Optimal Manipulation of Voting Rules

Svetlana Obraztsova
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
SVET0001@ntu.edu.sg

Edith Elkind
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
eelkind@ntu.edu.sg

ABSTRACT

Complexity of voting manipulation is a prominent research topic in computational social choice. In this paper, we study 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 poly-time algorithms for all scoring rules and Bucklin and hardness results for Copeland and Maximin.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms, Theory

Keywords

voting, manipulation, swap distance, footrule distance

1. INTRODUCTION

Mechanisms for aggregating the preferences of heterogeneous agents play an important role in the design of multi-agent systems [9]. Such mechanisms are typically implemented by *voting rules*, i.e., mappings that, given the rankings of the available alternatives by all agents, output an alternative that best reflects the collective opinion. There are many different voting rules that are used for group decision making; see, e.g., [3] for an overview.

A weakness shared by all reasonable voting rules is their susceptibility to *manipulation*: for any voting rule over a set of alternatives C, $|C| \geq 3$, that is not a dictatorship, there are voting situations where some voter would be better off if, instead of submitting her true ranking of the alternatives, she submitted a vote that did not quite match her true preferences. This was observed by Gibbard [11] and, independently, by Satterthwaite [18] more than 30 years ago, and a lot of research effort since then has been spent on identifying voting rules that are at least somewhat resistant to manipulation.

In their pioneering paper [2], Bartholdi, Tovey and Trick proposed to use computational complexity as a roadblock in the way of manipulative behavior: they observed that, in practice, the manipulator needs an efficient method to find a successful manipulative vote, and a voting rule that does not admit such a method may be viewed as being relatively less vulnerable to manipulation. However, most classic voting rules, with the notable exception of STV, turn out to be susceptible to manipulation in this sense [2, 1].

In this paper, 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* [12] (also known as bubble-sort distance, Kendall distance, etc.) and the *footrule distance* [20] (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 [8]; however, our question is not addressed by existing complexity results for swap bribery [8, 7, 6, 19] (see

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

Section 7 for a discussion). 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. [2] 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.

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 Section 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 provide a fairly complete picture of the complexity of finding an optimal manipulative vote for the three distances and four types of voting rules that we consider. Interestingly, they indicate that scoring rules (and the Bucklin rule, which is closely related to a subfamily of scoring rules known as k-approval) are fundamentally easier to manipulate than Copeland and Maximin; we remark that this observation is also suggested by the recent work of Obraztsova et al. [16, 15] on the complexity of manipulation under randomized tie-breaking. Thus, we believe that, besides being interesting for its own sake, our work contributes to the broad agenda of understanding the intrinsic complexity—and, therefore, practical applicability—of various voting rules.

2. PRELIMINARIES

An *election* is given by a set of candidates $C = \{c_1, \dots, c_m\}$ and a vector $\mathcal{R} = (R_1, \dots, R_n)$, where each R_i, $i = 1, \dots, n$, is a linear order over C; R_i is called the *preference order* (or, *vote*) of voter i. For readability, we will sometimes write $\succ_i$ in place of R_i. If $a \succ_i b$ 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: $r(c_j, R_i) = |\{c \in C \mid c \succ_i c_j\}| + 1$. We denote the space of all linear orders over C by $\mathcal{L}(C)$. We denote by $(\mathcal{R}_{-i}, L)$ the preference profile obtained from $\mathcal{R}$ by replacing R_i with L.

A *voting correspondence* $\mathcal{F}$ is a mapping that, given a candidate set C and a preference profile $\mathcal{R}$ over C outputs a non-empty subset of candidates $S \subseteq C$; we write $S = \mathcal{F}(\mathcal{R})$. The candidates in S are called the *winners* of election $(C, \mathcal{R})$. A *voting correspondence* $\mathcal{F}$ is said to be a *voting rule* if it always produces a unique winner, i.e., $|\mathcal{F}(\mathcal{R})| = 1$ for any profile $\mathcal{R}$.

A voting correspondence can be transformed into a voting rule with the help of a *tie-breaking rule*. 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 *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 $a \in S \setminus \{c\}$. In the context of single-voter manipulation problems, where there is one voter that considers lying about his vote to obtain a better outcome, of particular interest are *benevolent* and *adversarial* tie-breaking rules: the former breaks ties in the manipulator's favor while the latter breaks ties against the manipulator's wishes (i.e., tie-breaking is lexicographic with respect to, respectively, the manipulator's true preference ordering and its inverse). In the traditional computational social choice terminology benevolent and adversarial tie-breaking correspond to, respectively, non-unique and unique winner settings.

