Essays on the Complexity of Voting Manipulation

Svetlana Obraztsova

Nanyang Technological University,
School of Physical and Mathematical Sciences

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for
the degree of Doctor of Philosophy

2012

Abstract

In their groundbreaking paper, Bartholdi, Tovey and Trick [6] argued that many well-known voting rules, such as Plurality, Borda, Copeland and Maximin are easy to manipulate. Following the direction proposed by this paper we examine the influence of features to which attention was not paid previously, namely, tie-breaking rules, and additional constraints, namely, the distance to the manipulator's true preferences, on the complexity of manipulating elections.

In Chapter 3 we show that all scoring rules, (simplified) Bucklin and Plurality with Runoff are easy to manipulate if the winner is selected from all tied candidates uniformly at random. This result extends to Maximin under an additional assumption on the manipulator's utility function that is inspired by the original model of [6]. In contrast, we show that manipulation under randomized tie-breaking is hard for Copeland, Maximin, STV and Ranked Pairs. In Chapter 4 we demonstrate that Plurality, Maximin, Copeland and Borda, as well as many families of scoring rules, become hard to manipulate if we allow arbitrary polynomial-time deterministic tie-breaking rules.

In Chapter 5, we investigate the complexity of optimal manipulation, i.e., finding a manipulative vote that achieves the manipulator's goal yet deviates as little as possible from her true ranking. We study this problem for three natural notions of closeness, namely, swap distance, footrule distance, and maximum displacement distance, and a variety of voting rules, such as scoring rules, Bucklin, Copeland, and Maximin. For all three distances, we obtain polynomial-time algorithms for all scoring rules and Bucklin and hardness results for Copeland and Maximin.

Acknowledgments

I am very grateful to my supervisors Dmitrii Pasechnik and Edith Elkind. I am especially thankful to Edith for her friendly support and endless patience which helped me to survive through the program

Contents

Chapter 1

Introduction

Social choice studies the aggregation of individual preferences to determine an overall collective decision. The history of social choice theory begins from the voting paradox, which was found by Marquis de Condorcet. He pointed out that transitivity, which exists for individual preferences, can be easily lost in the aggregation process. Another person who influenced social choice a lot almost at the same time as Marquis de Condorcet, was Chevalier de Borda (for example, he proposed the voting rule that is now named after him). They both considered elections as the most natural tool for aggregating individual preferences. More recently, Arrow stated his famous theorem, which is now known as Arrow's impossibility theorem (see [2]). He proved that it is impossible to design a voting rule that satisfies some very appealing properties. This theorem is often taken as a beginning of modern social choice theory.

Evidently, voting is not the only possible method for preference aggregation. For the survey and discussion of the other approaches we refer the reader to [32] and [8]. However, there are many settings where voting is the most appropriate approach to aggregating preferences, and in this thesis we will focus on voting, and, more specifically, algorithmic properties of various voting procedures. For a detailed description of voting procedures and mechanisms see [1].

Commonly, in an election we have some set of candidates and preferences (or votes), i.e., linear orders over the entire set of candidates, of all voters. A voting rule takes these preferences and gives us the winner of the election. Mostly, voting rules grant points to the candidates and the winner of an election is someone who has the highest score. Still, procedures of granting

points can have very different nature. For example, the widely used Plurality rule gives each candidate one point for every vote where he is ranked in the first position. On the other hand, Copeland is based on a completely different principle: In this case we consider all possible pairwise elections over the set of candidates and a candidate obtains a point for every pairwise elections that he wins.

It is easy to construct preferences that give us different outcomes of the election under different voting rules. For example, consider the set of candidates $\{Putin, Zuganov, Mironov, Medvedev\}$ and the following set of votes:

$$Putin \succ Zuganov \succ Mironov \succ Medvedev$$

$$Putin \succ Zuganov \succ Mironov \succ Medvedev$$

$$Zuganov \succ Mironov \succ Medvedev \succ Putin$$

$$Mironov \succ Zuganov \succ Medvedev \succ Putin$$

$$Medvedev \succ Zuganov \succ Mironov \succ Putin$$

Clearly, if we use Plurality to determine the winner then it would be Putin. In case of Copeland the victory would be shifted to Zuganov.

Therefore, we can see that careful choice of voting rule is really important. Understanding weak points of different voting rules as well as properties of election outcomes under these voting rules can help us make an optimal choice of voting method.

The example above also illustrates that under Plurality the last voter has a great incentive to lie about his real preferences, at least assuming his knowledge about the votes of others. If the last voter submits a vote with Zuganov at the first place and ties are broken in favor of Zuganov then he can change the outcome of the election. We can see that he prefers this outcome to the original one. That gives us an example of possibility of manipulating the election.

Evidently, it would be a great advantage for the voting rule to be resistant to manipulation. Unfortunately, such voting rules do not exist. It was proved by Gibbard [26] and Satterthwaite [39] that for elections with at least 3 candidates any non-dictatorial and surjective voting rule is manipulable. Even after this result hope remains that voting rules can withstand manipulation in practice if it is computationally difficult to find a manipulative vote. That was the motivation of one of the most influential early contributions

to computational social choice, namely the paper by Bartholdi, Tovey, and Trick entitled "The computational difficulty of manipulating an election" [6]. In this paper, the authors suggested that computational complexity can serve as a barrier to dishonest behavior by the voters, and proposed classifying voting rules according to how difficult it is to manipulate them. In particular, they argued that such well-known voting rules as Plurality, Borda, Copeland and Maximin are easy to manipulate, yet a variant of the Copeland rule known as second-order Copeland is computationally resistant to manipulation. In a subsequent paper, Bartholdi and Orlin [5] showed that another well-known voting rule, namely, STV, is NP-hard to manipulate as well.

Since then, the computational complexity of manipulation under various voting rules, either by a single voter or by a coalition of voters, received considerable attention in the literature, both from the theoretical and from the experimental perspective (see, in particular, [49, 48] and the recent survey [21, 23] for the former, and [44, 12] for the latter). While it has been argued that worst-case complexity does not provide adequate protection against malicious behavior (see, e.g. [37, 46, 24, 29]), determining whether a given voting rule is NP-hard to manipulate is still a natural first step in evaluating its resistance to manipulation in realistic scenarios.

This thesis continues and refines this line of research. We examine the influence of features to which attention was not paid previously, namely, tie-breaking rules, and additional constraints, namely, the distance to the manipulator's true preferences, on the complexity of manipulating elections.

1.1 Tie-breaking rules

Many common voting rules operate by assigning scores to candidates, so that the winner is the candidate with the highest score. Now, in elections with a large number of voters and a small number of candidates there is usually only one candidate that obtains the top score. However, this does not necessarily hold when the alternative space is large, as may be the case when, e.g., agents in a multiagent system use voting to decide on a joint plan of action [20]. This does not hold either in the elections where the number of voters is small and the number of alternatives is not large. If, nevertheless, a single outcome needs to be selected, such ties have to be broken. In the context of manipulation, this means that the manipulator should take the tie-breaking rule into account when choosing his actions. Much of the existing

literature on voting manipulation circumvents the issue by assuming that the manipulator's goal is to make some distinguished candidate p one of the election winners, or, alternatively, the unique winner. The former assumption can be interpreted as a tie-breaking rule that is favorable to the manipulator, i.e., given a tie that involves p, always selects p as the winner; similarly, the latter assumption corresponds to a tie-breaking rule that is adversarial to the manipulator. In fact, most of the existing algorithms for finding a manipulative vote work for any tie-breaking rule that selects the winner according to a given ordering on the candidates (such tie-breaking rules are known as *lexicographic*); the two cases considered above correspond to this order being, respectively, the manipulator's preference order or its inverse.

In Chapter 3 we study an equally appealing approach to breaking ties, namely, selecting the winner among all tied candidates uniformly at random. Note that under randomized tie-breaking the outcome of the election is a random variable, so it is not immediately clear how to compare two outcomes: is having your second-best alternative as the only winner preferable to the lottery in which your top and bottom alternatives have equal chances of winning? In this thesis we propose to deal with this issue by augmenting the manipulator's preference model: we assume that the manipulator assigns a numeric utility to all candidates, and his goal is to vote so as to maximize his expected utility, where the expectation is computed over the random choices of the tie-breaking procedure; this approach is standard in the social choice literature (see, e.g., [27]) and has also been used in [14]. We show that in this setting any scoring rule and Bucklin are easy to manipulate, and so is the Maximin rule, assuming that the manipulator assigns 1 unit of utility to one candidate and utility 0 to all other candidates. In contrast, we provide NP-hardness results on the complexity of manipulating Maximin for general utilities as well as Copeland and several iterative voting rules. Our results for randomized tie-breaking can be summarized by Table 1.1.

In Chapter 4, we focus on deterministic tie-breaking rules. The easiness results obtained in [6] after careful examination can be extended to arbitrary lexicographic tie-breaking rules. Given these easiness results, it is natural to ask whether all (efficiently computable) deterministic tie-breaking rules produce easily manipulable rules when combined with the voting correspondences considered in [6]. Now, paper [6] shows that for Copeland this is not the case, by proving that the second-order Copeland rule is hard to manipulate. However, prior to our work, no such result was known for other rules considered in [6]. We demonstrate that Maximin and Borda, as well

P	NP-hard
Scoring rules	Copeland
Maximin (restricted)	Maximin (general)
simplified Bucklin	STV
classic Bucklin	Ranked Pairs
Plurality w/Runoff	

Table 1.1: Summary of results of Chapter 3.

as many families of scoring rules, become hard to manipulate if we allow arbitrary polynomial-time deterministic tie-breaking rules. This holds even if we require that the tie-breaking rule only depends on the set of the tied alternatives, rather than the voters' preferences over them; we will refer to such tie-breaking rules as *simple*. Our proof also works for Copeland, thus strengthening the hardness result of [6] to simple tie-breaking rules. We remark, however, that our hardness result is not universal: Plurality and other scoring rules that correspond to scoring vectors with a bounded number of non-zero coordinates are easy to manipulate under any polynomial-time simple tie-breaking rule, However, if non-simple tie-breaking rules are allowed, Plurality can be shown to be hard to manipulate as well.

1.2 Minimizing the distance to the true preferences

In Chapter 5 we study a refinement of the question asked by Bartholdi, Tovey and Trick. We observe that, while the manipulator is willing to lie about her preferences, she may nevertheless prefer to submit a vote that deviates as little as possible from her true ranking. Indeed, if voting is public (or if there is a risk of information leakage), and a voter's preferences are at least somewhat known to her friends and colleagues, she may be worried that voting non-truthfully can harm her reputation—yet hope that she will not be caught if her vote is sufficiently similar to her true ranking. Alternatively, a voter who is uncomfortable about manipulating an election for ethical reasons may find a lie more palatable if it does not require her to re-order more than a few candidates. Finally, a manipulator may want to express support for candidates she truly likes, even if these candidates have no chances of

winning; while she may lie about her ranking, she would prefer to submit a vote where her most preferred candidates are ranked close to the top.

These scenarios suggest the following research question: does a voting rule admit an efficient algorithm for finding a manipulative vote that achieves the manipulator's goals, yet deviates from her true ranking as little as possible? To make this question precise, we need to decide how to measure the discrepancy between the manipulator's true preferences and her actual vote. Mathematically speaking, votes are permutations of the candidate set, and there are several *distances* on permutations that one can use. In our work, we consider what is arguably the two most prominent distances on votes, namely, the *swap distance* [30] (also known as bubble-sort distance, Kendall distance, etc.) and the *footrule distance* [42] (also known as the Spearman distance), as well as a natural variation of the footrule distance, which we call the *maximum displacement* distance.

In more detail, the swap distance counts the number of candidate pairs that are ranked differently in two preference orderings. Thus, when the manipulator chooses her vote based on the swap distance, she is trying to minimize the number of swaps needed to transform her true ranking into the manipulative vote. We remark that for swap distance, our problem can be viewed as a special case of the swap bribery problem [18]; however, our question is not addressed by existing complexity results for swap bribery [18, 17, 16, 40]. The footrule distance and the maximum displacement distance are based on computing, for each candidate, the absolute difference between his positions in the two votes; the footrule distance then computes the sum of these quantities, over all candidates, while the maximum displacement distance returns the largest of them. We believe that each of these distances captures a reasonable approach to defining what it means for two votes to be close to each other; therefore, we are interested in analyzing the complexity of our manipulation problem for all of them.

We study our problem for several classic voting rules, namely, Bucklin, Copeland, Maximin, as well as all scoring rules. For all these rules, the algorithm of Bartholdi et al. [6] finds a successful manipulation if it exists. However, this algorithm does not necessarily produce a vote that is optimal with respect to any of our distance measures: in particular, it always ranks the manipulator's target candidate first, even if this is not necessary to achieve the manipulator's goal. Thus, we need to devise new algorithms—or prove that finding an optimal manipulation is computationally hard.

We investigate the complexity of optimal voting manipulation for three

distance measures on votes and four types of voting rules. For all three distances, we obtain the same classification of these rules with respect to the complexity of finding an optimal manipulation: our problem is easy for Bucklin and all polynomial-time computable families of scoring rules (see Chapter 2 for definitions), but hard for Copeland and Maximin. For swap distance and footrule distance, we strengthen these hardness results to show that our problem is, in fact, hard to approximate up to a factor of $\Omega(\log m)$, where m is the number of candidates.

Our results can be summarized by the following table:

	Sc. rules	Bucklin	Copeland	Maximin
$\mathrm{d}_{\mathrm{swap}}$	P	P	$\Omega(\log m)$-inapp.	$\Omega(\log m)$-inapp.
d_{fr}	P	P	$\Omega(\log m)$-inapp.	$\Omega(\log m)$-inapp.
d_{md}	P	P	NPC	NPC

Table 1.2: Summary of results of Chapter 5

1.3 Thesis structure

This thesis is organized as follows. Chapter 2 contains the preliminaries and consists mostly of definitions and discussion of different approaches to defining manipulation problems in social choice. Chapter 3 and Chapter 4 describe the influence of tie-breaking rules on the complexity of voting manipulation. In Chapter 3 we focus on randomized tie-breaking. Arbitrary deterministic polynomial-time computable tie-breaking rules are analyzed in Chapter 4. Chapter 5 describes algorithms and hardness results for the optimal manipulation problem. In Chapter 6 we discuss directions for future work.

Chapter 2

Preliminaries

An *election* is given by a set of candidates $C = \{c_1, \ldots, c_m\}$ and a vector $\mathcal{R} = (R_1, \ldots, R_n)$, where each R_i, $i = 1, \ldots, n$, is a linear order over C; R_i is called the *preference order* (or, *vote*) of voter i. We denote the space of all linear orderings over C by $\mathcal{L}(C)$. The vector $\mathcal{R} = (R_1, \ldots, R_n)$ is called a *preference profile*. For readability, we will sometimes denote R_i by $\succ_i$. When $a \succ_i b$ for some $a, b \in C$, we say that voter i prefers a to b. We denote by $r(c_j, R_i)$ the *rank* of candidate c_j in the preference order R_i: $r(c_j, R_i) = |\{c \in C \mid c \succ_i c_j\}| + 1$.

A *voting rule* $\mathcal{F}$ is a mapping that, given a preference profile $\mathcal{R}$ over C, outputs a candidate $c \in C$; we write $c = \mathcal{F}(\mathcal{R})$. Many classic voting rules, such as the ones defined below, are, in fact, *voting correspondences*, i.e., they map a preference profile $\mathcal{R}$ to a non-empty subset S of C. Voting correspondences can be transformed into voting rules using tie-breaking rules.

A *tie-breaking rule* for an election $(C, \mathcal{R})$ is a mapping $T = T(\mathcal{R}, S)$ that for any $S \subseteq C$, $S \neq \emptyset$, outputs a candidate $c \in S$. A tie-breaking rule T is called *simple* if it does not depend on $\mathcal{R}$, i.e., the value of $T(\mathcal{R}, S)$ is uniquely determined by S. Such rules have the attractive property that if a manipulator cannot change the set of tied candidates, he cannot affect the outcome of the election. Further, we say that T is *lexicographic* with respect to a preference ordering $\succ$ over C if for any preference profile $\mathcal{R}$ over C and any $S \subseteq C$ it selects the most preferred candidate from S with respect to $\succ$, i.e., we have $T(S) = c$ if and only if $c \succ a$ for all $a \in S \setminus \{c\}$.

A *composition* of a voting correspondence $\mathcal{F}$ and a tie-breaking rule T is a voting rule $T \circ \mathcal{F}$ that, given a preference profile $\mathcal{R}$ over C, outputs $T(\mathcal{R}, \mathcal{F}(\mathcal{R}))$. Clearly, $T \circ \mathcal{F}$ is a voting rule and $T \circ \mathcal{F}(\mathcal{R}) \in \mathcal{F}(\mathcal{R})$.

2.1 Voting rules

We will now describe the voting rules (correspondences) considered in this thesis. All these rules assign scores to candidates; the winners are the candidates with the highest scores.

Scoring rules Any vector $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^m$ with $\alpha_1 \geq \cdots \geq \alpha_m$ defines a *scoring rule* $\mathcal{F}_\alpha$. Under this rule, each voter grants α_i points to the candidate it ranks in the i-th position; the score of a candidate is the sum of the scores it receives from all voters. The vector α is called a *scoring vector*. A scoring rule is said to be *faithful* if $\alpha_1 > \cdots > \alpha_m$. We are interested in scoring rules that are succinctly representable; therefore, throughout this paper we assume that the coordinates of α are nonnegative integers given in binary. We remark that scoring rules are defined for a fixed number of candidates. Therefore, we will often consider families of scoring rules, i.e., collections of the form $(\alpha^m)_{m=1}^{\infty}$, where $\alpha^m = (\alpha_1^m, \ldots, \alpha_m^m)$. We require such families to be polynomial-time computable, i.e., we only consider families of voting rules $(\alpha^m)_{m=1}^{\infty}$ for which there exists a polynomial-time algorithm that given an $m \in \mathbb{N}$ outputs $\alpha_1^m, \ldots, \alpha_m^m$. A well-known example of a polynomial-time computable family of scoring rules is *Borda*, given by $\alpha^m = (m-1, \ldots, 1, 0)$.

k-approval, Plurality and Bucklin Under the k-approval rule, a candidate gets one point for each voter that ranks him in the top k positions; 1-approval is also known as Plurality. It is easy to see that k-approval and Plurality are examples of families of scoring rules. Let k^* be the smallest value of k such that some candidate's k-approval score is at least $\lfloor n/2 \rfloor + 1$; we will say that k^* is the *Bucklin winning round*. Under the simplified Bucklin rule, the winners are all candidates whose k^*-approval score is at least $\lfloor n/2 \rfloor + 1$; under the Bucklin rule, the winners are all k^*-approval winners. A candidate's *Bucklin score* is his k^*-approval score.

Copeland and second-order Copeland We say that a candidate a wins a *pairwise election* against b if more than half of the voters prefer a to b; if exactly half of the voters prefer a to b, then a is said to *tie* his pairwise election against b. Given a rational value $\alpha \in [0, 1]$, under the

Copeland$^\alpha$ rule each candidate gets 1 point for each pairwise election he wins and α points for each pairwise election he ties.

A candidate's *second-order Copeland score* is the sum of the Copeland scores of the competitors he defeats. Under the second-order Copeland rule, the winner is chosen among the candidates with the highest Copeland scores, breaking ties according to the second-order Copeland score. We follow the definition of second-order Copeland voting rule in [6]. In this paper the examples of using second-order Copeland can be also found.

Maximin and Ranked Pairs

For every pair of candidates $(c, d) \in C$, we define $s(c, d)$ as $|\{i \mid c \succ_i d\}|$.

The Maximin score of a candidate $c \in C$ is equal to the number of votes he gets in his worst pairwise election; in other words, his score equals $\min_{d \in C \setminus \{c\}} s(c, d)$.

Ranked Pairs was firstly considered by T. N. Tideman in [43]. For Ranked Pairs the election proceeds in several steps. This rule first creates an entire ranking of all the candidates, as follows. In each step, we consider a pair of candidates c, d that we have not previously considered (as a pair): specifically, we choose the remaining pair with the highest $s(c, d)$ (note that there may be several such pairs; we will comment on this issue in Chapter 3). We then fix the order c, d, unless this contradicts previous orders that we fixed (that is, it violates transitivity). We continue until we have considered all pairs of candidates (hence, in the end, we have a full ranking). The candidate at the top of the ranking wins.

Plurality with Runoff and STV Under the STV rule, the election proceeds in rounds. During each round, the candidate with the lowest Plurality score is eliminated, and the candidates' Plurality scores are recomputed. The winner is the candidate that survives till the last round. Plurality with Runoff can be thought of as a compressed version of STV: we first select two candidates with the highest Plurality scores, and then output the winner of the pairwise election between them. Note that these definitions are somewhat ambiguous, as several candidates may have the lowest/highest Plurality score; we will comment on this issue in Section 3.6.

2.2 Manipulation

Given a preference profile $\mathcal{R}$ over a set of candidates C, for any preference order L over C we denote by $(\mathcal{R}_{-i}, L)$ the preference profile obtained from $\mathcal{R}$ by replacing R_i with L. We say that a voter $i \in \{1, \ldots, n\}$ can successfully *manipulate* an election $(C, \mathcal{R})$ with a preference profile $(R_1, \ldots, R_n)$ with respect to a voting rule $\mathcal{F}$ if $\mathcal{F}(\mathcal{R}_{-i}, L) \succ_i \mathcal{F}(\mathcal{R})$. We will now define the computational problem that corresponds to this notion.

All voting rules defined in Section 2.1 are anonymous, so, we can fix any candidate as the manipulator. Thus, we fix voter n as the manipulator and we will make this assumption throughout the thesis.

Definition 2.2.1. *Let $\mathcal{F}$ be a voting rule. An instance of the $\mathcal{F}$-*MANIPULATION$^{\succ}$ *problem is given by a set of candidates C and a preference profile $\mathcal{R}$. The question is whether there exists a vote $L \in \mathcal{L}(C)$ such that $\mathcal{F}(\mathcal{R}_{-n}, L) \succ_n \mathcal{F}(\mathcal{R})$.*

Our definition of $\mathcal{F}$-MANIPULATION$^{\succ}$ is modeled after the standard social choice definition, see, e.g. [26, 39]. However, in the computational social choice literature it is usual to consider the decision problem where a manipulator focuses on a candidate p and his goal is to make p elected; we will refer to this problem as $\mathcal{F}$-MANIPULATION (see, e.g., [6]).

These problems are closely related. First, a polynomial-time algorithm for $\mathcal{F}$-MANIPULATION can be converted into a polynomial-time algorithm for $\mathcal{F}$-MANIPULATION$^{\succ}$ if the number of candidates is polynomial in the size of the output: we can simply run $\mathcal{F}$-MANIPULATION on all candidates ranked by the manipulator above the current winner, and pick the best among the candidates for which $\mathcal{F}$-MANIPULATION outputs "yes". Thus, if $\mathcal{F}$-MANIPULATION$^{\succ}$ is hard, $\mathcal{F}$-MANIPULATION is hard, too.

On the other hand, having a polynomial-time algorithm for the optimization version of $\mathcal{F}$-MANIPULATION$^{\succ}$, in which we ask who is the best candidate (from the manipulator's perspective) that can be made a winner, is sufficient for solving $\mathcal{F}$-MANIPULATION. Indeed, suppose that we are given an instance of $\mathcal{F}$-MANIPULATION. Consider the election in which the manipulator n submits an arbitrary vote L that ranks p first. If p wins in this election, then we are done. Otherwise, let w be the election winner. If w is ranked second in L, then p can be made the winner if and only if our election is a "yes"-instance of $\mathcal{F}$-MANIPULATION$^{\succ}$. Otherwise, consider the election obtained by promoting w into the second position in

L, i.e., one where voter n ranks p first and w second. We know that the manipulator can make w the winner by voting L. Thus, n's favorite candidate that can be made the winner is either p or w, and therefore we can solve $\mathcal{F}$-MANIPULATION using an algorithm for the optimization version of $\mathcal{F}$-MANIPULATION$^{\succ}$. Further, if $\mathcal{F}$ is monotone, i.e., the election winner continues to win if we move him up in all voters' preferences without changing the relative order of other candidates, then it suffices to have an algorithm for the decision version of $\mathcal{F}$-MANIPULATION$^{\succ}$: in this case, w remains the winner after we push him upwards in L, and hence the resulting instance is a "yes"-instance of $\mathcal{F}$-MANIPULATION$^{\succ}$ if and only if n can make p the winner. However, it remains unclear if in general $\mathcal{F}$-MANIPULATION can be reduced to $\mathcal{F}$-MANIPULATION$^{\succ}$ and hence the existing NP-hardness results for second-order Copeland [6] and STV [5] do not directly imply that second-order Copeland-MANIPULATION$^{\succ}$ and STV-MANIPULATION$^{\succ}$ are NP-hard.

Both problems stated above are defined for voting rules. The manipulation problems for voting correspondences are also widely discussed. There are two standard approaches to extending the manipulation problem to voting correspondences. Under the first approach, we assume that the manipulator's goal is to make the specific candidate p the only winner of the election.

Definition 2.2.2. *Let $\mathcal{F}$ be a voting correspondence. In the* $\mathcal{F}$-UNIQUEWINNERMANIPULATION *problem, we are given an election $E = (C, \mathcal{R})$ with a preference profile $\mathcal{R} = (R_1, \ldots, R_n)$, and a preferred candidate $p \in C$. The question is whether there exists a vote $L \in \mathcal{L}(C)$ such that the preference profile $\mathcal{R}' = (R_1, \ldots, R_{n-1}, L)$ satisfies $\{p\} = \mathcal{F}(\mathcal{R}')$.*

Under the second approach, it is assumed that it is enough for the manipulator to make candidate p one of the election winners.

Definition 2.2.3. *Let $\mathcal{F}$ be a voting correspondence. In the* $\mathcal{F}$-COWINNERMANIPULATION *problem, we are given an election $E = (C, \mathcal{R})$ with a preference profile $\mathcal{R} = (R_1, \ldots, R_n)$, and a preferred candidate $p \in C$. The question is whether there exists a vote $L \in \mathcal{L}(C)$ such that the preference profile $\mathcal{R}' = (R_1, \ldots, R_{n-1}, L)$ satisfies $p \in \mathcal{F}(\mathcal{R}')$.*

It is easy to see that $\mathcal{F}$-UNIQUEWINNERMANIPULATION is equivalent to $\mathcal{F}'$-MANIPULATION, where the voting rule $\mathcal{F}'$ is obtained by combining $\mathcal{F}$ with the lexicographic tie-breaking rule that is adversarial to manipulator, i.e., breaks ties according to the ordering obtained by reversing R_n. Similarly,

in the case of $\mathcal{F}$-CoWinnerManipulation we take the composition of $\mathcal{F}$ with the lexicographic tie-breaking rule that favors the manipulator, i.e., breaks ties according to the preference order R_n.

