables in puzzle(C).

It is immediate that control is only possible if p is the lexicographically first or the lexicographically last candidate. First assume that p is the lexicographically last candidate. If $V = \emptyset$, the lexicographically first candidate will always be the only winner, and control is not possible. If $V \neq \emptyset$, then let v be some element from V. The lexicographically last candidate is the only winner of $(C, \{v\})$, since if puzzle$(C) \in B$, puzzle(C) has at least two variables, and so v will not specify an assignment for puzzle(C). It follows that partition $(\{v\}, V - \{v\})$ makes p the winner.

For the remainder of the proof, assume that p is the lexicographically first candidate. We will first show that control is possible in all four cases listed above. If $V = \emptyset$, the lexicographically first candidate will always be the only winner. If there exist two distinct voters in V that have the same candidate as their top choice, then the partition $(V, \emptyset)$ makes p the winner. If puzzle$(C) \in B$ and V specifies a satisfying assignment for puzzle(C) or the

complement of a satisfying assignment for puzzle(C), then the partition $(V, \emptyset)$ makes p the winner. If puzzle(C) $\in B$ and the set of candidates that are the top choice of some voter in V is the set of the first $2d$ candidates in the lexicographic order, where d is the number of variables in puzzle(C), then we can partition V into V_1 and V_2 such that V_1 specifies a satisfying assignment for puzzle(C) and V_2 specifies the complement of this satisfying assignment. p will be the only winner of both subelections, and wins the final election.

It remains to show that if we are not in one of the four cases listed above, control is not possible. Suppose for a contradiction that control is possible and that this control is witnessed by partition (V_1, V_2). Because of the definition of election system $\mathcal{E}_3$, this implies that p is the winner of (C, V_1) and the winner of (C, V_2). It follows that puzzle(C) $\in B$ and that V_1 and V_2 specify a satisfying assignment or the complement of a satisfying assignment for puzzle(C). Since the top choice of each voter in V is different, it follows that V_1 and V_2 specify complementary assignments for puzzle(C). This implies that the set of candidates that are the top choice of some voter in V is the set of the first $2d$ candidates in the lexicographic order, where d is the number of variables in puzzle(C), which contradicts our assumption that we are not in one of the four cases listed above.

Finally we need to show that the CC-PV search problem is not in FP. Let F be a formula from B. We will construct a CC-PV instance that will allow us to find a satisfying assignment for F in polynomial time using a polynomial-time algorithm for the CC-PV search problem. We will build a candidate set $\widehat{C}$ encoding F as its puzzle. The voter set will consist of $2d$ voters, where $d \geq 2$ is the number of distinct variables in F, and the top choice of the ith voter will be the lexicographically ith candidate. The preferred candidate p will be the lexicographically first candidate. Now, we can query this instance to a hypothetical algorithm for the CC-PV search problem. We know it will succeed, as this is a positive instance of CC-PV, as explained above. Furthermore, the partition that the algorithm finds will necessarily give a satisfying assignment for F. We will have to arbitrarily pick one side of the partition and check whether the specified assignment or its complement is a satisfying assignment, but this can be done easily in polynomial time. Thus the existence of a polynomial-time algorithm for the CC-PV search problem leads to the existence of a polynomial-time algorithm that for every formula in B finds a satisfying assignment for that formula. Such an algorithm cannot exist, so the CC-PV search problem must not be in FP for $\mathcal{E}_3$. □

6.3.6 Theorem 6.3.1, Destructive Control by Partition of Voters (DC-PV) Case

In this case we can reuse the voting system $\mathcal{E}_3$ and many of the arguments from the constructive case.

Proof. Theorem 6.3.1, destructive control by partition of voters case.

We have already shown that the winner problem is in P. To show that the DC-PV decision problem is in P for $\mathcal{E}_3$, it suffices to observe that since every $\mathcal{E}_3$ election has exactly one winner, destructive control by partition of voters on input $((C, V), p)$ is possible if and only if there exists a candidate $q \in C - \{p\}$ such that constructive control by partition of voters on input $((C, V), p)$ is possible. Finally, to show that the DC-PV search problem is not in FP, we encode $F \in B$ in the same way as in the constructive case. Let p be the lexicographically last candidate. As in the constructive case, a successful control action is possible, and we can easily compute a satisfying assignment for F from a partition witnessing this success. □

6.3.7 Theorem 6.3.1, Constructive Control by Partition of Candidates, Ties Promote (CC-PC-TP) Case

The control by partition of candidates cases of control have a strange asymmetry unlike the partition of voters cases and the run-off partition of candidates cases. This makes this proof and the next perhaps the trickiest we have here. The basic idea is the same as the others in this section, that we design an election system that encodes a Borodin-Demers puzzle in a way that we must (in some limited cases at least) partition according to a satisfying assignment in order to succeed at control. There are however a number of differences here, as now we will use the candidate names to specify both the puzzle and the solution. And in some cases we will have many winners in a given election, unlike our other partition cases that only ever had one. This limits this proof to only work for the TP model but allows us to deal with the asymmetry of the control action.

Proof. Theorem 6.3.1, constructive control by partition of candidates, ties promote case.

We assume that P $\neq$ NP $\cap$ coNP, we let B be a set that fulfills the conditions of the Borodin-Demers Theorem, and we assume that our encoding of SAT is such that for each string $x \in$ SAT, the number of distinct variables in formula x is at most $|x|$. We will now describe the voting system which we will call $\mathcal{E}_4$. As we mentioned, we will now use candidate names to specify both the puzzle and the attempted solutions. Some candidates are of the form $0puz0$, where puz is the Borodin-Demers puzzle. Other candidates are of the form $10puz010^{5k} + 2i + \alpha_i$, where k is the length of puz and "+" denotes the lexicographic

increment. This candidate would be interpreted as specifying both the puzzle and the assignment of a value of α_i to the ith variable in puz. We will now specify the action of the voting system. If $C = \{0puz0\} \cup \bigcup_{1 \leq i \leq d}\{10puz010^{5k} + 2i + \alpha_i\}$, where $puz \in B$, k is the length of puz, d is the number of variables in puz, $\alpha_i \in \{0, 1\}$, and the full string α composed of all the bits $\alpha_1, \ldots, \alpha_d$ is either a satisfying assignment for puz or the complement of one, then $0puz0$ will be the only winner. If C consists of $0puz0$ along with any subset of $\bigcup_{1 \leq i \leq d}\{10puz010^{5k} + 2i, 10puz010^{5k} + 2i + 1\}$, where $puz \in B$, k is the length of puz, d is the number of variables in puz, $\alpha_i \in \{0, 1\}$, and we are not in the previous case, then every candidate except $0puz0$ will be a winner. In every other case every candidate will be a winner. This completes our specification of $\mathcal{E}_4$. Note that unlike some of our other systems we totally ignore any and all of the voters in this system.

First, we must show that the winner problem for $\mathcal{E}_4$ is in P. This is easy, as though the specific cases are complicated in form they are simple to recognize. The additional checks of detecting whether puz is in B and checking whether α is a satisfying assignment for puz or the complement of one can also be accomplished in polynomial time.

Next we must show that the CC-PC-TP decision problem is in P for $\mathcal{E}_4$. If $C \not\subseteq \{0puz0\} \cup \bigcup_{1 \leq i \leq d}\{10puz010^{5k} + 2i, 10puz010^{5k} + 2i + 1\}$, where $puz \in B$, k is the length of puz, d is the number of variables in puz, and $\alpha_i \in \{0, 1\}$, then control will always be possible, since all candidates are winners if we put no candidates in the first-round election. So suppose that $C \subseteq \{0puz0\} \cup \bigcup_{1 \leq i \leq d}\{10puz010^{5k} + 2i, 10puz010^{5k} + 2i + 1\}$, where $puz \in B$, k is the length of puz, d is the number of variables in puz, and $\alpha_i \in \{0, 1\}$. If p is not $0puz0$, then control can easily succeed as we could just put all candidates in the first round, eliminating only $0puz0$ in the first round and letting every other candidate win in the final election. If our distinguished candidate is $0puz0$, control can only succeed in a few specific but easy to recognize situations. One case is if we have the complete set of variable assignment candidates along with $0puz0$. In this case we can make $0puz0$ win by putting it together with a satisfying assignment of variable assignment candidates in the first round. Then it alone will move on to the final round and face the complement of this satisfying assignment in the final election and come out a winner. The only other case in which $0puz0$ can be made a winner through this type of control is if the candidate set consists of $0puz0$ along with exactly a satisfying assignment (or the complement of one) of the variable assignment candidates. In this case we can just give every candidate a bye and $0puz0$ will win the final election. These cases are all easy to recognize so the decision problem is easily in P.

Finally we need to show that the CC-PC-TP search problem is not in FP. Let puz be an element from B, let k be the length of puz, and let d be the number of variables in puz. We will then construct a voter set C as $\{0puz0\} \cup \bigcup_{1 \leq i \leq d}\{10puz010^{5k} + 2i, 10puz010^{5k} + 2i + 1\}$. The voter set can remain empty as this voting system totally ignores the voters. Our

distinguished candidate p will of course be $0puz0$. The only way to make $0puz0$ a winner through this type of control is to put it in the first-round election with variable assignment candidates specifying a satisfying assignment or the complement of a satisfying assignment. Thus a polynomial-time search function for CC-PC-TP will allow us to find a satisfying assignment for puz and thus give us a polynomial-time algorithm that for every formula in B finds a satisfying assignment. Such an algorithm cannot exist, so CC-PC-TP search must not be in FP for $\mathcal{E}_4$. □

6.3.8 Theorem 6.3.1, Constructive Control by Partition of Candidates, Ties Eliminate (CC-PC-TE) Case

Unlike all our other proofs in this section, for partition of candidates we do need a separate proof for the ties-promote and ties-eliminate cases. The system $\mathcal{E}_4$ in our previous proof often selects large numbers of winners, so the behavior of that system will be different under the TP and TE models. In the system we use here, we ignore the votes completely, add several special candidates, and use candidates to specify both the puzzle and the assignment.

Proof. Theorem 6.3.1, constructive control by partition of candidates, ties eliminate case.

We will use the variable assignment candidates from the previous proof to specify the puzzle and mark the variable assignments. We also have three special candidates named a, b, and c. For concreteness, let $a = 0$, $b = 01$, and $c = 10$. We assume that $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$, we let B be a set that fulfills the conditions of the Borodin-Demers Theorem, we assume that our encoding of SAT is such that for each string $x \in \mathrm{SAT}$, the number of distinct variables in formula x is at most $|x|$, and we assume that every $x \in B$ contains at least one variable. We will now describe the behavior of voting system $\mathcal{E}_5$ on input (C, V).

- If $a \in C$ but $b, c \notin C$, winners $= \emptyset$.
- If $C = \{b\}$ or $C = \{c\}$, winners $= \emptyset$.
- If $b \in C$ but $a, c \notin C$ and at least one other candidate is in C, winners $= \{b\}$.
- If $c \in C$ but $a, b \notin C$ and at least one other candidate is in C, winners $= \{c\}$.
- If $b, c \in C$, winners $= \{b, c\}$.
- If $a, b \in C$ but $c \notin C$:

  If the other candidates in C specify $puz \in B$ and a satisfying assignment for puz, then winners $= \{a\}$.

  If the other candidates in C specify $puz \in B$ and a nonsatisfying assignment for puz, then winners $= \{a, b\}$.

Else winners $= \emptyset$.

- If $a, c \in C$ but $b \notin C$:

  If the other candidates in C specify $puz \in B$ and a satisfying assignment for puz, then winners $= \{a\}$.

  If the other candidates in C specify $puz \in B$ and a nonsatisfying assignment for puz, then winners $= \{a, c\}$.

  Else winners $= \emptyset$.

- In every other case winners $= \emptyset$.

This system is more complicated than the others we have used so far and there are a few things to note. First of all, only a, b, and c can ever win the election. No candidate wins an election where it is the only candidate. And a can only ever win if it is together with exactly one of b or c and with a variable assignment. And a can only be a unique winner if they are together with a satisfying variable assignment.

We will now show that the winner problem for $\mathcal{E}_5$ is in P. Although there are many separate cases specifying $\mathcal{E}_5$'s winner problem, most of them only require very simple checks to determine the presence of the special candidates. Only in the last two cases do we have to test of membership in our Borodin-Demers set B or deal check for boolean formula satisfaction, but those things are easily in P as well. So the winner problem as whole is easily in P.

We will show next that the CC-PC-TE decision problem is in P for $\mathcal{E}_5$. Most of the work and special cases for this proof were directed towards making sure this would be the case. The main idea is that many instances of control are positive in this voting system, but they are very simple to detect (and for most of them it is simple to produce the exact control action as well). If the distinguished candidate is either b or c and there is at least one other candidate that is not a, then control will always be possible. The only circumstances that cause b or c to not be a winner is if they are alone in the election, or if they are in a particular case alongside a. As long as there is one other candidate, we can ensure b or c is a winner by giving every candidate except a (if they are present) a bye. However if a is a candidate, we can ensure it does not interfere with b or c becoming a winner by simply putting a alone in the first-round subelection where it will be eliminated, thus protecting b or c in the final round. If the distinguished candidate is either b or c and there is not at least one other candidate that is not a, then control will never be possible.