Voting rules We will now describe the voting correspondences considered in this paper. All these correspondences assign scores to candidates; the winners are the candidates with the highest scores. In what follows, we will assume that these correspondences are transformed into voting rules by breaking ties adversarially; however, all of our results can be adapted in a straightforward manner to benevolent or, more generally, lexicographic tie-breaking.

Scoring rules Any vector $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{R}^m$ such that $\alpha_1 \geq \cdots \geq \alpha_m$ defines a *scoring rule* $\mathcal{F}_\alpha$ as follows. Each voter grants α_i points to the candidate she ranks in the i-th position; the score of a candidate is the sum of the scores he receives from all voters. The vector α is called a *scoring vector*; we assume without loss of generality that the coordinates of α are nonnegative integers given in binary. We remark that scoring rules are defined for a fixed number of candidates, and therefore do not quite fit our definition of a voting rule. Thus, one needs to consider *families* of scoring rules (one for every possible number of candidates). From the algorithmic perspective, it is natural to restrict our attention to *polynomial-time computable families of scoring rules*, where the scoring vector α^m for an m-candidate election can be computed in time $\mathrm{poly}(m)$. Two well-known examples of such families are *Borda*, given by $\alpha = (m-1, \dots, 1, 0)$, and *$k$-approval*, given by $\alpha_i = 1$ if $i \leq k$, $\alpha_i = 0$ if $i > k$.

Bucklin Given an n-voter election, the *Bucklin winning round* is the smallest value of r such that the r-approval score of at least one candidate exceeds $n/2$. The *Bucklin score* of a candidate $c \in C$ is his r-approval score, where r is the Bucklin winning round.

Copeland A candidate a is said to win 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. Under the Copeland$^\alpha$ rule, $\alpha \in \mathbb{Q} \cap [0, 1]$, each candidate gets 1 point for each pairwise election he wins and α points for each pairwise election he ties.

Maximin The *Maximin score* of a candidate $c \in C$ is given by the number of votes c gets in his worst pairwise election, i.e., $\min_{d \in C \setminus \{c\}} |\{i \mid c \succ_i d\}|$.

Distances 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) nonnegativity: $d(x, y) \geq 0$; (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 paper, 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 $\{c_1, \dots, c_m\}$. Specifically, we will consider three such families (in the following definitions, $C = \{c_1, \dots, c_m\}$ 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. 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 [5] 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 $L,R\in\mathcal{L}(C)$.

3. OUR MODEL

We will now formally describe our computational problem.

DEFINITION 3.1. *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 voter $i\in\{1,\ldots,n\}$, 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 $\mathcal{F}(C,(\mathcal{R}_{-i},L))=\{p\}$ and $d^m(R_i,L)\leq k$, and a "no"-instance otherwise.*

REMARK 3.2. The problem $(\mathcal{D},\mathcal{F})$-OPTMANIPULATION is in NP as long as all distances in $\mathcal{D}$ and the rule $\mathcal{F}$ are poly-time computable: one can guess a vote L and check that $\mathcal{F}(C,(\mathcal{R}_{-i},L))=\{p\}$ and $d^m(R_i,L)\leq k$. In particular, it is in NP for all distance families and voting rules considered in this paper.

REMARK 3.3. 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 $(C,\mathcal{R})$, a voter i 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_i that satisfies $\mathcal{F}(C,(\mathcal{R}_{-i},L))=\{p\}$ (k is assumed to be $+\infty$ if there is no vote L with $\mathcal{F}(C,(\mathcal{R}_{-i},L))=\{p\}$). 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 approximation results) for Copeland and Maximin under swap distance (Sections 4) and footrule distance (Section 5).

REMARK 3.4. 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 $d^m(R_i,L)\leq k$ and $\mathcal{F}(C,(\mathcal{R}_{-i},L))\succ_i\mathcal{F}(C,\mathcal{R})$; we will refer to this problem as OPTMANIPULATION$'$. Clearly, an efficient algorithm for OPTMANIPULATION can be used to solve OPTMANIPULATION$'$, 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$'$. 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$'$: 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 7).

4. SWAP DISTANCE

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

4.1 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 manipulator's preferred candidate p, we know his final score. 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 voter i, a candidate p, and a position f in i's vote, finds an optimal manipulation for i among all votes that rank p in position f. More formally, let

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

our subroutine outputs $\perp$ if $\mathcal{L}^f(\alpha)$ is empty and a vote $\hat{L}$ such that $\mathrm{d}_{\mathrm{swap}}(\hat{L},R_i)\leq\mathrm{d}_{\mathrm{swap}}(L,R_i)$ 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 $\mathrm{d}_{\mathrm{swap}}(\hat{L},R_i)\leq k$. Thus the running time of our algorithm is m times the running time of $\mathcal{A}$. It remains to describe $\mathcal{A}$.