2.2.1 The model and the algorithm of Bartholdi, Tovey and Trick

We will now describe the algorithm for $\mathcal{F}$-CoWinnerManipulation proposed in [6]. This algorithm can be used for any voting correspondence $\mathcal{F}$ that assigns scores to candidates, so that the winners are the candidates with the highest scores. The algorithm places the manipulator's preferred candidate p first, and then fills in the remaining positions in the vote from top to bottom, searching for a candidate that can be placed in the next available position in the n-th vote so that his score does not exceed that of p. This approach works as long as the voting correspondence $\mathcal{F}$ is monotone and we can determine a candidate's final score given his position in the manipulator's vote and the identities of the candidates that the manipulator ranks above him. It is not hard to show that Plurality and Borda (and, in fact, all scoring rules), as well as Plurality with Runoff, Copeland and Maximin have this property. Simplified Bucklin and Bucklin do not satisfy this property, but they are easy to manipulate as well. For example, the algorithm for $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Bucklin})$-OptManipulation described in Chapter 5 can be used for proving the easiness of the manipulation problem.

We can easily modify this algorithm to make it work for $\mathcal{F}$-UniqueWinnerManipulation: in that case, when the manipulator fills a position j in his vote, $j > 1$, he needs to ensure that the score of the candidate in that position is strictly less than that of p. Generally, if ties are broken according to a lexicographic ordering $\succ$ over the candidates, when placing a candidate c with $c \succ p$, the manipulator needs to make sure that c's score is less than that of p, and when placing a candidate c with $c \prec p$, he needs to make sure that c's score does not exceed that of p.

In the same paper Bartholdi, Tovey and Trick argued that second-order Copeland is computationally resistant to manipulation. In a subsequent paper, Bartholdi and Orlin [5] showed that another well-known voting rule, namely, STV, is NP-hard to manipulate as well. In [49] Xia et al. showed that Ranked Pairs is hard to manipulate (for the defenition of Ranked Pairs given in this thesis).

2.3 McGarvey theorem

Many proofs in this thesis make use of McGarvey theorem. We will now state this theorem and give a sketch of its proof.

Let C be a set of candidates and let $\mathcal{R} = (R_1, \ldots, R_n)$ be a preference profile.

Definition 2.3.1. *We say that a candidate c_i* wins *the pairwise election against a candidate c_j if more that half of the voters in $\mathcal{R}$ rank c_i above c_j.*

Definition 2.3.2. *The digraph H is said to be* induced by $\mathcal{R}$ *if C is a set of vertices of digraph H and H contains an arc (c_i, c_j) if and only if c_i is the winner of the pairwise election between c_i and c_j.*

Now we can state McGarvey theorem (see also [33] or [35]).

Theorem 2.3.3. *For any digraph H_m on m vertices without cycles of length 2 there exists a preference profile $\mathcal{R}$ such that $\mathcal{R}$ consists of at most $m(m-1)$ votes and H_m is induced by $\mathcal{R}$.*

Proof. We give only a sketch of the proof of this theorem. Let the vertex set of digraph H_m be $C = \{c_1, \ldots, c_m\}$.

First consider the preference profile that consists of votes described as follows. For every arc (c_i, c_j) of H_m we take votes $c_i \succ c_j \succ c_1 \succ \ldots \succ c_m$ and $c_m \succ \ldots \succ c_1 \succ c_j \succ c_i$. For this profile we have ties in every pairwise election. It is easy to see that we can make either c_i or c_j the winner of their pairwise election by swapping these candidates in the suitable vote. As a result, the score of the winner in the pairwise election would exceed the score of the loser by exactly 2. □

Consider the simple example of using McGarvey theorem. We construct the preference profile with candidates $\{c_1, c_2, c_3, c_4\}$ for the graph on 4 vertices with adjacency matrix defined as follows:

$$(a_{i,j}) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0. \end{pmatrix}$$

At the first step of algorithm we obtain the preference profile which gives tie in every pairwise election. We set

$$R_{1,2} = c_1 \succ c_2 \succ c_3 \succ c_4 R_{2,1} = c_4 \succ c_3 \succ c_2 \succ c_1 R_{1,3} = c_1 \succ c_3 \succ c_2 \succ c_4 R_{3,1} = c_4 \succ c_2 \succ c_3 \succ c_1 R_{1,4} =$$

Afterward following the algorithm we swap c_i, c_j in $R_{j,i}$ for all $a_{i,j} = 1$. For example, consider $a_{1,2} = 1$ we obtain $R_{2,1} = c_4 \succ c_3 \succ c_1 \succ c_2$ instead of $R_{2,1} = c_4 \succ c_3 \succ c_2 \succ c_1$. Thereby we come up with the following profile as a result of the algorithm.

$$R_{1,2} = c_1 \succ c_2 \succ c_3 \succ c_4 R_{2,1} = c_4 \succ c_3 \succ c_1 \succ c_2 R_{1,3} = c_1 \succ c_3 \succ c_2 \succ c_4 R_{3,1} = c_4 \succ c_2 \succ c_1 \succ$$

It is easy to see that the following corollary can be proved similarly to Theorem 2.3.3.

Corollary 2.3.4. *For any digraph H_m with vertex set $C = \{c_1, \ldots, c_m\}$ that does not have cycles of length 2 there exists an election $E = (C, \mathcal{R})$ with a preference profile $\mathcal{R} = (R_1, \ldots, R_n)$, where n is even and polynomial in m, such that if H_m contains the arc (c_i, c_j) candidate c_i obtains exactly $\frac{n}{2} + 1$ points in the pairwise election against c_j, and if H_m does not contain an arc between c_i and c_j then there is a tie in the pairwise election.*

Also we can derive the following corollary from the proof of Theorem 2.3.3.

Corollary 2.3.5. *For any digraph H_m with vertex set $C = \{c_1, \ldots, c_m\}$ and any set of numbers $\{k_{i,j} \mid i, j : 1, \ldots, m\}$ there exists an election $E = (C, \mathcal{R})$ with a preference profile $\mathcal{R} = (R_1, \ldots, R_n)$ where n is even and polynomial in m and $\max_{i,j=1\ldots m} k_{i,j}$, such that for any arc (c_i, c_j) candidate c_i obtains exactly $\frac{n}{2} + k_{i,j}$ points in the pairwise election against c_j and if H_m does not contain an arc between c_i and c_j then there is a tie in the pairwise election.*

Better bounds on the number of voters needed to construct a profile that induces a given digraph can be found in [35]. In [13] author have considered the case of some given preorders for the voters and gave the upper bound for the number of voter which we need to add to the election due to obtaining a given digraph. this result also fall into the realm of corollaries above.

2.4 Gibbard-Satterthwaite Theorem

We remark, that one of the reason which support interest in computational complexity of voting manipulation is the famous Gibbard-Satterthwaite theorem. Informally, this theorem said that manipulation is almost always exists. The exact formulation of this theorem is as follows.

Definition 2.4.1. *A voting rule $\mathcal{F}$ is dictatorial if there is a voter i (the dictator) such that $\mathcal{F}(\mathcal{R}) \succ_i c_j$ for all $c_j \in C \setminus \{\mathcal{F}(\mathcal{R})\}$.*

Definition 2.4.2. *A voting rule $\mathcal{F}$ is onto if for each a candidate c_i there exists a preference profile $\mathcal{R}$ such that $\mathcal{F}(\mathcal{R}) = c_i$.*

Now we can state Gibbard-Satterthwaite theorem (see [26, 39]).

Theorem 2.4.3. *A non-manipulable voting rule that is onto is dictatorial if the number of candidates is at least three.*

Chapter 3

Randomized Tie-Breaking Rules

3.1 The model

In this chapter we discuss the complexity of manipulating elections under a very common approach to tie-breaking, namely, choosing the winner uniformly at random among all tied candidates. In this case, knowing the manipulator's preference ordering is not sufficient to determine his optimal strategy. For example, suppose that voter n prefers a to b to c, and by voting strategically he can change the output of the voting correspondence from b to $\{a, c\}$. It is not immediately clear if this manipulation is beneficial. Indeed, if voter n strongly prefers a, but is essentially indifferent between b and c, then the answer is probably positive, but if voter n strongly dislikes c and slightly prefers a to b, the answer is likely to be negative (of course, this also depends on n's risk attitude).

Thus, to model this situation appropriately, we need to know the utilities that the manipulator assigns to all candidates. Under the natural assumption of risk neutrality, the manipulator's utility for a set of candidates is equal to his expected utility when the candidate is drawn from this set uniformly at random, or, equivalently, to his *average* utility for a candidate in this set. Since we are interested in computational issues, it is reasonable to assume that all utilities are rational numbers; by scaling, we can assume that all utilities are positive integers given in binary.

Formally, given a set of candidates C, we assume that the manipulator is endowed with a utility function $u : C \to \mathbb{N}$. This function can be extended to sets of candidates by setting $u(S) = \frac{1}{|S|} \sum_{c \in S} u(c)$ for any $S \subseteq C$.

Definition 3.1.1. *Given a voting correspondence $\mathcal{F}$ and an election $(C, \mathcal{R})$, we say that a vote L is* optimal *for a manipulating voter n with a utility function $u : C \to \mathbb{N}$ with respect to $\mathcal{F}$ combined with the randomized tie-breaking rule if $u(\mathcal{F}(\mathcal{R}_{-n}, L)) \geq u(\mathcal{F}(\mathcal{R}_{-n}, L'))$ for all $L' \in \mathcal{L}(C)$. We say that the manipulator has a* successful manipulation *if his optimal vote L satisfies $u(\mathcal{F}(\mathcal{R}_{-n}, L)) > u(\mathcal{F}(\mathcal{R}))$.*

Now we can state the problem of finding a successful manipulation.

Definition 3.1.2. *An instance of the $\mathcal{F}$-*RandManipulation *problem is a tuple (E, u, q), where $E = (C, \mathcal{R})$ is an election, $u : C \to \mathbb{N}$ is the manipulator's utility function such that $u(c) \geq u(c')$ if and only if $c \succ_n c'$, and q is a non-negative rational number. It is a "yes"-instance if there exists a vote L such that $u(\mathcal{F}(\mathcal{R}_{-n}, L)) \geq q$ and a "no"-instance otherwise.*

*In the optimization version of $\mathcal{F}$-*RandManipulation*, the goal is to find an optimal vote.*

We remark that $\mathcal{F}$-RandManipulation is in NP for any polynomial-time computable voting correspondence $\mathcal{F}$: it suffices to guess the manipulative vote L, determine the set $S = \mathcal{F}(\mathcal{R}_{-n}, L)$, and compute the average utility of the candidates in S.

In the rest of this chapter, we will explore the complexity of finding an optimal vote with respect to scoring rules, Bucklin, Maximin, Copeland and several iterative rules under the randomized tie-breaking rule.

3.2 Scoring rules

All scoring rules turn out to be easy to manipulate under randomized tie-breaking.

Theorem 3.2.1. *For any scoring vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ $\mathcal{F}_\alpha$-*RandManipulation *is in* P.

Proof. Recall that we assume that the manipulator is the last voter n with a utility function u, and let $\mathcal{R}'$ denote the preference profile consisting of all other voters' preferences. Let s_i denote the score of candidate c_i after all voters other than n have cast their vote. Let us renumber the candidates in order of increasing score, and, within each group with the same score, in order of decreasing utility. That is, under the new ordering we have

$s_1 \leq \cdots \leq s_m$ and if $s_i = s_j$ for some $i < j$ then $u(c_i) \geq u(c_j)$. We say that two candidates c_i, c_j with $s_i = s_j$ belong to the same *level*. Thus, all candidates are partitioned into $h \leq m$ levels $H_1, \ldots, H_h$, so that if $c_i \in H_k$ and $c_j \in H_\ell$, $k < \ell$, then $s_i < s_j$.

Consider first the vote L_0 given by $c_1 \succ \ldots \succ c_m$, and let T be the number of points obtained by the winner(s) in $(\mathcal{R}', L_0)$. We claim that for any $L \in \mathcal{L}(C)$, in the preference profile $(\mathcal{R}', L)$ the winner(s) will get at least T points. Indeed, let c_i be the last candidate to get T points in $(\mathcal{R}', L_0)$, and suppose that there exists a vote L such that c_i gets less than T points in $(\mathcal{R}', L)$. By the pigeonhole principle, this means that L assigns at least α_i points to some c_j with $j > i$, and we have $s_j + \alpha_i \geq s_i + \alpha_i = T$, i.e., some other candidate gets at least T points, as claimed. We will say that a vote L is *conservative* if the winners' score in $(\mathcal{R}', L)$ is T.

We will now describe possible optimal votes for the manipulator.

Lemma 3.2.2. *If L maximizes the utility of voter n, then either L is conservative or it can be chosen so that $\mathcal{F}_\alpha$ has a unique winner under $(\mathcal{R}', L)$.*

Proof. Suppose that this is not the case, i.e., any vote L that maximizes the manipulator's utility is such that the set $S = \mathcal{F}_\alpha(\mathcal{R}', L)$ is of size at least 2, and all candidates in S get $T' > T$ points. Let c_i be n's most preferred candidate in S; we have $u(c_i) \geq u(S)$. Suppose that L grants α_j points to c_i. Since we have $s_i + \alpha_j > T$, it follows that $j < i$. Now, consider the vote obtained from L_0 by swapping c_i and c_j. Clearly, all candidates in $C \setminus \{c_i, c_j\}$ get at most T points, and c_i gets $T' > T$ points. Further, c_j gets $s_j + \alpha_i \leq s_j + \alpha_j \leq T$ points. Thus, in this case c_i is a unique winner and $u(c_i) \geq u(S)$, a contradiction. □

Therefore, to find an optimal manipulation, it suffices to (i) check for each candidate $c \in C$ whether c can be made the unique winner with a score that exceeds T and (ii) find an optimal conservative vote. The optimal manipulation can then be selected from the ones found in (i) and (ii).

Step (i) is easy to implement. Indeed, a candidate c_j can be made the unique winner with a score that exceeds T if and only if $s_i + \alpha_1 > T$. To see this, observe that if $s_i + \alpha_1 > T$, we can swap c_1 and c_i in L_0: c_i will get more than T points, and all other candidates will get at most T points. Conversely, if $s_i + \alpha_1 \leq T$, then the score of c_i is at most T no matter how voter n votes.

Thus, it remains to show how to implement (ii). Intuitively, our algorithm proceeds as follows. We start with the set of winners produced by L_0; we will later show that this set is minimal, in the sense that if it contains x candidates from some level, then for any vote the set of winners will contain at least x candidates from that level. Note also that due to the ordering of the candidates we select the best candidates from each level at this step. We then try to increase the average utility of the set of winners. To this end, we order the remaining candidates by their utility, and try to add them to the set of winners one by one as long as this increases its average utility. We will now give a formal description of our algorithm and its proof of correctness.

Let $S_0 = \mathcal{F}_\alpha(\mathcal{R}', L_0)$. We initialize S and L by setting $S = S_0$, $L = L_0$. Let $\succ^*$ be some ordering of the set C that ranks the candidates in S_0 first, followed by the candidates in $C \setminus S_0$ in the order of decreasing utility, breaking ties arbitrarily. We order the candidates from $C \setminus S_0$ according to $\succ^*$, and process the candidates in this ordering one by one. For each candidate c_j, we check if $u(c_j) > u(S)$; if this is not the case, we terminate, as all subsequent candidates have even lower utility. Otherwise, we check if we can swap c_j with another candidate that is currently not in S and receives $T - s_j$ points from L (so that c_j gets T points in the resulting vote). If this is the case, we update L by performing the swap and set $S = S \cup \{c_j\}$. We then proceed to the next candidate on the list.

We claim that the vote L obtained in the end of this process is optimal for the manipulator, among all conservative votes. We remark that at any point in time S is exactly the set of candidates that get T points in $(\mathcal{R}', L)$. Thus, we claim that any conservative vote $\hat{L}$ satisfies $u(\mathcal{F}_\alpha(\mathcal{R}', \hat{L})) \leq u(S)$.

Assume that this is not the case. Among all optimal conservative votes, we will select one that is most "similar" to L in order to obtain a contradiction. Formally, let $\mathcal{L}_0$ be the set of all optimal conservative votes, and let $\mathcal{L}_1$ be the subset of $\mathcal{L}_0$ that consists of all votes L' that maximize the size of the set $\mathcal{F}_\alpha(\mathcal{R}', L') \cap S$. The ordering $\succ^*$ induces a lexicographic ordering on the subsets of C. Let $\hat{L}$ be the vote such that the set $\mathcal{F}_\alpha(\mathcal{R}', \hat{L})$ is minimal with respect to this ordering, over all votes in $\mathcal{L}_1$. Set $\hat{S} = \mathcal{F}_\alpha(\mathcal{R}', \hat{L})$; by our assumption we have $u(\hat{S}) > u(S)$.

Observe first that our algorithm never removes a candidate from S: when we want to add c_j to S and search for an appropriate swap, we only consider candidates that have not been added to S yet. Also, at each step of our algorithm the utility of the set S strictly increases. These observations will be important for the analysis of our algorithm.

We will first show that $\hat{S} \setminus S$ is empty.

Lemma 3.2.3. *We have $\hat{S} \setminus S = \emptyset$.*

Proof. Suppose that the lemma is not true, and let c_i be a candidate in $\hat{S} \setminus S$. Suppose that c_i appears in the j-th position in our ordering of $C \setminus S_0$. If our algorithm terminated at or before the j-th step, we have $u(c_i) < u(S) < u(\hat{S})$, and hence $u(\hat{S} \setminus \{c_i\}) > u(\hat{S})$. Also, it is easy to see that if we swap c_i and c_j in $\hat{L}$ than we obtain $S \setminus \{c_i\}$ as a set of winners. So, this is a contradiction with the optimality of $\hat{L}$.

Thus, when our algorithm considered c_i, it could not find a suitable swap. Since $c_i \in \hat{S}$, it has to be the case that there exists an entry of the scoring vector that equals $T - s_i$; however, when our algorithm processed c_i it was unable to place c_i in a position that grants $T - s_i$ points. This could only happen if all candidates that were receiving $T - s_i$ points from L at that point were in S at that time; denote the set of all such candidates by B_i. Note that all candidates in B_i belong to the same level as c_i. Also, all candidates in $B_i \cap S_0$ have the same or higher utility than c_i, because initially we order the candidates at the same level by their utility, so that L_0 grants a higher score to the best candidates at each level. On the other hand, all candidates in $B_i \setminus S_0$ were added to S at some point, which means that they have been processed before c_i. Since at this stage of the algorithm we order the candidates by their utility, it means that they, too, have the same or higher utility than c_i.

Now, since $\hat{L}$ grants $T - s_i$ points to c_i, it grants less than $T - s_i$ points to one of the candidates in B_i. Let c_k be any such candidate, and consider the vote $\hat{L}'$ obtained from $\hat{L}$ by swapping c_i and c_k. Let $\hat{S}' = \mathcal{F}_\alpha(\mathcal{R}', \hat{L}')$; we have $\hat{S}' = (\hat{S} \setminus \{c_i\}) \cup \{c_k\}$. By the argument above, we have either $u(c_k) > u(c_i)$ or $u(c_k) = u(c_i)$. In the former case, we get $u(\hat{S}') > u(\hat{S})$. In the latter case, we get $u(\hat{S}') = u(\hat{S})$ and $|\hat{S}' \cap S| > |\hat{S} \cap S|$. In both cases, we obtain a contradiction with our choice of $\hat{L}$. □

Thus, we have $\hat{S} \subseteq S$, and it remains to show that $S \subseteq \hat{S}$. We will first show that $\hat{S}$ contains all candidates in S_0.

Lemma 3.2.4. *We have $S_0 \subseteq \hat{S}$.*

Proof. Suppose that $|S_0 \cap H_k| = m_k$ for $k = 1, \ldots, h$. We will first show that $|\hat{S} \cap H_k| \geq m_k$ for $k = 1, \ldots, h$. Indeed, fix a $k \leq h$, and suppose that the first candidate in the k-th level is c_i. Then in $(\mathcal{R}', L_0)$ the scores of the

candidates in H_k are $s_i + \alpha_i, \ldots, s_i + \alpha_j$ for some $j \geq i$. If $s_i + \alpha_i < T$, then $m_k = 0$ and our claim is trivially true for this value of k. Otherwise, by the pigeonhole principle, if it holds that in $(\mathcal{R}', \hat{L})$ less than m_k voters in H_k get T points, it has to be the case that at least one candidate in $H_{k+1} \cup \cdots \cup H_h$ receives at least α_i points from $\hat{L}$. However, for any $c_\ell \in H_{k+1} \cup \cdots \cup H_h$ we have $s_\ell > s_i$, so $s_\ell + \alpha_i > T$, a contradiction with our choice of $\hat{L}$.

Now, suppose that $S_0 \cap H_k \not\subseteq \hat{S} \cap H_k$ for some $k \leq h$, and consider a candidate $c_\ell \in (S_0 \cap H_k) \setminus (\hat{S} \cap H_k)$. Since we have argued that $|\hat{S} \cap H_k| \geq m_k$, it must be the case that there also exists a candidate $c_j \in (\hat{S} \cap H_k) \setminus (S_0 \cap H_k)$. It is easy to see that S_0 contains the m_k best candidates from H_k, so $u(c_\ell) \geq u(c_j)$. The rest of the proof is similar to that of Lemma 3.2.3: Consider the vote $\hat{L}'$ obtained from $\hat{L}$ by swapping c_ℓ and c_j and let $\hat{S}' = \mathcal{F}_\alpha(\mathcal{R}', \hat{L}')$. Since c_ℓ and c_j belong to the same level, we have $\hat{S}' = (\hat{S} \setminus \{c_\ell\}) \cup \{c_k\}$. Thus, either $u(\hat{S}') > u(\hat{S})$ or $u(\hat{S}') = u(\hat{S})$ and $|\hat{S}' \cap S| > |\hat{S} \cap S|$. In both cases we get a contradiction. Thus, we have $S_0 \cap H_k \subseteq \hat{S} \cap H_k$. Since this holds for every value of k, the proof is complete. □

Given Lemma 3.2.3 and Lemma 3.2.4, it is easy to complete the proof. Suppose that $\hat{S}$ is a strict subset of S. Observe first that for any subset S' of S there is a vote L' such that $\mathcal{F}_\alpha(\mathcal{R}', L') = S'$: we can simply ignore the candidates that are not members of S' when running our algorithm, as this only increases the number of "available" swaps at each step. Now, order the candidates in $C \setminus S_0$ according to $\succ^*$. Let c_i be the first candidate in this order that appears in S, but not in $\hat{S}$. If there is a candidate c_j that appears later in the sequence and is contained in both S and $\hat{S}$, consider the set $S' = \hat{S} \setminus \{c_j\} \cup \{c_i\}$. As argued above, there is a vote L' such that $\mathcal{F}_\alpha(\mathcal{R}', L') = S'$. Now, if $u(c_i) > u(c_j)$, this set has a higher average utility that $\hat{S}$. Thus, this is a contradiction with our choice of $\hat{L}$. On the other hand, if $u(c_j) = u(c_i)$, then we have $u(S') = u(\hat{S})$, $|S \cap S'| = |S \cap \hat{S}|$, and S' precedes $\hat{S}$ is the lexicographic ordering induced by $\succ^*$, a contradiction with the choice of $\hat{L}$ again. Therefore, none of the candidates in S that appear after c_i in the ordering belongs to $\hat{S}$. Now, when we added c_i to S, we did so because its utility was higher than the average utility of S at that point. However, by construction, the latter is exactly equal to $u(\hat{S})$. Thus, $u(\hat{S} \cup \{c_i\}) > u(\hat{S})$, a contradiction again. Therefore, the proof is complete. □

Example 1. *It can be easily seen from the algorithm of finding manipulation for scoring rules under randomized tie-breaking that for the initializing of*

algorithm we do not need to know $\mathcal{R}$, the sufficient information is scores of candidates before submission of n-th vote and utility function of the last voter. Thus, consider the following example with 5 candidates and Borda rule for determination of the winner. Recall that s_i denotes the score of candidate i before last voter votes.

$$s_1 = 10, s_2 = 6, s_3 = 12, s_4 = 4, s_5 = 10,$$

$$u(c_1) = 2, u(c_2) = 3, u(c_3) = 1, u(c_4) = 7, u(c_5) = 4.$$

Step 1. We renumber candidates in order of increasing score. For candidates of the same score we use order of decreasing utility. After this step candidate $c_1^{'}$ has score 4 and $u(c_1^{'}) = 7$, $c_2^{'}$ has score 6 and $u(c_2^{'}) = 3$, $c_3^{'}$ and $c_4^{'}$ have scores 10, $u(c_3^{'}) = 4$ and $u(c_4^{'}) = 2$ and $c_5^{'}$ has score 12 and $u(c_5^{'}) = 1$.

Step 2. Consider vote $L_0 = (c_1^{'}, c_2^{'}, c_3^{'}, c_4^{'}, c_5^{'})$. After this vote candidates have scores as follows.

$$s(c_1^{'}) = 8, s(c_2^{'}) = 9, s(c_3^{'}) = 12, s(c_4^{'}) = 11, s(c_5^{'}) = 12.$$

Here we also found $T = 12$.

Step 3. Now we determine the best candidate who can become the only winner. The set of candidates whose score can exceed 12 is $\{c_3^{'}, c_4^{'}, c_5^{'}\}$. The candidate $c_3^{'}$ has the largest utility among these candidates. By swapping $c_1^{'}$ and $c_3^{'}$ in L_0 we obtain vote L. If the last voter submits L then $c_3^{'}$ is the only winner of the election and the utility of last voter is 4.

Step 4. Now we find the optimal conservative vote. It is easy to see that candidates from first and second levels $(c_1^{'}, c_2^{'})$ cannot be among tied candidates and $c_5^{'}$ is always among the tied candidates. From the level 3 exactly one candidate who receive 2 points from the last voter is in the set of tied candidates. Thus, L_0 is the optimal conservative vote. After the submitting L_0 the set of tied candidates is $\{c_3^{'}, c_5^{'}\}$ and utility of last voter is 2,5.

Thus, vote L maximizes utility of manipulator. In the original notation the best manipulative vote is $(c_5, c_2, c_4, c_1, c_3)$.

3.3 Bucklin

In this section, we describe a polynomial-time algorithm for Bucklin-Rand-Manipulation. In the first subsection we will focus on the simplified Bucklin rule, and omit the term "simplified" throughout this section; in the second

subsection, we will explain how to extend our algorithm to the classic Bucklin rule.

3.3.1 Simplified Bucklin

We first need some additional notation. Consider an election $E = (C, \mathcal{R})$ with $|C| = m$ and the preference profile $\mathcal{R} = (R_1, \ldots, R_n)$, and suppose that the manipulating voter has utility function u. Set $E' = (C, \mathcal{R}')$ and $\mathcal{R}' = (R_1, \ldots, R_{n-1})$. For any $c \in C$, let $s_k(c)$ denote c's k-approval score in $\mathcal{R}'$. Given an $L \in \mathcal{L}(C)$, let $S(L)$ be the set of Bucklin winners in $(\mathcal{R}_{-n}, L)$.