If the distinguished candidate is a, we can succeed in a somewhat narrower set of circumstances. The perfect case is when C consists of $\{a, b, c\}$ along with the complete set of variable assignment candidates (0 and 1 for every variable) for some $puz \in B$. In this

case we put a, b, and a satisfying assignment for puz in the first round. Candidate a will thus be the only winner of the first-round election, and a will also be a winner of the final election with c and the complement of this satisfying assignment (which may or may not itself be a satisfying assignment). Another similar case is if C consists of $\{a, b, c\}$ along with the a satisfying assignment for some $puz \in B$ and an assignment for a different $puz' \in B$. Another case where control can succeed for a is if the candidate set contains a, and only one of b or c, and C contains an assignment for some $puz \in B$, by giving a bye to all candidates other than a, b, and c, and an assignment for $puz \in B$. Similarly we can make a win if C consists of a, b, and c and an assignment for some $puz \in B$, by putting c (or b) alone in the first round, thus eliminating it and ending up in a situation like the previous case. It is easy to check that these are the only cases in which control is possible. Though we have a large number of cases here, none of them are difficult to check and most simply require checking for the presence of the special candidates.

Finally we can show that CC-PC-TE search is not in FP. Let puz be an element from B. We then construct an instance of CC-PC-TE search with an empty voter set, a candidate set consisting of $\{a, b, c\}$ together with a complete set of variable assignment candidates encoding puz and having 0 and 1 assignments for every variable in puz, and let our preferred candidate be a. This follows the pattern of our "perfect" case mentioned earlier, so there must be a successful control action. The action in fact must be to partition either a and b or a and c along with assignment candidates specifying a satisfying assignment for puz into the first-round election. Thus a polynomial-time algorithm for CC-PC-TE search would give us a polynomial-time algorithm for finding satisfying assignments for elements of B as well. Such an algorithm cannot exist, so CC-PC-TE search must not be in FP for $\mathcal{E}_5$. □

6.3.9 Theorem 6.3.1, Constructive Control by Run-Off Partition of Candidates (CC-RPC) Case

This case is similar to the proofs for control by partition of voters except that since we are partitioning candidates we use the candidate names to specify both the puzzle and the assignment, and the voters are not even used.

Proof. Theorem 6.3.1, constructive control by run-off partition of candidates case.

We again have schemes for interpreting some candidate names as the Borodin-Demers puzzle and some others as an assignment to the variables of this puzzle. We distinguish between the two by padding the assignment candidate names out to be much longer than the puzzle candidate names. The puzzle candidates will simply consist of the puzzle x followed by a single bit, either a zero or a one, while the assignment candidates all will be five times the length of the puzzle candidates. Given that the puzzle x is of length k,

each of these will consist of a bitstring 0^{5k} incremented $2i$ places past to denote that the ith variable should have value 0, or incremented $2i+1$ places past to denote that the ith variable should have value 1.

We again of course assume that P $\neq$ NP $\cap$ coNP and let B be a Borodin-Demers set. Without loss of generality, we assume that every element of B contains at least one variable and that $\epsilon \notin B$. We assume that our encoding of SAT is such that for each string $x \in$ SAT, the number of distinct variables in formula x is at most $|x|$. We will now specify the behavior of the voting system $\mathcal{E}_5$ on an election (C, V).

- If the lexicographically smallest candidate is $x0$ with $x \in B$, and the remainder of the candidate set consists of d variable assignment candidates (where x contains d variables) giving exactly one value to each variable in x, and with the full set representing a satisfying assignment for x, then only $x0$ will win.
- If the lexicographically smallest candidate is $x1$ with $x \in B$, and the remainder of the candidate set consists of d variable assignment candidates (where x contains d variables) giving exactly one value to each variable in x (but with no requirement that we have a satisfying assignment), then only $x1$ will win.
- If the entire candidate set consists of two candidates $x0$ and $x1$ with $x \in B$, then only $x0$ wins the election.
- In every other case no one wins.

Note that in no case do we ever have more than a single winner, so the behavior of this system is exactly the same in the ties-promote and ties-eliminate models.

We can see that the winner problem for this voting system is easily in P. Only a few cases produce any winners at all and they are easy to check for. In the first we have to test whether the puzzle boolean formula is satisfied by a particular assignment but this is easily doable in polynomial time, and in the others we only have to test for the presence of certain candidates and test for membership in B, which is in P by definition.

The decision problem for this case of control is in P as well. First of all, no candidate other than $x0$ or $x1$ for $x \in B$ can ever win, and so any instance with any other preferred candidate will always be negative. Since we are dealing with RPC, every candidate must go through a first-round subelection and then win in the final election to win overall. And since we never have more than one winner, if a candidate is going to win overall it must win in a final election containing either one or two candidates. The only case where that may happen is in our third winner case above, where $x0$ wins among $x0$ and $x1$. So the only way to make any candidate win following RPC is to have $x0$ win one subelection, and $x1$ win the other, and to then have $x0$ win the final election. This can only happen if our entire

candidate set consists of $x0$ (with $x \in B$), $x1$, and the entire set of variable assignment candidates for x, with candidates assigning 0 or 1 to every variable in x. And in this case $x0$ can win as described by being put together with a satisfying assignment of assignment candidates in the first round and with $x1$ being put together with the other candidates in the first round, leading to $\{x0, x1\}$ as the candidates in the final election. This situation is easily detectable so this form of control is in P.

Finally, the CC-RPC search problem is not in FP in either tie handling model. Let x be an element from B and let d be the number of variables in x. We can then easily use a polynomial-time algorithm for the CC-RPC search problem to find a satisfying assignment for x. We can do this simply by building an election with $C = \{x0, x1\} \cup \bigcup_{1 \leq i \leq d} \{0^{5k} + 2i, 0^{5k} + 2i + 1\}$ and let $p = x0$. That is, we create the one election described above where CC-RPC has a chance to work. We then could use a polynomial-time search algorithm for this problem to find the successful partition, and this will yield a satisfying assignment for x. Thus a polynomial-time algorithm for the CC-RPC search problem would give us a polynomial-time algorithm that for every formula in B finds a satisfying assignment for that formula. Such an algorithm cannot exist, so the CC-RPC search problem in either tie model must not be in FP for this voting system. □

Chapter 7

Conclusions

This thesis has sought to expand what is known about the computational behavior of several popular election systems under manipulative actions. The systems studied here, normalized range voting, Schulze voting, and ranked-pairs voting, are all quite popular and have been lauded for their good conventional social choice properties [Smi00, Tid87, Sch11]. The work presented here has resolved many questions about their computational properties, aiding choosers of election systems interested in these systems.

Additionally, the work on search vs. decision has sought to reframe the standard notions of computational social choice in a way that more accurately captures when manipulative actions are hard, so that the practical importance of the results obtained is as great as possible. We hope that the work presented in this thesis, taken as a whole, provides useful results relating to important voting systems and helps to make the field of computational social choice more practically useful as well.

There are many open avenues regarding the systems we studied here. There are still several open control cases for Schulze voting. There has not yet been a thorough study of the worst-case hardness of control in ranked-pairs voting. There is still no known natural voting system that resists every case of control, and finding one would be of immense practical and theoretical interest. It would be of interest to see whether the complexities of search and decision collapse under some reasonable set of voting system properties.

The field of computational social choice remains vibrant and there is much work to be done.

Appendix A

Manipulation Can Be Hard in Tractable Voting Systems Even for Constant-Sized Coalitions

A.1 Introduction

Research in voting theory has become increasingly important due to the ubiquity of voting systems. While voting is most typically used for political or organizational elections, recently it has become increasing important as a tool in multiagent systems and distributed artificial intelligence. From recommender systems [GMHS99, PHG00] such as those seen in Netflix which makes recommendations based on user activity, to consensus mechanisms for planning in artificial intelligence [ER91] and search engine and metasearch engine design [Lif00, DKNS01], mechanisms that aggregate individual 'votes' have far-reaching applications. Many fields of study in computer science such as mechanism design and algorithmic game theory have become intertwined with research in voting theory. In this chapter, we concentrate on results in manipulation, that is, strategically changing one's vote so as to change the result of the election.

Manipulation, as opposed to other ways of influencing election results, does not change the structure of the election (as in control [BTT92] and cloning [Tid87]) or involve an external actor bribing voters to change their preferences (as in bribery [FHH09] and campaign management [EF10a, SFE11]). Manipulation does not require going outside of the election model but merely involves voters picking their optimal votes. Thus manipulation is the most immediate and frequently relevant of these problems.

The most representative subproblem of manipulation is unweighted coalitional manipulation (UCM). The model is simpler than one with weighted voters and so the focus is squarely on the voting rule itself, not on the interplay of differently weighted voters forming a coalition. As such, results in UCM are a purer test of a voting system's vulnerability to manipulation.

We will first explore the history and key results of voting theory that imply significant issues with all voting systems, and subsequently we will show how we can cope with some of these difficulties through complexity theory. In the interest of presenting this history as a self-contained section, there will be some overlap with the previous discussion of these topics.

Voting Theory

Voting has long been used as a tool for collaborative decision making, with democratic government known to have existed at least as far back as 6th century BCE in ancient Greece. For nearly as long people have studied voting in an attempt to find the best election methods and solve problems related to voting.

One milestone in the study of voting is the work of 13th century mystic Ramon Llull. Llull, a prominent Franciscan, was involved with the Catholic church and researched methods for electing church officials.

Among his extenstive body of work, encompassing at least 265 titles on subjects ranging from controversial theological viewpoints to romantic fiction, Llull's contributions to voting theory stem from three works: *Artifitium electionis personarum*, *En qual manera Natana fo eleta a abadessa* (Chaper 24 of his novel *Blaquerna*) and *De arte eleccionis*, all featuring variants of a pairwise election system we now know as Condorcet voting systems [HP01, Szp10].

In the eighteenth century, the Marquis de Condorcet developed one of the first criteria for evaluating voting systems. Condorcet's criterion is that, for a given election, if there exists a candidate that beats all other candidates in pairwise contests, then that candidate must be the winner of the election. It is a somewhat surprising result that such a candidate will not always exist due the possibility of cycles in the pairwise societal preferences. Condorcet proposed this criteria and showed that many popular voting systems do not possess the property. Those that do are called Condorcet methods, or Condorcet-consistent voting systems. Elections using Condorcet methods can be viewed as a number of pairwise majority-rule elections[1] or in the case of an election with two candidates, exactly equivalent to a majority-rule election. Hence the literature often refers to Condorcet methods as

[1] A majority rule decides on the alternative with the majority of the votes.

variants of the majority rule [Ris05].

Condorcet methods have often been contrasted with Borda voting, which takes complete preference orderings as the votes and gives points to each candidate based on the number of candidates they are ranked above in each vote. For instance a vote denoting a is preferred to b is preferred to c would give two points to a, one to b, and none to c. There are fervent arguments between the two systems, debating the importance of the Condorcet criterion [New92, Saa06, Ris05], in a rivalry dating back to the Marquis de Concorcet's criticism of Borda voting when it was first introduced [Szp10].

One persistent concern in elections is that some of the participants may be able to vote strategically and unfairly gain an advantage over honest voters. Pliny the Younger's attempts to manipulate the Roman senate circa 105 CE is possibly the earliest recorded instance of strategic behavior in elections [Szp10]. The senate, presiding over a murder trial, were divided into three blocs: those who favored acquittal, banishment, or death for the accused. The senators were more or less evenly distributed among the three positions, with the acquittal bloc (headed by Pliny) being slightly larger than the other two. The normal method of voting was similar to the current justice system in most countries: The senate would first vote on the guilt of the accused, followed by a vote on the punishment (banishment or death). Considering how the blocs were aligned, the probable outcome of the first election would be a decision of guilty, followed by banishment. To give his faction an edge, Pliny proposed the senators vote for acquittal, banishment or death in a single ternary-choice election. However, his strategy backfired. The death-penalty faction, fearing an acquittal, voted for banishment.

Pliny's story holds more than just strategy and counter-strategy: Pliny convinced the senate of the fairness of a single ternary-choice election by stating it aligned naturally with the principle of voting *qua sentitits*, or according to your true preferences. His attempt proved unsuccessful but serves as an excellent example of the problem of getting people to vote honestly. While this problem was recognized by voting theoreticians through history, it was either dismissed or attempts to solve it were limited at best. For instance, Llull documents that voters were required to give an oath to vote sincerely, and Jean-Charles de Borda famously dismissed criticism of his system's vulnerability to manipulation by saying "My scheme is only intended for honest men" [Bla58].

Later work drew from game theory to more formally model voter strategy and to analyze it's possibilities, especially in the work of Allan Gibbard [Gib73]. We will first explore the work of Kenneth Arrow, who initiated the modern study of voting theory and introduced the election model that has become the standard.

Arrow's Impossibility Theorem

Arrow's seminal work in modern social choice theory [Arr50, Arr63] originated in an attempt to formalize an aggregation function for social opinion. Aggregate mechanisms existed previously in welfare economics. Called welfare functions, they attempted to measure societal welfare for a number of alternative possibilities by aggregating the utility or welfare of individuals (measuring the impact of say, a change in fiscal policy or tax rates). These mechanisms all shared the assumption that as subjective a concept as individual utility could be compared or quantified. A breakthrough came with the Bergson-Samuelson social welfare function [Ber38] which inspired Arrow's own aggregate mechanism, also called a social welfare function[2]. Like the Bergson-Samuelson model, Arrow broke from previous economic models by considering votes (individual preferences) to be ranked preferences rather than a collection of numerical utilities over the alternatives. Thus the output of the social welfare function is a ranked ordering of the alternatives as well. Arrow argues that this is a more appropriate model for aggregate functions due to the difficulty of interpersonal comparisons of utility.