THEOREM 4.1. *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 voter $i\in\{1,\ldots,n\}$, 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 $\mathrm{d}_{\mathrm{swap}}(\hat{L},R_i)\leq\mathrm{d}_{\mathrm{swap}}(L,R_i)$ for all $L\in\mathcal{L}^f(\alpha)$ otherwise, and runs in time $O(m^2\log(n\alpha_1))$.*

PROOF. For convenience, let us renumber the candidates in C so that $c_m=p$ and $c_1\succ_i\ldots\succ_i c_{m-1}$. 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 i'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 4.2. *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 4.3. *If the subroutine $\mathcal{A}(C, \mathcal{R}, i, p, f)$ 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 $\mathcal{F}_\alpha(C, (\mathcal{R}_{-i}, \hat{L})) = \{p\}$, and hence $\hat{L} \in \mathcal{L}^f(\alpha)$.

Now, suppose that $\mathcal{A}(C, \mathcal{R}, i, p, f) = \perp$. 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 4.2, 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 4.4. *If $\mathcal{A}(C, \mathcal{R}, i, p, f) = \hat{L}$, then $\mathrm{d}_{\mathrm{swap}}(\hat{L}, R_i) \leq \mathrm{d}_{\mathrm{swap}}(L, R_i)$ 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 $\mathrm{d}_{\mathrm{swap}}(L, R_i) \leq \mathrm{d}_{\mathrm{swap}}(L', R_i)$ 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 4.2 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 $\mathrm{d}_{\mathrm{swap}}(L', R_i) < \mathrm{d}_{\mathrm{swap}}(L, R_i)$, 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 $\mathrm{d}_{\mathrm{swap}}(L', R_i) < \mathrm{d}_{\mathrm{swap}}(L, R_i)$. 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 $r(c, R_i) < r(c', R_i)$, but $r(c, R) > r(c', R)$. Clearly, the swap distance from R to R_i 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:

- $c_\ell \succ_i c \succ_i c_j$. In this case, both (c_j, c) and (c, c_ℓ) are inversions in L, but neither of them is an inversion in L'.
- $c_\ell \succ_i c_j \succ_i c$. 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.
- $c \succ_i c_\ell \succ_i c_j$. 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 $\mathrm{d}_{\mathrm{swap}}(L', R_i) < \mathrm{d}_{\mathrm{swap}}(L, R_i)$, 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 4.3 and 4.4. □

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

COROLLARY 4.5. *For every polynomial-time computable family $\hat{\mathcal{F}} = (\mathcal{F}_\alpha^m)_{m=1,\ldots}$ 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 4.6. $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Bucklin})$-OPTMANIPULATION *is in* P.

PROOF SKETCH. Consider an election $(C, \mathcal{R})$ and a manipulator i. Just as in the proof of Theorem 4.1, 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 $(C, (\mathcal{R}_{-i}, L_f))$ 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 $(C, \mathcal{R}_{-i})$, and let s be p's r^*-approval score in $(C, (\mathcal{R}_{-i}, L_f))$; 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 $C_1 = \{c \in C \setminus \{p\} \mid s_{r^*}(c) = s - 1\}$, $C_2 = \{c \in C \setminus (C_1 \cup \{p\}) \mid s_r(c) = \lfloor \frac{n}{2} \rfloor$ 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 (a) $r(c, L) > r^*$ for all $c \in C_1$ and (b) $r(c, L) \geq r^*$ for all $c \in C_2$. We will say that a position j is *safe* for a candidate $c \in C \setminus \{p\}$ if (1) $c \notin C_1 \cup C_2$ or (2) $c \in C_1$ and $j > r^*$ or (3) $c \in C_2$ and $j \geq r^*$. The argument above shows that p is the unique Bucklin winner in $(C, (\mathcal{R}_{-i}, L))$ 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 4.1; 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). □

4.2 Maximin and Copeland

For both Maximin and Copeland, finding an optimal manipulation with respect to the swap distance turns out to be computationally hard. In fact, we will prove that the optimization versions of these problems (see Remark 3.3) 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 3.2, NP-complete).

We provide reductions from the optimization version of the SET COVER problem [10]. 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 [17]. 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 [17]. Thus, in what follows, we only consider instances of SET COVER that satisfy conditions (1)–(3).

THEOREM 4.7. *There exists a $\delta > 0$ s. t.* $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-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 $S = \{s_1, \ldots, s_r\}$.