Let $\ell = \min\{k \mid s_k(c) \geq \lfloor \frac{n}{2} \rfloor + 1 \text{ for some } c \in C\}$, and set

$$D = \{c \in C \mid s_\ell(c) \geq \lfloor \frac{n}{2} \rfloor + 1\}.$$

Clearly, for any $L \in \mathcal{L}(C)$, if k is the Bucklin winning round in $(\mathcal{R}_{-n}, L)$, then $k \leq \ell$. For each $i = 1, \ldots, m$, let

$$C_i = \{c \in C \mid s_i(c) = \lfloor \frac{n}{2} \rfloor, s_{i-1}(c) < \lfloor \frac{n}{2} \rfloor\},$$

and set $C_{<i} = \bigcup_{j<i} C_j$ if $i < \ell$ and $C_{<\ell} = (\bigcup_{j<\ell} C_j) \setminus D$.

Suppose that $i \leq \ell$. If the manipulator ranks a candidate $c \in C_i$ in position i or higher, and ranks each candidate in $C_{<i}$ in position i or lower, in the resulting election i is the Bucklin winning round, and c is a Bucklin winner. Conversely, if $i \leq \ell$ is the Bucklin winning round in $(\mathcal{R}_{-n}, L)$ and a candidate c is a Bucklin winner, then one of the following conditions holds: (a) $c \in C_i$ and c is ranked in position i or higher in L, or (b) $c \in C_{<i}$ and c is ranked in position i in L, or (c) $i = \ell$ and $c \in D$.

For $i \leq \ell$ and $s \leq m$, let $\mathcal{L}_{i,s}$ denote the set of all votes $L \in \mathcal{L}(C)$ such that (a) i is the Bucklin winning round in $(\mathcal{R}_{-n}, L)$ and (b) $|S(L) \cap C_i| = s$. Also, let $\mathcal{L}^*_{i,s} = \arg\max\{u(S(L)) \mid L \in \mathcal{L}_{i,s}\}$ be the set of utility-maximizing votes in $\mathcal{L}_{i,s}$.

We will now explain how to find a vote in $\mathcal{L}^*_{i,s}$. First, we will show that if $L^* \in \mathcal{L}^*_{i,s}$ and the set $S(L^*)$ contains some candidate $c \in C_{<i}$, then c is the top candidate in $C_{<i}$.

Lemma 3.3.1. *If $L^* \in \mathcal{L}^*_{i,s}$ for some $i \leq \ell$ and $s \leq m$ and $S(L^*) \cap C_{<i} \neq \emptyset$, then $|S(L^*) \cap C_{<i}| = 1$ and $S(L^*) \cap C_{<i} \in \arg\max\{u(c) \mid c \in C_{<i}\}$.*

Proof. Fix a vote $L^* \in \mathcal{L}^*_{i,s}$, and let c be a candidate in $S(L^*) \cap C_{<i}$. Since i is the Bucklin winning round for L^* and $c \in C_{<i}$, c cannot be ranked in position $i-1$ or higher in L^*. Further, since $c \in S(L^*)$ and i is the Bucklin winning round for L^*, c cannot be ranked in position $i+1$ or lower in L^* (here, for $i = \ell$ it is crucial that the set $C_{<\ell}$ does not contain candidates in D). Hence, c is ranked in position i in L^*, so $|S(L^*) \cap C_{<i}| = 1$. Now, if $c \notin \arg\max\{u(c) \mid c \in C_{<i}\}$, consider the vote L' obtained from L^* by swapping c with some candidate $b \in \arg\max\{u(c) \mid c \in C_{<i}\}$. We have $L' \in \mathcal{L}_{i,s}$. Further, the argument above shows that $b \notin S(L^*)$, so $S(L') = (S(L^*) \setminus \{c\}) \cup \{b\}$ and hence $u(S(L')) > u(S(L^*))$, a contradiction. □

Now, we use Lemma 3.3.1 to find a vote in $\mathcal{L}^*_{i,s}$.

Lemma 3.3.2. *For any $i \leq \ell$ and any $s \leq |C|$, there is a polynomial-time algorithm that checks whether $\mathcal{L}_{i,s}$ is non-empty, and, if so, identifies a vote $L^* \in \mathcal{L}^*_{i,s}$.*

Proof. Let $\mathcal{L}^1_{i,s}$ be the set of all votes L in $\mathcal{L}_{i,s}$ such that $S(L) \cap C_{<i} \neq \emptyset$, and let $\mathcal{L}^2_{i,s} = \mathcal{L}_{i,s} \setminus \mathcal{L}^1_{i,s}$. We will identify the best vote in $\mathcal{L}^1_{i,s}$ and $\mathcal{L}^2_{i,s}$ and output the better of the two. Observe that either or both of $\mathcal{L}^1_{i,s}$ and $\mathcal{L}^2_{i,s}$ can be empty: if both are empty, then so is $\mathcal{L}_{i,s}$, and if $\mathcal{L}^j_{i,s}$ is empty, but $\mathcal{L}^{3-j}_{i,s}$ is not, we output the best vote in $\mathcal{L}^{3-j}_{i,s}$.

If $C_{<i} \neq \emptyset$, let b_i be some candidate in $\arg\max\{u(c) \mid c \in C_{<i}\}$. By Lemma 3.3.1, to find the best vote in $\mathcal{L}^1_{i,s}$, we place b_i in position i. Now, we need to place s candidates from C_i in top $i-1$ positions. Clearly, if $|C_i| < s$ or if $s > i-1$, this is impossible, so $\mathcal{L}^1_{i,s} = \emptyset$. Otherwise, we pick s candidates in C_i with the highest utility, breaking ties arbitrarily, and rank them in top s positions in the vote. We then fill the remaining $i-1-s$ positions above i with candidates from $C \setminus (C_i \cup C_{<i})$; again, if $|C \setminus (C_i \cup C_{<i})| < i-1-s$, then $\mathcal{L}^1_{i,s} = \emptyset$. The remaining candidates can be ranked arbitrarily. It is easy to see that the resulting vote L_1 is in $\mathcal{L}^1_{i,s}$, and, moreover, $u(S(L_1)) \geq u(S(L'))$ for any $L' \in \mathcal{L}^1_{i,s}$.

The procedure for finding the best vote in $\mathcal{L}^2_{i,s}$ is similar. By the same argument as in the previous case, if $|C_i| < s$ or $s > i$ or $|C \setminus C_{<i+1}| < i - s$, then $\mathcal{L}^2_{i,s}$ is empty. Otherwise, we pick s candidates in C_i with the highest utility, rank them in top s positions in the vote, rank some candidates from $C \setminus (C_i \cup C_{<i})$ in the next $i - s$ positions, and then rank the remaining candidates arbitrarily. The resulting vote L_2 satisfies $u(S(L_2)) \geq u(S(L'))$ for any $L' \in \mathcal{L}^2_{i,s}$. □

Using Lemma 3.3.2, we can simply find the best vote in $\mathcal{L}_{i,s}$ for all $i = 1, \ldots, \ell$, $s = 0, \ldots, m$; while for many values of i and s the set $\mathcal{L}_{i,s}$ is empty, we have $\mathcal{L}_{i,s} \neq \emptyset$ for some $i \leq \ell$, $s \leq m$. We obtain the following result.

Theorem 3.3.3. Simplified Bucklin-RandManipulation *is in* P.

Example 2. *Suppose* $\mathcal{R}' = \begin{pmatrix} a & a & b & b & d \\ b & d & a & c & c \\ d & b & d & d & a \\ c & c & c & a & b \end{pmatrix}$ *and* $u(d) = 10, u(b) = 5, u(c) = 2, u(a) = 1$.

It is easy to see that $\lfloor \frac{n}{2} \rfloor = 3$ *and* $l = 3$ *as well. Clearly,* $C_1 = C_{<2} = C_3 = \emptyset$, $C_2 = \{a, b\}$ *and* $D = \{a, b, d\}$. *Therefore,* $C_{<3} = C_2 \setminus D = \emptyset$.

Step 1. Evidently, $\mathcal{L}_{1,s} = \emptyset$.

Step 2. Clearly, $\mathcal{L}_{2,s} = \emptyset$ *if* $s > 2$. *It follows from* $C_{<2} = \emptyset$ *that* $\mathcal{L}^1_{2,s} = \emptyset$ *for* $s = 1, 2$. *Now we will identify the best votes in* $\mathcal{L}^2_{2,1}$ *and* $\mathcal{L}^2_{2,2}$. *At first place we consider* $\mathcal{L}^2_{2,1}$. *We take the candidate with highest utility from* C_2 *and this candidate is* b. *Afterward we take arbitrary candidate from* $C \setminus (C_2 \cup C_{<2}) = \{c, d\}$, *for example,* c. *The following candidates can be placed in arbitrary order. Thus, we obtain* $(b, c, d, a) \in \mathcal{L}^*_{2,1}$. *Similarly we find* $(b, a, d, c) \in \mathcal{L}^*_{2,2}$. *It is easy to see that* (b, c, d, a) *gives a better outcome of the election.*

Step 3. It follows from $C_{<3} = C_3 = \emptyset$ *that only* $\mathcal{L}^2_{3,0}$ *can be non-empty. Similarly to the previous step we obtain* $(c, d, b, a) \in \mathcal{L}^*_{3,0}$.

Comparing (b, c, d, a) *and* (c, d, b, a) *we can see that* (c, d, b, a) *gives the best outcome of the election.*

3.3.2 Classic Bucklin

To extend our algorithm to the classic Bucklin rule, observe that if $L \in \mathcal{L}_{i,s}$ for some $i < \ell$, then each Bucklin winner in $(\mathcal{R}_{-n}, L)$ has the same i-approval score (namely, $\lfloor \frac{n}{2} \rfloor + 1$), therefore, any simplified Bucklin winner in $(\mathcal{R}_{-n}, L)$ is also a Bucklin winner in $(\mathcal{R}_{-n}, L)$. Thus, only the case $i = \ell$ has to be handled differently. In this case, it matters which candidates in D are ranked in top ℓ positions by the manipulator, as this affects their ℓ-approval score. Therefore, for this case we denote by $\hat{\mathcal{L}}_{\ell,s}$ the set of all votes $L \in \mathcal{L}(C)$ such that (a) ℓ is the Bucklin winning round in $(\mathcal{R}_{-n}, L)$ and (b) $|S(L) \cap (C_\ell \cup D)| = s$. As in the previous case, we define the set of votes $\hat{\mathcal{L}}^*_{\ell,s}$ as the set of utility-maximizing votes in $\hat{\mathcal{L}}_{\ell,s}$, and let $\hat{L}^*$ be a vote in $\hat{\mathcal{L}}^*_{\ell,s}$.

Lemma 3.3.4. *There is a polynomial-time algorithm that checks whether $\hat{\mathcal{L}}_{\ell,s}$ is non-empty, and, if so, identifies a vote $L^* \in \hat{\mathcal{L}}^*_{\ell,s}$.*

Proof. Divide the set $\hat{\mathcal{L}}^*_{\ell,s}$ into three subsets as follows:

- let $\mathcal{L}_1$ be the set of all votes L in $\hat{\mathcal{L}}^*_{\ell,s}$ such that $S(L) \subset D$;
- let $\mathcal{L}_2$ be the set of all votes L in $\hat{\mathcal{L}}^*_{\ell,s}$ such that $S(L) \cap C_{<\ell} \neq \emptyset$;
- let $\mathcal{L}_3 = \hat{\mathcal{L}}^*_{\ell,s} \setminus (\mathcal{L}_1 \cup \mathcal{L}_2)$.

Let $d = \max\{s_\ell(c) \mid c \in D\}$. Evidently, the Bucklin winning score in $(\mathcal{R}_{-n}, L)$ for a $L \in \hat{\mathcal{L}}^*_{\ell,s}$ is either d or $d+1$.

If $d > \lfloor \frac{n}{2} \rfloor + 1$ then $S(L) \subset D$ for any $L \in \hat{\mathcal{L}}^*_{\ell,s}$ and, so, $\mathcal{L}_2 = \emptyset$ and $\mathcal{L}_3 = \emptyset$. Therefore, in this case our algorithm needs to check whether $\mathcal{L}_1 = \emptyset$ and if it is not the case then find $L^* \in \mathcal{L}_1$. If $d = \lfloor \frac{n}{2} \rfloor + 1$, all three sets can be nonempty, and the algorithm will identify the best vote in $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ and output the better of the three. Thus, we divide proof into two cases: $d > \lfloor \frac{n}{2} \rfloor + 1$ and $d = \lfloor \frac{n}{2} \rfloor + 1$.

Denote the set of candidates whose score $s_\ell(c)$ is equal to d by D_d, and the set of candidates whose score $s_\ell(c)$ is equal to $d-1$ by D_{d-1}.

Case $d > \lfloor \frac{n}{2} \rfloor + 1$**.** It is easy to see that $\mathcal{L}_1 = \emptyset$ if $|C_{<\ell}| > m - (\ell - 1)$. Therefore, we can consider only the case $|C_{<\ell}| \leq m - (\ell - 1)$. Now we will construct the best vote L_1 such that the Bucklin winning score in $(\mathcal{R}_{-n}, L_1)$ is equal to $d+1$ and the best vote L_2 such that the Bucklin winning score in $(\mathcal{R}_{-n}, L_2)$ is equal to d, and output the better of the two. If both votes do not exist then $\mathcal{L}_1 = \emptyset$.

First suppose that the Bucklin winning score is $d+1$. In this case $S(L_1) \subset D_d$. It is easy to see that the utility of any s-element subset of D_d is at most the sum of utilities of s most valuable candidates. Thus, if $|D_d| - s \leq m - \ell - \max\{|C_{<\ell}| - 1, 0\}$ and $s < \ell$ then we rank s most valuable (ties is broken arbitrary) candidates from D_d at top s positions, rank one of the candidates from $C_{<\ell}$ at place ℓ, and rank all other candidates in D_d and $C_{<\ell}$ at the $|D_d| + |C_{<\ell}| - s - 1$ bottom places; the remaining candidates can be ranked arbitrarily. The output is the vote L_1. In this case ℓ is the Bucklin winning round in $(\mathcal{R}_{-n}, L_1)$ and only the s candidates from D_d who are ranked above ℓ-th position

have score $d+1$ in this round, while other candidates have lower scores. So, they are the only winners and the utility of this set of candidates is larger than the utility of any subset of D_d. If $s = \ell$ the procedure will be almost the same excluding the placement of one candidate from $C_{<\ell}$ at position ℓ.

Suppose $|D_d| - s > m - \ell - \max\{|C_{<\ell}| - 1, 0\}$. Then all candidates in the set $D_d \cup C_{<\ell}$ cannot be ranked at places ℓ (this position can be allocated to a candidate in $C_{<\ell}$ only) and below. Therefore $\mathcal{L}_1 = \emptyset$.

Now consider a vote L_2 such that d is the Bucklin winning score for $(\mathcal{R}_{-n}, L_2)$. Evidently, if $|D_d| > s$ then there are no such votes, because $D_d \subset S(L_2)$. So, we can assume $|D_d| \leq s$. By our assumption $d > \lfloor \frac{n}{2} \rfloor + 1$, so, it is evident that L_2 can be obtained almost exactly as in the previous case with two modifications: We will rank $s - |D_d|$ candidates from D_{d-1} above ℓ instead of candidates from D_d and place all remaining candidates from D_{d-1} and the candidates from $D_d \cup C_{<\ell}$ below ℓ. One candidate in $C_{<\ell}$ can still be ranked at position ℓ.

Case $d = \lfloor \frac{n}{2} \rfloor + 1$**.** Notice that $D_{d-1} = \emptyset$ in this case.

It is easy to see that in this case the best vote L_1 such that the Bucklin winning score in $(\mathcal{R}_{-n}, L_1)$ is equal to $d+1$ can be found exactly as in the previous case. Now we will find an optimal vote $\hat{L}^*$ such that the Bucklin winning score in $(\mathcal{R}_{-n}, \hat{L}^*)$ is equal to d.

First consider L_1. Obviously, $D_d \subset S(L_1)$. By definition of L_1 we also have $S(L_1) \subset D_d$, therefore, either $S(L_1) = D_d$ and $|D_d| = s$, or $\mathcal{L}_1 = \emptyset$. If $S(L_1) = D_d$ then neither the candidates from C_ℓ, nor those from $C_{<\ell}$ can be among the winners of the election and, therefore, all candidates $D_d \cup C_\ell \cup C_{<\ell}$ are ranked below ℓ. Therefore, if $|D_d| + |C_\ell| + |C_{<\ell}| \leq m - \ell$ then we can put candidates from $D_d \cup C_\ell \cup C_{<\ell}$ at bottom places, and all others can be ranked arbitrary. Otherwise, $\mathcal{L}_1 = \emptyset$.

Second consider L_2. By definition of L_2 one candidate from $C_{<\ell}$ and at least one candidate from C_ℓ can be among the winners of the election. Thus, at most $s - 2$ winners can be from the set D_d. Therefore, all candidates from D_d and $|C_{<\ell}| - 1$ candidate from $C_{<\ell}$ are ranked below ℓ as well as $|D_d| + |C_\ell| - s$ candidates from C_ℓ. So, if $|C_\ell| + |C_{<\ell}| + |D_d| - s \leq m - \ell + 1$ then put $|D_d| + |C_\ell| - s$ best candidates from C_ℓ (ties are broken arbitrary) at top places, the best candidate from $C_{<\ell}$ at the

place ℓ, all remaining candidates from $C_\ell \cup C_{<\ell}$ and candidates from D_d at bottom places, and all others candidates can be ranked arbitrary. Otherwise, $\mathcal{L}_2 = \emptyset$.

The third vote L_3 can be handled almost exactly as L_2.

□

Using this lemma we can easily obtain the following theorem.

Theorem 3.3.5. Bucklin-RandManipulation *is in* P.

Proof. Using Lemmas 3.3.2 and 3.3.4, we can simply find the best vote in $\mathcal{L}_{i,s}$ and in $\hat{\mathcal{L}}_{\ell,s}$ for all $i = 1, \ldots, \ell-1$, $s = 0, \ldots, m$; while for many values of i and s the sets $\mathcal{L}_{i,s}$ and $\hat{\mathcal{L}}_{\ell,s}$ are empty, we have either $\mathcal{L}_{i,s} \neq \emptyset$ or $\hat{\mathcal{L}}_{\ell,s} \neq \emptyset$ for some $i \leq \ell$, $s \leq m$. □

3.4 Maximin

3.4.1 General utilities

In this section, we show that Maximin-RandManipulation is NP-hard. In fact, our hardness result holds even for a fairly simple utility function, namely if we set $u(w) = 0$, $u(c) = 1$ for $c \in C \setminus \{w\}$, then Maximin-RandManipulation becomes NP-complete. Observe that if the manipulator has this utility function, and w is the Maximin winner irrespective of manipulator's vote, then the manipulator's goal is to maximize the overall number of Maximin winners.

Our hardness proof proceeds by a reduction from Feedback Vertex Set [25]. Recall that an instance of Feedback Vertex Set is given by a directed graph $\mathcal{G}$ with s vertices $\{\nu_1, \ldots, \nu_s\}$ and a parameter $t \leq s$; it is a "yes"-instance if it is possible to delete at most t vertices from $\mathcal{G}$ so that the resulting graph contains no directed cycles and a "no"-instance otherwise. It will be convenient to assume that $\mathcal{G}$ contains no directed cycles of length 2. It is easy to see that Feedback Vertex Set remains NP-hard under this assumption. Indeed, given an arbitrary instance $(\mathcal{G}, t)$ of Feedback Vertex Set with r arcs, we can introduce r new vertices $\nu'_1, \ldots, \nu'_r$ and replace each arc of the form $e_i = (\nu_j, \nu_\ell)$ with a path of length 2 that consists of arcs (ν_j, ν'_i) and (ν'_i, ν_ℓ); denote the resulting graph by $\mathcal{G}'$. Clearly, $\mathcal{G}'$

contains no directed cycles of length 2, and if $(\mathcal{G}, t)$ is a "yes"-instance of FEEDBACK VERTEX SET, so is $(\mathcal{G}', t)$: if the removal of a vertex set X eliminates all directed cycles in $\mathcal{G}$, its removal also eliminates all directed cycles in $\mathcal{G}'$. Conversely, if we can eliminate all directed cycles in $\mathcal{G}'$ by removing a set of vertices Y, consider the set Y' obtained from Y by replacing each vertex $\nu_i' \in Y$ with a vertex ν_j such that (ν_j, ν_i') is an arc of $\mathcal{G}'$. It is easy to see that removing Y' eliminates all directed cycles in $\mathcal{G}$ and $|Y'| \leq |Y|$. Thus, from now on, we will assume that $\mathcal{G}$ contains no directed cycles of length 2.

Theorem 3.4.1. Maximin-RANDMANIPULATION *is* NP*-complete.*

Proof. We have argued that Maximin-RANDMANIPULATION is in NP. For the hardness proof, suppose that we are given an instance $(\mathcal{G}, t)$ of FEEDBACK VERTEX SET, where $\mathcal{G}$ is an s-vertex graph with the vertex set $\{\nu_1, \ldots, \nu_s\}$ that has no directed 2-cycles. We will now construct an instance of our problem with $C = \{c_1, c_2, \ldots, c_s, w\}$.

By Corollary 2.3.4, there exists an election $E = (C, \mathcal{R}')$ with a preference profile $\mathcal{R}' = (R_1, \ldots, R_{n-1})$, where n is odd, such that

- for $i = 1, \ldots, s$, if the indegree of ν_i in $\mathcal{G}$ is at least 1, then exactly $\frac{n-1}{2}$ voters rank w above c_i; otherwise, exactly $\frac{n-1}{2} + 1$ voter ranks w above c_i.

- if $(\nu_i, \nu_j) \in \mathcal{G}$ (and hence, since $\mathcal{G}$ contains no directed cycles of length 2, $(\nu_j, \nu_i) \notin \mathcal{G}$), exactly $\frac{n-1}{2} + 1$ voters rank c_i above c_j.

- if $(\nu_i, \nu_j) \notin \mathcal{G}$ and $(\nu_j, \nu_i) \notin \mathcal{G}$, exactly $\frac{n-1}{2}$ voters rank c_i above c_j.

Moreover, $\mathcal{R}' = (R_1, \ldots, R_{n-1})$ can be constructed in time polynomial in s. We will say that c_i is a *parent* of c_j if exactly $\frac{n-1}{2} + 1$ voter ranks c_i above c_j. Observe that in the resulting election the Maximin score of w is $\frac{n-1}{2}$, and the Maximin score of any other candidate is $\frac{n-1}{2} - 1$.

Recall that n is the manipulator, and consider the election $E'' = (C, \mathcal{R}'')$ with a preference profile $\mathcal{R}'' = (\mathcal{R}', L) = (R_1, \ldots, R_{n-1}, L)$, where L is the manipulator's vote. Since w is the unique Maximin winner before the manipulator votes, and w's score exceeds the score of any other candidate by 1, a candidate c_i is a winner of $(\mathcal{R}', L)$ if and only if (a) the manipulator ranks c_i above all of her parents and (b) w's Maximin score does not increase;

on the other hand, w will remain the Maximin winner no matter how the manipulator votes.

Let the manipulator's utility be given by $u(w) = 0$, $u(c) = 1$ for any $c \in C \setminus \{w\}$. Under this utility function, the manipulator's utility is 0 if w is the only Maximin winner, 1 if w is not among the Maximin winners, and $r/(r+1)$ if the Maximin winners are w and r candidates from $C \setminus \{w\}$. Let R_n be some preference order over C that is consistent with u, and set $\mathcal{R} = (R_1, \ldots, R_{n-1}, R_n)$ and $E' = (C, \mathcal{R})$. We claim that $(\mathcal{G}, t)$ is a "yes"-instance of Feedback Vertex Set if and only if $(E', u, (s-t)/(s-t+1))$ is a "yes"-instance of Maximin-RandManipulation.

Suppose $(\mathcal{G}, t)$ is a "yes"-instance of Feedback Vertex Set. Then we can delete t vertices from $\mathcal{G}$ so that the resulting graph $\mathcal{G}'$ is acyclic, and hence can be topologically sorted. Let $\nu_{i_1}, \ldots, \nu_{i_{s-t}}$ be the vertices of $\mathcal{G}'$, listed in the sorted order, i.e., so that any edge of $\mathcal{G}$ is of the form $(\nu_{i_j}, \nu_{i_\ell})$ with $j < \ell$. Consider the vote L obtained by ranking the candidates that correspond to vertices of $\mathcal{G}'$ first, in reverse topological order (i.e., $c_{i_{s-t}}, \ldots, c_{i_1}$), followed by the remaining candidates in $C \setminus \{w\}$, followed by w. By construction, each of the first $s-t$ candidates is ranked above all of its parents, so its Maximin score in $(\mathcal{R}', L)$ is $\frac{n-1}{2}$. On the other hand, w's score remains equal to $\frac{n-1}{2}$. Thus, the manipulator's utility in the resulting election is at least $(s-t)/(s-t+1)$.

Conversely, suppose the manipulator submits a vote L' so that in the preference profile $(\mathcal{R}_{-n}, L')$ his utility is at least $(s-t)/(s-t+1)$. We have argued that w is a Maximin winner in $(\mathcal{R}_{-n}, L')$, and therefore $(\mathcal{R}_{-n}, L')$ has at least $s-t+1$ Maximin winners (including w). Let C' be a set of some $s-t$ candidates in $C \setminus \{w\}$ that are Maximin winners in $(\mathcal{R}_{-n}, L')$, and suppose they appear in L' ordered as $c_{i_1}, \ldots, c_{i_{s-t}}$. Let $\mathcal{G}'$ be the induced subgraph of $\mathcal{G}$ with the set of vertices $\nu_{i_1}, \ldots, \nu_{i_{s-t}}$. Each of the candidates in C' appears in L' before all of its parents. Therefore, in the ordering $\nu_{i_1}, \ldots, \nu_{i_{s-t}}$ of the vertices of $\mathcal{G}'$ all arcs are directed from right to left, i.e., $\mathcal{G}'$ contains no directed cycles. Since $\mathcal{G}'$ has $s-t$ vertices, this means that $(\mathcal{G}, t)$ is a "yes"-instance of Feedback Vertex Set. □

3.4.2 A tractable special case

In the previous section, we have shown that Maximin with randomized tie-breaking does not admit an efficient algorithm for finding an optimal manipulation in the general utility model. However, we will now present a

polynomial-time algorithm for this problem assuming that the manipulator's utility function has a special structure. Specifically, recall that in the model of [6] the manipulator's goal is to make a specific candidate p a winner. This suggests that the manipulator's utility can be modeled by setting $u(p) = 1$, $u(c) = 0$ for all $c \in C \setminus \{p\}$. We will now show that for such utilities there exists a polynomil-time algorithm for finding an optimal manipulation under Maximin combined with the randomized tie-breaking rule.

Theorem 3.4.2. *If the manipulator's utility function is given by $u(p) = 1$, $u(c) = 0$ for $c \in C \setminus \{p\}$, the problem of finding an optimal manipulation under Maximin combined with the randomized tie-breaking rule is in* P.