Arrow's key result, known as Arrow's Impossibility Theorem, shows that no social welfare function can satisfy all of a set of five reasonable criteria whenever there are more than two alternatives.By reasonable criteria we mean these conditions "... accord with common sense and with our intuition about fairness and the democratic process" [Szp10]. In formalizing his aggregate mechanism, Arrow laid down two postulates that directed the construction of individual preferences, and outlined the aforementioned five characteristics.

The first postulate states that for any pair of alternatives a, b, every individual will always have some opinion between them: Individuals can be indifferent between them (generally represented as aIb or bIa, denoting indifference between a and b), or prefer one alternative to the other (generally represented as aPb in the case that a is preferred to b). The second postulate is that an individual's preferences must be transitive, thus disallowing cycles in individual preference orderings. Note that this requirement is not universally held in voting theory, and intransitive preferences are sometimes considered reasonable when voters use different criteria to decide between different pairs of candidates [Hug80]. Consider the example of an individual Jeff who has to rent a car, and is willing to pay a little more for additional space. Between a compact and a midsize car, Jeff always chooses the midsize, since he has to pay just a little more for additional comfort. Similarly, between a midsize and a fullsize car, Jeff prefers a fullsize car. But between a fullsize car and a compact, Jeff finds the price difference to be too great, and chooses the compact car instead.

Arrow defines five natural and reasonable criteria for social welfare functions: unre-

[2]Differences between the two functions are elaborated on throughout Arrow's paper [Arr50].

stricted domain, monotonicity, nonimposition, independence of irrelevant alternatives and nondictatorship.

Unrestricted Domain By the two aforementioned postulates, an individual's preferences are represented as an ordering complete over the set of alternatives. Any restriction on which sets of orderings are permitted as input to the function violates the democratic nature of the mechanism and violates this criterion. Unrestricted domain would be violated in the example of elections held in a despotic state where only votes with the current ruler ranked first are allowed.

Nonimposition The second criterion states that the function should not allow inclusion or preclusion of outcomes irrespective of the preferences of the electorate. This criterion implies the social outcome should depend entirely on the set of individual preference orderings. An example of imposition could be an election in a theocracy where only candidates from the state religion are permitted to be elected.

Monotonicity The third property, monotonicity, states that an individual cannot harm a alternative's position by ranking it higher. In other words, if an aggregate preference ordering holds that alternative Lane is preferred to alternative Jeff, then, an individual cannot harm Lane's aggregate position by ranking Lane above Jeff in his or her ordering, all other individual orderings being constant. This property implies that that the aggregate decision must be responsive to and representative of the individual's preferences.

In the later version of his work, Arrow replaced monotonicity and nonimposition with the combined property of the Pareto criterion, or unanimity [Arr63]. The Pareto criterion is slightly stronger than monotonicity since it incorporates nonimposition. It states that for any two alternatives a and b, if an individual preference ordering prefers a to b with all other individual preferences indifferent between these two alternatives, then the social outcome also prefers a to b.

Independence of Irrelevant Alternatives The fourth property of Independence of Irrelevant Alternatives (IIA) implies that individual preferences for any pair of alternatives should not be influenced by other alternatives. A famous anecdote attributed to Sidney Morgenbesser [Pou08] illustrates IIA: Morgenbesser, ordering dessert in a restaurant, was informed by the waitress that the dessert choices were apple pie and blueberry pie. Morgenbesser chose apple pie. A few minutes later, the waitress returned and informed him that cherry pie was also available. "In that case," said Morgenbesser to the utter confusion of the poor waitress, "I'll have blueberry."

IIA and the possibility of strategic behavior in a voting system are mutually exclusive: The presence of one in a voting model indicates the absence of the other. While any honest preference relation between a pair of alternatives would not be influenced by extraneous alternatives, strategic behavior often requires them. Consider a Borda election between two alternatives Jeff and Mike where a certain voter prefers Mike, the stronger candidate, to Jeff. A new candidate, Lane, is introduced that said voter prefers to all others. The voter, then, instead of his true preference ordering Lane > Mike > Jeff (where Lane > Mike implies that Lane is preferred to Mike), might misrepresent his preferences as Lane > Jeff > Mike, in order to weaken Lane's strongest opponent, Mike. In other words, the introduction of Lane results in the voter switching his preferences for Jeff and Mike.

Nondictatorship The fifth property states the function should not permit an individual who is a dictator, i.e., for a given profile of individuals, the function cannot reflect any one individual's preferences, irrespective of the preferences of all others in the profile.

Arrow proved that any thusly defined acceptable social welfare function, meeting all these criteria, cannot decisively aggregate the preferences of a profile of individuals if there are more than two alternatives. This game-changing result essentially meant that any social welfare function violated the most basic thresholds for acceptability, thus any such function would have to compromise on meeting at least one of these five criteria. Arrow, in discussing this problem opined that compromising on an unrestricted domain was the only reasonable alternative [Arr50].

One of the more notable approaches to this problem actually preceded Arrow's work: In 1948 Scottish economist Duncan Black wrote about an intriguing property of societal preferences called single-peakedness [Bla48, Bla58]. Consider plotting a preference ordering with the horizontal axis representing a linear ordering of alternatives and the vertical axis representing their rank in the ordering. If the resulting curve (drawn from joining all alternative-representing points together) has a single peak (defined to be a point flanked by either lower-ranking points on both sides, or only on one side if the peak starts or ends the curve) then we can state that the preference ordering is single-peaked with respect to the linear ordering on the horizontal axis. For a given profile of preferences, if there exists at least one linear ordering such that all votes are single-peaked with reference to this linear ordering, then we pronounce the profile to be single-peaked, or admitting the property of single-peakedness.

Black showed that aggregate functions admitting single-peaked preference profiles (with respect to some linear ordering) meet all of Arrow's criteria except for unrestricted domain. We lose this criterion as a linear ordering that induces single-peakedness does not exist for

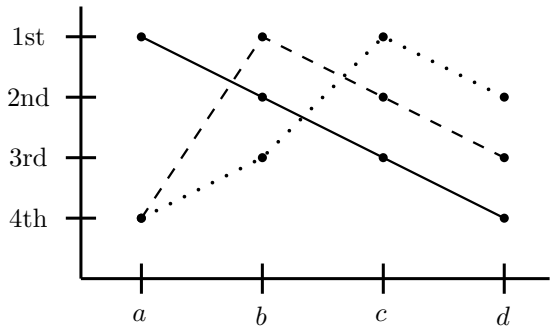

Figure A.1: A preference profile that is single peaked for the ordering *abcd*. The votes are $a > b > c > d$ (solid), $b > c > d > a$ (dashed), and $c > d > b > a$ (dotted).

every preference profile[3]. Therefore, if an aggregate function is to admit only single-peaked input, it would have to exclude certain preference orderings, restricting the domain.

Another approach by Amartya Sen [Sen69] showed the existence of aggregate mechanisms that have all of Arrow's criteria but implement a relaxation of transitivity called quasi-transitivity. This permits the existence of certain preferences that violate standard transitivity—for example, for alternatives a, b, c you can have the following preference profile: aIb, bIc, aPc. Sen also discussed an aggregate function where the input, instead of individual preference orderings, was individual utility functions.

Brams and Fishburn showed that approval voting similarly has very desirable properties when we restrict the preference domain [Nie84]. In the case that voters have dichotomous preferences (that is, they can divide the candidates into two groups: one they they approve of and one they do not) approval voting has many positive properties, including immunity to strategic voting and the Condorcet criterion [BF83]. In the general case with unrestricted preferences, the system no longer has these properties [Nie84].

The impact of Arrow's work can be summed up in a quote attributed to Paul Samuelson [Pou08], "What Kenneth Arrow proved once and for all is that there cannot possibly be ... an ideal voting scheme." The Gibbard-Satterthwaite Theorem held even more dismal news: In addition to being less-than-ideal, all voting schemes were also vulnerable to manipulation, unless they admitted dictators.

[3]The existence of a single-peaked profile would (partially) depend on the set of least-preferred alternatives across all given orderings. For example, in the case of exactly three alternatives a, b, c, a profile of two or more preference orderings having a total of two alternatives ranked last (say a, b), linear orderings *acb* and *bca* exist relative to which the profile is single-peaked. For methods to determine single-peakedness, we refer the reader to the work of Ballester and Haeringer [BH11] and Escoffier, Lang, and Öztürk [ELÖ08].

Gibbard-Satterthwaite Theorem

In the early-to-mid 1970s Alan Gibbard [Gib73] and Mark Satterthwaite [Sat75] independently extended Arrow's Theorem for voting systems in a model that incorporated strategic misrepresentation of preferences. They proved that no *strategy-proof* voting system existed for elections with three or more alternatives, unless the voting system allowed dictators. A strategy-proof voting system is one where no manipulating strategy can exist in elections using this voting system. While this result revolutionized voting theory, it had been speculated previously: In 1960 William Vickrey when discussing individuals strategically misrepresenting their preferences in Arrow's model [Vic60], stated that "it is clear that social welfare functions that satisfy the non-perversity [monotonicity] and the independence [IIA] postulates and are limited to rankings as arguments, are also immune to strategy." In addition, the Dummett-Farquharson conjecture of 1961[DF61] parallels the Gibbard-Satterthwaite Theorem.

The Gibbard-Satterthwaite Theorem transformed voting theory for two reasons: One was the aforementioned result that nondictatorial voting rules were susceptible to strategic voting (manipulation) in cases with three or more outcomes, and the second was the adoption of the arcane science of game theory from Neumann and Morgenstern's *Theory of Games and Economic Behavior*[vNM44], published just four years prior to Arrow's work. While the influence of game theory was implied (and acknowledged) in Arrow's social welfare function [Wil72], the Gibbard-Satterthwaite Theorem proofs were more explicit in their treatment of voting functions as game-theoretic mechanisms. This provided the field of voting theory with a set of tools to examine a whole range of scenarios – for example, the motivations for the electorate to vote dishonestly, or their reaction to changes in the structure of the voting rule, or attempts to form coalitions to further their individual utility.

In Gibbard's work, a voting scheme or *social choice function* is built upon a construct called a *game form*. A game form is similar to a game construct in game theory [vNM44] applied to a voting model: Players are voters, and any player's strategies are the set of all possible orderings of preferences (unless restricted for specific scenarios), over a set of alternatives or candidates. Game forms, unlike games, do not have functions assign utilities for each player for a given action (or chosen strategy) or for a scenario of chosen strategies of all the players. Instead the social choice function has the concept of an honest or sincere strategy: Among the set of strategies for each player is a specially marked strategy denoted to be the honest representation of preferences for that player. This can be used to compare outcomes (which may be a preference ordering or a single alternative) for a voter's different strategies, to see if one is less or more preferable to another in the honest ranking of alternatives.

This social choice function will be immune to manipulation only if each voter has a

dominant strategy, a strategy that will be at least as good as any other for that voter no matter what any other voter does. Otherwise, if a voter does not have a dominant strategy, then they might possibly be motivated to change their vote from their true preferences in order to obtain a better outcome. A social choice function where each voter has a dominant strategy (and hence is immune to manipulation) is said to be *straightforward*.

The proof of the Gibbard-Satterthwaite Theorem relies on the Vickrey conjecture: A voting scheme (defined with the property of unrestricted domain) is strategy-proof iff it has the properties of IIA and unanimity. Since Arrow showed that no aggregate function with more than two outcomes can satisfy all of his model's criteria, such a voting system then is necessarily a dictatorship. In other words, we have that any voting system with at least three outcomes will either be a dictatorship or it will be manipulable. A similar result, the Duggan-Schwartz Theorem [DS00], exists for voting systems that elect multiple candidates.

The Gibbard-Satterthwaite Theorem presents a problem and accepts a solution similar to Arrow's Theorem. By Black's results, voting schemes that permit only single-peaked preferences restrict the domain of the function, but can have all other Arrow criteria, including unanimity and IIA, which together imply strategy-proofness. Thus, relaxing the condition of unrestricted domain is necessary for voting schemes that resist manipulation. Another example of this is Gibbard's development of probabilistic mechanisms [Gib77, Gib78]. Procaccia took a similar approach by designing strategy-proof probabilistic voting systems that are similar to standard deterministic voting systems [Pro10].

Computational Difficulty of Manipulation

Another solution to the problem of inherent manipulability in voting was proposed by Bartholdi along with Tovey, Trick, and Orlin in a series of papers that started the field of computational social choice [BTT89a, BTT89b, BO91]. Their approach was to select voting schemes where manipulation is computationally difficult to carry out, i.e. where the manipulation problem is NP-hard. Our definition of the manipulation problem is that of constructive coalitional manipulation, i.e., does there exist a set of votes for the manipulating coalition that causes their preferred candidate to win the election? This subsumes the case of a single manipulator and contrasts with destructive manipulation, which is concerned with preventing a certain candidate from winning.

Bartholdi, Tovey, and Trick's initial work also highlighted the problems of selecting systems with complexity. Their Impracticality Theorem [BTT89b] is another instance of systems meeting seemingly reasonable attributes thereby inducing undesirable properties.

The theorem states that any *fair* voting system requires excessive computation to determine the winner, making it impractical—a highly disturbing result. This theorem followed

work by Kemeny [Kem59], Young and Levenglick [YL78] and Gardenfors [Gar76].

According to Bartholdi, Tovey, and Trick, a voting system is *fair* if it has the Condorcet criterion, the property of *neutrality* (i.e., it is symmetric in its treatment of candidates, a property implied by IIA [GPP09]), and the property of *consistency* (i.e., if disjoint subsets of the voters voting separately arrive at the same preference ordering, then voting together always produces this same preference ordering as well). Their theorem states that computation of the winner in any such fair voting system is NP-hard.