The proof of McGarvey theorem [14] 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 (the manipulator) 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 $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Maximin})$-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 with $h(G, \mathcal{S}) = h$.

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 $h = h(G, \mathcal{S})$. Then 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), $r \leq t^K$ (where K is a constant whose value can be extracted from the proof in [17]), 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 $\delta' > 0$, then there exists a polynomial-time $(2\delta'\gamma \log t)$-approximation algorithm for SET COVER. By [17], for small enough δ this implies P=NP. □

The argument for Copeland is similar.

THEOREM 4.8. *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 SKETCH. 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 $S = \{s_1, \ldots, s_r\}$. There exists a tournament over the candidate set C such that that:

- p beats all candidates in $G \cup S$ as well as $6r - (t + 1)/2$ 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 $X \cup S$.

Thus, by McGarvey theorem [14], we can construct a preference profile $\mathcal{R}'$ with n' voters that generates this tournament; moreover, we can assume that n' is even and polynomially bounded in t and r, We let $n = n' + 1$, $i = n$ and set our preference profile to be $\mathcal{R} = (\mathcal{R}', R_n)$, where voter n (the manipulator) ranks the candidates as

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

This completes the description of our $(\mathrm{d}_{\mathrm{swap}}, \mathrm{Copeland}^\alpha)$-OPTMANIPULATION instance; note that n is even 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 4.7, and, from this point on, the proof repeats the proof of Theorem 4.7 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 4.7). □

5. 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.

5.1 Scoring Rules and Bucklin

The overall structure of our argument is similar to the one in Section 4: 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 $|C| = m$, a voter i, 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) $\mathcal{F}(C, (\mathcal{R}_{-i}, L)) = \{p\}$; (b) $r(p, L) = f$; (c) $\mathrm{d}_{\mathrm{fr}}(L, R_i) \leq k$, 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 $c_1 \succ_i \ldots \succ_i c_m$ (note that this is different from the assumption we made in Section 4), and denote by $s_\alpha(c)$ the score of a candidate $c \in C$ in election $(C, \mathcal{R}_{-i})$ under the voting rule $\mathcal{F}_\alpha$. Let r be the rank of p in i'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 $\mathcal{F}_\alpha(C, (\mathcal{R}_i, L)) = \{p\}$ 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 $\mathrm{d_{fr}}(L_M, R_i) = x + |r - f|$. 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 $O(m^2 \log(n\alpha_1))$, and a minimum-cost matching can be found in time $O(m^3)$ [4].

We summarize these observations as follows.

THEOREM 5.1. *For every polynomial-time computable family $\hat{\mathcal{F}} = (\mathcal{F}_\alpha^m)_{m=1,\ldots}$ of scoring rules, the problem* $(\mathrm{d_{fr}}, \hat{\mathcal{F}})$-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 4.6. We obtain the following corollary.

COROLLARY 5.2. $(\mathrm{d_{fr}}, \mathrm{Bucklin})$-OPTMANIPULATION *is in* P.

5.2 Maximin and Copeland

In Section 2 we have mentioned that for any candidate set C and any pair of votes $L, R \in \mathcal{L}(C)$ we have $\mathrm{d_{swap}}(L, R) \leq \mathrm{d_{fr}}(L, R) \leq 2\mathrm{d_{swap}}(L, R)$ [5].

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

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

If $\mathcal{L}' \neq \emptyset$, let $k = \min\{\mathrm{d_{fr}}(L, R_i) \mid L \in \mathcal{L}'\}$. 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 $\mathrm{d_{fr}}(L, R_i) = k'$.

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

$$\mathrm{d_{swap}}(L', R_i) \geq \frac{1}{2}\mathrm{d_{fr}}(L', R_i) \geq \frac{k}{2}.$$

On the other hand, for L we obtain

$$\mathrm{d_{swap}}(L, R_i) \leq \mathrm{d_{fr}}(L, R_i) = k' \leq \rho k.$$

Now, consider an algorithm $\mathcal{A}_{\mathrm{swap}}$ for $(\mathrm{d_{swap}}, \mathcal{F})$-OPTMANIPULATION that, given an instance of the problem, runs $\mathcal{A}_{\mathrm{fr}}$ on it and returns the value reported by $\mathcal{A}_{\mathrm{fr}}$. The computation above proves that $\mathcal{A}_{\mathrm{swap}}$ is a 2ρ-approximation algorithm for $(\mathrm{d_{swap}}, \mathcal{F})$-OPTMANIPULATION (note that $\mathcal{A}_{\mathrm{swap}}$ returns $+\infty$ if and only if $\mathcal{L}' = \emptyset$). Combining this observation with Theorems 4.7 and 4.8, we obtain the following corollaries.