Proof. Consider an election $E = (C, \mathcal{R})$ with the candidate set $C = \{c_1, \ldots, c_m\}$ and recall that n is the manipulating voter. In this proof, we denote by $s(c_i)$ the Maximin score of a candidate $c_i \in C$ in the election $E' = (C, \mathcal{R}')$, where $\mathcal{R}' = \mathcal{R}_{-n}$. Let $s = \max_{c_i \in C} s(c_i)$.

For any $c_i \in C$, the manipulator's vote increases the score of c_i either by 0 or by 1. Thus, if $s(p) < s - 1$, the utility of the manipulator will be 0 irrespective of how he votes.

Now, suppose that $s(p) = s - 1$. The manipulator can increase the score of p by 1 by ranking p first. Thus, his goal is to ensure that after he votes (a) no other candidate gets $s + 1$ point and (b) the number of candidates in $C \setminus \{p\}$ with s points is as small as possible. Similarly, if $s(p) = s$, the manipulator can ensure that p gets $s + 1$ points by ranking him first, so his goal is to rank the remaining candidates so that in $C \setminus \{p\}$ the number of candidates with $s+1$ points is as small as possible. We will now describe an algorithm that works for both of these cases.

We construct a directed graph G with the vertex set C that captures the relationship among the candidates. Namely, we have an edge from c_i to c_j if there are $s(c_j)$ votes in $\mathcal{R}'$, where c_j is ranked above c_i. Observe that, by construction, each vertex in G has at least one incoming edge. We say that c_i is a *parent* of c_j in G whenever there is an edge from c_i to c_j. We remark that if the manipulator ranks one of the parents of c_j above c_j in his vote, then c_j's score does not increase. We say that a vertex c_i of G is *purple* if $s(c_i) = s(p) + 1$, *red* if $s(c_i) = s(p)$ and $c_i \neq p$, and *green* otherwise; note that by construction p is green. Observe also that if $s(p) = s$, there are no purple vertices in the graph. We will say that a candidate c_j is *dominated* in an ordering L (with respect to G) if at least one of c_j's parents in G appears before c_j in L. Thus, our goal is to ensure that the set of dominated

candidates includes all purple candidates and as many red candidates as possible.

Our algorithm is based on a recursive procedure $\mathcal{A}$, which takes as its input a graph H with a vertex set $U \subseteq C$ together with a coloring of U into green, red and purple; intuitively, U is the set of currently unranked candidates. It returns "no" if the candidates in U cannot be ranked so that all purple candidates in U are dominated by other candidates in U with respect to H. Otherwise, it returns an ordered list L of the candidates in U in which all purple candidates are dominated, and a set S consisting of all red candidates in U that remain undominated in L with respect to H.

To initialize the algorithm, we call $\mathcal{A}(G)$. The procedure $\mathcal{A}(H)$ is described below.

1. Set $L = \emptyset$.
2. If H contains p, set $L = [p]$, and remove p from H.
3. While H contains a candidate c that is green or has a parent that has already been ranked, set $L :: [c]$ (where :: denotes the list concatenation operation) and remove c from H.
4. If H is empty, return $(L, \emptyset)$.
5. If there is a purple candidate in H with no parents in H, return "no".
6. If there is a red candidate c in H with no parents in H, let H' be the graph obtained from H by coloring c green. Compute $\mathcal{A}(H')$. If $\mathcal{A}(H')$ returns "no", return "no". Otherwise, if $\mathcal{A}(H')$ returns (L', S'), return $(L :: L', S' \cup \{c\})$.
7. At this point in the algorithm, each vertex of H has a parent. Hence, H contains a cycle. Let T be some such cycle. Collapse T, i.e., (a) replace T with a single vertex t, and (b) for each $y \notin T$, add an edge (t, y) if H contained an edge (x, y) for some $x \in T$ and add an edge (y, t) if H contained a vertex z with $(y, z) \in H$. Color t red if T contains at least one red vertex, and purple otherwise. Let H' be the resulting graph and call $\mathcal{A}(H')$. If $\mathcal{A}(H')$ returns "no", return "no". Now, suppose that $\mathcal{A}(H')$ returns (L', S').

   Suppose that $t \in S'$. At any point in the algorithm, we only put a vertex in S if it is red, so t must be red, and hence T contains a red

vertex. Let c be some red vertex in T, and let $\hat{L}$ be an ordering of the vertices in T that starts with c and follows the edges of T. Let L'' be the list obtained from L' by replacing t with $\hat{L}$ (i.e., if $L' = L_1 :: [t] :: L_2$, then $L'' = L_1 :: \hat{L} :: L_2$). Return $(L :: L'', (S' \setminus \{t\}) \cup \{c\})$.

If $t \notin S'$, then by Lemma 3.4.3 (see below) t is dominated in H'. Let a be a parent of t that precedes it in L'. Then T contains a child of a. Let c be some such child, and let $\hat{L}$ be an ordering of the vertices in T that starts with c and follows the edges of T. Let L'' be the list obtained from L' by replacing t wish $\hat{L}$. Return $(L :: L'', S')$.

We will now argue that our algorithm outputs "no" if and only if no matter how manipulator n votes, some candidate in $C \setminus \{p\}$ gets $s(p) + 2$ points. Moreover, if $\mathcal{A}(G) = (L, S)$ and the set S contains r red candidates, then whenever manipulator n votes so that after his vote all other candidates have at most $s(p)+1$ points, there are at least r red candidates with $s(p)+1$ points.

We will split the proof into several lemmas.

Lemma 3.4.3. *At any point in the execution of the algorithm, if $\mathcal{A}(H) = (L, S)$, then each candidate in $U \setminus S$ is dominated in H.*

Proof. The proof is by induction on the recursion depth. Consider a candidate $x \in U \setminus S$. Clearly, if there are no recursive calls, $\mathcal{A}$ ranks x at Step 3, and the claim is obviously true.

For the induction step, suppose that the claim is true if we have d nested recursive calls, and consider an execution that makes $d+1$ nested calls. Again, consider a candidate $x \in U \setminus S$. As in the base case, if x has been ranked in Step 3 the claim is clearly true. If x was ranked in Step 6, it follows that $x \notin S'$, and the claim follows by the inductive assumption. Now, suppose that x was ranked in Step 7 when we collapsed some cycle T. If $x \notin T$, then $x \notin S'$ and the claim follows by the inductive assumption. In particular, if x was ranked after t before the expansion, there is some vertex y in T such that H contains the edge (y, x), so after expansion x will be dominated by y.

Now, suppose that $x \in T$. If t was in S', but x was not added to S, it means that x was not the first vertex of T to appear in the ranking, i.e., x was ranked after its predecessor in T. If t was not in S', then by the inductive assumption t was ranked after its parent in H', i.e., there is a $z \in H' \setminus \{t\}$ such that z is ranked before t in L' and there is an edge (z, t) in H'. By

construction of t, this means that there is a vertex $y \in T$ such that there is an edge (z, y) in H. Thus, when we expanded t into T, the first vertex of T to be ranked was placed after its parent, and all subsequent vertices of T were placed after their predecessors in T. Thus, all vertices in T and, in particular, x, are dominated. □

We are now ready to prove that our algorithm correctly determines whether the manipulator can ensure that no candidate gets more than $s(p)+1$ points.

Lemma 3.4.4. *The algorithm outputs "no" if and only if for any vote L there is a purple candidate that is undominated.*

Proof. Observe that the algorithm only outputs "no" if it finds a purple candidate with no parents. Let c be some such candidate. Now, in the original graph G each vertex has a parent. Further, if there was an edge from some x to c, and we collapsed a cycle T that contains x, but not c, there is still an edge from the resulting vertex t to c. Thus, the only way to obtain a purple vertex with no incoming edges is by collapsing a cycle T such that T contains purple vertices only, and no vertex of T has an incoming edge. By induction on the execution of the algorithm, it is easy to see that if we obtained a purple vertex with no incoming edges at some point, then in the original graph there was a group of purple vertices such that there was no edge from any red or green vertex to any of the vertices in the group. Now, in any ordering on C one of the candidates in this group would have to be ranked first. By construction, this candidate would be ranked before all its parents, so it is undominated.

Conversely, suppose that the algorithm does not answer "no", and outputs a pair (L, S) instead. We have observed that S consists of red vertices only. Thus, by Lemma 3.4.3 each purple vertex is dominated. □

It remains to show that the set S output by the algorithm contains as few candidates as possible.

Lemma 3.4.5. *At any point in the execution of the algorithm, if $\mathcal{A}(H) = (L, S)$, then in any ordering of the candidates in U in which each purple vertex in U is dominated, at least $|S|$ red vertices in U are undominated.*

Proof. The proof is by induction on the recursion depth. Suppose first that we make no recursive calls. Then our algorithm outputs $S = \emptyset$, and our claim is trivially true. Now, suppose that our claim is true if we make d

nested calls. Consider an execution of $\mathcal{A}$ which makes $d+1$ nested call, and suppose that when we call $\mathcal{A}(H')$ within this execution, it returns (L', S').

Suppose first that we made the recursive call in Step 6 of the algorithm, and therefore set $S = S' \cup \{c\}$. Suppose for the sake of contradiction that there exists a ranking of the candidates in U such that at most $|S| - 1$ candidate is undominated. Since c has no parents in H, there are at most $|S| - 2$ other red candidates that are undominated. In other words, if we recolor c green, in the resulting instance (which is exactly the instance passed to $\mathcal{A}$ during the recursive call), there are at most $|S| - 2$ undominated red candidates. Since $|S'| = |S| - 1$, this is a contradiction with the inductive assumption.

Now, suppose that we made the recursive call in Step 7 of the algorithm, and collapsed a cycle T into a vertex t. Again, assume for the sake of contradiction that there exists a ranking $\bar{L}$ of the candidates in U such that at most $|S| - 1$ candidates are undominated. Let c be the first vertex of T to appear in $\bar{L}$. Consider the ranking of U' obtained by removing all vertices of $T \setminus \{c\}$ from $\bar{L}$ and replacing c with t; denote this ranking by $\bar{L}'$. We claim that in $\bar{L}'$ at most $|S| - 1$ vertices of H' are undominated. Indeed, any parent of c in H is a parent of t in H', so t is undominated if and only if c was. On the other hand, if for some vertex x the only parent that preceded it in $\bar{L}$ was a vertex $y \in T \setminus \{c\}$, then in H' there is an edge from t to x, i.e., x is preceded by its parent t in $\bar{L}'$. For all other vertices, if they were preceded by some parent z in $\bar{L}$, they are preceded by the same parent in $\bar{L}'$. Since $|S| = |S'|$, we have shown that U' can be ordered so that at most $|S'| - 1$ vertices are undominated, a contradiction with the inductive assumption. □

Combining Lemma 3.4.4 and Lemma 3.4.5, we conclude that if our algorithm outputs (L, S), then L is the optimal vote for the manipulator and if our algorithm outputs "no", then the manipulator's utility is 0 no matter how he votes. Also, it is not hard to see that the algorithm runs in polynomial time. Thus, the proof is complete. □

Example 3. *In this example we begin from the graph G, which we obtain from the profile $\mathcal{R}_{-n}$. This graph can be seen at picture 3. We suppose that all vertices except p are red in this graph and p is green.*

Step 1-6. The only thing which algorithm does at these steps is adding p to L and deleting it from G. Step 7. After collapsing the cycle (c, d, f) to the single vertex cdf we obtain the graph at picture 3. Color of vertex cdf is red.

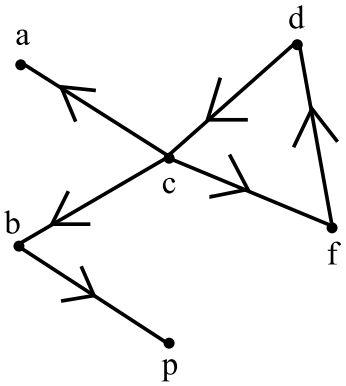

Figure 3.1: Graph G

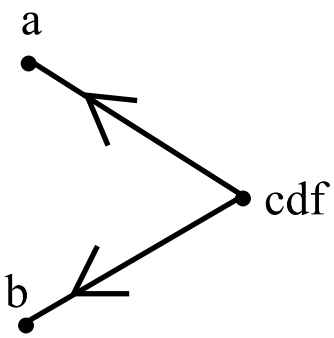

Figure 3.2: Graph G after contracting the cycle (c, d, f)

Now we return to the step 6 and color cdf green. Vertex cdf is the only green vertex in the graph, thus, add it to L. Now, both vertex a and b have parent in L and therefore can be added to L in arbitrary order (say, a, b). All vertices c, d, f are red, therefore we can replace cdf in L by vertices c, d, f in arbitrary order. For example, c, d, f.

Now we obtain the vote (p, c, d, f, a, b) and it is easy to see that there are 2 winners p and c. The utility of manipulator is $\frac{1}{2}$.

3.5 Copeland

For the Copeland rule, we give an NP-hardness reduction from the INDEPENDENT SET problem [25]. An instance of this problem is given by an undirected graph $\mathcal{G}$ and a positive integer t. It is a "yes"-instance if $\mathcal{G}$ contains an independent set of size at least t, i.e., if $\mathcal{G}$ has at least t vertices such that no two of them are connected by an edge; otherwise, it is a "no"-instance.

Our reduction makes use of a technical lemma, which essentially shows that any undirected graph $\mathcal{G}$ can be obtained as a graph of ties in an election whose size is polynomial in the size of $\mathcal{G}$; a similar result appears in [22] (Lemma 2.4).

Lemma 3.5.1. *Let $\mathcal{G}$ be an undirected graph with the vertex set $\nu_1, \ldots, \nu_s$, $s \geq 3$. Let $d(\nu_i)$ denote the degree of vertex ν_i. Then there exists a directed graph $\mathcal{G}'$ with the vertex set $G \cup Z \cup \{w\}$, where $G = \{g_1, \ldots, g_s\}$, $Z = \{z_1, \ldots, z_{4s+1}\}$, such that the outdegree d_{out} and the indegree d_{in} of each vertex of $\mathcal{G}'$ satisfy*

- $d_{\text{out}}(w) = 4s + 1, d_{\text{in}}(w) = s$;
- $d_{\text{out}}(g_i) = 4s + 1 - d(\nu_i), d_{\text{in}}(g_i) = s$ *for* $i = 1, \ldots, s$;
- $d_{\text{in}}(z) + d_{\text{out}}(z) = 5s + 1$ *and* $d_{\text{out}}(z) \leq 3s + 1$ *for all* $z \in Z$,

$\mathcal{G}'$ contains no 2-cycles, and, furthermore for $i, j \in \{1, \ldots, s\}$, g_i and g_j are not connected by an arc in $\mathcal{G}'$ if and only if there is an edge between ν_i and ν_j in $\mathcal{G}$.

Proof. The graph $\mathcal{G}'$ is constructed as follows. First, we add an arc (g, w) for each $g \in G$ and an arc (w, z) for each $z \in Z$. Further, for any $i, j \in \{1, \ldots, s\}$, $i < j$, we add an arc (g_i, g_j) if and only $(\nu_i, \nu_j) \notin \mathcal{G}$. Also, we add an arc

(z_i, z_j) for any $i, j \in \{1, \ldots, 4s+1\}$ such that $j \in \{i+1, \ldots, i+2s\}$, where summation is taken modulo $4s+1$. At this point, w has the required indegree and outdegree, $d_{\text{out}}(g), d_{\text{in}}(g) \leq s$ for any $g \in G$, and $d_{\text{in}}(z) = d_{\text{out}}(z) = 2s$ for any $z \in Z$.

Now, for $i = 1, \ldots, s$, we pick an arbitrary subset $Z_i \subseteq Z$ of size $4s + 1 - d(\nu_i) - d_i$, where d_i is the current outdegree of g_i, add an arc (g_i, z) for all $z \in Z_i$, and add an arc (z, g_i) for all $z \in Z \setminus Z_i$. After this step, the vertices in G have the desired indegree and outdegree. Moreover, for each $z \in Z$ and each $x \in (Z \cup G \cup \{w\}) \setminus \{z\}$ we have either $(x, z) \in \mathcal{G}$ or $(z, x) \in \mathcal{G}$. Finally, we have $d_{\text{in}}(z) \geq 2s$, so $d_{\text{out}}(s) \leq 3s + 1$. Therefore, $\mathcal{G}'$ has the requested properties. □

We are now ready to present the main result of this subsection.

Theorem 3.5.2. Copeland$^\alpha$-RandManipulation *is* NP*-complete for any rational* $\alpha \in [0, 1]$.

Proof. Fix a rational $\alpha \in [0, 1]$. We have argued that Copeland$^\alpha$-RandManipulation is in NP. For the hardness proof, suppose that we are given an instance $(\mathcal{G}, t)$ of Independent Set, where $\mathcal{G}$ is a graph with the vertex set $\{\nu_1, \ldots, \nu_s\}$. We will now construct an instance of our problem with a set of candidates $C = G \cup Z \cup \{w\}$, where $G = \{g_1, g_2, \ldots, g_s\}$, $Z = \{z_1, \ldots, z_{4s+1}\}$.

Given two candidates $x, y \in C$ in an n-voter election, we say that x *safely wins* a pairwise election against y (and y *safely loses* a pairwise election against x) if at least $\lfloor n/2 \rfloor + 2$ voters prefer x to y. For any candidate $x \in C$, let $\text{SW}(x)$ and $\text{SL}(x)$ denote the number of pairwise elections that x safely wins and safely loses, respectively.

Let $d(\nu_i)$ denote the degree of the vertex ν_i in $\mathcal{G}$. By Lemma 3.5.1 and Corollary 2.3.5, we can construct an election $E' = (C, \mathcal{R}')$ with a preference profile $\mathcal{R}' = (R_1, \ldots, R_{n-1})$ that has the following properties:

- $\text{SW}(w) = 4s + 1$, $\text{SL}(w) = s$;
- $\text{SW}(g_i) = 4s + 1 - d(\nu_i), \text{SL}(g_i) = s$ for $i = 1, \ldots, s$;
- $\text{SW}(z) + \text{SL}(z) = 5s + 1$ and $\text{SW}(z) \leq 3s + 1$ for any $z \in Z$;
- there is a tie between two candidates c and c' if and only if $c = g_i$, $c' = g_j$ for some $i, j \in \{1, \ldots, s\}$ and there is an edge between ν_i and ν_j in $\mathcal{G}$.

Consider an election $E = (C, \mathcal{R})$ with $\mathcal{R} = (R_1, \ldots, R_{n-1}, R)$, where R is a preference ordering that is consistent with the utility function u given by $u(w) = 0$, $u(z) = 0$ for any $z \in Z$, $u(g) = 1$ for any $g \in G$. For any $L \in \mathcal{L}(C)$, in the preference profile $(\mathcal{R}_{-n}, L)$ the Copeland$^\alpha$ score of w is $4s + 1$, and the Copeland$^\alpha$ score of each candidate $z \in Z$ is at most $3s + 1$. Moreover, the Copeland$^\alpha$ score of each $g_i \in G$ is at least $4s + 1 - d(\nu_i)$ and at most $4s + 1$; to ensure that g_i's score is $4s + 1$, the manipulator must rank g_i above all of the candidates that g_i is tied with in E' (note that for $\alpha = 1$ all candidates in G are currently tied with w, but some of them will lose points after the manipulator votes). We claim that $(\mathcal{G}, t)$ is a "yes"-instance of Independent Set if and only if $(E, u, t/(t + 1))$ is a "yes"-instance of Copeland$^\alpha$-RandManipulation.

Indeed, let $J = \{\nu_{i_1}, \ldots, \nu_{i_t}\}$ be an independent set in $\mathcal{G}$. Consider a vote L that ranks the candidates $g_{i_1}, \ldots, g_{i_t}$ first (in any order), followed by the remaining candidates in $G \cup Z$, followed by w. Clearly, in the resulting election the Copeland$^\alpha$ score of the top t candidates in L is $4s + 1$, so the manipulator's utility is at least $t/(t + 1)$.

Conversely, suppose that for some $L' \in \mathcal{L}(C)$ the manipulator's utility is at least $t/(t + 1)$. Let S' be the set of all candidates in G whose Copeland$^\alpha$ score in $(\mathcal{R}_{-n}, L')$ is $4s + 1$; we have $|S'| \geq t$. As argued above, the manipulator ranks each candidate $g \in S'$ above all candidates that g is tied with in E'. This implies that two candidates in S' cannot be tied in E', i.e., S' corresponds to an independent set in $\mathcal{G}$. □

3.6 Iterative rules

Some of the common voting rules, such as, e.g., STV, do not assign scores to candidates. Rather, they are defined via multi-step procedures. When one computes the winner under such rules, ties may have to be broken during each step of the procedure. A natural approach to winner determination under such rules is to use the *parallel universes tie-breaking* [9]: a candidate c is an election winner if the intermediate ties can be broken so that c is a winner after the final step. Thus, any such rule defines a voting correspondence in a usual way, and hence the corresponding RandManipulation problem is well-defined. In this section, we consider three rules in this class, namely, Plurality with Runoff, STV, and Ranked Pairs.

For Plurality with Runoff, RandManipulation turns out to be in P.

The main idea of the proof is that if $\mathcal{L}_c$ is the set of all votes that rank a candidate $c \in C$ first, then the best vote in $\mathcal{L}_c$ ranks all candidates other than c according to their utility.

Theorem 3.6.1. Plurality with Runoff-RandManipulation *is in* P.

Proof. Consider an election $E = (C, \mathcal{R})$ and a manipulating voter n with a utility function u.

Evidently, if we know the optimal vote in $\mathcal{L}_c$ for all $c \in C$ we can find the manipulative vote in linear time. To complete the proof, we will now show that the best vote in $\mathcal{L}_c$ ranks all candidates other than c according to their utility. Consider the vote $L \in \mathcal{L}_c$ that ranks all candidates other than c according to their utility. We will prove that the utility that the manipulator obtains in election $(C, (\mathcal{R}_{-n}, L))$, is at least the utility, that the manipulator obtains in election $(C, (\mathcal{R}_{-n}, L'))$ for any other vote L'. It is easy to see that all candidates have the same score in both elections at the first step. Therefore, a pair of candidates c_1, c_2 can be obtained as the pair of candidates at the second step in election $(C, (\mathcal{R}_{-n}, L'))$ if and only if they can be obtained as a pair of candidates at the second step in election $(C, (\mathcal{R}_{-n}, L))$.

Now, consider the second step. Suppose, candidates c_1, c_2 remain at this stage. Evidently, if they are ranked in the n-th vote in the same way in both elections, then the utility obtained by the manipulator is the same in both cases. Therefore, we only need to consider the case when c does not survive until the second step and c_1, c_2 are ranked differently with respect to each other in L and L'. Without loss of generality we can assume $u(c_1) > u(c_2)$. Denote the scores of c_1 and c_2 at the second step of the election $(C, (\mathcal{R}_{-n}, L))$ by s_1 and s_2, respectively. Thus, the scores of c_1 and c_2 at the second step of the election $(C, (\mathcal{R}_{-n}, L'))$ are $s_1 - 1$ and $s_2 + 1$, respectively. It is easy to see that the score of the candidate of larger utility increases in $(C, (\mathcal{R}_{-n}, L))$ compared to $(C, (\mathcal{R}_{-n}, L'))$. Therefore, the manipulator's utility is at least as large in $(C, (\mathcal{R}_{-n}, L))$ as in $(C, (\mathcal{R}_{-n}, L'))$. □

For STV and Ranked Pairs, RandManipulation is NP-hard. The proof of this fact hinges on an observation that allows us to inherit hardness results from the standard model of voting manipulation.

For STV and Ranked Pairs CoWinnerManipulation is known to be NP-hard (see, respectively, [5] and [49]). It is easy to see that this implies that for these rules RandManipulation is hard as well.

Proposition 3.6.2. *For any voting correspondence $\mathcal{F}$, the problems* $\mathcal{F}$-CoWinnerManipulation *many-one reduces to* $\mathcal{F}$-RandManipulation.

Proof. Given an instance (E', p) of $\mathcal{F}$-CoWinnerManipulation with $E' = (C, \mathcal{R}')$, where $\mathcal{R}' = (R_1, \ldots, R_n)$, we construct an instance (E, u, q) of $\mathcal{F}$-RandManipulation as follows. Set $E = (C, \mathcal{R})$ with $\mathcal{R} = (\mathcal{R}'_{-n}, R)$, where R ranks p first, followed by all other candidates in an arbitrary order. Also, set $u(p) = 1$, $u(c) = 0$ for $c \in C \setminus \{p\}$, and $q = 1/|C|$. It is easy to see that a "yes"-instance of $\mathcal{F}$-CoWinnerManipulation corresponds to a "yes"-instance of $\mathcal{F}$-RandManipulation and vice versa. □

Corollary 3.6.3. STV-RandManipulation *and* Ranked Pairs-RandManipulation *are* NP*-hard.*

We remark that it is not clear if these problems are in NP, since the respective winner determination problem is not known to be polynomial-time solvable; in fact, for STV it is known to be NP-hard [9].

For iterative rules one can also use randomness to break the intermediate ties. The manipulator's goal is then to maximize the expected utility with respect to the resulting distribution. Generally speaking, this problem is different from RandManipulation: while the set of candidates that win with non-zero probability is the same in both settings, the probability distribution on these candidates can be different.

3.7 Related work

In this chapter we assume that the manipulator assigns utilities to all candidates, and his goal is to vote so as to maximize his expected utility. This approach is standard in the social choice literature (see, e.g.,[27]) and has also been used in [14].

Also, randomized tie-breaking has been considered in the context of convergence to equilibria under Plurality voting [34].

There exists very recent results on Maximin under randomized tie-breaking. In [51] it was proved that when the manipulator's utilities for the candidates are given by the vector $(1, \ldots, 1, 0, \ldots, 0)$, with k ones and $m - k$ zeros, then the problem of finding an optimal vote for the manipulator is fixed-parameter tractable when parameterized by k.

3.8 Summary

We have determined the complexity of finding an optimal manipulation under the randomized tie-breaking rule for several prominent voting rules, namely, scoring rules, Maximin, Copeland$^\alpha$ for any rational $\alpha \in [0, 1]$, two variants of the Bucklin rule, Plurality with Runoff, STV, and Ranked Pairs. This provides an essentially complete picture of the complexity of RandManipulation for commonly studied voting rules (Table 3.1).

P	NP-hard
Scoring rules	Copeland
Maximin (restricted)	Maximin (general)
simplified Bucklin	STV
classic Bucklin	Ranked Pairs
Plurality w/Runoff	

Table 3.1: Complexity of RandManipulation for classic voting rules.

To compare the results for manipulation problem under randomized tie-breaking and lexicographic tie-breaking consider the following table.

	Lexicographic tie-breaking	Randomized tie-breaking
Scoring rules	P	P
(classic) Bucklin	P	P
Plurality w/Runoff	P	P
Maximin (restricted)	P	P
Maximin (general)	P	NP-hard
Copeland	P	NP-hard
STV	NP-hard	NP-hard
Ranked Pairs	NP-hard	NP-hard

Table 3.2: Complexity of Manipulation for classic voting rules under lexicographic and randomized tie-breaking.