The above-mentioned property of consistency (also called convexity [Woo94] and separability [Smi73]) has been proved to be present in ranked voting systems (those in which a vote is a ranking or ordering of preferences) only if they happen to be scoring protocols as well (where the alternatives receive a certain number of points depending on their position in the ordering) [You75]. Scoring protocols are often incompatible with the Condorcet criterion (refer the aforementioned debate on Condorcet versus Borda) thus unsurprisingly so far we know of only one voting system, Kemeny scoring [Kem59], that meets all three conditions [YL78]. Kemeny scoring is then NP-hard, but additionally it has been shown to be complete for parallel access to NP [HSV05].

However, this does not imply other voting systems are immune to having an intractable-winner problem: Systems such as Dodgson meet only two of the three fairness conditions—that of the Condorcet criterion and neutrality but not consistency—but computation of the winner in Dodgson is known to be not only NP-hard [BTT89b] but complete for parallel access to NP [HHR97], similar to the result for Kemeny scoring.

Research focusing on tractable voting systems[4] was more promising: While Bartholdi, Tovey, and Trick gave us a greedy algorithm that finds a manipulating vote for several tractable voting systems in polynomial time [BTT89a], two voting systems—second-order Copeland and single transferable vote [BTT89a, BO91]—proved to be resistant and were shown to have a manipulation problem that is NP-hard. Research in this field remained dormant for the next fifteen years until a revival starting in 2006 brought about results for most common tractable voting systems.

In this chapter we are concerned with a restricted version of the manipulation problem. We survey tractable voting systems that resist manipulation in the unweighted coalitional manipulation (UCM) model with only a constant number of manipulators. This limited case subsumes hardness results in the weighted coalitional manipulation (WCM) model or with variably sized coalitions, thus making a case for UCM being a stronger class of manipulation. We include both the initial work of Bartholdi, Tovey, and Trick [BTT89a] and Bartholdi and Orlin [BO91] that achieved the first results in this area and the recent resurgence of interest in this problem that has resulted in a number of new outcomes.

[4] A voting system is tractable if calculating the winner takes at most polynomial time.

Terms Defined

An *election* is defined to be an instance of a voting system, comprising a voting rule vr, a set of candidates C and a set of votes V. A *voting rule* is a function that takes as input a set of votes and a set of candidates and outputs a set of winners. Unless explicitly stated otherwise, references to n and m are defined as follows: $n = ||V||$ and $m = ||C||$. A *vote* is defined to be a linear ordering over the set of candidates. The *advantage* of a candidate c_i over c_j (hereafter referred to as $adv(c_i, c_j)$) is the number of votes that rank c_i ahead of c_j with values ranging from 0 to n. The *net advantage* of a candidate c_i over c_j is $adv(c_i, c_j) - adv(c_j, c_i)$, with values range from $-n$ to n. A *netadv* score between two candidates c_i and c_j is represented as $netadv(c_i, c_j)$. By definition we can see that $netadv(c_i, c_j) = -netadv(c_j, c_i)$, thus one *netadv* score can represent both directions.

UCM (unweighted coalitional manipulation) is defined to be a decision problem as follows.

Given An election, namely, a voting rule vr, a set of voters V such that $V = V_{NM} \cup V_M$, where V_M is the subset of voters that form the manipulating coalition and V_{NM} is all other voters, and a set of candidates C containing a distinguished candidate c.

Question Does there exist a set of votes for V_M such that vr over the complete set of votes yields c as the winner?

We use the format $\text{UCM}_{\text{2Cope}}$ to refer to the UCM problem in second-order Copeland and likewise for other election systems.

A.2 UCM in Single Transferable Vote

Single transferable vote (henceforward STV) is a voting system with a long history. As esteemed a figure as John Stuart Mill said it was "among the greatest improvements yet made in the theory and practice of government." It determines the winners with a simple multiround procedure that redistributes votes placed for less popular candidates. Also, unlike many of the esoteric voting systems studied in voting theory, STV actually has a history of being used for real-world political elections, in the United States and around the world.

The STV vote tallying procedure is as follows. Give a point to each candidate for each first-place vote he or she receives. If any candidate is the majority winner (i.e. with more than half the total points), that candidate will be the only winner of the election. If no majority winner exists, then select the candidates with the fewest number of points, remove them from consideration, and for the voters who currently give their support to

these candidates, reallocate their support by giving their points to the next highest ranked candidate on their ballots still under consideration. Repeat this procedure until a winner is chosen or all candidates are removed. If the latter occurs, then all of candidates that were removed in the last round will be the winners.

The complex, shifting behavior of STV with multiple candidates is what gives it the resistance to manipulation we discuss here, but it also leads to STV failing to possess some very desirable voting system characteristics. Notably it does not possess the property of monotonicity: It is possible for a voter to increase his or her ranking of a candidate and for that candidate to subsequently do worse in the election. This was enough for Doron and Kronick [DK77] to refer to it as a "perverse social choice function," and it certainly is a flaw of concern.

We show resistance to manipulation through a conventional, if difficult, reduction based on Bartholdi and Orlin's work [BO91]. Their work was actually directed towards showing that the EFFECTIVE PREFERENCE problem is NP-complete. EFFECTIVE PREFERENCE is the problem of finding if a single voter can cause the preferred candidate to win an election. This is effectively the same as UCM with a single manipulator, and we will prove that this problem is NP-hard for STV with a proof based on the aforementioned work [BO91]. The proof is structured as a reduction from the exact cover by three-sets problem.

Exact Cover by Three-sets (X3C)

Given A set $D = \{d_1, \ldots, d_{3k}\}$ and a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of sets of size three of elements from D.

Question Is it possible to select k sets from $\mathcal{S}$ such that their union is exactly D?

In other words, the goal of the problem is to find if there is an appropriately sized set of subsets which covers each of the elements in D. Since each S_i has exactly three elements and due to the required size of the subset, it must be an exact cover with no repeated elements in all subsets chosen.

Proof. We will describe a reduction from an instance of X3C to a instance of the manipulation problem for STV. Note that since the reduction will only require a single manipulator, this shows that the manipulation problem for STV is NP-hard even for only a single manipulator.

Given an instance of X3C $(D, \mathcal{S})$ we construct the election as follows. Let the following comprise the candidate set C:

- The possible winners c and w;

- The set of "first losers" $a_1, \ldots, a_n$ and $\overline{a}_1, \ldots \overline{a}_n$, one of each corresponding to each subset S_i;
- The "second line" $b_1, \ldots, b_n$ and $\overline{b}_1, \ldots, \overline{b}_n$, one of each corresponding to each subset S_i;
- The w-bloc $d_0, \ldots, d_{3k}$, each of whose voters will just prefer them to w;
- The "garbage collector" candidates $g_1, \ldots, g_n$.

We will now describe the set of voters. Where we use ellipses, the remainder of a vote is arbitrary for our purposes and will not effect the result of the election.

- $12n$ voters with preference $(c, \ldots)$;
- $12n - 1$ voters with preference $(w, c, \ldots)$;
- $10n + 2k$ voters with preferences $(d_0, w, c, \ldots)$;
- For each $i \in \{1, \ldots, 3k\}$, $12n - 2$ voters with preference $(d_i, w, c, \ldots)$;
- For each $i \in \{1, \ldots, n\}$, $12n$ voters with preference $(g_i, w, c, \ldots)$;
- For each $i \in \{1, \ldots, n\}$, $6n + 4i - 5$ voters with preference $(b_i, \overline{b}_i, w, c, \ldots)$ and for the three j such that $d_j \in S_i$, 2 voters with preference $(b_i, d_j, w, c, \ldots)$;
- For each $i \in \{1, \ldots, n\}$, $6n + 4i - 1$ voters with preference $(\overline{b}_i, b_i, w, c, \ldots)$ and 2 voters with preference $(\overline{b}_i, d_0, w, c, \ldots)$;
- For each $i \in \{1, \ldots, n\}$, $6n + 4i - 3$ voters with preference $(a_i, g_i, w, c, \ldots)$, 1 voter with preference $(a_i, b_i, w, c, \ldots)$, and 2 voters with preference $(a_i, \overline{a}_i, w, c, \ldots)$;
- For each $i \in \{1, \ldots, n\}$, $6n + 4i - 3$ voters with preference $(\overline{a}_i, g_i, w, c, \ldots)$, 1 voter with preference $(\overline{a}_i, \overline{b}_i, w, c, \ldots)$, and 2 voters with preference $(\overline{a}_i, a_i, w, c, \ldots)$.

This reduction works by requiring the elimination order of a subset of the candidates to correspond to an exact cover over B in order for c to win the election. Namely, c will win the election iff $I = \{i \mid b_i$ is one of the first $3n$ candidates to be eliminated $\}$ is an exact cover. Furthermore, there is a preference order for a single manipulator that will force I to be an exact cover if one exists. We will now consider the relevant properties of the election and show that this is the case.

Since this election is a single-winner STV election with more than two candidates, the scoring process will proceed for a number of rounds and a number of candidates will be

eliminated as the rounds progress. The first $3n$ candidates to be eliminated will be $a_1, \ldots, a_n$, $\overline{a}_1, \ldots, \overline{a}_n$, and exactly one of b_i or $\overline{b}_i$ for every $i \in 1, \ldots, n$.

Candidate c initially has $12n$ votes, while c's primary rival w has $12n - 1$. Every voter that does not have c as their first choice ranks them directly below w, and so c can only gain more votes if w is eliminated. In order to do so, the manipulator must ensure that w does not gain additional votes before he or she is eliminated, as otherwise w would have gained votes against c and c would have been eliminated first. The manipulator must consequently make sure that no candidate is eliminated such that any voter most prefers that candidate and prefers w second most. This is the case with every voter that prefers one of the d candidates, so w will gain a large number of votes if one of them is eliminated. Therefore if any d candidate is eliminated before w, c cannot possibly win. For every b_i candidate that is eliminated, $\overline{b}_i$ gains a large number of votes, pushing it higher than $12n$ in score and preventing them from being eliminated early. Also, for each $d_j \in S_i$, d_j gains two votes, pushing them high enough to not be eliminated early. Thus eliminating b_i protects the d candidates associated with the set S_i. Conversely, for every $\overline{b}_i$ that is eliminated, d_0 gains two votes and b_i gains a large number of votes, preventing b_i from being eliminated early. Thus for every i only one of b_i or $\overline{b}_i$ can be eliminated before c or w.

The a candidates are the other candidates that can be eliminated early. For every a_i that is eliminated, $\overline{a}_i$ gains two votes, b_i gains one vote, and g_i gains the rest of a_i's votes. The effect of this is that now $\overline{a}_i$ has been promoted above b_i and $\overline{b}_i$ in the overall ranking, and since b_i also gained a point over $\overline{b}_i$, $\overline{b}_i$ will be the next to be eliminated instead of b_i. Thus by controlling which of a_i or $\overline{a}_i$ is eliminated first, we control which of b_i or $\overline{b}_i$ is eliminated early.

Thus we can show that the candidate c will win the election iff $I = \{i \mid b_i$ is one of the first $3n$ candidates to be eliminated $\}$ is an exact cover. We know that either b_i or $\overline{b}_i$ will be among the first $3n$ eliminated candidates. If b_i is the one eliminated, then for every $d_j \in S_i$, d_j will gain two votes and will have at least $12n$ votes total, preventing them from early elimination. If I is an exact cover, this will be true for every d_j as each of them is covered by some selected b_i and so each of them will win over w. Also, since d_0 gains two points for every one of the $\overline{b}_i$ eliminated, d_0 will gain at least $2(n - k)$ votes and will receive at least $12n$ votes overall, pushing them over the score of w as well. Thus after the first $3n$ candidates have been eliminated, w will have the least score with $12n - 1$, having gained no votes other than his or her initial first-place votes, and he or she will be the next candidate to be eliminated. The candidate c will then gain a large number of votes from the elimination of w and will go on to win the election.

If the set I defined above does not correspond to an exact cover, c cannot win the election. If I is not an exact cover, some candidate d_j will not gain the two points from a corresponding

b_i being eliminated. Thus d_j will only have $12n-2$ votes after the first $3n$ candidates are eliminated while the other remaining candidates have at least $12n-1$, leaving d_j the next to be eliminated. The candidate w then gains points from d_j's elimination, preventing c from gaining points against w and winning the election.

If an exact cover exists, a single manipulator can construct his or her preference order as follows to ensure c is a winner. For an exact cover I, if $i \in I$, let the ith candidate in the preference order be a_i and otherwise $\overline{a}_i$. The rest of the preference order is arbitrary. This will result in the following order of elimination of the first $3n$ candidates: For $i \in I$, $\overline{a}_i, b_i, a_i$ will be eliminated in that order as the $3i-2, 3i-1, 3i$ candidates. For $i \notin I$, $a_i, \overline{b}_i, \overline{a_i}$ will instead be the candidates to be eliminated. For $i \in I$, since this preference ranks a_i over $\overline{a}_i$, a_i will have one more vote and thus $\overline{a}_i$ will be eliminated first. This then gives two votes to a_i and one more to $\overline{b}_i$, making b_i the next least preferred candidate. When b_i is eliminated, $\overline{b}_i$ gains a large number of votes, so a_i is now the least preferred candidate and is eliminated next. The rounds proceed similarly in the case that $i \notin I$ and $\overline{a}_i$ is preferred instead. Thus the set $\{i \mid b_i$ is one of the first $3n$ candidates to be eliminated$\}$ will correspond to an exact cover and c will win the election.