COROLLARY 5.3. *There exists a $\delta > 0$ s. t.* $(\mathrm{d_{fr}}, \mathrm{Maximin})$-OPTMANIPULATION *does not admit a poly-time $\delta \log |C|$-approximation algorithm unless* P=NP.

COROLLARY 5.4. *There exists a $\delta > 0$ such that for any $\alpha \in \mathbb{Q} \cap [0, 1]$,* $(\mathrm{d_{fr}}, \mathrm{Copeland}^\alpha)$-OPTMANIPULATION *does not admit a poly-time $\delta \log |C|$-approximation algorithm unless* P=NP.

6. MAX 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 essentially 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.

6.1 Scoring Rules and Bucklin

For scoring rules, we can use a simplified variant of the min-cost matching argument given in Section 5.1. 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 manipulator i, a preferred candidate p and a distance bound k. We assume that the manipulator ranks the candidates as $c_1 \succ_i \ldots \succ_i c_m$. For each $f = 1, \ldots, m$ we try to find a successful manipulative vote L with $\mathrm{d_{md}}(L, R_i) \leq k$ that ranks p in position f; in fact, it suffices to consider only values of f that satisfy $|f - r(p, R_i)| \leq k$. 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 5.1) and $|\ell - j| \leq k$. In contrast to the construction in Section 5.1, 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_i. Thus, we obtain the following result.

THEOREM 6.1. *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, i.e., combine the matching-based algorithm with the definition of a safe position given in the proof of Theorem 4.6. This results in following corollary.

COROLLARY 6.2. $(\mathrm{d_{md}}$,Bucklin$)$-OPTMANIPULATION *is in* P.

6.2 Maximin and Copeland

For Maximin and Copeland, finding an optimal manipulation with respect to the maximum displacement distance is computationally hard; however, in contrast with our results in Sections 4 and 5, we are only able to show the NP-hardness of the decision version of this problem (rather than inapproximability of its optimization version). We omit the proofs of the following two theorems due to space constraints; both proofs are based on (somewhat involved) reductions from SET COVER.

THEOREM 6.3. $(\mathrm{d_{md}}, \mathrm{Maximin})$-OPTMANIPULATION *is* NP*-complete.*

THEOREM 6.4. *For any $\alpha \in \mathbb{Q} \cap [0, 1]$,* $(\mathrm{d_{md}}, \mathrm{Copeland}^\alpha)$-OPTMANIPULATION *is* NP*-complete.*

7. 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 [8]. 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 4 identify a new family of easy instances of the swap bribery problem, thus complementing the results of [7, 6, 19]. 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 4 can be adapted to handle this setting.

On the other hand, one may wonder if the hardness results of Section 4 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 [21], 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. [2] 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 4 strengthen the existing hardness results for swap bribery with respect to these rules.

8. CONCLUSIONS AND FUTURE WORK

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 1 (where "NPC" stands for "NP-complete" and "$(\log m)$-inapp." stands for "inapproximable up to a factor of $\Omega(\log m)$").

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 for 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 [13] 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.

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

Table 1: Summary of results

Acknowledgments This research was supported by National Research Foundation (Singapore) under grant 2009-08 (Edith Elkind) and by Russian Foundation for Basic Research grant 11-01-12135 ofi-m (Svetlana Obraztsova). We would like to thank the AAMAS reviewers for their very useful feedback.

9. REFERENCES

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

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

[3] S. Brams and P. Fishburn. Voting procedures. In K. Arrow, A. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare, Volume 1*, pages 173–236. Elsevier, 2002.

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

[5] 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.

[6] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. In *IPEC'10*, pages 107–122, 2010.

[7] E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *WINE'10*, pages 473–482, 2010.

[8] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *SAGT'09*, pages 299–310, 2009.

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

[10] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[11] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41(4):587–601, 1973.

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

[13] R. Kumar and S. Vassilvitskii. Generalized distances between rankings. In *WWW'10*, pages 571–580, 2010.

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

[15] S. Obraztsova and E. Elkind. On the complexity of voting manipulation under randomized tie-breaking. In *IJCAI'11*, pages 319–324, 2011.

[16] S. Obraztsova, E. Elkind, and N. Hazon. Ties matter: Complexity of voting manipulation revisited. In *AAMAS'11*, pages 71–79, 2011.

[17] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC'97*, pages 475–484, 1997.

[18] 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.

[19] I. Schlotter, P. Faliszewski, and E. Elkind. Campaign management under approval-driven voting rules. In *AAAI'11*, pages 726–731, 2011.

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

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