Chapter 4

Deterministic Tie-Breaking Rules

In [6] it was proved that several well-known voting correspondences are easy to manipulate if ties are broken in manipulator's favor. In Chapter 2, we have argued that the algorithm of [6] can be modified to work for an arbitrary lexicographic tie-breaking rule. Given these easiness results, it is natural to ask whether all (efficiently computable) tie-breaking rules produce easily manipulable rules when combined with the voting correspondences considered in [6]. In this chapter we show that Maximin and Borda, as well as many families of scoring rules, become hard to manipulate if we allow arbitrary computable deterministic tie-breaking rules. This holds even if we require that the tie-breaking rule only depends on the set of the tied alternatives, rather than the voters' preferences over them; we refer to such tie-breaking rules as *simple*. Now we give the formal defenition of simple tie-breaking.

As it was defined in the preliminaries a tie-breaking rule for an election $(C, \mathcal{R})$ is a mapping $T = T(\mathcal{R}, S)$ that for any $S \subseteq C$, $S \neq \emptyset$, outputs a candidate $c \in S$.

Definition 4.0.1. *A tie-breaking rule T is called* simple *if it is polynomial-time computable and the value of $T(\mathcal{R}, S)$ is uniquely determined by S .*

Our proof also works for Copeland, thus strengthening the hardness result for second-order Copeland proved in [6] to simple tie-breaking rules. We remark, however, that our hardness result is not universal: Plurality and other scoring rules that correspond to scoring vectors with a bounded number of non-zero coordinates are easy to manipulate under any polynomial-time simple tie-breaking rule, However, if non-simple tie-breaking rules are allowed, Plurality can be shown to be hard to manipulate as well.

We will first present a specific simple tie-breaking rule T. We will then show that manipulating the composition of this rule with Borda and Maximin is NP-hard. We then show that by tweaking this tie-breaking rule a little, we can also obtain an NP-hardness result for Copeland.

Recall that an instance $\mathcal{C}$ of 3-SAT is given by a set of s variables $X = \{x_1, \ldots, x_s\}$ and a collection of t clauses $Cl = \{c_1, \ldots, c_t\}$, where each clause $c_i \in Cl$ is a disjunction of three *literals* over X, i.e., variables or their negations; we denote the negation of x_i by $\overline{x}_i$. It is a "yes"-instance if there is a truth assignment for the variables in X such that all clauses in Cl are satisfied, and a "no"-instance otherwise. This problem is known to be hard even if we assume that all literals in each clause are distinct, so from now on we assume that this is the case. Now, given d variables $x_1, \ldots, x_d$, there are exactly $\ell = \binom{2d}{3}$ 3-literal clauses that can be formed from these variables (this includes clauses of the form $x_1 \vee \overline{x}_1 \vee x_2$). Ordering the literals as $x_1 < \overline{x}_1 < \cdots < x_d < \overline{x}_d$ induces a lexicographic ordering over all 3-literal clauses. Let ϕ_i denote the i-th clause in this ordering. Thus, we can encode an instance $\mathcal{C}$ of 3-SAT with d variables as a binary string $\sigma(\mathcal{C})$ of length ℓ, where the i-th bit of $\sigma(\mathcal{C})$ is 1 if and only if ϕ_i appears in $\mathcal{C}$.

We are ready to describe T. Given a set $S \subseteq C$ of candidates, where $|C| = m$, T first checks if $m = \ell + 2s + 4$ for some $s > 0$ and $\ell = \binom{2s}{3}$. If this is not the case, it outputs the lexicographically first candidate in S and stops. Otherwise, it checks whether $c_m \in S$ and for every $i = 1, \ldots, s$, the set S satisfies $|S \cap \{c_{\ell+2i-1}, c_{\ell+2i}\}| = 1$. If this is not the case, it outputs the lexicographically first candidate in S and stops. If the conditions above are satisfied, it constructs an instance $\mathcal{C} = (X, Cl)$ of 3-SAT by setting $X = \{x_1, \ldots, x_s\}$, $Cl = \{\phi_i \mid 1 \leq i \leq \ell, c_i \in S\}$. Next, it constructs a truth assignment $(\xi_1, \ldots, \xi_s)$ for $\mathcal{C}$ by setting $\xi_i = \top$ if $c_{\ell+2i-1} \in S$, $c_{\ell+2i} \notin S$ and $\xi_i = \bot$ if $c_{\ell+2i-1} \notin S$, $c_{\ell+2i} \in S$. Finally, if $\mathcal{C}(\xi_1, \ldots, \xi_s) = \top$, it outputs c_m and otherwise it outputs the lexicographically first candidate in S. Clearly, T is simple and polynomial-time computable, and hence the problem $T \circ \mathcal{F}$-MANIPULATION is in NP for any polynomial-time computable rule $\mathcal{F}$ (and, in particular, for Borda, Maximin and Copeland). In the rest of this section, we will show that $T \circ \mathcal{F}$-MANIPULATION is NP-hard for all these rules.

4.1 Borda and other scoring rules

We will first consider the Borda rule. We will then show that essentially the same proof works for a large class of scoring rules. To simplify notation, in the proof of Lemma 4.1.1 and Theorem 4.1.2 we will denote the Borda score of a candidate x in a preference profile $\mathcal{R}$ by $s(\mathcal{R}, x)$.

Lemma 4.1.1. *For any set of candidates $C = \{c_1, \ldots, c_m\}$ with $m \geq 4$ and any vector $(\beta_1, \ldots, \beta_{m-1})$ with $\beta_i \in \{0, 1, \ldots, m\}$ for $i = 1, \ldots, m-1$ and $\beta_1 > 0$, we can efficiently construct a preference profile $\mathcal{R} = (R_1, \ldots, R_{n'})$ with $n' = m(m-1)$ voters such that for some $K \geq m^2 + m + 1$ and some $u \leq m(m-1)$ the Borda scores of all candidates satisfy $s(\mathcal{R}, c_i) = K + \beta_i$ for $i = 1, \ldots, m-1$ and $s(\mathcal{R}, c_m) = u$.*

Proof. For convenience, we will reformulate our problem as a bin-packing problem by associating the candidates with bins, and scores distributed by each voter with items. We will first argue that if we have $m(m-1)$ items of size i for each $i = 0, \ldots, m-1$, then we can place the same number of items in each bin so that the size of the i-th bin is the desired value of $s(\mathcal{R}, c_i)$. We will then show that given any assignment of items into bins such that every bin holds the same number of items, we can construct a corresponding preference profile (i.e., match items to the voters so that no voter places two items in the same bin). We remark that the latter result has also been proved in Theorem 3.1 in [12]; however, we provide a much simpler proof that generalizes easily to arbitrary scoring rules.

Let us assume that we have $m(m-1)$ items of each size. First, for each $i = 2, \ldots, m-1$, let us place m items of size i into each of the first $m-1$ bins. At this point, each of these bins contains $m(m-2)$ items and its size is $\frac{m^2(m-1)}{2} - m$. We now place β_i items of size 1 and $m - \beta_i$ items of size 0 into the i-th bin for $i = 1, \ldots, m-1$, bringing each of the first $m-1$ bins to its target size. We are left with $m(m-1)$ items of size 0 and 1, which we will place in the last bin. Clearly, at this point each bin contains $m(m-1)$ item. Set $K = \frac{m^2(m-1)}{2} - m$, and let u be the current size of the last bin. Clearly, this placement of items satisfies our constraints.

We will now show how to extract a preference profile from this assignment of items to bins. Our proof works for any assignment in which the number of items in each bin is the same. We proceed in stages: at each stage we construct one voter and remove the respective items from all bins. We need to argue that at each stage we can construct a new voter, i.e., pick one

item from each bin so that all these items have different sizes. Then, after $m(m-1)$ stages this algorithm produces the target preference profile.

The proof proceeds by induction. The basis of induction is easy: we can pick one item of size 0 from the last bin, one item of size 1 from the first bin (recall that we assume that $\beta_1 > 0$), and each other item size is guaranteed to appear in all bins, so we can pick the remaining items easily.

For the induction step, suppose that we have already constructed i voters. Then currently we have $m(m-1)-i$ items of each size, and each bin contains $m(m-1)-i$ items. We construct a bipartite graph whose vertices are item sizes and bins, and there is an edge from an item size to a bin if this bin currently contains an item of that size. We claim that this graph satisfies the conditions of Hall's theorem [45]. Indeed, pick a $k \leq m$ and consider a collection of k different item sizes. If all items with these sizes appear in at most $k-1$ bins, then these bins contain $k(m(m-1)-i)$ items, so some bin must contain at least $\frac{k}{k-1}(m(m-1)-i)$ items, a contradiction. Thus, by Hall's theorem, this graph contains a complete bipartite matching. Clearly, any such matching corresponds to a voter. Thus, we construct the $(i+1)$-st voter and remove the corresponding items from the bins. □

Theorem 4.1.2. *$T \circ$ Borda-Manipulation$^{\succ}$ is NP-hard.*

Proof. Suppose that we are given an instance $\mathcal{C}$ of 3-SAT with s variables. Note that this instance can be encoded by a binary vector $(\sigma_1, \ldots, \sigma_\ell)$, where $\ell = \binom{2s}{3}$, as described in the construction of T: $\sigma_i = 1$ if and only if $\mathcal{C}$ contains the i-th 3-variable clause with respect to the lexicographic order. We will now construct an instance of our problem with $m = \ell + 2s + 4$ candidates $c_1, c_2, \ldots, c_m$. For readability, we will also denote the first ℓ candidates by $u_1, \ldots, u_\ell$, the next $2s$ candidates by $x_1, y_1, \ldots, x_s, y_s$, and the last four candidates by d_1, d_2, w, and c.

Let $U = \{u_1, \ldots, u_\ell\}$, let $Q = \{c_i \in U \mid \sigma_i = 1\}$, and let $q = |Q|$. For convenience, we renumber the candidates in U so that $Q = \{u_1, \ldots, u_q\}$.

We will now use Lemma 4.1.1 to construct a preference profile $\mathcal{R}' = (R_1, \ldots, R_{n-1})$ with the following scores:

- $s(\mathcal{R}', w) = K + m$, $s(\mathcal{R}', c) = K + 1$;
- $s(\mathcal{R}', u_i) = K + m - i$ for $i = 1, \ldots, q$;
- $s(\mathcal{R}', u_i) = K$ for $i = q+1, \ldots, \ell$;
- $s(\mathcal{R}', x_i) = s(\mathcal{R}', y_i) = K + i + 1$ for $i = 1, \ldots, s$;

- $s(\mathcal{R}', d_1) = K$, $s(\mathcal{R}', d_2) = u$,

where $K > m^2 + m + 1$ and $u \leq m(m-1)$.

Now, consider an election with the set of candidates C and the preference profile $R_1, \ldots, R_n$, where $R_1, \ldots, R_{n-1}$ are constructed above and the preferences R_n of the last voter (who is also the manipulating voter) are given by

$$c \succ w \succ x_1 \succ y_1 \succ \ldots \succ x_s \succ y_s \succ u_1 \succ \ldots \succ u_\ell \succ d_1 \succ d_2.$$

Observe that if the manipulator votes truthfully, then w wins. Thus, a manipulation is successful if and only if voter n manages to vote so that c gets elected.

Suppose first that we have started with a "yes"-instance of 3-SAT, and let $(\xi_1, \ldots, \xi_s) \in \{\top, \bot\}^s$ be the corresponding truth assignment. For $i = 1, \ldots, s$, set $z_i = x_i$ if $\xi_i = \top$ and $z_i = y_i$ if $\xi_i = \bot$. Suppose that the manipulator submits a vote L in which he ranks $c, z_1, \ldots, z_s$ in the top $s+1$ positions (in this order), $u_q, \ldots, u_1, w$ in the bottom $q+1$ positions (in this order), and all other candidates in the remaining positions in between.

It is not hard to see that in this case the candidates c, w, $z_1, \ldots, z_s$ and all candidates in Q get $K+m$ points, while all other candidates get less than $K+m$ points. Thus, the set of tied candidates S is $Q \cup \{c, w, z_1, \ldots, z_s\}$. Therefore, given the set S, our tie-breaking rule will reconstruct $\mathcal{C}$, check whether $z_1, \ldots, z_s$ encode a satisfying truth assignment for $\mathcal{C}$ (which is indeed the case), and output $c_m = c$. Thus, in this case L is a successful manipulation.

Conversely, suppose that n submits a vote L so that c gets elected. Since we have $s(\mathcal{R}', w) - s(\mathcal{R}', c) = m - 1$, it follows that L ranks c first and w last, and hence both of them get $K+m$ points. Similarly, we can show by induction on i that for all $i = 1, \ldots, q$ it holds that n ranks u_i in the $(m-i)$-th position; thus, each candidate in Q also gets $K+m$ points. Moreover, all other candidates in $U \setminus Q$ get less than $K+m$ points, i.e., the manipulator cannot change the formula encoded by the set of tied candidates. Let S be the set of all candidates with the top score. Since c wins the election, it has to be the case that the set $S \cap \{x_1, y_1, \ldots, x_s, y_s\}$ encodes a satisfying truth assignment for $\mathcal{C}$, i.e., $\mathcal{C}$ is satisfiable. Thus, the proof is complete. □

It is not hard to generalize the result of Theorem 4.1.2 to all families of scoring vectors $(\alpha^m)_{m=1}^{\infty}$, where $\alpha^m = (\alpha_1^m, \ldots, \alpha_m^m) \in \mathbb{N}^m$, and the coordinates of each scoring vector satisfy the following conditions:

(1) $\alpha_1^m > \cdots > \alpha_m^m$;

(2) $\alpha_{m-1}^m = 1$, $\alpha_m^m = 0$;

(3) there exists a polynomial $p = p(m)$ such that $\alpha_i^m \leq p(m)$ for all $m \geq 1$ and all $i \leq m$.

That is, we require each scoring vector to be faithful and polynomially bounded, as well as to satisfy $\alpha_{m-1}^m = 1$, $\alpha_m^m = 0$.

Indeed, in the proof of Theorem 4.1.2, we can modify the construction of the non-manipulators' preference profile $\mathcal{R}'$ by requiring

- $s(\mathcal{R}', w) = K + \alpha_1^m + 1$;
- $s(\mathcal{R}', c) = K + 1$;
- $s(\mathcal{R}', u_i) = K + \alpha_1^m - \alpha_{m-i}^m + 1$ for $i = 1, \ldots, q$;
- $s(\mathcal{R}', u_i) = K$ for $i = q+1, \ldots, \ell$;
- $s(\mathcal{R}', x_i) = s(\mathcal{R}', y_i) = K + \alpha_1^m - \alpha_i^m + 1$ for $i = 1, \ldots, s$;
- $s(\mathcal{R}', d_1) = K$;
- $s(\mathcal{R}', d_2) = u$,

where $K - u > \alpha_1^m$, and $s(\mathcal{R}', z)$ denotes the score of a candidate z with respect to the scoring rule $\mathcal{F}_\alpha$. Assuming that such a profile can be constructed in polynomial time, the rest of the proof of Theorem 4.1.2 goes through as long as the scoring vector is faithful. To construct $\mathcal{R}'$, we modify the statement of Lemma 4.1.1 by requiring $0 \leq \beta_i \leq \alpha_1^m + 1$ and $K - u > \alpha_1^m$, and set the number of voters to $(\alpha_1^m + 1)(m-1)$ (which is polynomial by condition (3)). This version of Lemma 4.1.1 can be proved in essentially the same way as the original Lemma 4.1.1; the proof uses condition (2) and the observation that condition (1) implies $\alpha_i^m \geq m - i$. Thus, we obtain the following corollary.

Corollary 4.1.3. *For any family of scoring rules $(\mathcal{F}_{\alpha^m})_{m=1}^\infty$ such that the corresponding family of scoring vectors $(\alpha^m)_{m=1}^\infty$ satisfies conditions (1)–(3) it holds that $T \circ \mathcal{F}_{\alpha^m}$-*MANIPULATION$^\succ$ *is* NP*-complete.*

Theorem 4.1.2 can also be extended to scoring rules with non-faithful scoring vectors that satisfy conditions (2) and (3), as long as they have sufficiently many non-zero coordinates. However, the proof will have to be

modified by adding dummy candidates. In particular, we can show that manipulating the composition of T and k-approval is NP-hard as long as k and m are polynomially related (i.e., $m \leq q(k)$ for some polynomial q). Similarly, we can remove condition (2) by altering the tie-breaking rule T. We omit the formal proofs of these statements, as they are tedious, yet conceptually similar to the proof of Theorem 4.1.2.

On the other hand, while condition (3) does not seem essential, in the sense that it is plausible that scoring rules with exponentially large coordinates are also hard to manipulate under suitable polynomial-time computable tie-breaking rules, the current proof strategy will not work for this case. Indeed, for such scoring vectors the preference profiles constructed in the proof of (an analogue of) Lemma 4.1.1 may have exponentially many voters.

Note, however, that we cannot hope to prove an analogue of Theorem 4.1.2 for *all* scoring rules as long as we insist that the tie-breaking rule is simple: we have to require that the scoring vector has a superlogarithmic number of non-zero coordinates. Indeed, if the number of non-zero coordinates k satisfies $k = O(\log m)$, the manipulator can simply try all possible placements of the candidates into the top k positions in polynomial time. This strategy works for any simple polynomial-time tie-breaking rule, since the set of tied candidates only depends on the top k positions in the manipulator's vote. This remark can be applied for example to k-approval. On the other hand, if we drop the simplicity requirement, there exists a tie-breaking rule T' for which even Plurality is hard to manipulate. Informally, T' interprets the set of winners as a boolean formula and views the manipulator's vote as a truth assignment.

Theorem 4.1.4. *There exists a tie-breaking rule T' such that $T' \circ$ Plurality-*Manipulation$^{\succ}$ *is* NP*-complete.*

Proof. Suppose $|C| = m$. First we will construct the tie-breaking rule T'. Given a set $S \subseteq C$ of candidates, T' first checks if $m = \ell + s + 2$ for some $s > 0$ and $\ell = \binom{2s}{3}$. If this is not the case, it outputs the lexicographically first candidate in S and stops. After that T' checks whether $c_m \in S$ and if this is not the case, it outputs the lexicographically first candidate in S and stops. Otherwise, it considers the n-th (last) vote R_n. Set S' consists of all candidates that are ranked before c_{m-1} with the exception of the candidate at the first position.

Recall that an instance $\mathcal{C}$ of 3-SAT with s variables can be encoded by a binary vector $(\sigma_1, \ldots, \sigma_\ell)$ as described in Theorem 4.1.2. If $c_m \in S$, then

T' constructs an instance $\mathcal{C} = (X, Cl)$ of 3-SAT where $X = \{x_1, \ldots, x_s\}$ and Cl is encoded by the string $\{\sigma_i = 1 \mid 1 \leq i \leq \ell, c_i \in S\}$. Next, it constructs a truth assignment $(\xi_1, \ldots, \xi_s)$ for $\mathcal{C}$ by setting $\xi_i = \top$ if $c_{\ell+i} \in S'$ and $\xi_i = \bot$ otherwise. Finally, if $\mathcal{C}(\xi_1, \ldots, \xi_s) = \top$, it outputs c_m and otherwise it outputs the lexicographically first candidate in S. This completes the description of T'.

Given an instance $\mathcal{C}$ of 3-SAT with s variables such that $(\bot, \ldots, \bot)$ is not a satisfying assignment for $\mathcal{C}$ we construct an instance of $T' \circ$ Plurality-Manipulation$^{\succ}$ with $\ell + s + 2$ candidates as follows. For readability, we denote the first ℓ candidates by $u_1, \ldots u_\ell$, the next s candidates by $x_1, \ldots, x_s$, and the last two candidates by w and c.

Let $U = \{u_1, \ldots, u_\ell\}$, let $Q = \{c_i \in U \mid \sigma_i = 1\}$, and let $q = |Q|$. For convenience, we renumber the candidates in U so that $Q = \{u_1, \ldots, u_q\}$ and u_1 precedes $u_2, \ldots, u_q$ in the lexicographic ordering of the candidates.

Using at most $3\ell + 2$ votes we can obtain a profile $\mathcal{R}' = (R_1, \ldots, R_{n-1})$ such that the scores of candidates in Q are equal to 3, the score of c is equal to 2 and the scores of all other candidates are 0. Let the manipulator's preference order R_n be

$$c \succ w \succ x_1 \succ \ldots \succ x_s \succ u_1 \succ \ldots \succ u_\ell.$$

The instance of $T' \circ$ Plurality-Manipulation$^{\succ}$ is $(C, \mathcal{R})$ with $\mathcal{R} = (\mathcal{R}', R_n)$.

Suppose that the manipulator submits his truthful vote. In this case the set of tied candidates is $Q \cup \{c\}$ and $S' = \emptyset$. Therefore, u_1 is the winner of the election. Evidently, for any vote L the winner of the election $(C, (\mathcal{R}_{-n}, L))$ has at least 3 points and, thus, he is a candidate from the set $Q \cup \{c\}$. Thus, our election is a "yes"-instance of $T' \circ$ Plurality-Manipulation$^{\succ}$ if and only if c can be made the winner.

Suppose first that we have started with a "yes"-instance of 3-SAT, and let $(\xi_1, \ldots, \xi_s) \in \{\top, \bot\}^s$ be the corresponding truth assignment. Set $X' = \{x_i \mid \xi_i = \top\}$. Suppose that the manipulator submits a vote L in which he ranks c in top position followed by candidates from X' in positions $2, \ldots, |X'| + 1$ (ranked according to his preference order), w in position $|X'| + 2$, and all other candidates in the remaining positions in arbitrary order.

It is not hard to see that in this case candidate c and all candidates in Q get 3 points, while all other candidates get 0 points. Thus, the set of tied candidates S is $Q \cup \{c\}$. Therefore, given the set S, our tie-breaking rule will reconstruct $\mathcal{C}$, check whether X' encodes a satisfying truth assignment

for $\mathcal{C}$ (which is indeed the case), and output $c_m = c$. Thus, in this case L is a successful manipulation.

Conversely, suppose that n submits a vote L so that c gets elected. Since the score of c in $\mathcal{R}'$ is 2 and the score of each candidate in Q is equal to 3, it follows that L ranks c first and hence all candidates in $Q \cup \{c\}$ get 3 points in $(\mathcal{R}', L)$. All other candidates in $C \setminus (Q \cup \{c\})$ get 0 points, i.e., the manipulator cannot change the formula encoded by the set of tied candidates. Let X' be the set of all candidates who are ranked between c and w. Since c wins the election, it has to be the case that the set X' encodes a satisfying truth assignment for $\mathcal{C}$, i.e., $\mathcal{C}$ is satisfiable. Thus, the proof is complete.

□

4.2 Maximin

We will now show that $T \circ$ Maximin is hard to manipulate using essentially the same construction as in the proof of Theorem 4.1.2.

Theorem 4.2.1. *$T \circ$ Maximin-Manipulation$^{\succ}$ is* NP*-hard.*

Proof. Given a 3-SAT formula $\mathcal{C}$, we construct an election $E = (C, \mathcal{R})$ where C, U, Q and q are as in the proof of Theorem 4.1.2.

We can encode an election over a set of candidates C as a matrix $\{a(i,j)\}_{i,j \in C}$, where for all $i \neq j$ the entry $a(i,j)$ equals the number of voters that prefer i to j. By Corollary 2.3.5, for some $n = \text{poly}(m)$ we can efficiently construct a preference profile $\mathcal{R}' = (R_1, \ldots, R_{n-1})$ corresponding to the following matrix:

- $a(u_i, u_{i+1}) = b + 1$ for $i = 1, \ldots, q$;
- $a(u_i, u_{i+1}) = b - 1$ for $i = q+1, \ldots, \ell$;
- $a(x_i, y_i) = a(y_i, u_i) = b$ for $i = 1, \ldots, s$;
- $a(c, w) = a(d_1, c) = b$, $a(w, d_1) = b + 1$;
- $a(c, d_2) = g + 1$;
- $a(x, y) = g + b - a(y, x)$ if $a(y, x)$ has been defined above;
- $a(x, y) = \frac{b+g}{2}$ for all other pairs $(x, y) \in C \times C$,

where $u_{\ell+1} := u_1$, $b < m$, $g > 2m$, and $b + g = n - 1$. Now, consider an election with the set of candidates C and n voters, where for $i \leq n-1$ the preferences of the i-th voter are given by R_i, and the preferences of the last voter (who is also the manipulating voter) are given by

$$c \succ w \succ d_1 \succ d_2 \succ x_1 \succ y_1 \succ \ldots \succ x_s \succ y_s \succ u_\ell \succ \ldots \succ u_1.$$

Observe that if voter n votes truthfully, then $a(w, d_1) = b+2$, $a(w, x) > b+2$ for all $x \in C \setminus \{d_1\}$, while the Maximin score of any other candidate is at most $b + 1$, so w is the election winner. Hence, a manipulation is successful if and only if n manages to vote so that c gets elected. We will now show that this is possible if and only if we have started with a "yes"-instance of 3-SAT. Suppose first that we have started with a "yes"-instance of 3-SAT, and let $(\xi_1, \ldots, \xi_s) \in \{\top, \bot\}^s$ be the corresponding truth assignment. For $i = 1, \ldots, s$, set $z_i = x_i$ if $\xi_i = \top$ and $z_i = y_i$ if $\xi_i = \bot$. Suppose that voter n submits a vote L given by

$$c \succ z_1 \succ \ldots \succ z_s \succ u_\ell \succ \ldots \succ u_1 \succ \ldots \succ w,$$

where the candidates in $C \setminus (U \cup \{c, w, z_1, \ldots, z_s\})$ are ranked in an arbitrary order between u_1 and w. It is easy to see that after this vote the Maximin scores of c, w, $z_1, \ldots, z_s$ and the candidates in Q are $b + 1$, while all other candidates have at most b Maximin points. As argued in the proof of Theorem 4.1.2, this implies that L is a successful manipulation.

Conversely, suppose that voter n submits a vote L so that c gets elected. Before the manipulator votes, there are candidates whose Maximin score is $b + 1$. Therefore, the manipulator needs to ensure that c's Maximin score is $b + 1$, and the set of the tied candidates includes all candidates whose Maximin score is $b+1$ prior to n's vote. That is, $w, u_1, \ldots, u_q$ will be among the winners and $u_{q+1}, \ldots, u_\ell$ will not, because their scores prior to n's vote do not exceed $b - 1$. Hence, c is not the unique winning candidate. Let S be the set of tied candidates. Since c wins the election, it has to be the case that the set $S \cap \{x_1, y_1, \ldots, x_s, y_s\}$ encodes a satisfying truth assignment for the formula encoded by Q, i.e., $\mathcal{C}$, and thus $\mathcal{C}$ is satisfiable. Thus, the proof is complete. □

4.3 Copeland

For our proof for Copeland$^\alpha$ we need to change the tie-breaking rule a little. The tie-breaking rule T'' differs from T in the number of dummy candidates.

T'' checks whether the number of candidates equals to $\ell+2s+6$ and afterwards it works exactly as T.

Now we prove the following technical lemma, which is needed for the proof of the hardness result for Copeland.