If no exact cover exists, no matter how a manipulator votes, at least one of the d candidates will not receive the protective boost from the elimination of the corresponding b candidate as previously described. This candidate will then be eliminated early, leading to w being boosted past c in votes and preventing c from winning.

Thus even just setting the first n rankings for the ballot of a single manipulator for STV is NP-hard, and the system is resistant to this very limited case of manipulation. □

A.3 UCM in Borda Voting

Borda voting is a classic voting system dating back at least to the eighteenth century. It was introduced by the French mathematician and engineer Jean-Charles de Borda to remedy the failure of plurality to reflect the wishes of the electorate when used with more than two candidates: The candidate that gets the most votes will not necessarily be preferred to the other candidates. It is very similar to a system introduced in the 15th century by Cardinal Nicolaus Cusanus [Szp10].

It has rich and varied history of real-world use: In some form it has been used in political elections in Slovenia and the micronesian countries of Kiribati and Nauru, in the Eurovision contest, the election of the board of directors of the X.Org foundation, and even in sports, in the election of the Most Valuable Player award in Major League Baseball. Borda is one of a class of systems known as scoring protocols, where each vote awards points to each candidate depending on their ranking in the vote. The winners are candidates with the

highest sum of points over all the votes. In the case of Borda voting, candidates receive linearly descending points for progressively less favorable positions in the votes, with the top position awarding $m-1$ points, the next position awarding $m-2$ points, and on down to 0 points for the lowest position.

It was long an open problem whether $\text{UCM}_{\text{Borda}}$ was hard in general, though manipulation with a single manipulator has long been known to be easy [BTT89a]. A greedy algorithm that can find a set of successful manipulating votes in polynomial time that is at most one larger than the optimum manipulative coalition size is known as well [ZPR09]. Recently, Betzler, Niedermeier, and Woeginger [BNW11] and Davies, Katsirelos, Narodytska, and Walsh [DKNW11] proved Borda-manipulation to be NP-hard for instances with two or more manipulators.

Both Betzler, Niedermeier, and Woeginger [BNW11] and Davies et al. [DKNW11] prove their results by reduction from the problem of 2-Numerical Matching with Target Sums, a known NP-hard problem [YHL04] that closely corresponds to the problem of allocating points to the nonfavored candidates in the election.

2-Numerical Matching with Target Sums (2NMTS)

Given A sequence $a_1, \ldots, a_k$ of positive integers with $\sum_{i=1}^{k} a_i = k(k+1)$ and $1 \leq a_i \leq 2k$.

Question Are there permutations ψ_1 and ψ_2 of $1, \ldots, k$ such that $\psi_1(i) + \psi_2(i) = a_i$ for $1 \leq i \leq k$?

Preliminaries For manipulation to work, nonfavored candidates must be ranked low enough in manipulative votes such that the number of points they gain by said votes do not prevent the preferred candidate from winning. To that end we define the *gap* to be the maximum number of points nonfavored candidates can gain by all manipulative votes while still allowing the preferred candidate to win. In any Borda instance with a favored candidate c^*, m other candidates, and t manipulators, the gap g_i for a candidate c_i is $score(c^*) + t * m - score(c_i)$. Here $score(c)$ refers to the Borda score for the candidate c over the nonmanipulator votes. We assume these gap values $g_1, \ldots, g_m$ to be ordered in a nondecreasing fashion.

Result 1 Recall that the awarded points for the last j candidates in a vote will range from $j-1$ to 0, hence the sum of their points equals $j(j-1)/2$. Thus for any successful manipulation instance, we must have that $\sum_{i=1}^{j} g_i \geq t \cdot j(j-1)/2$ for each $j \in \{1, \ldots, m\}$. We define an instance to be *tight* if $\sum_{i=1}^{j} g_i = t \cdot j(j-1)/2$. Thus, in a tight instance, for manipulation to be successful, the number of points a nonfavored candidate gains from manipulative votes must be exactly equal to its gap value.

Proof. Given any instance of 2NMTS we construct a $\text{UCM}_{\text{Borda}}$ instance (C, V, p) as follows. The candidate set C consists of candidates $c_1, \ldots, c_k$ and the preferred candidate p, and thus

the range of Borda points is from 0 to k. The set of votes V consists of a manipulating coalition M of size two and a set NM of nonmanipulating votes of size three. The instance is constructed such that the gap for any nonfavored candidate $g_i = 2k - a_i$. In this context, the Borda problem can be considered as follows: For every nonfavored candidate c_i, can we assign a position in each of the manipulating votes such that the points c_i gains from said votes is $\leq g_i$? This constructed instance of Borda manipulation will have a solution if and only if the 2NMTS instance has a solution.

Direction 1 Given a solution to 2NMTS, we can obtain a solution for the Borda instance as follows: Preferred candidate p is placed in the first position in both manipulating votes. A solution to 2NMTS exists so we have two orderings $\psi_1(i), \psi_2(i)$ such that $\psi_1(i) + \psi_2(i) = a_i$. For every candidate $c_i, (1 \leq i \leq k)$ set its position to $\psi_1(i) + 1$ in the first manipulative vote, and $\psi_2(i) + 1$ in the second manipulative vote. The corresponding Borda points are obtained from subtracting this position number from $||C|| = k + 1$. Therefore the points c_i has gained from both manipulative votes is $(k+1-(\psi_1(i)+1)) + (k+1-(\psi_2(i)+1))$ which equals $2k - a_i = g_i$, permitting p to win.

Direction 2 Given a solution to the Borda instance, we have a solution for 2NMTS as follows: By construction, $\sum_{i=1}^{k} g_i = \sum_{i=1}^{k} (2k - a_i)$.

Since $\sum_{i=1}^{k} a_i = k(k+1)$, $\sum_{i=1}^{k} (2k - a_i) = k(k-1)$.

Hence, this Borda instance is tight for $j = k$ and $t = 2$ (from Result 1) and consequently each nonfavored candidate C_i gains exactly g_i points from the manipulative votes. If $pos_i(1), pos_i(2)$ are the positions for c_i, points gained from these positions total $(k + 1 - pos_i(1)) + (k + 1 - pos_i(2)) = g_i = 2k - a_i$, which yields $pos_i(1) + pos_i(2) = a_i + 2$. Therefore setting $\psi_1(i) = pos_i(1) - 1$ and $\psi_2(i) = pos_i(2) - 1$ gives us a solution for 2NMTS.

As mentioned earlier, we assume by construction $g_i = 2k - a_i$. This requires constructing the set of nonmanipulator votes such that the deficits for each candidate relative to the preferred candidate map precisely to the target sums in the original problem. Executing this involves complicated construction and the addition of a large number of "dummy" candidates to pad out the remaining positions and to precisely set the required deficits for the primary candidate set [BNW11] (in Davies et al., such padding is done with voters [DKNW11]). Thus the constructed instance has a much larger candidate set than a voter set (though it remains polynomially bounded), but it suffices to prove the desired hardness result. □

A.4 UCM in Copeland Elections

Copeland voting is a voting system with a long history. One version of the system was discovered by the 13th century mystic Ramon Llull, and then another variation was discovered by A.H. Copeland in the 1950's. It is a Condorcet voting system.

Copeland voting is in fact a family of voting systems, parametrized on how ties are handled. In Copeland$^\alpha$, the score of a candidate c in an election E is $\text{wins}_E(c) + \alpha * \text{ties}_E(c)$, where $\text{wins}_E(c)$ and $\text{ties}_E(c)$ denote the number of pairwise victories and ties of the candidate c in the election E. Llull's system is Copeland^1, while "Copeland voting" has been used to describe Copeland$^{0.5}$ or to describe Copeland^0. Different parameter values subtly alter the behavior of the system and complicate the task of analyzing its computational properties, as we will explore.

When constructing an election for a reduction, we do not have to explicitly construct the voter set. It suffices to specify the number of voters that prefer every candidate over each other candidate, or even just the net advantage function—the difference of the number of voters that prefer one candidate over the other and vice versa. This relation can be conveniently represented as a weighted directed graph. It is also possible to go in the other direction and map from an election graph to a set of voters that will express that election graph, provided that all the edge weights are even. We can construct such a set in polynomial time in the size of the set of candidates and in the maximum *value* of the net advantage function. As long as our construction can be limited to a net advantage function polynomially bounded in value, we can subsequently construct a concrete election in polynomial time and complete the reduction [FHS10].

It is also not necessary to specify the entire election graph in order to construct an consistent election. Using this technique greatly aids in the construction of hardness proofs, as we can then more easily construct an election graph with the properties we require without immediately specifying the details of the votes.

$\text{UCM}_{\text{Cope}^\alpha}$ is in P when there is only one manipulator [BTT89a], but it is known to be resistant to manipulation even with two manipulators for $\alpha \in [0, 0.5) \cup (0.5, 1]$ [FHS10, FHS08] Different proofs were required to show resistance to manipulation for different parameter ranges, as the behavior of the system changes in subtle but significant ways with different values of the parameter. We will describe the proof that was used to show hardness for Copeland$^\alpha$ for $\alpha \in \{0, 1\}$.

Both of these cases were proved by Faliszewski, Hemaspaandra, and Schnoor [FHS10] through similar reductions from X3C defined in section A.2. The proof constructs a $\text{UCM}_{\text{Cope}^\alpha}$ instance from an X3C instance $(D = \{d_1, \ldots, d_3k\}, \mathcal{S} = \{S_1, \ldots, S_n\})$

This reduction is greatly aided by the technique mentioned earlier of building a simplified

election representation rather than specifying the exact votes that make up the election. In fact, we do not need to describe the entire set of candidates. It is enough to describe a partial set of significant candidates, the net advantage score between each pair of these candidates, and the lead in Copeland score that each specified candidate has over the distinguished candidate. It suffices that we can then easily construct a complete election that matches the specified behavior.

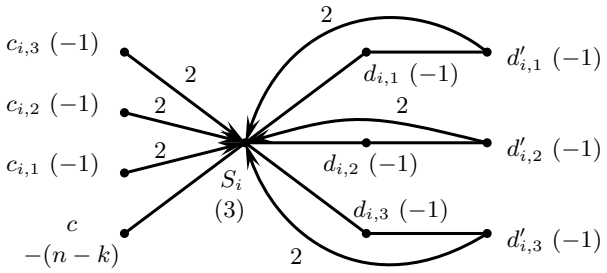

Figure A.2: Gadget used in Copeland[1] manipulation NP-hardness proof

The set of candidates includes primary and other auxiliary candidates corresponding to each set S_i, two for each element d_j, and one primary adversary candidate c. The election is constructed, mapping from the X3C instance, by constructing a gadget for each set S_i composed of the specified information. All of the d candidates are given one point of advantage of Copeland score over the distinguished candidate, so each must be made to lose one pairwise contest. However, in order to do so one must lose to the primary candidate for a set S_i of which that element is a member. The S_i candidates, however, must also be made to defeat the adversary candidate c as c will start with a significant advantage over the distinguished candidate. Thus we must find an exact cover so we only have to use k of the S_i candidates to beat all the d candidates so we will still have enough left to beat c.

A.5 UCM in Second-Order Copeland

Solutions for breaking ties in Copeland voting attempt to choose the more "powerful" candidate as the winner, which can be defined in a number of ways. One such method is second-order Copeland. It selects the candidate whose set of defeated opponents (hereafter referred to as DO_c for a candidate c) has the higher sum of Copeland scores. We will let $sum\_score(S)$ for a set of candidates S be the sum of the Copeland scores for candidates in S, and so $sum\_score(DO_c)$ gives the second-order Copeland score for a candidate c.

Second-order Copeland has been used by the National Football League and the United

States Chess Federation to break ties, and has a special place in voting theory: It was the first of the tractable voting systems for which the problem of manipulation was shown to be NP-complete, even in the case of just one manipulator [BTT89a].

The problem of unweighted coalitional manipulation in second-order Copeland, hereafter referred to as $\text{UCM}_{\text{2Cope}}$, is NP-complete even for one manipulator [BTT89a]. Verifying a given solution is clearly in P as we simply have to calculate Copeland scores and second-order Copeland scores for each candidate. To prove $\text{UCM}_{\text{2Cope}}$ is NP-hard we show a polynomial-time reduction from 3,4-SAT, a known NP-hard problem [Tov84].

3,4-SAT

Given A set U of boolean variables, a collection of clauses Cl, each clause composed of disjunctions of exactly three *literals*, which may be a variable or its complement, and each variable occurs in exactly four clauses.

Question Does there exist a boolean assignment over U such that each clause in Cl contains at least one term set to true?

To facilitate the reduction we construct a graph representation of a second-order Copeland election that encodes the given 3,4-SAT instance. We will use a election graph representation with vertices representing candidates and directed edges representing the result of pairwise contests.

Proof. Given a 3,4-SAT instance, we create a second-order Copeland election graph as follows: Every clause (C_1 to $C_{||Cl||}$) and every literal is a candidate, represented as a vertex in our graph. The manipulating coalition's chosen candidate is a separate candidate C_0. All pairs of vertices have directed edges between them, representing decided pairwise contests, except for any variable and its complement. The decided contests cannot be overturned by our manipulators, while undecided contests can be shifted in either direction according to the manipulating vote. Clauses beat (that is, have a directed edge to) literals they contain, and lose to all other literals.

In addition to these candidates derived from the SAT instance, we pad the election with a number of auxiliary candidates in such a way to achieve the desired Copeland scores and second-order Copeland scores for each of the candidates. We will have that all the clause candidates and C_0 are tied with the highest Copeland score. We will also have that each clause candidate C_i has $sum\_score(DO_{C_i}) = sum\_score(DO_{C_0}) - 3$.