Lemma 4.3.1. *Let $m = \ell + 2s + 6$, $\ell = \binom{2s}{3}$ and $s \geq 3$. Then for any $0 < q \leq \frac{\ell}{2} - 5$ there exists a directed graph $\mathcal{G}$ with the vertex set $G \cup X \cup Y \cup \{w, c, d_1, d_2, d_3, d_4\}$, where $G = \{g_1, \ldots, g_\ell\}$, $X = \{x_1, \ldots, x_s\}$, $Y = \{y_1, \ldots, y_s\}$, such that the outdegree d_{out} and the indegree d_{in} of each vertex of $\mathcal{G}$ satisfy*

- $d_{\text{out}}(g_i) = \frac{m}{2} + 2, d_{\text{in}}(g_i) = \frac{m}{2} - 3$ *for* $i = 1, \ldots, q$;
- $d_{\text{out}}(g_i) = \frac{m}{2}, d_{\text{in}}(g_i) = \frac{m}{2} - 1$ *for* $i = q+1, \ldots, \ell$;
- $d_{\text{out}}(x) = d_{\text{out}}(y) = \frac{m}{2} + 1$ *and* $d_{\text{in}}(x) = d_{\text{in}}(y) = \frac{m}{2} - 3$ *for all* $x \in X, y \in Y$;
- $d_{\text{out}}(w) = \frac{m}{2} + 2, d_{\text{in}}(w) = \frac{m}{2} - 8$;
- $d_{\text{out}}(c) = \frac{m}{2} + 2, d_{\text{in}}(c) = \frac{m}{2} - 3$;
- $d_{\text{out}}(d_1) = d_{\text{out}}(d_2) = \frac{\ell}{2} - q, d_{\text{in}}(d_1) = d_{\text{in}}(d_2) = \frac{\ell}{2} + q + 2s + 4$;
- $d_{\text{out}}(d_3) = \frac{\ell}{2}, d_{\text{in}}(d_3) = \frac{\ell}{2} + 2s + 4$;
- $d_{\text{out}}(d_4) = 1, d_{\text{in}}(d_3) = m - 2$.

Also, $\mathcal{G}$ does not contain arcs between x_i and y_i for any $i = 1, \ldots, s$ as well as arcs between w and c, d_1, d_2, d_3, d_4 and $\mathcal{G}$ does not contain cycles of length 2.

Proof. Order the vertices in $G \cup X \cup Y \cup \{c\}$ as

$$g_1 \prec \cdots \prec g_\ell \prec x_1 \prec y_1 \prec \cdots \prec x_s \prec y_s \prec c$$

For each vertex u in this order, add arcs to the next $\frac{\ell}{2} + s \bmod \ell + 2s + 1$ vertices. Then for each $j = 1, \ldots, s$ remove the arc (x_j, y_j).

We obtain $d_{\text{out}}(g_i) = d_{\text{out}}(y_j) = d_{\text{out}}(c) = \frac{\ell}{2} + s$ for all $i = 1, \ldots, \ell$, $j = 1, \ldots, s$ and $d_{\text{out}}(x_j) = \frac{\ell}{2} + s - 1$ for all $j = 1, \ldots, s$.

Next, we add arcs from w to y for all $y \in Y$ and to g_i for $i = \frac{\ell}{2} - 4, \ldots, \ell$ and from each remaining vertex of $G \cup X$ to w. Now the indegree and the outdegree of w are as in the statement of the lemma.

Next, we add arcs from all $x \in X$ and $y \in Y$ to d_1, d_2, d_3, d_4. Now all the vertices in $X \cup Y$ have desired properties.

At this stage we only need to set arcs between $G \cup \{c\}$ and d_1, d_2, d_3, d_4 and within the set d_1, d_2, d_3, d_4. First, we add arcs from all vertices of $G \cup \{c\}$ to d_4. Second, we add arcs from g_i for $i = 1, \ldots, q$ and $i = \frac{\ell}{2} + 1, \ldots, \ell$ to d_1, d_2; from g_i for $i = 1, \ldots, \frac{\ell}{2}$ to d_3. Also, we add arcs from $g_{\frac{\ell}{2}-4}, g_{\frac{\ell}{2}-3}, g_{\frac{\ell}{2}-2}$ to d_1 and from $g_{\frac{\ell}{2}-1}, g_{\frac{\ell}{2}}$ to d_2. (These arcs were not set earlier, because by our assumption $q < \frac{\ell}{2} - 5$.) For all remaining pairs g_i, d_j for $i = 1, \ldots, \ell$ and $j = 1, 2, 3$ we add arcs from d_j to g_i. Also, we add arcs from c to d_1, d_2, d_3. Now, the vertices in $G \cup \{c\}$ have the requested properties.

The remaining arcs are set as follows. We add arcs from d_1 to d_2, d_3, d_4, from d_2 to d_3, d_4 and from d_4 to d_3. It is easy to see that d_1, d_2, d_3, d_4 have the desired degrees.

Therefore, $\mathcal{G}$ has the requested properties. □

Theorem 4.3.2. *$T'' \circ$ Copeland$^\alpha$-Manipulation$^\succ$ is NP-hard for any $\alpha \in [0, 1]$.*

Proof. Given a 3-SAT formula $\mathcal{C}$, we construct an election $E = (C, \mathcal{R})$ where C, U, Q and q are the same as in the proof of Theorem 4.1.2. Without loss of generality we assume that $q \leq \frac{\ell}{2} - 5$. We say that x *safely wins* a pairwise election against y (and y *safely loses* a pairwise election against x) if at least $\frac{n}{2} + 2$ voters prefer x to y. For any candidate $x \in C$, let $\mathrm{SW}(x)$ and $\mathrm{SL}(x)$ denote the number of pairwise elections that x safely wins and safely loses, respectively. By Corollary 2.3.5 and Lemma 4.3.1, we can construct a preference profile $\mathcal{R}' = (R_1, \ldots, R_{n-1})$ with the following properties:

- $\mathrm{SW}(u_i) = \frac{m}{2} + 2$, $\mathrm{SL}(u_i) = \frac{m}{2} - 3$ for $i = 1, \ldots, q$;
- $\mathrm{SW}(u_i) = \frac{m}{2}$, $\mathrm{SL}(u_i) = \frac{m}{2} - 1$ for $i = q + 1, \ldots, \ell$;
- $\mathrm{SW}(x_i) = \mathrm{SW}(y_i) = \frac{m}{2} + 1$ for $i = 1, \ldots, s$;
- $\mathrm{SL}(x_i) = \mathrm{SL}(y_i) = \frac{m}{2} - 3$ for $i = 1, \ldots, s$;
- $\mathrm{SW}(c) = \frac{m}{2} + 1$, $\mathrm{SL}(c) = \frac{m}{2} - 3$;
- $\mathrm{SW}(w) = \frac{m}{2} + 2$, $\mathrm{SL}(w) = \frac{m}{2} - 8$;
- $\mathrm{SW}(d_1) = \mathrm{SW}(d_2) = \frac{\ell}{2} - q$, $\mathrm{SL}(d_1) = \mathrm{SL}(d_2) = \frac{\ell}{2} + q + 2s + 4$;
- $\mathrm{SW}(d_3) = \frac{\ell}{2}$, $\mathrm{SL}(d_3) = \frac{\ell}{2} + 2s + 4$;

- $\mathrm{SW}(d_4) = 1$, $\mathrm{SL}(d_4) = m - 3$;
- there is a tie between c and w, w and d_1, d_2, d_3, d_4, and x_i and y_i for $i = 1, \ldots, s$.

Now, consider an election with the set of candidates C and a set of n voters, where for $i \leq n-1$ the preferences of the i-th voter are given by R_i, and the preferences of the last voter (who is also the manipulating voter) are given by

$$c \succ w \succ d_1 \succ d_2 \succ d_3 \succ d_4 \succ x_1 \succ y_1 \succ \ldots \succ x_s \succ y_s \succ u_\ell \succ \ldots \succ u_1.$$

If the manipulator votes truthfully, then w wins. Hence, a manipulation is successful if and only if n manages to vote so that c gets elected. We will now show that this is possible if and only if we started with a "yes"-instance of 3-SAT.

Suppose first that we have started with a "yes"-instance of 3-SAT, and let $(\xi_1, \ldots, \xi_s) \in \{\top, \bot\}^s$ be the corresponding truth assignment. For $i = 1, \ldots, s$, set $z_i = x_i$ if $\xi_i = \top$ and $z_i = y_i$ if $\xi_i = \bot$. Suppose that the manipulator submits a vote L in which he ranks $c, z_1, \ldots, z_s$ in top $s+1$ positions and places w last. It is easy to see that in the resulting election $c, w, z_1, \ldots z_s, u_1, \ldots, u_q$ have $\frac{m}{2} + 2$ points and all others candidates have at most $\frac{m}{2} + 1$ points. Thus, the set of tied candidates S is $Q \cup \{c, w, z_1, \ldots, z_s\}$. Therefore, given the set S, our tie-breaking rule will reconstruct $\mathcal{C}$, check whether $z_1, \ldots, z_s$ encode a satisfying truth assignment for $\mathcal{C}$ (which is indeed the case), and output $c_m = c$. Thus, in this case L is a successful manipulation.

Conversely, suppose that c wins. Since prior to the manipulator's vote c has $\frac{m}{2} + 1$ points and w and the candidates in Q have $\frac{m}{2} + 2$ points, it follows that voter n ranks c above w and the set of tied candidates S contains c, w, and all candidates in Q. On the other hand, it cannot contain any candidates in $U \setminus Q$, as n's vote cannot affect their scores. Thus, for c to win it has to be the case that $S \cap \{x_1, y_1, \ldots, x_s, y_s\}$ encodes a satisfying assignment for the formula that corresponds to Q, i.e., $\mathcal{C}$. □

4.4 Related work

One can view the results in this chapter as a continuation of the line of work suggested in [10, 19], namely, identifying minor tweaks to voting rules that

make them hard to manipulate. Indeed, here we propose to "tweak" a voting rule by combining it with an appropriate tie-breaking rule; arguably, such a tweak affects the original rule less than the modifications proposed in [10] and [19] (i.e., combining a voting rule with a preround or taking a "hybrid" of the rule with itself or another rule).

4.5 Summary

We have explored the complexity of manipulating many common voting rules under arbitrary polynomial-time tie-breaking procedures. We have shown that Borda (and a large class of scoring rules), Maximin and Copeland are NP-hard to manipulate under simple tie-breaking. Moreover, all of our hardness reductions directly show hardness of both variants of the problem, namely, $\mathcal{F}$-MANIPULATION$^{\succ}$ and $\mathcal{F}$-MANIPULATION. Also we have demonstrated that there exists voting rules which are manipulable in polynomial time under any simple tie-breaking rule, namely, Plurality. However, there exists a (non-simple) tie-breaking rule such that its combination with Plurality is NP-hard to manipulate.

Chapter 5

Optimal Voting Manipulation

5.1 The model

In this chapter, we study the problem of finding a successful manipulative vote that minimizes the distance to the manipulator's true preference order. We consider this problem for three distances on votes, namely, the swap distance, the footrule distance and the maximum displacement distance (defined below) and the following voting rules: scoring rules, Bucklin, Copeland, and Maximin.

We begin by giving the definition of a distance.

Definition 5.1.1. *A* distance *on a space X is a mapping $d : X \times X \to \mathbb{R}$ that has the following properties for all $x, y, z \in X$:*

(1) non-negativity: $d(x, y) \geq 0$;

(2) identity of indiscernibles: $d(x, y) = 0$ if and only if $x = y$;

(3) symmetry: $d(x, y) = d(y, x)$;

(4) triangle inequality: $d(x, y) + d(y, z) \geq d(x, z)$.

In this thesis, we will be interested in distances over votes, i.e., mapping of the form $d : \mathcal{L}(C) \times \mathcal{L}(C) \to \mathbb{R}$. In fact, since we are interested in asymptotic complexity results, we will consider *families of distances* $(d^m)_{m \geq 1}$, where d^m is a distance over the space of all linear orderings of the set $\{c_1, \ldots, c_m\}$. Specifically, we will consider three such families (in the following definitions, $C = \{c_1, \ldots, c_m\}$ and R and L are two preference orders in $\mathcal{L}(C)$, also denoted as $\succ_R$ and $\succ_L$):

Swap distance. The *swap distance* $\mathrm{d}_{\mathrm{swap}}(L, R)$ is given by

$$\mathrm{d}_{\mathrm{swap}}(L, R) = |\{(c_i, c_j) \mid c_i \succ_L c_j \text{ and } c_j \succ_R c_i\}|.$$

This distance counts the number of swaps of adjacent candidates needed to transform L into R.

Footrule distance. Recall that $r(c_j, R_i)$ denotes the rank of candidate c_j in the preference order R_i: $r(c_j, R_i) = |\{c \in C \mid c \succ_i c_j\}| + 1$.

The *footrule distance* $\mathrm{d}_{\mathrm{fr}}(L, R)$ is given by

$$\mathrm{d}_{\mathrm{fr}}(L, R) = \sum_{i=1}^{m} |r(c_i, L) - r(c_i, R)|.$$

This distance calculates by how much each candidate needs to be shifted to transform L into R, and sums up all shifts.

Maximum displacement distance. The *maximum displacement distance* $\mathrm{d}_{\mathrm{md}}(L, R)$ is given by

$$\mathrm{d}_{\mathrm{md}}(L, R) = \max_{i=1,\ldots,m} |r(c_i, L) - r(c_i, R)|.$$

This distance is similar to the footrule distance; the only difference is that instead of summing up all shifts it only considers the maximum shift.

It is not hard to verify that the swap distance, the footrule distance, and the maximum displacement distance fulfill all distance axioms. It is also known [15] that the swap distance and the footrule distance are always within a factor of two from each other: we have $\mathrm{d}_{\mathrm{swap}}(L, R) \leq \mathrm{d}_{\mathrm{fr}}(L, R) \leq 2\mathrm{d}_{\mathrm{swap}}(L, R)$ for any space of candidates C and any $L, R \in \mathcal{L}(C)$.

Recall that we assume that voter n is the manipulator. In this chapter we assume that ties are broken adversarially, i.e., against the manipulator's wishes.

We will now formally describe our computational problem.

Definition 5.1.2. *Let $\mathcal{D} = (d^m)_{m \geq 1}$ be a family of integer-valued distances, where d^m is a distance over $\mathcal{L}(\{c_1, \ldots, c_m\})$. Let $\mathcal{F}$ be a voting rule. An instance of $(\mathcal{D}, \mathcal{F})$-*OptManipulation *is given by an election $(C, \mathcal{R})$ with $C = \{c_1, \ldots, c_m\}$, $\mathcal{R} = (R_1, \ldots, R_n)$, a candidate $p \in C$, and a positive integer k. It is a "yes"-instance if there exists a vote $L \in \mathcal{L}(C)$ such that $\mathcal{F}(\mathcal{R}_{-n}, L) = \{p\}$ and $d^m(R_n, L) \leq k$, and a "no"-instance otherwise.*

A few remarks are in order.

Remark 5.1.3. The problem $(\mathcal{D}, \mathcal{F})$-OPTMANIPULATION is in NP as long as all distances in $\mathcal{D}$ and the rule $\mathcal{F}$ are polynomial time computable: one can guess a vote L and check that $\mathcal{F}(\mathcal{R}_{-n}, L) = \{p\}$ and $d^m(R_n, L) \leq k$. In particular, it is in NP for all distance families and voting rules considered in this thesis.

Remark 5.1.4. We formulated OPTMANIPULATION as a decision problem. However, it also admits a natural interpretation as an optimization problem: in this case, we are given an election $(C, \mathcal{R})$ and a candidate p, and the goal is to find the smallest value of k such that there exists a vote $L \in \mathcal{L}(C)$ at distance at most k from R_n that satisfies $\mathcal{F}(\mathcal{R}_{-n}, L) = \{p\}$ (k is assumed to be $+\infty$ if there is no vote L with $\mathcal{F}(\mathcal{R}_{-n}, L) = \{p\}$). In this version of the problem, one can relax the optimality condition, and ask for an *approximately optimal* manipulative vote: an algorithm is said to be a *ρ-approximation algorithm* for $(\mathcal{D}, \mathcal{F})$-OPTMANIPULATION, $\rho \geq 1$, if, given an instance of the problem for which the correct answer is $k \in \mathbb{R} \cup \{+\infty\}$, it outputs a value k' that satisfies $k \leq k' \leq \rho k$. We will consider the optimization version of OPTMANIPULATION (and prove hardness of approximaton results) for Copeland and Maximin under swap distance (Sections 5.2) and footrule distance (Section 5.3).

Remark 5.1.5. In our definition of OPTMANIPULATION, the manipulator wants to make a specific candidate elected; the identity of this candidate is given as a part of the instance description. An alternative approach would be to ask if the manipulator can obtain what he considers a better outcome by submitting a non-truthful vote, i.e., whether there is a vote $L \in \mathcal{L}(C)$ such that $d^m(R_n, L) \leq k$ and $\mathcal{F}(\mathcal{R}_{-n}, L) \succ_n \mathcal{F}(\mathcal{R})$; we will refer to this problem as OPTMANIPULATION$^{\succ}$ (see discussion in Chapter 2). Clearly, an efficient algorithm for OPTMANIPULATION can be used to solve OPTMANIPULATION$^{\succ}$, by determining the winner w under truthful voting, and then running the OPTMANIPULATION algorithm for all candidates that the manipulator ranks above w. Hence, OPTMANIPULATION is at least as hard as OPTMANIPULATION$^{\succ}$, In what follows, we will provide polynomial-time algorithms for the "harder" problem OPTMANIPULATION. On the other hand, all our NP-hardness results apply to the "easier" problem OPTMANIPULATION$^{\succ}$: in fact, in all our hardness proofs the manipulator's goal will be to make his favorite candidate the election winner. Using OPTMANIPULATION as our

base problem allows for a direct comparison between the problem of finding the optimal manipulation and the swap bribery problem (see Section 5.5).

5.2 Swap distance

We start by considering optimal manipulability with respect to what is perhaps the best known distance on votes, namely, the swap distance $\mathrm{d}_{\mathrm{swap}}$.

5.2.1 Scoring rules and Bucklin

The main result of this section is a simple polynomial-time algorithm that solves OptManipulation for swap distance and an arbitrary scoring rule; we then show that this algorithm can be adapted to work for the Bucklin rule.

An observation that will be important for our analysis of scoring rules in this and subsequent sections is that once we select the position of the candidate p whom manipulator tries to make the winner, we know final score of p. Thus, once p's position is fixed, it remains to rank other candidates so that their scores remain strictly lower than that of p (recall that we use adversarial tie-breaking). More formally, let $s_\alpha(c)$ be the total number of points a candidate c receives from non-manipulators under a voting rule $\mathcal{F}_\alpha$; we will say that a position j is *safe* for a candidate c_ℓ given that p is ranked in position f if $s_\alpha(c_\ell) + \alpha_j < s_\alpha(p) + \alpha_f$. Clearly, for a manipulation to be successful, all candidates other than p should be ranked in positions that are safe for them.

Fix a scoring rule $\mathcal{F}_\alpha$ with $\alpha = (\alpha_1, \ldots, \alpha_m)$. Our algorithm relies on a subroutine $\mathcal{A}$ that given an election $(C, \mathcal{R})$ with $|C| = m$, a candidate p, and a position f in n's vote, finds an optimal manipulation for n among all votes that rank p in position f. More formally, let

$$\mathcal{L}^f(\alpha) = \{L \in \mathcal{L}(C) \mid \mathcal{F}_\alpha(\mathcal{R}_{-n}, L) = \{p\}, r(p, L) = f\};$$

our subroutine outputs

- $\perp$ if $\mathcal{L}^f(\alpha)$ is empty;
- a vote $\hat{L}$ such that $\mathrm{d}_{\mathrm{swap}}(\hat{L}, R_n) \leq \mathrm{d}_{\mathrm{swap}}(L, R_n)$ for all $L \in \mathcal{L}^f(\alpha)$ otherwise.

Given $\mathcal{A}$, we can easily solve $(\mathrm{d}_{\mathrm{swap}}, \mathcal{F}_\alpha)$-OPTMANIPULATION: we run $\mathcal{A}$ for all values of f between 1 and m and output "yes" if at least one of these calls returns a vote $\hat{L}$ with $\mathrm{d}_{\mathrm{swap}}(\hat{L}, R_n) \leq k$. Thus the running time of our algorithm is m times the running time of $\mathcal{A}$. It remains to describe $\mathcal{A}$.

Theorem 5.2.1. *For any $\alpha = (\alpha_1, \ldots, \alpha_m) \in {\mathbb{Z}_+}^m$ there exists a procedure $\mathcal{A}$ that takes an n-voter m-candidate election $(C, \mathcal{R})$, a candidate $p \in C$, and a position $f \in \{1, \ldots, m\}$ as its input, outputs $\perp$ if $\mathcal{L}^f(\alpha) = \emptyset$ and a vote $\hat{L}$ that satisfies $\mathrm{d}_{\mathrm{swap}}(\hat{L}, R_n) \leq \mathrm{d}_{\mathrm{swap}}(L, R_n)$ for all $L \in \mathcal{L}^f(\alpha)$ otherwise, and runs in time $O(m^2 \log(n\alpha_{\max}))$.*

Proof. For convenience, let us renumber the candidates in C so that $c_m = p$ and $c_1 \succ_n \ldots \succ_n c_{m-1}$. Our algorithm proceeds in $m-1$ rounds. In the ℓ-th round, $\ell = 1, \ldots, m-1$, we determine the final position of candidate c_ℓ; we then say that this candidate is *pinned* to that position, and the position becomes *unavailable*. Initially, all candidates are unpinned and all positions are available.

Initialization: We pin p to position f (thus f becomes unavailable), and then fill the remaining positions with the candidates in $C \setminus \{p\}$, in the order of n's preferences, i.e., placing c_1 in the highest available position and c_{m-1} in the lowest available position. In what follows, we will shift the candidates around in order to make p the winner.

Round ℓ, $\ell = 1, \ldots, m-1$ Suppose that in the beginning of the round candidate c_ℓ is ranked in position j. If j is safe for c_ℓ, we pin c_ℓ to position j (which then becomes unavailable) and proceed to the next round. Otherwise, we find the smallest value of h such that position h is available and safe for c_ℓ; if no such value of h can be found, we terminate and return $\perp$. If a suitable value of h has been identified (note that $h > j$), then c_ℓ gets pinned to position h, and all unpinned candidates in positions $j+1, \ldots, h$ are shifted one available position upwards.

If $\mathcal{A}$ does not abort (i.e., return $\perp$), it terminates at the end of the $(m-1)$-st round and returns the vote obtained at that point. Each round involves $O(m)$ score comparisons and shifts, and each comparison can be performed in time $O(\log(n\alpha_1))$; this implies the bound of $O(m^2 \log(n\alpha_1))$ on the running time. It remains to argue that $\mathcal{A}$ works correctly.

The following observation will be useful for our analysis.

Lemma 5.2.2. *Suppose that at the beginning of round ℓ candidate c_ℓ is ranked in position j. Then positions $1, \ldots, j-1$ are not available at that point.*

Proof. An easy inductive argument shows that the set of candidates ranked above c_ℓ at the beginning of round ℓ is a subset of $\{c_1, \ldots, c_{\ell-1}\}$. For each $t = 1, \ldots, \ell - 1$, candidate c_t is pinned in round t and therefore by the beginning of round ℓ his position is unavailable. As this holds for all positions above j, the lemma is proved. □

We split the rest of proof into two lemmas.

Lemma 5.2.3. *If the subroutine $\mathcal{A}(C, \mathcal{R}, p, f)$ outputs a vote $\hat{L}$ then $\hat{L} \in \mathcal{L}^f(\alpha)$, and if it outputs $\perp$ then $\mathcal{L}^f(\alpha) = \emptyset$.*

Proof. By construction, if $\mathcal{A}$ outputs a vote $\hat{L}$, then $r(p, \hat{L}) = f$. Moreover, every other candidate c_j can only be pinned to a position that is safe for him. Since $\mathcal{A}$ returns $\hat{L}$ only when all candidates in C are pinned, we have $\mathcal{F}_\alpha(\mathcal{R}_{-n}, \hat{L}) = \{p\}$, and hence $\hat{L} \in \mathcal{L}^f(\alpha)$.

Now, suppose that $\mathcal{A}(C, \mathcal{R}, p, f) = \perp$. This means that for some candidate c_ℓ, $\ell \leq m - 1$, our algorithm was unable to find an available safe position. Let $\hat{L}$ be the vote constructed by the algorithm by the beginning of round ℓ, and let h be the lowest available position at the beginning of round ℓ.

Suppose for the sake of contradiction that $\mathcal{L}^f(\alpha) \neq \emptyset$, and let L be some vote in $\mathcal{L}^f(\alpha)$. Since the algorithm has output $\perp$, position h is not safe for c_ℓ. Thus, in L candidate c_ℓ is ranked in position $h + 1$ or lower. Consequently, some candidate c_t that is ranked in position $h + 1$ or lower in $\hat{L}$ must be ranked in position h or higher in L. Since positions $h + 1, \ldots, m$ are not available at the beginning of round ℓ, they are occupied by candidates who were pinned to these positions in earlier rounds (and, possibly, by p), i.e., $t < \ell$. This means that position h was available when c_t was processed, but the algorithm chose not to place c_t in position h. By Lemma 5.2.2, it was not the case that c_t was pinned to the position it was in at the beginning of round t. Hence, the reason why c_t was ranked in position $h+1$ or lower was that h (and, *a forteriori*, any position above h) was not safe for c_t. On the other hand, we have argued that c_t is ranked in position h or higher in L, a contradiction with $L \in \mathcal{L}^f(\alpha)$. Thus it has to be the case that $\mathcal{L}^f(\alpha) = \emptyset$. □

Lemma 5.2.4. *If $\mathcal{A}(C, \mathcal{R}, p, f) = \hat{L}$, then $\mathrm{d}_{\mathrm{swap}}(\hat{L}, R_n) \leq \mathrm{d}_{\mathrm{swap}}(L, R_n)$ for all $L \in \mathcal{L}^f(\alpha)$.*

Proof. We will prove a somewhat stronger statement: there is a unique optimal vote in $\mathcal{L}^f(\alpha)$, and this vote coincides with $\hat{L}$. Suppose for the sake of contradiction that there exists a vote $L \in \mathcal{L}^f(\alpha)$ such that $\mathrm{d}_{\mathrm{swap}}(L, R_n) \leq \mathrm{d}_{\mathrm{swap}}(L', R_n)$ for all $L' \in \mathcal{L}^f(\alpha)$ and $L \neq \hat{L}$. Let c_ℓ be the first candidate ranked differently by L and $\hat{L}$, i.e., $\ell = \min\{j \mid r(c_j, L) \neq r(c_j, \hat{L})\}$.

Suppose first that $r(c_\ell, \hat{L}) > r(c_\ell, L)$. It cannot be the case that c_ℓ remains in place during round ℓ: by Lemma 5.2.2 all positions above c_ℓ in $\hat{L}$ are filled with candidates in $\{c_1, \ldots, c_{\ell-1}\}$, and $r(c_j, \hat{L}) = r(c_j, L)$ for $j < \ell$. Hence, c_ℓ has to move during round ℓ. Now, $r(c_\ell, \hat{L})$ is the highest available position that is safe for c_ℓ. Since $r(c_\ell, L)$ is necessarily safe, it follows that $r(c_\ell, L)$ must be unavailable at the beginning of round ℓ. However, this means that there is a candidate c_j, $j < \ell$, pinned to this position in $\hat{L}$, and all such candidates are ranked in the same positions in L and $\hat{L}$, a contradiction.

Thus, it has to be the case that $r(c_\ell, \hat{L}) < r(c_\ell, L)$. Let c_j be the candidate ranked in position $r(c_\ell, \hat{L})$ in L; we have $j > \ell$ by our choice of ℓ. Let L' be the vote obtained from L by swapping c_ℓ and c_j. We claim that $L' \in \mathcal{L}^f(\alpha)$ and $\mathrm{d}_{\mathrm{swap}}(L', R_n) < \mathrm{d}_{\mathrm{swap}}(L, R_n)$, thus contradicting our choice of L.

To see that $L' \in \mathcal{L}^f(\alpha)$, observe that after the swap the scores of all candidates other than c_ℓ do not go up, and $r(c_\ell, L') = r(c_j, L) = r(c_\ell, \hat{L})$, so position $r(c_\ell, L')$ is safe for c_ℓ. It remains to prove that $\mathrm{d}_{\mathrm{swap}}(L', R_n) < \mathrm{d}_{\mathrm{swap}}(L, R_n)$. To this end, we need and additional definition: we say that a pair of candidates (c, c') is an *inversion* in a vote R if $r(c, R_n) < r(c', R_n)$, but $r(c, R) > r(c', R)$. Clearly, the swap distance from R to R_n is simply the number of inversions in R. Thus, our goal is to show that L' has fewer inversions than L.

Observe first that (c_j, c_ℓ) is an inversion in L, but not in L'. Among all other pairs of candidates, it suffices to consider pairs of the form (c_j, c) and (c, c_ℓ), where c is ranked between c_j and c_ℓ in L; any other pair of candidates is an inversion in L if and only if it is an inversion in L'.

Since $j > \ell$, we have three possibilities:

$c_\ell \succ_n c \succ_n c_j$. In this case, both (c_j, c) and (c, c_ℓ) are inversions in L, but neither of them is an inversion in L'.

$c_\ell \succ_n c_j \succ_n c$. In this case, (c, c_ℓ) is an inversion in L, but (c_j, c) is not. On the other hand, (c, c_j) is an inversion in L', but (c_ℓ, c) is not.

$c \succ_n c_\ell \succ_n c_j$. In this case, (c_j, c) is an inversion in L, but (c, c_ℓ) is not. On the other hand, (c_ℓ, c) is an inversion in L', but (c, c_j) is not.

Thus, for any candidate c ranked between c_j and c_ℓ in L the pairs involving c contribute at least as much to the inversion count of L as to that of L'. By taking into account the pair (c_j, c_ℓ) itself, we conclude that $\mathrm{d}_{\mathrm{swap}}(L', R_n) < \mathrm{d}_{\mathrm{swap}}(L, R_n)$, a contradiction.

It follows that $\hat{L}$ is the optimal vote in $\mathcal{L}^f(\alpha)$ and the proof of the lemma is complete. □

The theorem now follows easily from Lemmas 5.2.3 and 5.2.4. □

We have already explained how to convert the subroutine $\mathcal{A}$ into an algorithm for OptManipulation. Thus, we obtain the following corollary.

Corollary 5.2.5. *For every polynomial-time computable family* $\hat{\mathcal{F}} = (\mathcal{F}^m_\alpha)_{m=1,\ldots}$ *of scoring rules, the problem* $(\mathrm{d}_{\mathrm{swap}}, \hat{\mathcal{F}})$-OptManipulation *is in* P.

For the Bucklin rule, the algorithm is essentially the same; the only difference is in the definition of a safe position.

Theorem 5.2.6. $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Bucklin})$-OptManipulation *is in* P.

Proof. Consider an election $(C, \mathcal{R})$. Just as in the proof of Theorem 5.2.1, it suffices to design a procedure that, for a given value of $f \in \{1, \ldots, m\}$, searches for the best manipulative vote that ranks p in position f and returns $\perp$ if no such vote can make p the unique winner.

Fix a particular value of f, and let $\mathcal{L}_f = \{L \in \mathcal{L}(C) \mid r(p, L) = f\}$. Let L_f be an arbitrary vote in $\mathcal{L}_f$. Let r^* be the smallest value of r such that p's r-approval score in $(C, (\mathcal{R}_{-n}, L_f))$ is greater than $n/2$; note that r^* does not depend on the choice of L_f. For every candidate $c \in C$, and every $r = 1, \ldots, m$, let $s_r(c)$ denote c's r-approval score in $(C, \mathcal{R}_{-n})$, and let s be p's r^*-approval score in $(C, (\mathcal{R}_{-n}, L_f))$; note that $s > n/2$.

To make p the winner, we need to ensure that r^* is the Bucklin winning round and that the r^*-approval score of any candidate $c \in C \setminus \{p\}$ does not exceed s. Thus, if there is a candidate $c \in C \setminus \{p\}$ such that $s_r(c) > n/2$ for some $r < r^*$ or $s_{r^*}(c) \geq s$, then there is no vote in $\mathcal{L}_f$ that makes p the unique election winner, so we return $\perp$ and stop.

Now, suppose that this is not the case. Set

$$C_1 = \{c \in C \setminus \{p\} \mid s_{r^*}(c) = s - 1\},$$

$$C_2 = \{c \in C \setminus (C_1 \cup \{p\}) \mid s_r(c) = \lfloor \frac{n}{2} \rfloor \text{ for some } r < r^*\}.$$

Intuitively, candidates from C_1 can prevent p from winning by receiving the same r^*-approval score as p, which happens if they are ranked in the top r^* positions. Similarly, candidates from C_2 can prevent p from winning by receiving a strict majority vote in an earlier round; this happens if they are ranked in the top $r^* - 1$ positions. Thus, p is the unique Bucklin winner in the election where the manipulator submits a vote $L \in \mathcal{L}_f$ if and only if

- $r(c, L) > r^*$ for all $c \in C_1$ and
- $r(c, L) \geq r^*$ for all $c \in C_2$.

We will say that a position j is *safe* for a candidate $c \in C \setminus \{p\}$ if

- $c \notin C_1 \cup C_2$ or
- $c \in C_1$ and $j > r^*$ or
- $c \in C_2$ and $j \geq r^*$.

The argument above shows that p is the unique Bucklin winner in the election $(C, (\mathcal{R}_{-n}, L))$ if and only if in L each candidate $c \neq p$ is ranked in a position that is safe for him.

Given this definition of a safe position, we can apply the algorithm for scoring rules described in the proof of Theorem 5.2.1; note that this algorithm operates in terms of safe positions rather than actual scores. The proofs of correctness and optimality are identical to those for scoring rules (these proofs, too, are phrased in terms of safe positions). □

5.2.2 Maximin and Copeland

For both Maximin and Copeland, finding an optimal manipulation with respect to the swap distance turns out to be NP-hard. In fact, we will prove that the optimization versions of these problems (see Remark 5.1.4) cannot be approximated up to a factor of $\delta \log |C|$ for some $\delta > 0$ unless P=NP; this implies, in particular, that the decision versions of these problems are NP-hard (and hence, by Remark 5.1.3, NP-complete).

We provide reductions from the optimization version of the Set Cover problem [25]. Recall that an instance of Set Cover is given by a ground

set $G = \{g_1, \ldots, g_t\}$ and a collection $\mathcal{S} = \{S_1, \ldots, S_r\}$ of subsets of G. In the optimization version of the problem, the goal is to find the smallest value of h such that G can be covered by h sets from $\mathcal{S}$; we denote this value of h by $h(G, \mathcal{S})$. More formally, we are interested in the smallest value of h such that $G = \cup_{S' \in \mathcal{S}'} S'$ for some collection of subsets $\mathcal{S}' \subseteq \mathcal{S}$ with $|\mathcal{S}'| = h$. A *ρ-approximation algorithm* for Set Cover is a procedure that, given an instance $(G, \mathcal{S})$ of set cover, outputs a value h' that satisfies $h(G, \mathcal{S}) \leq h' \leq \rho \cdot h(G, \mathcal{S})$. There exists a $\delta > 0$ such that Set Cover does not admit a polynomial-time $\delta \log t$-approximation algorithm unless *P=NP* [38]. The inapproximability result still holds if we assume that (1) $G = \cup_{S \in \mathcal{S}} S$; (2) $t \leq r$; and (3) $r \leq t^K$ for some positive constant K. Indeed, if (1) fails, the instance does not admit a solution, (2) can be achieved by duplicating sets in $\mathcal{S}$, and (3) follows by a careful inspection of the proof in [38]. Thus, in what follows, we only consider instances of Set Cover that satisfy conditions (1)–(3).

Theorem 5.2.7. *There exists a $\delta > 0$ s. t.* ($\mathrm{d}_{\mathrm{swap}}$, Maximin)-OptManipulation *does not admit a polynomial-time $\delta \log |C|$-approximation algorithm unless* P=NP.

Proof. Suppose that we are given an instance $(G, \mathcal{S})$ of Set Cover with $G = \{g_1, \ldots, g_t\}$, $\mathcal{S} = \{S_1, \ldots, S_r\}$ that satisfies conditions (1)–(3).

In our election, the candidate set is $C = \{p\} \cup G \cup X \cup S$, where $X = \{x_1, \ldots, x_{2r}\}$ and $S = \{s_1, \ldots, s_r\}$.

Corollary 2.3.5 implies that we can construct a preference profile $\mathcal{R}'$ with n' voters, where n' is polynomially bounded in t and r, so that n' is even and:

- For any $c \in C \setminus \{p\}$ exactly $n'/2 - 2$ voters prefer p to c.
- For any $S_j \in \mathcal{S}$ and any $g_\ell \in S_j$ exactly $n'/2 - 2$ voters prefer g_ℓ to s_j.
- For any other pair of candidates $(c, c') \in G \cup S \times G \cup S$, exactly $n'/2$ voters prefer c to c'.
- For $j = 1, \ldots, 2r - 1$ exactly $n'/2 - 4$ voters prefer x_j to x_{j+1}, and $n'/2 - 4$ voters prefer x_{2r} to x_1.
- For any $g_j \in G$ and any $x \in X$ exactly $n'/2$ voters prefer g_j to x.
- For any $s_j \in S$ and any $x \in X$ exactly $n'/2 - 4$ voters prefer s_j to x.

Denote the Maximin score of candidate c in election $(C, \mathcal{R}')$ by $s(c)$. We have $s(p) = n'/2 - 2$, $s(g_j) = n'/2 - 2$ for any $g_j \in G$ (this follows from condition (1)), $s(s_j) = n'/2 - 4$ for any $s_j \in S$, and $s(x_j) = n'/2 - 4$ for any $x_j \in X$.

We let $n = n' + 1$, $i = n$ and set our preference profile to be $\mathcal{R} = (\mathcal{R}', R_n)$, where voter n ranks the candidates as

$$p \succ g_1 \succ \ldots \succ g_t \succ x_1 \succ \ldots \succ x_{2r} \succ s_1 \succ \ldots \succ s_r.$$

This completes the description of our $(\mathrm{d}_{\mathrm{swap}}, \text{Maximin})$-OptManipulation instance (as we consider the optimization version of the problem, we need not specify k).

Observe that p's final Maximin score is $n'/2 - 1$ if and only if the manipulator ranks p first. Further, the final Maximin score of any candidate in $X \cup S$ is at most $n'/2 - 3$. Finally, the final Maximin score of a candidate $g_j \in G$ is $n'/2 - 1$ if in the manipulator's vote g_j appears above all candidates s_ℓ such that $g_j \in S_\ell$ and $n'/2 - 2$ otherwise. Thus, to make p the unique winner, the manipulator should rank him first, and rank each candidate $g_j \in G$ below a candidate representing a set that covers g_j.

Suppose that $h(G, \mathcal{S}) = h$, i.e., there exists a collection of subsets $\mathcal{S}' = \{S_{i_1}, \ldots, S_{i_h}\}$ with $i_1 < \ldots < i_h$ such that $\cup_{S' \in \mathcal{S}'} S' = G$. Consider a vote L that ranks p first, followed by candidates $s_{i_1}, \ldots, s_{i_h}$ (in this order), followed by candidates in $X \cup G$ (in the order of their appearance in R_n), followed by the remaining candidates in S (in the order of their appearance in R_n). By the argument above, p is the unique Maximin winner of $(C, (\mathcal{R}', L))$. Furthermore, we have $\mathrm{d}_{\mathrm{swap}}(L, R_n) \leq h(t + 2r + (r - h))$: to transform R_n into L, we swap each of the candidates s_{i_j}, $j = 1, \ldots, h$, with (a) t candidates in G, (b) $2r$ candidates in X and (c) at most $r - h$ candidates in S. By condition (2), we obtain $\mathrm{d}_{\mathrm{swap}}(L, R_n) \leq 4hr$.

On the other hand, consider an arbitrary vote L' such that p is the unique Maximin winner of $(C, (\mathcal{R}', L'))$. Construct a bipartite graph with the vertex set $G \cup S$ in which there is an edge between g_j and s_ℓ if and only if s_ℓ is ranked above g_j in L'. We claim that this graph contains a matching of size h. To see this, consider a greedy algorithm that constructs a matching by inspecting the vertices in G one by one and matching each vertex to one of its previously unmatched neighbors in S; if some vertex in G cannot be matched, the algorithm proceeds to the next vertex. If this algorithm terminates without finding h edges, it means that the matched vertices in

S correspond to a cover of size at most $h-1$, a contradiction. Consider a pair of candidates (g_j, s_ℓ) that corresponds to an edge of this matching, and an arbitrary candidate $x \in X$. It cannot be the case that L' ranks g_j above x and x above s_ℓ: otherwise, by transitivity, L' would rank g_j above s_ℓ. Therefore, at least one of the pairs (g_j, x) and (x, s_ℓ) is ordered differently in R_n and L', and therefore each edge of the matching contributes at least $2r$ to the swap distance between L' and R_n. Summing over all edges of the matching, we obtain that $\mathrm{d}_{\mathrm{swap}}(R_n, L') \geq 2hr$.

Now, suppose that there is a polynomial-time ρ-approximation algorithm $\mathcal{M}$ for $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OptManipulation: given an instance of $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OptManipulation that admits a successful manipulative vote L with $\mathrm{d}_{\mathrm{swap}}(L, R_i) = k$, this algorithm outputs a value k' that satisfies $k \leq k' \leq \rho k$. Consider the following algorithm $\mathcal{M}'$ for Set Cover: given an instance $(G, \mathcal{S})$ of Set Cover with $|G| = t$, $|\mathcal{S}| = r$, $\mathcal{M}'$ transforms it into an instance of $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OptManipulation as described above, applies $\mathcal{M}$, and divides the returned value by $2r$. Clearly, $\mathcal{M}'$ runs in polynomial time. We claim that it provides a 2ρ-approximation algorithm for Set Cover.

Indeed, let $h = h(G, \mathcal{S})$. In this case for the corresponding instance of $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OptManipulation there exists a successful manipulative vote L with $\mathrm{d}_{\mathrm{swap}}(L, R_i) \leq 4hr$ and hence $\mathcal{M}$ outputs a value k' that satisfies $k' \leq 4\rho hr$. On the other hand, for any successful manipulative vote L' we have $\mathrm{d}_{\mathrm{swap}}(L', R_i) \geq 2hr$, and hence the value k' output by $\mathcal{M}$ satisfies $k' \geq 2hr$. Thus, $\mathcal{M}$ produces a value h' that satisfies $h \leq h' \leq 2\rho h$.

Since $|C| = O(t + r)$ and, by condition (3), $r \leq t^K$, we have $\log |C| \leq \gamma \log t$ for a suitable constant $\gamma > 0$. Therefore, if there exists a polynomial-time $\delta \log |C|$-approximation algorithm for $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-OptManipulation for some constant $\delta > 0$, then there exists a polynomial-time $\delta' \log t$-approximation algorithm for Set Cover for some constant $\delta' > 0$, and, by [38], this implies P=NP. □

The argument for Copeland is similar.

Theorem 5.2.8. *There exists a $\delta > 0$ such that for any $\alpha \in \mathbb{Q} \cap [0, 1]$, $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Copeland}^\alpha)$-OptManipulation does not admit a polynomial-time $\delta \log |C|$-approximation algorithm unless* P=NP.

Proof. Suppose that we are given an instance $(G, \mathcal{S})$ of Set Cover with $G = \{g_1, \ldots, g_t\}$, $\mathcal{S} = \{S_1, \ldots, S_r\}$ that satisfies conditions (1)–(3); we will

additionally assume that t and r are odd.

In our election, the candidate set is $C = \{p\} \cup G \cup X \cup S$, where $X = \{x_1, \ldots, x_{6r}\}$ and $S = \{s_1, \ldots, s_r\}$.

It is easy to see that using a variant of the construction in the proof of Lemma 3.5.1 (with summation modulo t for adding the arcs between vertices of G and summation modulo $7r$ for adding the arcs between vertices of $X \cup S$), we can construct a graph that induces the following outcomes of pairwise elections between the candidates:

- Candidate p beats all candidates in $G \cup S$ as well as $(t-1)/2$ candidates in X, and loses to all other candidates in X.
- Every candidate $g_i \in G$ is tied with all candidates s_ℓ such that $g_i \in S_\ell$ and beats all other candidates in $X \cup S$.
- Every candidate in G beats exactly $(t-1)/2$ other candidates in G.
- Every candidate in $X \cup S$ beats exactly $(7r-1)/2$ other candidates in $X \cup S$.

By Corollary 2.3.5, we can construct a preference profile $\mathcal{R}'$ with n' voters, where n' is polynomially bounded in t and r, so that n' is even and the outcomes of pairwise elections between candidates are as described above.

We let $n = n' + 1$ and set our preference profile to be $\mathcal{R} = (\mathcal{R}', R_n)$, where voter n (the manipulator) ranks the candidates as

$$c \succ g_1 \succ \ldots \succ g_t \succ x_1 \succ \ldots \succ x_{6r} \succ s_1 \succ \ldots \succ s_r.$$

This completes the description of our $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Copeland}^\alpha)$-OPTMANIPULATION instance; note that n is odd and therefore the value of α is unimportant for our analysis.

Observe that in $\mathcal{R}$ the Copeland score of p is $(t-1)/2 + 7r$, the Copeland score of each $g_j \in G$ is $(t-1)/2 + 7r$, and the Copeland score of each candidate in $X \cup S$ is at most $(7r-1)/2 + 1 < 4r$. Thus, under truthful voting p is not the unique winner; indeed, for p to be the unique winner, in the manipulator's vote every candidate $g_j \in G$ must be ranked below some candidate s_ℓ such that $g_j \in S_\ell$. Note also that the manipulator's vote can only affect the outcomes of pairwise elections for candidate pairs of the form (g_j, s_ℓ), $g_j \in S_\ell$. Thus, no matter how the manipulator votes, the Copeland score of every candidate $x \in X$ is at most $4r < (t-1)/2 + 7r$, and the

Copeland score of every candidate $s_\ell \in S$ is at most $4r + t < (t-1)/2 + 7r$ (recall that we assume $t < r$), and hence candidates in $X \cup S$ are not among the election winners. We conclude that L is a successful manipulative vote if and only if it ranks each candidate $g_j \in G$ below a candidate representing a set that covers g_j. This condition is almost identical to the one in the proof of Theorem 5.2.7, and, from this point on, the proof repeats the proof of Theorem 5.2.7 almost verbatim; the reader can verify that the analysis is not negatively impacted by the fact that the set X contains $6r$ candidates (rather than $2r$ candidates, as in the proof of Theorem 5.2.7). □

5.3 Footrule distance

For the footrule distance our analysis turns out to be much easier than for the swap distance: for scoring rules and Bucklin, we design a simple matching-based algorithm, and for Copeland and Maximin we can use the fact that the swap distance and the footrule distance are always within a factor of 2 from each other, as this allows us to inherit the hardness results of the previous section.

5.3.1 Scoring rules and Bucklin

The overall structure of our argument is similar to the one in Section 5.2: for any scoring rule $\mathcal{F}_\alpha$ with $\alpha = (\alpha_1, \ldots, \alpha_m)$ we will design a procedure $\mathcal{A}'$ that, given an election $(C, \mathcal{R})$ with $|C| = m$, the preferred candidate p, a target position f for the preferred candidate, and a bound k on the distance, constructs a vote L such that (a) $\mathcal{F}(\mathcal{R}_{-n}, L) = \{p\}$; (b) $r(p, L) = f$; (c) $\mathrm{d}_{\mathrm{fr}}(L, R_n) \leq k$, or returns $\perp$ if no such vote exists. We then run this procedure for $f = 1, \ldots, m$ and return "yes" if at least one of these calls *does not* return $\perp$.

We assume without loss of generality that the manipulator ranks the candidates as $c_1 \succ_n \ldots \succ_n c_m$ (note that this is different from the assumption we made in Section 5.2), and denote by $s_\alpha(c)$ the score of a candidate $c \in C$ in election $(C, \mathcal{R}_{-n})$ under the voting rule $\mathcal{F}_\alpha$. Let r be the rank of p in n's truthful vote, i.e., $p = c_r$.

$\mathcal{A}'$ proceeds by constructing a bipartite graph G with parts $X = C \setminus \{p\}$ and $Y = \{1, \ldots, m\} \setminus \{f\}$; there is an edge from c_j to ℓ if and only if position ℓ is safe for c_j, i.e., $s_\alpha(c_j) + \alpha_\ell < s_\alpha(p) + \alpha_f$, Each edge has a

weight: the weight of the edge (c_j, ℓ) is simply $|j - \ell|$. Clearly, there is a one-to-one correspondence between votes L that rank p in position f and satisfy $\mathcal{F}_\alpha(\mathcal{R}_{-n}, L) = \{p\}$ and perfect matchings in this graph. Furthermore, the cost of a matching M is x if and only if the corresponding vote L_M satisfies $\mathrm{d}_{\mathrm{fr}}(L_M, R_n) = x + |r - f|$. Thus, it suffices to find a minimum cost perfect matching in G; our algorithm returns the vote L that corresponds to this matching if its cost does not exceed $k - |r - f|$ and $\perp$ otherwise. The graph G can be constructed in time $O(m^2 \log(n\alpha_{\max}))$, and a minimum-cost matching can be found in time $O(m^3)$ [11].

We summarize these observations as follows.

Theorem 5.3.1. *For every polynomial-time computable family $\hat{\mathcal{F}} = (\mathcal{F}^m_\alpha)_{m=1,\ldots}$ of scoring rules, the problem $(\mathrm{d}_{\mathrm{fr}}, \hat{\mathcal{F}})$-*OptManipulation *is in* P.

For the Bucklin rule, it suffices to combine the matching-based algorithm given above with the definition of a safe position given in the proof of Theorem 5.2.6. We obtain the following corollary.

Corollary 5.3.2. $(\mathrm{d}_{\mathrm{fr}}, \mathrm{Bucklin})$-OptManipulation *is in* P.

5.3.2 Maximin and Copeland

In Section 5.1 we have mentioned that for any candidate set C and any pair of votes $L, R \in \mathcal{L}(C)$ we have $\mathrm{d}_{\mathrm{swap}}(L, R) \leq \mathrm{d}_{\mathrm{fr}}(L, R) \leq 2\mathrm{d}_{\mathrm{swap}}(L, R)$ [15].

Now, suppose that there exists a ρ-approximation algorithm $\mathcal{A}_{\mathrm{fr}}$ for $(\mathrm{d}_{\mathrm{fr}}, \mathcal{F})$-OptManipulation for some voting rule $\mathcal{F}$. Consider an instance $(C, \mathcal{R}, p)$ of (the optimization version of) this problem, and let

$$\mathcal{L}' = \{L \in \mathcal{L}(C) \mid \mathcal{F}(\mathcal{R}_{-n}, L) = \{p\}\}.$$

If $\mathcal{L}' \neq \emptyset$, let $k = \min\{\mathrm{d}_{\mathrm{fr}}(L, R_n) \mid L \in \mathcal{L}'\}$. On this instance $\mathcal{A}_{\mathrm{fr}}$ outputs a value k' that satisfies $k \leq k' \leq \rho k$; this value corresponds to a vote $L \in \mathcal{L}'$ such that $\mathrm{d}_{\mathrm{fr}}(L, R_n) = k'$.

Now, for any vote $L' \in \mathcal{L}'$ we have

$$\mathrm{d}_{\mathrm{swap}}(L', R_n) \geq \frac{1}{2}\mathrm{d}_{\mathrm{fr}}(L', R_n) \geq \frac{k}{2}.$$

On the other hand, for L we obtain

$$\mathrm{d}_{\mathrm{swap}}(L, R_n) \leq \mathrm{d}_{\mathrm{fr}}(L, R_n) = k' \leq \rho k.$$

Now, consider an algorithm $\mathcal{A}_{\text{swap}}$ for $(\text{d}_{\text{swap}}, \mathcal{F})$-OptManipulation that, given an instance of the problem, runs $\mathcal{A}_{\text{fr}}$ on it and returns the value reported by $\mathcal{A}_{\text{fr}}$. The computation above proves that $\mathcal{A}_{\text{swap}}$ is a 2ρ-approximation algorithm for $(\text{d}_{\text{swap}}, \mathcal{F})$-OptManipulation (note that $\mathcal{A}_{\text{swap}}$ returns $+\infty$ if and only if $\mathcal{L}' = \emptyset$). Combining this observation with Theorems 5.2.7 and 5.2.8, we obtain the following corollaries.

Corollary 5.3.3. *There exists a $\delta > 0$ s. t. $(\text{d}_{\text{fr}}, \text{Maximin})$-OptManipulation does not admit a polynomial $\delta \log |C|$-approximation algorithm unless* P=NP.

Corollary 5.3.4. *There exists a $\delta > 0$ such that for any $\alpha \in \mathbb{Q} \cap [0, 1]$, $(\text{d}_{\text{fr}}, \text{Copeland}^{\alpha})$-OptManipulation does not admit a polynomil-time $\delta \log |C|$-approximation algorithm unless* P=NP.

5.4 Maximum displacement distance

Maximum displacement distance is fairly generous to the manipulator. Indeed, the optimal manipulation problems for swap distance and footrule distance become trivial if the maximum distance k is bounded by a constant: in this case, there are only polynomially many possible manipulative votes, and the manipulator can try all of them. In contrast, for the maximum displacement distance, there are exponentially many votes even at distance 2 from the true vote (to see this, cut the manipulator's vote into segments of length 3; within each segment, the candidates can be shuffled independently). Nevertheless, from the algorithmic perspective maximum displacement distance exhibits the same behavior as swap distance and footrule distance: we can design efficient algorithms for all scoring rules and the Bucklin rule, and derive NP-hardness results for Copeland and Maximin.