The second-order Copeland score for C_0 will be independent of the variable-complement contests. For all possible outcomes of the variable-complement contests, C_0 still beats every candidate except for the clause candidates. Recall that the elements for every clause candidate's defeated-opponent set are their component literals. Their second-order Copeland

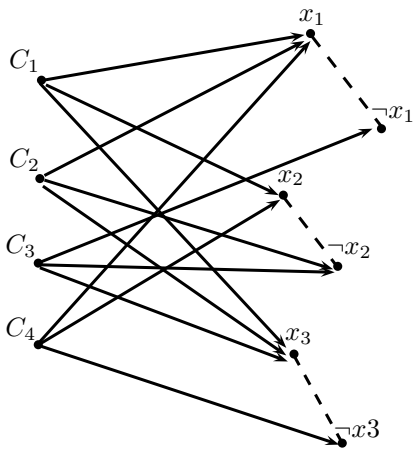

Figure A.3: A partial representation of the resultant election-graph. Clauses represented are $C_1(x_1 \vee x_2 \vee x_3)$, $C_2(x_1 \vee \neg x_2 \vee x_3)$, $C_3(\neg x_1 \vee \neg x_2 \vee x_3)$ and $C_4(x_1 \vee x_2 \vee \neg x_3)$. Each clause vertex beats the variables (or their complements) that are its component literals. The dotted lines indicate undecided (second-order Copeland) contests between variables and their complements.

scores and the final result of the election will then depend on how each of the variable-complement contests are decided.

Consider if any clause candidate C_i's literals win all their contests. The $sum\_score(DO_{C_i})$ increases by 3 points and C_i is tied with C_0 for first place. Therefore, in order for C_0 to be the unique winner, at least one element of each clause candidate's defeated-opponent set must lose at least one of their contests. Recall that the elements of any such defeated-opponent set are variables and their complements. For any variable x, we can interpret a directed edge from x to $\neg x$ as setting x to *true*. A vote that allows C_0 to win, then, would correspond exactly to each clause having at least one true term. In other words, we have a solution for $\text{UCM}_{\text{2Cope}}$ if and only if we have solution for 3,4-SAT. Therefore $\text{UCM}_{\text{2Cope}}$ is NP-hard with just a single manipulator. □

A.6 UCM in Maximin

Maximin voting, also known as the Simpson-Kramer method, is a typical Condorcet voting system. As such it deals with contests between pairs of candidates, specifically their *net advantage* scores. To find the winner under Maximin, given a *netadv* function over the set of candidates, we first select the lowest *netadv* score for each candidate k in C, i.e., we select the minimum score for $netadv(k, k')$ for all k' in C such that $k' \neq k$. The winner is the candidate with the highest such score. We can trivially see that a candidate with a

minimum *netadv* score greater than 0 will be the Condorcet winner, and there can only be one such candidate, and so this is a Condorcet voting system.

Calculating the Maximin winner is easily seen to be polynomial in the size of the election, but Xia, Zuckerman, Procaccia, Conitzer, and Rosenschein [XZP+09] prove that for two or more manipulators the problem of $\text{UCM}_{\text{Maximin}}$ is NP-complete.

$\text{UCM}_{\text{Maximin}}$ is NP-complete for two or more manipulators [XZP+09]: Verifying an instance is easily seen to be polynomial in the number of candidates as we can calculate the winner in polynomial time. To prove $\text{UCM}_{\text{Maximin}}$ is NP-hard we construct a polynomial-time reduction from the *vertex-disjoint-two-path* problem, known to be NP-complete [LR78].

Vertex-Disjoint-Two-Path problem (VDP_2)

Given A directed graph G and two sets of vertices u, u' and v, v' such that all four vertices are unique.

Question Do there exist two paths $u \to u_1 \to \ldots \to u_j \to u'$ and $v \to v_1 \to \ldots \to v_k \to v'$ in G such that each path is a set of vertices disjoint from the other?

Proof. To facilitate our reduction-construction from a graph problem such as the vertex-disjoint-two-path problem to Maximin, we first construct a graph-representation of Maximin elections. A complete set of votes is not required to represent an election.[5] We can construct the same given just a *netadv* (or *adv*) function. Also, there exists a bijection between the *netadv* function and directed edges of an antisymmetric graph such that given one set, we can represent it in terms of the other.[6]

P_{coal} is the set of votes of the manipulating coalition and $P_{noncoal}$ is the set of all other votes. The term $netadv_{noncoal}$ indicates the *netadv* score obtained by considering only the noncoalitional votes with $netadv_{coal}$ similarly defined. M is the size of the coalition and c is the candidate supported by the manipulating coalition.

Given a VDP_2 instance, that is a graph $G(V_G, E)$ and vertices $u, u', v, v' \in V$, we obtain a graph G' using the following constructions and assumptions:

- Every vertex in our graph is reachable from u or v.
- There are no directed edges $u \to v'$ or $v \to u'$.
- We add special edges $u' \to v$ and $v' \to u$ such that $E_{G'} = E \cup \{(u', v), (v', u)\}$.

Our $\text{UCM}_{\text{Maximin}}$ instance then is as follows:

- Set of candidates $C = V_G$.

[5] See Section A.9.
[6] See Section A.10.

- The set of $P_{noncoal}$ preferences.[7]
  - $\forall c' \in C : c' \neq c, netadv(c, c') = -4M$
  - $netadv(u, v') = netadv(v, u') = -4M$
  - For all other edges (x, y) in E, $netadv(y, x) = -2M - 2$
  - For all other vertex pairs a, b, $netadv(a, b) = 0$

Regarding P_{coal} votes, we can freely assume that every coalition vote will rank c first, thus giving $netadv_{coal}(c, c') = M$ and $netadv_{noncoal \cup coal}(c, c') = -3M$. Thus, in our construction, scores for $netadv(c, c')$ are fixed for all candidates $c' \in C$. In order for c to be the winner, at least one $netadv$ score must be less than $-3M$ for every other candidate. We will see that in our construction this can occur if and only if there are two vertex-disjoint paths in G'.

Direction 1

The existence of vertex-disjoint paths $u \to u_1 \to \ldots \to u_j \to u'$ and $v \to v_1 \to \ldots \to v_k \to v'$ yields a P_{coal} that makes c the winner. In order to construct the manipulator preferences, we will make use of a connected subgraph over G' containing all the vertices, but with $u \to \ldots \to u' \to v \to \ldots \to v' \to u$ as the only cycle in the graph.

We can construct P_{coal} votes in 3 parts as follows: Each manipulator vote will rank c the highest, followed by the vertex-disjoint-path vertices, followed by the other vertices.

Other-vertex ordering: These vertices will be ordered in the votes based on a linear order extracted from the single-cycle subgraph.

Vertex-disjoint-path orderings: We have two vertex-disjoint-path orderings: $u \to \ldots \to u' \to v \to \ldots \to v'$ and $v \to \ldots \to v' \to u \to \ldots \to u'$ and thus two possible vote constructions for P_{coal}. We construct $m - 1$ votes as per the first ordering and 1 vote as per the second.[8] Thus $netadv(c, c')$ increases by m points but every other $netadv$ score increases by less than m points, making c the winner. The calculations are as follows for the complete (coalitional and noncoalitional) set of votes:

- $netadv(u, v') = -4M + (M - 1) - 1 = -3M - 2$
- $netadv(v, u') = -4M + 1 - (M - 1) = -5M + 2$

For any other candidate $c' \notin \{c, u, v\}$, we can see that there exists some candidate d in every vote of P_{coal} that beats c', i.e., the lowest $netadv$ score for c' is $netadv_{coal}(c', d) = -M$, and thus for the complete set of votes the (lowest) $netadv$ for any such candidate c' is no

[7] As mentioned in Section A.9, we can easily convert from a $netadv$ function over the set of candidates to an equivalent set of votes. Such a $P_{noncoal}$ would also be polynomial in size relative to the number of candidates.

[8] Switching these orderings results in the same outcome.

more than $-2M - 2 - M = -3M - 2$. All the above *netadv* scores are less than $-3M$ for all values of $m \geq 2$, thus c is the winner.

Direction 2

The existence of a P_{coal} that makes c a winner yields a positive VDP$_2$ instance in the graph G':

Since c is the winner, we know that for any other candidate c' in C, there exists a candidate d that beats c' such that:

- $netadv(c', d) < -3M$;
- There exists an edge (d, c') in G';
- d is ranked higher than c' in a majority of the total votes and in at least one vote in P_{coal}—the proof of this is as follows.

Consider such an edge (d, c'):[9] either (d, c') is one of the special edges (v', u), (u', v') or $(d, c') \in E$. If (d, c') is a special edge, then at least one vote in P_{coal} must prefer d to c' (since $netadv(c', d) < -3M$). If $(d, c') \in E$, then all M votes in P_{coal} must prefer d to c'.

For this d, we can choose a candidate that beats it with sufficient margin, and continue to find such a candidate for the previous choice of d.[10] That is, we find a d for c' starting with say, u and then continue to find such a candidate after setting d to c'. There is only one possible d for (each case of) $c' = u, v$. Thus, we obtain a set of chained pairs. This, coupled with the facts that any vertex is reachable from u, v and the existence of special edges (both by construction), we obtain a cycle of vertices which breaks into disjoint sets along the special edges. Obtaining such a set of pairs with *netadv* less than $-3M$ is not possible without the existence of (c-winner-making) P_{coal} (as per the third item above). □

A.7 UCM in Tideman-Ranked-Pairs

Tideman-Ranked-Pairs (TRP) was conceived by Nicolaus Tideman in 1987 when attempting to define a voting system that "almost always" has the property of independence of clones [Tid87]. It is defined as follows: Given a *netadv* function over the set of candidates, create a list by ranking the pairs in descending order of their scores. In the case of a tie between two *netadv* pairs, e.g., $netadv(a, b) = netadv(x, y)$, we break ties by ordering the pairs lexicographically by some arbitrary ordering of the candidates. We add the top-scoring pair to an election-graph G[11] and remove it from the list if the resultant graph does not

[9] If there exists more than one such d we choose one arbitrarily.
[10] Choosing such a d is formalized in the proof of Xia et al. as a composite function f.
[11] As usual, $V_G = C$ and directed edges represent *netadv*.

contain a cycle. Otherwise we skip this pair and move on to the next one in the list. We continue until have considered all pairs. Since we now have a directed acyclic graph, there must exist a source vertex, which we state to be the TRP winner.

Xia, Zuckerman, Procaccia, Conitzer, and Rosenschein [XZP+09] found that UCM_{TRP} is NP-complete even for one manipulator. We can easily see that verifying a given solution to UCM_{TRP} is in P. UCM_{TRP} was proven to be NP-hard by a polynomial-time reduction from 3SAT.

Three-Conjunctive-Normal-Form Satisfiability (3SAT)

Given A set U of boolean variables, a collection of clauses Cl, each clause composed of disjunctions of exactly three *literals*, which may be a variable or its complement.

Question Does there exist a boolean assignment over U such that each clause in Cl contains at least one literal set to true?

Proof. Given a 3SAT instance, we construct a UCM_{TRP} election graph as follows. Clauses $C_1 \dots C_{||Cl||}$ are vertices as is the coalition's preferred candidate c. For each clause C_i, we construct six other special-clause candidates—three for the literals it contains and three for their complements. C_i beats (has a directed edge to) the special-clause candidates corresponding to the literals it contains, which in turn beat the literals they correspond to. We also have a C_i' such that it is beaten by the complement of the literals in C_i. Candidate c starts out beating the C_i candidates but gets defeated (though by a smaller margin) by the C_i' candidates. The intuition of this correspondence is that the edges beating c are so weak and so far down the ordering that they are not added, leaving c to be the source vertex (and TRP winner) iff there exists a solution to 3SAT. Thus the UCM problem is NP-complete even in the case of a single manipulator. The proof of Xia et al. relies on mathematical gadgets to achieve this correspondence [XZP+09]. □

A.8 Conclusion

Our survey of UCM results can be seen as qualifying election systems by a single metric. Determining which election system is superior is an ongoing debate often reflecting differing philosophies. Pierre-Simon Laplace in his lectures at the Ecole Normale Superieure in 1795 attacked the *Éléction par ordre de mérite* (Election by ranking of merit) system of his contemporary Jean-Charles Borda [Szp10], later proposing a variation of a majority-rule in its place. Much of the modern literature on voting theory is still devoted to advocacy for particular voting systems, arguing their superiority by one metric or another.

Voting rule	Coalition size = 1	Coalition size ≥ 2
Copeland$^{\alpha}(0 < \alpha < 0.5)(0.5 < \alpha < 1)$	P [BTT89a]	NP-complete [FHS08]
Copeland$^{\alpha}(\alpha = \{0, 1\})$	P [BTT89a]	NP-complete [FHS10]
Copeland$^{\alpha}(\alpha = \{0.5\})$	P [BTT89a]	?
Second-order Copeland	NP-complete [BTT89a]	NP-complete [BTT89a]
Single Transferable Vote	NP-complete [BO91]	NP-complete [BO91]
Maximin	P [BTT89a]	NP-complete [XZP^{+}09]
Tideman Ranked Pairs	NP-complete [XZP^{+}09]	NP-complete [XZP^{+}09]
Borda	P [BTT89a]	NP-complete [BNW11, DKNW11]
Bucklin	P [XZP^{+}09]	P [XZP^{+}09]
Plurality with Runoff	P [ZPR09]	P [ZPR09]
Veto	P [BTT89a]	P [ZPR09]
Cup	P [CSL07]	P [CSL07]

Table A.1: Table of UCM results for common tractable voting systems.