5.4.1 Scoring rules and Bucklin

For scoring rules, we can use a simplified variant of the min-cost matching argument given in Section 5.3.1. Again, suppose that we are given a scoring rule $\mathcal{F}_\alpha$ with $\alpha = (\alpha_1, \ldots, \alpha_m)$, an election $(C, \mathcal{R})$ with $|C| = m$, a preferred candidate p and a distance bound k. We assume that the manipulator ranks the candidates as $c_1 \succ_n \ldots \succ_n c_m$. For each $f = 1, \ldots, m$ we try to find a successful manipulative vote L with $\text{d}_{\text{md}}(L, R_n) \leq k$ that ranks p in position

f; in fact, it suffices to consider only values of f that satisfy $|f - r(p, R_n)| \leq k$. For each such f, we construct a bipartite graph G with parts $C \setminus \{p\}$ and $\{1, \ldots, m\} \setminus \{f\}$. In this graph, there is an edge from c_j to ℓ if and only if ℓ is safe for c_j (we use the same definition of a safe position as in Section 5.3.1) and $|\ell - j| \leq k$. In contrast to the construction in Section 5.3.1, the graph is unweighted. It is immediate that there is a one-to-one correspondence between perfect matchings in G and successful manipulative votes at distance at most k from R_n. Thus, we obtain the following result.

Theorem 5.4.1. *For every polynomial-time computable family* $\hat{\mathcal{F}} = (\mathcal{F}_\alpha^m)_{m=1,\ldots,}$ *of scoring rules, the problem* $(\mathrm{d_{md}}, \hat{\mathcal{F}})$-OPTMANIPULATION *is in* P.

For the Bucklin rule, we use the same approach as in Section 5.3, i.e., combine the matching-based algorithm with the definition of a safe position given in the proof of Theorem 5.2.6. This results in the following corollary.

Corollary 5.4.2. ($\mathrm{d_{md}}$,Bucklin)-OPTMANIPULATION *is in* P.

5.4.2 Copeland and Maximin

Theorem 5.4.3. $(\mathrm{d_{md}}, \text{Maximin})$-OPTMANIPULATION *is* NP*-hard.*

Proof. We provide a reduction from SET COVER to $(\mathrm{d_{md}}, \text{Maximin})$-OPTMANIPULATION. Suppose that we are given an instance $(G, \mathcal{S}, k)$ of SET COVER with $G = \{g_1, \ldots, g_t\}$ and $\mathcal{S} = \{S_1, \ldots, S_r\}$ that satisfies conditions (1) $G = \cup_{S \in \mathcal{S}} S$; (2) $t \leq r$.

We will now construct an instance of $(\mathrm{d_{md}}, \text{Maximin})$-OPTMANIPULATION with a set of candidates $C = \{p\} \cup B \cup G \cup X \cup S$, where

$$B = \{b_1, \ldots, b_{t+1}\}, \quad S = \{s_1, \ldots, s_r\},$$

$$X = \{x_{1,1}, \ldots, x_{1,2r}, \ldots, x_{t+1,1}, \ldots, x_{t+1,2r}\}.$$

Corollary 2.3.5 implies that we can construct a preference profile $\mathcal{R}'$ with n' voters, where n' is polynomially bounded in t and r, so that n' is even and:

- For any $c \in C \setminus \{p\}$ exactly $n'/2 - 2$ voters prefer p to c.
- For any $S_j \in \mathcal{S}$ and any $g_\ell \in S_j$ exactly $n'/2 - 2$ voters prefer g_ℓ to s_j.
- For any $x_{j,1}$ and g_j exactly $n'/2 - 2$ voters prefer $x_{j,1}$ to g_j.

- For any $x_{j,i+1}$ and $x_{j,i}$ exactly $n'/2-2$ voters prefer $x_{j,i+1}$ to $x_{j,i}$.
- For any $x_{j,2r}$ and b_{j+1} where $j=1,\ldots,t$ exactly $n'/2-2$ voters prefer $x_{j,2r}$ to b_{j+1}, and exactly $n'/2-2$ voters prefer $x_{t+1,2r}$ to b_1.
- For any other pair of candidates $(c,c') \in B\cup G\cup X \times B\cup G\cup X$, exactly $n'/2$ voters prefer c to c'.
- For any $s_j \in S$ and any $c \in B\cup G\cup X$ exactly $n'/2-4$ voters prefer s_j to c.

Denote the Maximin score of candidate c in election $(C,\mathcal{R}')$ by $s(c)$. We have $s(p)=n'/2-2$, $s(b_j)=n'/2-2$ for any $b_j\in B$, $s(g_j)=n'/2-2$ for any $g_j\in G$ (this follows from condition (1)), $s(x_{i,j})=n'/2-2$ for any $x_{i,j}\in X$, and $s(s_j)=n'/2-4$ for any $s_j\in S$.

We let $n=n'+1$, and set our preference profile to be $\mathcal{R}=(\mathcal{R}',R_n)$, where the manipulator ranks the candidates as

$$p \succ b_1 \succ \ldots \succ b_{t+1} \succ g_1 \succ \ldots g_t \succ$$

$$\succ x_{1,1} \succ \ldots \succ x_{1,2r} \succ \ldots \succ x_{t+1,1} \succ \ldots \succ x_{t+1,2r} \succ s_1 \succ \ldots \succ s_r.$$

We set $K=(t+1)(2r+1)+k-1$. That completes the description of our $(\mathrm{d}_{\mathrm{md}},\text{Maximin})$-OptManipulation instance.

Observe that p's final Maximin score is $n'/2-1$ if and only if the manipulator ranks p first. Further, the final Maximin score of any candidate in S is at most $n'/2-3$.

Here we will use same "parent"-terminology as in Section 3.4.2 (i.e., we say that c_i is a *parent* of c_j whenever c_j obtains exactly $n'/2-2$ points in his pairwise election against c_i.)

It can be easily seen that a candidate in the set $B\cup G\cup X$ has score $n'/2-2$ only if at least one of his parents is ranked above him in the manipulator's vote. Each candidate in the set $B\cup X$ has exactly one parent. Therefore, if p is the winner of the election then for $i=1,\ldots,t$ candidates $\{g_i,x_{i,1},\ldots,x_{i,2r}\}$ are ranked higher than b_{i+1} and candidates $\{x_{t+1,1},\ldots,x_{t+1,2r}\}$ are ranked higher than b_1.

Suppose that $h(G,\mathcal{S})=k$, i.e., there exists a collection of subsets $\mathcal{S}'=\{S_{i_1},\ldots,S_{i_k}\}$ with $i_1<\ldots<i_k$ such that $\cup_{S'\in\mathcal{S}'}S'=G$.

Consider a vote L that ranks p first,

- followed by candidates $x_{t+1,1},\ldots,x_{t+1,2r},b_1$,

- followed by candidates $s_{i_1}, \ldots, s_{i_k}$ (in this order),
- followed by candidates in G and remaining candidates in X (in the order of their appearance in R_n),
- followed by candidates $b_2, \ldots, b_{t+1}$ (in this order),
- followed by the remaining candidates in S (in the order of their appearance in R_n).

By the argument above, p is the unique Maximin winner of $(C, (\mathcal{R}', L))$. It is easy to see that $\mathrm{d}_{\mathrm{md}}(R, L) = (2r+1)(t+1) + k - 1$.

On the other hand, suppose $h(G, \mathcal{S}) > k$ and consider an arbitrary vote L' such that p is the unique Maximin winner of $(C, (\mathcal{R}', L'))$.

Consider a candidate b_ℓ such that $r(b_\ell, L') \geq p(b_i, L')$ for $i = 1, \ldots, t+1$. Consider candidates who are ranked higher than b_ℓ in L'. All candidates in the set B have position in L' which is at least as high as the position of b_ℓ. Therefore, all candidates in the set $X \cup G$ are ranked above the respective elements of B, because p is the only winner of the election. Also p occupies the first place. We assume that $h(G, \mathcal{S}) > k$, so, at least $k+1$ elements of S are ranked above elements of G, because every element of G has to be preceded by at least one of his parents. Therefore, $r(b_\ell, L') \geq 1+(k+1)+|B|+|G|+|X| = 2+k+t+(2r+1)(t+1)$. It is evident that the lowest position that is achievable for b_ℓ in R is $t+2$. Thus, $\mathrm{d}_{\mathrm{md}}(R, L') \geq (2r+1)(t+1) + k$.

Thus, we have constructed an election $(C, \mathcal{R})$ and a positive integer $K = (2r+1)(t+1)+k-1$ so that $(C, \mathcal{R}, K)$ is a "yes"-instance of $(\mathrm{d}_{\mathrm{md}}, \text{Maximin})$-OptManipulation if and only if $(G, \mathcal{S}, k)$ is a "yes"-instance of Set Cover. □

Theorem 5.4.4. $(\mathrm{d}_{\mathrm{md}}, \text{Copeland}^\alpha)$-OptManipulation *is* NP*-hard for any rational* $\alpha \in [0, 1]$.

Proof. We provide a reduction from Set Cover to $(\mathrm{d}_{\mathrm{md}}, \text{Copeland}^\alpha)$-OptManipulation. Suppose that we are given an instance $(G, \mathcal{S}, k)$ of Set Cover with $G = \{g_1, \ldots, g_t\}$ and $\mathcal{S} = \{S_1, \ldots, S_r\}$ that satisfies conditions (1) $G = \cup_{S \in \mathcal{S}} S$; (2) $t \leq r$; (3) t is odd.

We will now construct an instance of $(\mathrm{d}_{\mathrm{md}}, \text{Copeland}^\alpha)$-OptManipulation with a set of candidates $C = \{p\} \cup \{b\} \cup G \cup X \cup S \cup D$, where $X = \{x_1, \ldots, x_{2r+1}\}$, $D = \{d_1, \ldots, d_{11r}\}$ and $S = \{s_1, \ldots, s_r\}$.

It is easy to see that we can construct a digraph that induces the following outcomes of pairwise elections.

- Candidate p is tied with all candidates in $\{b\} \cup G \cup X$, beats $9r - t - 2$ candidates in D, and loses to all other candidates.
- Candidate b beats all candidates in $S \cup D$ and is tied with all other candidates.
- Every candidate $g_i \in G$ is tied with all candidates s_ℓ such that $g_i \in S_\ell$ and beats all other candidates in $X \cup S$ as well as $8r - \frac{t-1}{2} - 1$ candidates in D and $\frac{t-1}{2}$ in G.
- Every candidate in $X \cup S$ beats exactly $7r$ other candidates in $X \cup S \cup D$.
- Every candidate in $d_1, \ldots, d_{\frac{t-1}{2}}$ beats all candidates in $\{p\} \cup G$ and exactly $7r$ other candidates in $X \cup S \cup D$. Every candidate in $d_{\frac{t-1}{2}+1}, \ldots, d_{2r+t+2}$ beats $\{p\}$ and exactly $7r$ other candidates in $X \cup S \cup D$. All other candidates in D beat exactly $7r$ candidates in $X \cup S \cup D$.

The only thing that is not described among the results of pairwise elections is how to obtain the outdegrees $7r$ for the induced subgraph on vertices $X \cup S \cup D$. Evidently, $|X \cup S \cup D| = 14r + 1$ and we can use a construction with summation modulo $14r + 1$ similar to the one used in the proof of Lemma 3.5.1.

By Corollary 2.3.5, we can construct a preference profile $\mathcal{R}'$ with n' voters, where n' is even and polynomially bounded in t and r and the outcomes of pairwise elections between candidates are as described above.

We let $n = n' + 1$ and set our preference profile to be $\mathcal{R} = (\mathcal{R}', R_n)$, where voter n (the manipulator) ranks the candidates as

$$c \succ g_1 \succ \ldots \succ g_t \succ x_1 \succ \ldots \succ x_{2r+1} \succ s_1 \succ \ldots \succ s_r \succ d_1 \succ \ldots \succ d_{11r}.$$

Set $K = 2r+1+t+k$. This completes the description of our $(\mathrm{d}_{\mathrm{md}}, \mathrm{Copeland}^\alpha)$-OptManipulation instance; note that n is odd and therefore the value of α is unimportant for our analysis.

Observe that in $\mathcal{R}$ the Copeland score of p is $12r$, the Copeland score of b is $14r + t + 1$ the Copeland score of each $g_j \in G$ is $12r$, and the Copeland score of each candidate in $X \cup S \cup D$ is at most $7r + t + 1 \leq 9r$. Thus, under truthful voting p is not among the winners at all. For p to be the unique winner, in the manipulator's vote every candidate $g_j \in G$ must be ranked below some candidate s_ℓ such that $g_j \in S_\ell$ or below b and b must be ranked below all candidates in $\{p\} \cup G \cup X$.

Note also that the manipulator's vote can only affect the outcomes of pairwise elections for candidate pairs of the form

- (g_j, s_ℓ), $g_j \in S_\ell$;
- (b, g_i);
- (p, x), where x is any candidate in $\{b\} \cup G \cup X$.

Thus, no matter how the manipulator votes, the Copeland score of every candidate $x \in X \cup S \cup D$ is at most $7r + t \leq 8r$, and hence candidates in $X \cup S \cup D$ are not among the election winners.

So, we can conclude that L is a successful manipulative vote if and only if it ranks each candidate $g_j \in G$ below a candidate representing a set that covers g_j, candidate b below all candidates in $\{p\} \cup G \cup X$ and candidate p higher than all candidates in $\{b\} \cup G \cup X$.

Suppose that $h(G, \mathcal{S}) = k$, i.e., there exists a collection of subsets $\mathcal{S}' = \{S_{i_1}, \ldots, S_{i_k}\}$ with $i_1 < \ldots < i_k$ such that $\cup_{S' \in \mathcal{S}'} S' = G$.

Consider a vote L that ranks p first, followed by candidates $x_1, \ldots, x_{2r+1}$, followed by candidates $s_{i_1}, \ldots, s_{i_k}$ (in this order), followed by candidates in G and followed by the remaining candidates in S and candidates in D (in the order of their appearance in R_n).By the argument above, p is the unique Copeland winner of $(C, (\mathcal{R}', L))$. It is easy to see that $\mathrm{d_{md}}(R, L) = 2r + 1 + t + k$.

On the other hand, suppose $h(G, \mathcal{S}) > k$ and consider an arbitrary vote L' such that p is the unique Copeland winner of $(C, (\mathcal{R}', L'))$. Consider candidates who are ranked higher than b. All candidates in the set $\{p\} \cup X \cup G$ are ranked above candidate b, because p is the only winner of the election. We assume that $h(G, \mathcal{S}) > k$, so, at least $k + 1$ elements of S are ranked above elements of G, because every element of G has to be preceded by at least one of the elements s_ℓ such that $g_j \in S_\ell$. Therefore, $r(b, L') \geq k + 1 + 1 + |G| + |X| + 1 = 4 + k + t + 2r$. In R candidate b occupies position 2. Thus, $\mathrm{d_{md}}(R, L') \geq 2r + 1 + t + k + 1$.

Thus, we have constructed an election $(C, \mathcal{R})$ and a positive integer $K = 2r+1+t+k$ so that $(C, \mathcal{R}, K)$ is a "yes"-instance of $(\mathrm{d_{md}}, \mathrm{Copeland}^\alpha)$-OPTMANIPULATION if and only if $(G, \mathcal{S}, K)$ is a "yes"-instance of SET COVER. □

5.5 Related work

Our work can be placed in the broader context of *mechanism design with verification* [28, 41, 36]. This research area deals with the design of mechanisms (voting rules, auctions, etc.) for selfish agents in settings where agents cannot misrepresent their private information (type) arbitrarily, but rather are restricted to submit a report that is in some way related to their true type. In some settings (most notably, mechanism design for scheduling problems) imposing natural restrictions on possible misreports enables one to circumvent known impossibility results [4, 3]. We remark, however, that the Gibbard–Satterthwaite theorem has been recently shown to be very robust with respect to restrictions on misreporting: every non-dictatorial voting rule for 3 or more candidates remains manipulable even if we only allow the manipulative votes that only differ by a single swap of adjacent candidates from the manipulator's true preference ranking [7]. Viewed from the perspective of mechanism design with partial verification, the hardness results in this thesis provide a complexity-theoretic separation between the unrestricted manipulation problem for Copeland and Maximin and its version with partial verification (where the permissible misreports are required to be within a certain distance from the manipulator's true ranking). To the best of our knowledge, this is a first result of this type in the mechanism design with partial verification literature.

5.5.1 Optimal manipulability and swap bribery

The problem of finding an optimal manipulation with respect to the swap distance can be viewed as a special case of the swap bribery problem [18]. In the swap bribery model, there is an external party that wants to make a particular candidate the election winner. This party can pay the voters to change their preference orders, with a price assigned to swapping each pair of candidates in each vote. The goal is to decide whether the manipulator can achieve his goal given a budget constraint. Clearly, our problem is a special case of swap bribery, where for one voter each swap has unit cost, and for the remaining voters the prices are set to $+\infty$. Swap bribery is known to be hard, even to approximate, for almost all prominent voting rules, including such relatively simple rules as 2-approval. Thus, the easiness results of Section 5.2 identify a new family of easy instances of the swap bribery problem, thus complementing the results of [17, 16, 40]. It would be interesting to see if a

somewhat more general variant of the swap bribery problem for scoring rules, where only one voter can be bribed but swap bribery prices can be arbitrary, remains tractable; it is not clear if the algorithm given in Section 5.2 can be adapted to handle this setting.

On the other hand, one may wonder if the hardness results of Section 5.2 are implied by the existing hardness results for swap bribery. However, this does not seem to be the case: the hardness (and inapproximability) of swap bribery for Copeland and Maximin follows from the hardness results for the possible winner problem [47], and the latter problem is easy if all but one voter's preferences are fixed (it can be verified that the algorithm of Bartholdi et al. ([6]) works even if the positions of some candidates in the vote are already fixed). Thus, the hardness results for Copeland and Maximin given in Section 5.2 strengthen the existing hardness results for swap bribery with respect to these rules.

5.6 Summary

We have considered the problem of finding a successful manipulative vote that differs from the manipulators' preferences as little as possible, for three distance measures on votes and four types of voting rules. Our results are summarized in Table 5.1 (where "NPC" stands for "NP-complete" and "$\Omega(\log m)$-inapp." stands for "inapproximable up to a factor of $\Omega(\log m)$").

	Sc. rules	Bucklin	Copeland	Maximin
d_{swap}	P	P	$\Omega(\log m)$-inapp.	$\Omega(\log m)$-inapp.
d_{fr}	P	P	$\Omega(\log m)$-inapp.	$\Omega(\log m)$-inapp.
d_{md}	P	P	NPC	NPC
Single-voter manip.	P	P	P	P

Table 5.1: Summary of results and comparing with single-voter manipulation

Chapter 6

Future Work

6.1 Tie-breaking rules

We have determined the complexity of finding an optimal manipulation under the randomized tie-breaking rule for several prominent voting rules, namely, scoring rules, Maximin, Copeland$^\alpha$ for any rational $\alpha \in [0, 1]$, two variants of the Bucklin rule, Plurality with Runoff, STV and Ranked Pairs. This provides an essentially complete picture of the complexity of RandManipulation for commonly studied voting rules.

Still there is a number of open questions left by our work. For instance, it would be interesting to see whether the easiness results for coalitional manipulation under lexicographic tie-breaking proven by [50, 49] extend to randomized tie-breaking, or whether our algorithmic results hold under a more general definition of randomized tie-breaking, where different candidates may be selected with different probabilities; the latter question includes, in particular, the setting considered in the end of Section 3.6. Another promising research direction is designing approximation algorithms for the optimization version of RandManipulation. We conjecture that for Copeland it can be shown that this problem does not admit a constant-factor approximation algorithm, it is not the clear if this is the case for Maximin, STV, or Ranked Pairs.

Other interesting questions include identifying natural tie-breaking rules that make manipulation hard and extending our results to multi-winner elections.

6.2 Minimizing the distance to the true preferences

We have considered the problem of finding a successful manipulative vote that differs from the manipulator's preferences as little as possible, for three distance measures on votes and four types of voting rules.

A natural direction for future work is extending our results to other distances on votes; for instance, it should not be too hard to generalize our results to weighted variants of swap and footrule distances; such distances play an important role in several applications of rank aggregation, and have received considerable attention in the literature (see [31] and references therein). At a more technical level, we remark that for maximum displacement distance we only have NP-hardness results for Copeland and Maximin; it would be interesting to see if this variant of our problem admits efficient approximation algorithms.

Bibliography

[1] K. Arrow, A. Sen, and K. Suzumura. *Handbook of Social Choice and Welfare*. North Holland, 2002. (1)

[2] K. J. Arrow. A difficulty in the concept of social welfare. *Journal of Political Economy*, 58. (1)

[3] V. Auletta, R. D. Prisco, P. Penna, and G. Persiano. The power of verification for one-parameter agents. *Journal of Computer and System Sciences*, 75(3):190–211, 2009. (82)

[4] V. Auletta, R. D. Prisco, P. Penna, G. Persiano, and C. Ventre. New constructions of mechanisms with verification. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP'06)*, pages 596–607, 2006. (82)

[5] J. J. Bartholdi, III and J. B. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991. (3, 13, 14, 43)

[6] J. J. Bartholdi, III, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989. (1, 3, 4, 5, 6, 11, 12, 13, 14, 34, 47, 83)

[7] I. Caragiannis, E. Elkind, M. Szegedy, and L. Yu. Mechanism design: from partial to probabilistic verification. In *Proceedings of the 13th ACM Conference on Electronic Commerce (ACM EC'12)*. (82)

[8] V. Conitzer. Making decisions based on the preferences of multiple agents. *Communications of the ACM*, 53:84–94, 2010. (1)

[9] V. Conitzer, M. Rognlie, and L. Xia. Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 109–115, 2009. (42, 44)

[10] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*. (59, 60)

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2001. (75)

[12] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. An empirical study of Borda manipulation. In *Proceedings of the 3rd International Workshop on Computational Social Choice (COMSOC'10)*, pages 91–102, 2010. (3, 49)

[13] B. Debord. Caractérisation des matrices des préférences nettes et méthodes d'agrégation associées. *Mathématiques et Sciences Humaines*, 97:5–17, 1987. (16)

[14] Y. Desmedt and E. Elkind. Equilibria of plurality voting with abstentions. In *Proceedings of the 11th ACM Conference on Electronic Commerce (ACM EC'10)*, pages 347–356, 2010. (4, 44)

[15] P. Diakonis and R. Graham. Spearman footrule as a measure of disarray. *Journal of the Royal Statistical Society B (Methodological)*, 39(2):262–268, 1977. (62, 75)

[16] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. In *Proceedings of the 5th International Symposium on Parameterized and Exact Computation (IPEC'10)*, pages 107–122, 2010. (6, 82)

[17] E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *Proceedings of the 6th Workshop on Internet and Network Economics (WINE'10)*, pages 473–482, 2010. (6, 82)

[18] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *Proceedings of 3rd International Symposium on. Algorithmic Game Theory (SAGT'09)*, pages 299–310, 2009. (6, 82)

[19] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Proceedings of the 16th Annual International Symposium on Algorithms and Computation (ISAAC'05)*, pages 206–215, 2005. (59, 60)

[20] E. Ephrati and J. Rosenschein. A heuristic technique for multi-agent planning. *Annals of Mathematics and Artificial Intelligence*, 20(1–4):13–67, 1997. (3)

[21] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Using complexity to protect elections. *Communications of the ACM*, 53:74–82, 2010. (3)

[22] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: Ties matter. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, pages 983–990, 2008. (40)

[23] P. Faliszewski and A. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31:53–64, 2010. (3)

[24] E. Friedgut, G. Kalai, and N. Nisan. Elections can be manipulated often. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS'08)*, pages 243–249, 2008. (3)

[25] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979. (31, 40, 69)

[26] A. F. Gibbard. Manipulation of voting schemes. *Econometrica*, 41(4):587–601, 1973. (2, 12, 17)

[27] A. F. Gibbard. Manipulation of schemes that mix voting and chance. *Econometrica*, 45:665–681, 1977. (4, 44)

[28] J. Green and J.-J. Laffont. Partially verifiable information and mechanism design. *Review of Economic Studies*, 53:447–456, 1986. (82)

[29] M. Isaksson, G. Kindler, and E. Mossel. The geometry of manipulation—a quantitative proof of the Gibbard–Satterthwaite theorem. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 319–328, 2010. (3)

[30] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938. (6)

[31] R. Kumar and S. Vassilvitskii. Generalized distances between rankings. In *Proceedings of the 10th International World Wide Web Conference (WWW'10)*, pages 571–580, 2010. (86)

[32] C. List and C. Puppe. Judgment aggregation: A survey. *Oxford Handbook of Rational and Social Choice*, pages 84–94, 2009. (1)

[33] D. McGarvey. A theorem on the construction of voting paradoxes. *Econometrica*, 21(4):608–610, 1953. (15)

[34] R. Meir, M. Polukarov, J. S. Rosenschein, and N. R. Jennings. Convergence to equilibria in plurality voting. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI'10)*, pages 823–828, 2010. (44)

[35] J. W. Moon. *Topics on Tournaments*. Holt, Rinehart and Winston, 1968. (15, 16)

[36] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001. (82)

[37] A. Procaccia and J. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of AI Research*, 28:157–181, 2007. (3)

[38] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC'97)*, pages 475–484, 1997. (70, 72)

[39] 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. (2, 12, 17)

[40] I. Schlotter, P. Faliszewski, and E. Elkind. Campaign management under approval-driven voting rules. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI'11)*, pages 726–731, 2011. (6, 82)

[41] N. Singh and D. Wittman. Implementation with partial verification. *Review of Economic Design*, 6:63–84, 2001. (82)

[42] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. (6)

[43] T. N. Tideman. Independence of clones as a criterion for voting rules. *Social Choice and Welfare*, 4:185–206, 1987. (11)

[44] T. Walsh. Where are the really hard manipulation problems? The phase transition in manipulating the veto rule. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 324–329, 2009. (3)

[45] D. West. *Introduction to Graph Theory*. Prentice Hall, 2001. (50)

[46] L. Xia and V. Conitzer. A sufficient condition for voting rules to be frequently manipulable. In *Proceedings of the 9th ACM Conference on Electronic Commerce (ACM EC'08)*, pages 99–108, 2008. (3)

[47] L. Xia and V. Conitzer. Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research*, 41:25–67, 2011. (83)

[48] L. Xia, V. Conitzer, and A. Procaccia. A scheduling approach to coalitional manipulation. In *Proceedings of the 11th ACM Conference on Electronic Commerce (ACM EC'10)*, pages 275–284, 2010. (3)

[49] L. Xia, M. Zuckerman, A. Procaccia, V. Conitzer, and J. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 348–352, 2009. (3, 14, 43, 85)

[50] M. Zuckerman, A. Procaccia, and J. Rosenschein. Algorithms for the coalitional manipulation problem. *Artificial Intelligence Journal*, 173:392–412, 2009. (85)

[51] M. Zuckerman and J. S. Rosenschein. Manipulation with randomized tie-breaking under maximin (extended abstract). In *The 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*. (44)