In our survey we showcase manipulation results for a particular class of voting systems, namely those with a tractable winner problem but where unweighted coalitional manipulation is hard for a constant coalition size. The complexity of this case of UCM has been determined for most common voting rules, though a few remain: Copeland$^{0.5}$ remains unsolved even as results for all other parameter values have been found [FHS10].

Several related areas of research, however, remain more or less uncharted. The most significant simplification in the literature is that most of the hardness results achieved are just worst-case. Several papers have studied whether voting systems are difficult to manipulate in a large fraction of instances, finding that manipulation can be easy in the average case while it is hard in the worst case [CS06, PR07b, PR07a]. Additionally, approximation algorithms exist for several worst-case hardness results. Brelsford, Faliszewski, Hemaspaandra, Schnoor, and Schnoor [BFH^{+}08] formalized manipulation as an optimization problem and then studied whether this version of the problem is approximable. Zuckerman, Procaccia, and Rosenschein [ZPR09] discovered an approximation algorithm for Borda manipulation before it was known to be NP-hard, and Davies, Katsirelos, Narodytska, and Walsh [DKNW11] both proved that $\text{UCM}_{\text{Borda}}$ is NP-hard but also found several more approximation algorithms for this problem. Zuckerman, Lev, and Rosenschein [ZLR10] discovered an approximation algorithm for the maximin voting system. Lin explored the use of relatively efficient algorithms for the NP-complete integer partitioning problem to solve manipulation instances [Lin11].

Conitzer, Sandholm, and Lang qualified the manipulation problem with an additional metric: the minimum number of candidates that must be present for manipulation to be NP-hard [CSL07]. Additionally they broke from the standard model and studied whether manipulation is hard for cases where manipulators do not have complete information of all of the votes. Slinko explored how often elections will be manipulable based on the size of the manipulative coalition [Sli04].

Research into how often elections can be manipulated [FKN08], and more general areas such as parametrization of NP-hard problems [Nie10] and phase transitions [CKT91, KS94, Zha01] lead to a more nuanced approach to problem classification. Walsh has examined phase transitions for manipulation in the veto rule [Wal09].

More specific to computational social choice, votes in the literature are most commonly represented as transitive linear preference orderings over the set of candidates and the concept of irrational votes has only been sparsely dealt with. Irrational (by which we mean intransitive) votes, may be more apt for any number of real-world scenarios where voters tend to rank candidates according to multiple criteria. Irrational votes are not represented as a linear ordering but as a preference table which holds the voter's choice for any pair of candidates. For Copeland$^{\alpha}$ for $\alpha \in \{0, 0.5, 1\}$, manipulation is in P in the irrational voter model, while it is known to be NP-hard for $\alpha \in \{0, 1\}$ in the standard voter model [FHS10]. Thus voting systems may have different behavior with regard to manipulation in the irrational voter model and which deserves more study. Another convention is that the default definition of UCM is constructive—i.e., efforts are directed to making a preferred candidate a winner, rather than preventing a certain candidate from winning. Variations of UCM with a destructive approach is another area rich with possibilities.

UCM instances presented in this paper typically have a large number of candidates and a smaller constant-sized coalition of manipulators. In contrast, cases with a small number of candidates and a relatively large manipulating coalition might be considered more natural. Betzler, Niedermeier, and Woeginger mention a specific open area of research in this area: whether there exists a combinatorial algorithm to solve Borda efficiently with few candidates and an unbounded coalition size [BNW11]. They also mention that solving a $\text{UCM}_{\text{Borda}}$ instance having a coalition of size 2 in less than $O(||C||!)$ is still an open problem. Also results from areas such as scheduling are being applied to algorithms for manipulation problems [XCP10].

Another approach to the manipulation problem taken by Conitzer and Sandholm [CS03] and Elkind and Lipmaa [EL05] is modifying voting systems to give them greater resistance to manipulation. They add an extra initial round of subelections between subsets of the candidates. Conitzer and Sandholm describe techniques that can make manipulation NP-hard or even PSPACE-hard for these modified voting systems. Elkind and Lipmaa present a version of this technique that uses one-way functions to construct the initial round schedule from the set of votes. Reversing the one-way function is computationally hard, preventing election organizers from gaming the initial round and forcing a desired result in polynomial time. These techniques essentially construct new voting systems by structurally augmenting our standard systems to imbue them with complexity.

Other related work includes the study of electoral control, which encompasses attempts

by an election organizer to change the election result by modifying the structure of the election in various ways [BTT92, FHHR09, EFPR10, HHR09], and the problem of bribery, where a briber attempts to sway the result of an election by paying off a set of voters to change their votes [FHH09, FHHR09, Fal08, EFS09]. These too are problems endemic to many voting systems to which complexity can serve as a defense.

Another possible response to the classic voting theory results that limit the quality of our voting systems is to reconsider the standard election model. Balinski and Laraki [BL07] introduce a model where voters give candidates independent grades, such as the letter grades F to A or {good, average, bad}, rather than ranking them in a linear order, similar to approval voting or range voting. In a sense this represents a reversion to the pre-Bergson-Samuelson model of welfare functions. Balinski and Laraki's method gives the median grade over all voters as the grade of each candidate, unlike range voting where the aggregate grade is instead the average. We can give an complete aggregate preference ordering of the candidates with the help of a tie-breaking mechanism: Successively remove one of the median-score-awarding voters from the votes for each tied candidate and recompute the median grades until they are no longer tied. Balinski and Laraki's approach does not rely on complexity but instead redesigns the election model to become strategy-proof in a limited case defined by the authors.

Faliszewski, Hemaspaandra, Hemaspaandra, and Rothe [FHHR11] showed that with a restriction to single-peaked preferences, a wide range of manipulation and control instances that are NP-hard in the general case turn out to be easy (though not any of the results we describe here). For some voting systems these problems remain easy even with a partial relaxation of the single-peaked model that allows for a small number of "mavericks," whose votes are not aligned with the single-peaked ordering [FHH11a]. The single-peaked model is considered "*the* canonical setting for models of political institutions" [GPP09], so this work calls the significance of a number of hardness results into question.

It can be argued that as compared to WCM, UCM is a better test of a voting system's vulnerability to manipulation. UCM serves as a special case of WCM and hence subsumes its hardness results. In other words, if an election system is resistant to manipulation in the UCM case, it will resist manipulation in the WCM case, but the other direction does not necessarily follow. With this problem solved for most common voting systems, we look forward to the resolution of the remaining open problems as well as new avenues of research into the manipulability of voting systems.

A.9 Constructing an Election Given a Netadv Function: The McGarvey Method

While the traditional representation of an election requires a set of votes, we can construct these votes given a pairwise preference relation over the set of candidates. This method was given by McGarvey [McG53] and can be applied with very little modification to a *netadv* function.

A.9.1 From a Preference Pattern to a Set of Votes

Theorem A.9.1. (McGarvey's Theorem) *Given a* preference pattern *we can elicit a set of votes (defined to be strict and complete preference orderings over the set of candidates) such that (the ordering derived from) the preference pattern is the result of the election.*

Definition A.9.2. *A* preference pattern *is a set of relations over the set of candidates. The relations are a preference relation (expressed as aPb, i.e. a is preferred to b) and an indifference relation (aIb, i.e. a is neither preferred to b nor is b preferred to a).*

Both relations are *distinct* for any pair of candidates, i.e., aPb implies $\neg bPa$, and aIb implies bIa. Thus we will have $m(m-1)/2$ pairs over both relations where m is the cardinality of the set of candidates. McGarvey's method constructs a set of votes as follows:

For each pair aPb, we construct two preference orderings $abc_1 \dots c_{m-2}$ and $c_{m-2} \dots c_1 ab$.[12] For each pair aIb, we construct $abc_1 \dots c_{m-2}$ and $c_{m-2} \dots c_1 ba$. The idea is that on evaluation for these preferences, rankings of all candidates besides a, b from these orderings will be equal, and the rankings of a, b reflect the preference relation under consideration. Consider the example $C = \{a, b, c, d\}$. The six pairs we consider are $\{aPb, aPc, aPd, bPc, bPd, cPd\}$.

To represent aPb we construct two votes $abcd$ and $dcab$. Evaluating these two votes in the context of pairwise rankings leads to two votes for aPb and no votes for any other pair over the set of candidates. Similarly, for aPc, we construct $acbd$ and $dbac$ and so on. The idea is that for all candidates besides the ones under consideration, preferences for and against them cancel each other out. Hence the need for two votes for each pair. The total number of thus-constructed votes is twice the cardinality of the preference pattern. Thus in our example, we obtain a set of votes which yield exactly the set of pairwise relations in the given preference pattern.

[12] Where candidates are listed in order of decreasing preference.

A.9.2 From a Netadv Function to a Set of Votes

We consider the following useful property of the *netadv* function when constructing the corresponding election:

Theorem A.9.3. *For all pairs of candidates c_i, c_j where $c_i \neq c_j$, the values of $netadv(c_i, c_j)$ are either all even or all odd.*

Proof. Consider two blocks of votes a, b where each block represents one side taken in a pairwise election such that $a + b = n$.
If n is even:

- Then a, b are either both odd or either both even since the sum components of an even number are either both even, or both odd.
- The difference between two even numbers or two odd numbers is always even.

If n is odd:

- Then a, b are either odd and even, or even and odd, respectively, since the sum components of an odd number are always a combination of even and odd.
- The difference between an even and odd number is always odd.

Thus, the *netadv* values are either all even or all odd. □

We can see that *netadv* function corresponds to elements of a preference pattern: $netadv(c_i, c_j) > 0$ corresponds to $c_i P c_j$, $netadv(c_i, c_j) = 0$ corresponds to $c_i I c_j$, and $netadv(c_i, c_j) < 0$ corresponds to $c_j P c_i$. However, the key difference between the *netadv* functions and preference-pattern elements is that *netadv* has scores, which we must factor into our construction.

Since we construct two votes for each pair of candidates, the problem of applying McGarvey's method to a *netadv* relation with an even score is trivial—for each of the two preference orderings constructed we simply have $n/2$ such votes. In fact, given a *netadv* function, if one *netadv* score is even, then (1) the number of votes will be even, and (2) every *netadv* score in that set will be even. The converse also applies: If one *netadv* score in a given set is odd, then the number of votes will be odd, and every *netadv* score in the given set will be odd.

Applying McGarvey's method to an odd set of *netadv* scores requires a slight tweak. We first must select some arbitrary ordering of the candidates. Then for any *netadv* score $s = netadv(a, b)$ where a precedes b in the ordering, we construct $s - 1$ (net) votes (or units of score) exactly as in the case where the scores are all even (i.e., two preference orderings,

with $(s-1)/2$ such votes for each one). In the case of $s = netadv(b, a)$ where a precedes b, we will instead create $(s+1)/2$ pairs of votes to give a score of $s+1$ for b over a. To obtain the last points, we construct one preference ordering corresponding to the previously chosen ordering of the candidates. This will give one net vote to every pair a, b where a precedes b, and minus one vote where b precedes a. Thus we achieve the desired odd numbers for each *netadv* input.

Number of Constructed Votes In the above two cases (*netadv* scores being even or odd) we can see the upper bound on the number of votes constructed is one for every unit of score across all *netadv* values. The size of the set of votes will be bounded by $\sum_{c_i,c_j \in C} |netadv(c_i, c_j)|$. Thus given a *netadv* function with bounded value, we can construct a reasonably small set of votes.

A.10 Graph Representation of the Netadv Function

Consider a directed antisymmetric graph $G(V_G, E)$ where V_G is the set of vertices and E is the set of directed weighted edges. We can trivially see that there can exist a maximum of $m(m-1)/2$ directed edges, where $m = ||V_G||$. In other words, we can represent a *netadv* function with such a graph G where V_G is the set of candidates C and E encodes the *netadv* function. Thus, for example, given a Maximin election as input in the form of a *netadv* function and a set of candidates C containing a distinguished candidate c, we can construct a graph G where $V_G = C$ and we have directed edges for every positive element of the *netadv* function. For example, $netadv(c_i, c_j) = 4$ is represented as a directed edge from c_i to c_j of weight 4. Similarly, given such a graph input, we can obtain an equivalent *netadv* function.

Bibliography

[Arr50] K. Arrow. A difficulty in the concept of social welfare. *Journal of Political Economy*, 58(4):328–346, 1950.

[Arr63] K. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, 1951 (revised editon, 1963).

[BBCN12] N. Betzler, R. Bredereck, J. Chen, and R. Niedermeier. Studies in computational aspects of voting—A parameterized complexity perspective. In *The Multivariate Algorithmic Revolution and Beyond*, pages 318–363. Springer-Verlag *Lecture Notes in Computer Science #7370*, 2012.

[BD76] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR 76-284, Department of Computer Science, Cornell University, Ithaca, NY, July 1976.

[Ber38] A. Bergson. A reformulation of certain aspects of welfare economics. *The Quarterly Journal of Economics*, 52(2):310–334, February 1938.

[BF83] S. Brams and P. Fishburn. *Approval Voting*. Birkhäuser, Boston, 1983.

[BFH+08] E. Brelsford, P. Faliszewski, E. Hemaspaandra, H. Schnoor, and I. Schnoor. Approximability of manipulating elections. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 44–49. AAAI Press, July 2008.

[BH11] M. Ballester and G. Haeringer. A characterization of the single-peaked domain. *Social Choice and Welfare*, 36(2):305–322, February 2011.

[BL07] M. Balinski and R. Laraki. A theory of measuring, electing, and ranking. *Proceedings of the National Academy of Sciences*, 104(21):8720–8725, May 2007.

[Bla48] D. Black. On the rationale of group decision-making. *Journal of Political Economy*, 56(1):23–34, 1948.

[Bla58] D. Black. *The Theory of Committees and Elections*. Cambridge University Press, 1958.

[BNW11] N. Betzler, R. Niedermeir, and G. Woeginger. Unweighted coalitional manipulation under the Borda rule is NP-hard. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 55–60, 2011.

[BO91] J. Bartholdi, III and J. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[BS06] S. Brams and R. Sanver. Critical strategies under approval voting: Who gets ruled in and ruled out. *Electoral Studies*, 25(2):287–305, 2006.

[BTT89a] J. Bartholdi, III, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[BTT89b] J. Bartholdi, III, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.

[BTT92] J. Bartholdi, III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical and Computer Modeling*, 16(8/9):27–40, 1992.

[BU09] N. Betzler and J. Uhlmann. Parameterized complexity of candidate control in elections and related digraph problems. *Theoretical Computer Science*, 410(52):43–53, 2009.

[Bus87] S. Buss. The boolean formula value problem is in ALOGTIME. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 123–131, May 1987.

[CKT91] P. Cheeseman, B. Kanefsky, and W. Taylor. Where the really hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 331–340, 1991.

[CS03] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 781–788. Morgan Kaufmann, August 2003.

[CS06] V. Conitzer and T. Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 627–634. AAAI Press, July 2006.

[CSL07] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):Article 14, 2007.

[DF61] M. Dummett and R. Farquharson. Stability in voting. *Econometrica*, 29(1):33–43, 1961.

[DF95] R. Downey and M. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[DF99] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[DK77] G. Doron and R. Kronick. Single transferable vote: An example of a perverse social choice function. *American Journal of Political Science*, XXI(2):303–311, 1977.

[DKNS01] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference*, pages 613–622. ACM Press, March 2001.

[DKNW11] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for Borda manipulation. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 657–662, August 2011.

[DS00] J. Duggan and T. Schwartz. Strategic manipulability without resoluteness or shared beliefs: Gibbard-Satterthwaite generalized. *Social Choice and Welfare*, 17(1):85–93, 2000.

[DS12] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 64(1):126–151, 2012.

[EF10a] E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *Internet and Network Economics*, pages 473–482. Springer-Verlag *Lecture Notes in Computer Science #6484*, 2010.

[EF10b] G. Erdélyi and M. Fellows. Parameterized control complexity in bucklin voting and in fallback voting. In *Proceedings of the 3rd International Workshop on Computational Social Choice*, pages 163–174, 2010.

[EFPR10] G. Erdlyi, M. Fellows, L. Piras, and J. Rothe. Control complexity in Bucklin and fallback voting. In *Proceedings of the Third International Workshop on Computational Social Choice*, page 44, 2010.

[EFS09] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory*, pages 299–310. Springer-Verlag *Lecture Notes in Computer Science #5814*, October 2009.

[EL05] E. Elkind and H. Lipmaa. Small coalitions cannot manipulate voting. In *Proceedings of the 9th International Conference on Financial Cryptography and Data Security*, pages 285–297. Springer-Verlag *Lecture Notes in Computer Science #3570*, February/March 2005.

[ELÖ08] B. Escoffier, J. Lang, and M. Öztürk. Single-peaked consistency and its complexity. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 366–370, July 2008.

[ENR09] G. Erdélyi, M. Nowak, and J. Rothe. Sincere-strategy preference-based approval voting fully resists constructive control and broadly resists destructive control. *Mathematical Logic Quarterly*, 55(4):425–443, 2009.

[EPR11] G. Erdélyi, L. Piras, and J. Rothe. The complexity of voter partition in Bucklin and fallback voting: Solving three open problems. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 837–844, May 2011.

[ER91] E. Ephrati and J. Rosenschein. The Clarke Tax as a consensus mechanism among automated agents. In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 173–178. AAAI Press, July 1991.

[ER10] G. Erdélyi and J. Rothe. Control complexity in fallback voting. In *Proceedings of the 16th Australasian Theory Symposium*, pages 39–48, January 2010.

[Fal08] P. Faliszewski. Nonuniform bribery (short paper). In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 1569–1572. International Foundation for Autonomous Agents and Multiagent Systems, May 2008.

[FHH09] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. How hard is bribery in elections? *Journal of Artificial Intelligence*, 35(1):485–532, 2009.

[FHH11a] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. The complexity of manipulative attacks in nearly single-peaked electorates. Technical Report TR-968, Department of Computer Science, University of Rochester, Rochester, NY, May 2011.

[FHH11b] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Multimode control attacks on elections. *Journal of Artificial Intelligence*, 40:305–351, 2011.

[FHHR07] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Llull and Copeland voting broadly resist bribery and control. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 724–730, July 2007.

[FHHR09] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence*, 35:275–341, 2009.

[FHHR11] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. *Information and Computation*, 209(2):89–107, 2011.

[FHS08] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: Ties matter. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 983–990, May 2008.

[FHS10] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Manipulation of Copeland elections. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 367–374, 2010.

[FKN08] E. Friedgut, G. Kalai, and N. Nisan. Elections can be manipulated often. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science*, pages 243–249, October 2008.

[Gar76] P. Gardenfors. Manipulation of social choice functions. *Journal of Economic Theory*, 13:217–228, 1976.

[Gib73] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41(4):587–601, 1973.

[Gib77] A. Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica*, 45(3):665–681, 1977.

[Gib78] A. Gibbard. Straightforwardness of game forms with lotteries as outcomes. *Econometrica*, 46(3):595–614, 1978.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[GKNW13] S. Gaspers, T. Kalinowski, N. Narodytska, and T. Walsh. Coalitional manipulation for Schulzes rule. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, May 2013. To appear.

[GMHS99] S. Ghosh, M. Mundhe, K. Hernandez, and S. Sen. Voting for movies: The anatomy of recommender systems. In *Proceedings of the 3rd Annual Conference on Autonomous Agents*, pages 434–435. ACM Press, 1999.

[GPP09] S. Galmard, J. Patty, and E. Penn. Arrow's theorem on single-peaked domains. In *The Political Economy of Democracy*, pages 335–342. 2009.

[HHM12] E. Hemaspaandra, L. Hemaspaandra, and C. Menton. Search versus decision for election manipulation problems. Technical Report TR-971, Department of Computer Science, University of Rochester, 2012.

[HHM13] E. Hemaspaandra, L. Hemaspaandra, and C. Menton. Search versus decision for election manipulation problems. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science*, pages 377–388, February/March 2013.

[HHR97] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.

[HHR07] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(5–6):255–285, 2007.

[HHR09] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Hybrid elections broaden complexity-theoretic resistance to control. *Mathematical Logic Quarterly*, 55(4):397–424, 2009.

[Hil05] C. Hillinger. The case for utilitarian voting. Discussion Papers in Economics 653, University of Munich, Department of Economics, May 2005.

[HLM13] L. Hemaspaandra, R. Lavaee, and C. Menton. Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, May 2013.

[HP01] G. Hägele and F. Pukelsheim. The electoral writings of Ramon Llull. *Studia Lulliana*, 41(97):3–38, 2001.

[HSV05] E. Hemaspaandra, H. Spakowski, and J. Vogel. The complexity of Kemeny elections. *Theoretical Computer Science*, 349(3):382–391, 2005.

[Hug80] R. Hughes. Rationality and intransitive preferences. *Analysis*, 40:132–134, 1980.

[Kem59] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:577–591, 1959.

[KS94] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random Boolean expressions. *Science*, 264(5163):1297–1301, 1994.

[Len83] H. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[Lif00] M. Lifantsev. Voting model for ranking web pages. In *Proceedings of the International Conference on Internet Computing*, pages 143–148, 2000.

[Lin11] A. Lin. Solving election manipulation using integer partitioning problems. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 593–590, 2011.

[LR78] A. LaPaugh and R. Rivest. The subgraph homeomorphism problem. In *Proceedings of the 10th ACM Symposium on Theory of Computing*, pages 40–50, 1978.

[LZ10] H. Liu and D. Zhu. Parameterized complexity of control problems in maximin election. *Information Processing Letters*, 110(10):383–388, 2010.

[McG53] D. McGarvey. A theorem on the construction of voting paradoxes. *Econometrica*, 21(4):608–610, 1953.

[Men09] C. Menton. Range voting is resistant to control. Master's thesis, December 2009.

[Men12] C. Menton. Normalized range voting broadly resists control. *Theory of Computing Systems*, December 2012.

[MS12] C. Menton and P. Singh. Manipulation can be hard in tractable voting systems even for constant-sized coalitions. *Computer Science Review*, 6(23):71–87, 2012.

[MS13] C. Menton and P. Singh. Control complexity of Schulze voting. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, August 2013. To appear.

[New92] J. Van Newenhizen. The Borda method is most likely to respect the Condorcet principle. *Economic Theory*, 2(1):69–83, 1992.

[Nie84] R. Niemi. The problem of strategic behavior under approval voting. *American Political Science Review*, 78(4):952–958, December 1984.

[Nie06] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[Nie10] R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*, pages 17–32, 2010.

[PHG00] D. Pennock, E. Horvitz, and C. Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 729–734. AAAI Press, July/August 2000.

[Pou08] W. Poundstone. *Gaming the Vote: Why Elections Aren't Fair (and What We Can Do About It)*. Hill and Wang, 2008.

[PR07a] A. Procaccia and J. Rosenschein. Average-case tractability of manipulation in voting via the fraction of manipulators. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007.

[PR07b] A. Procaccia and J. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence*, 28:157–181, 2007.

[Pro10] A. Procaccia. Can approximation circumvent Gibbard-Satterthwaite? In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 836–841, 2010.

[PX12] D. Parkes and L. Xia. A complexity-of-strategic-behavior comparison between Schulze's rule and ranked pairs. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1429–1435. AAAI Press, August 2012.

[Ris05] M. Risse. Why the Count de Borda cannot beat the Marquis de Condorcet. *Social Choice and Welfare*, 25(1):95–113, 2005.

[Saa06] D. Saari. Which is better: the Condorcet or Borda winner? *Social Choice and Welfare*, 26(1):107–129, 2006.

[Sat75] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

[Sch11] M. Schulze. A new monotonic, clone-independent, reversal symmetric, and Condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36(2):267–303, 2011.

[Sen69] A. Sen. Quasi-transitivity, rational choice and collective decisions. *Review of Economic Studies*, pages 381–393, July 1969.

[SFE11] I. Schlotter, P. Faliszewski, and E. Elkind. Campaign management under approval-driven voting rules. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 726–731, 2011.

[Sli04] A. Slinko. How large should a coalition be to manipulate an election? *Mathematical Social Sciences*, 47(3):289–293, May 2004.

[Smi73] J. H. Smith. Aggregation of preferences with variable electorate. *Econometrica*, 41(6):1027–1041, 1973.

[Smi00] W. Smith. Range voting. `http://www.math.temple.edu/~wds/homepage/rangevote.pdf`, November 2000.

[Szp10] G. Szpiro. *Numbers Rule*. Princeton University Press, 2010.

[Tid87] T. Tideman. Independence of clones as a criterion for voting rules. *Social Choice and Welfare*, 4(3):185–206, 1987.

[Tov84] C. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.

[Vic60] W. Vickrey. Utility, strategy, and social decision rules. *The Quarterly Journal of Economics*, 74(4):507–535, November 1960.

[vNM44] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[Wal09] T. Walsh. Where are the really hard manipulation problems? The phase transition in manipulating the veto rule. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 324–329, July 2009.

[Wik12] Wikipedia. Schulze method. `en.wikipedia.org/wiki/Schulze_method`, 2012.

[Wil72] R. Wilson. The game-theoretic structure of Arrow's general possibility theorem. *Journal of Economic Theory*, 5(1):14–20, August 1972.

[Woo94] D. Woodall. Properties of preferential election rules. *Voting matters*, 22(3):8–15, December 1994.

[XCP10] L. Xia, V. Conitzer, and A. Procaccia. A scheduling approach to coalitional manipulation. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, pages 275–284. ACM Press, June 2010.

[XZP+09] L. Xia, M. Zuckerman, A. Procaccia, V. Conitzer, and J. Rosenschein. Complexity of unweighted manipulation under some common voting rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 348–353, July 2009.

[YHL04] W. Yu, H. Hoogeveen, and J. Lenstra. Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. *Journal of Scheduling*, 7(5):333–348, 2004.

[YL78] H. Young and A. Levenglick. A consistent extension of Condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35(2):285–300, 1978.

[You75] H. Young. Social choice scoring functions. *SIAM Journal on Applied Mathematics*, 28(4):824–838, 1975.

[Zha01] W. Zhang. Phase transitions and backbones of 3-SAT and maximum 3-SAT. In *Principles and Practice of Constraint Programming—CP 2001*, volume 2239 of *Lecture Notes in Computer Science*, pages 153–167. 2001.

[ZLR10] M. Zuckerman, O. Lev, and J. Rosenschein. An algorithm for the coalitional manipulation problem under maximin. In *Proceedings of the 3rd International Workshop on Computational Social Choice*, pages 67–78, September 2010.

[ZPR09] M. Zuckerman, A. Procaccia, and J. Rosenschein. Algorithms for the coalitional manipulation problem. *Artificial Intelligence*, 173(2):392–412, 2009.