Ties Matter: Complexity of Voting Manipulation Revisited

Svetlana Obraztsova **Edith Elkind**
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore

Noam Hazon
Robotics Institute
Carnegie Mellon University, USA

Abstract

In their groundbreaking paper, Bartholdi, Tovey and Trick [1989] argued that many well-known voting rules, such as Plurality, Borda, Copeland and Maximin are easy to manipulate. An important assumption made in that paper is that the manipulator's goal is to ensure that his preferred candidate is among the candidates with the maximum score, or, equivalently, that ties are broken in favor of the manipulator's preferred candidate. In this paper, we examine the role of this assumption in the easiness results of [Bartholdi *et al.*, 1989]. We observe that the algorithm presented in [Bartholdi *et al.*, 1989] extends to all rules that break ties according to a fixed ordering over the candidates. We then show that all scoring rules 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 [Bartholdi *et al.*, 1989]. In contrast, we show that manipulation becomes hard when arbitrary polynomial-time tie-breaking rules are allowed, both for the rules considered in [Bartholdi *et al.*, 1989], and for a large class of scoring rules.

1 Introduction

Computational social choice is an actively growing subarea of multiagent systems that provides theoretical foundations for preference aggregation and collective decision-making in multiagent domains. One of the most influential early contributions to this area is the paper by Bartholdi, Tovey, and Trick entitled "The computational difficulty of manipulating an election" [Bartholdi *et al.*, 1989]. 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 [1991] 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, [Xia *et al.*, 2009; 2010] and the recent survey [Faliszewski and Procaccia, 2010] for the former, and [Walsh, 2009; Davies *et al.*, 2010] for the latter). While it has been argued that worst-case complexity does not provide adequate protection against malicious behavior (see, e.g. [Procaccia and Rosenschein, 2007; Xia and Conitzer, 2008; Friedgut *et al.*, 2008; Isaksson *et al.*, 2010]), 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.

An important property of the voting rules discussed in [Bartholdi *et al.*, 1989] is that they may produce multiple winners, i.e., they are, in fact, voting correspondences (see Section 2 for the formal definitions). It is not immediately clear what it means for manipulation to be successful in such a case. [Bartholdi *et al.*, 1989] take a rather liberal approach: they define a manipulation to be successful if, as a result, the manipulator's preferred candidate is *one of the election winners*. This approach is equivalent to assuming that ties are broken in favor of the manipulator. Now, a careful examination of the algorithm in [Bartholdi *et al.*, 1989] shows that it works as long as ties are broken either adversarially to the manipulator or according to an arbitrary fixed lexicographic order over the candidates. However, in real-life settings, when an election ends in a tie, it is not uncommon to choose the winner using a tie-breaking rule that is non-lexicographic in nature. Indeed, perhaps the most common approach is to toss a coin, i.e., select the winner uniformly at random among all tied alternatives. A more sophisticated example is provided by the second-order Copeland rule studied in [Bartholdi *et al.*, 1989], which is effectively the Copeland rule combined with a rather involved tie-breaking method; despite its apparent complexity, the second-order Copeland is the voting rule of choice for several organizations [Bartholdi *et al.*, 1989]. Thus, it is natural to ask under what conditions on the tie-breaking rule the voting correspondences considered in [Bartholdi *et al.*, 1989] remain easy to manipulate.

In this paper, we make two contributions towards answer-

ing this question. We first consider the randomized tie-breaking rule, which chooses the winner uniformly at random among all tied candidates. Now, to formalize the notion of a successful manipulation under this rule, we need additional information about the manipulator's preferences: knowing the manipulator's preference order is insufficient for determining whether he prefers a tie between his top candidate and his least favorite candidate to his second choice becoming the unique winner. Thus, following [Desmedt and Elkind, 2010], we endow the manipulator with utilities for all candidates, and seek a manipulation that maximizes his expected utility, where the expectation is taken over the random bits used to select the winner. We demonstrate that for all scoring rules such a manipulation can be found in polynomial time. This is also true for Maximin as long as the manipulator's utility function has a special form that is inspired by the notion of manipulation employed in [Bartholdi *et al.*, 1989]: namely, the manipulator values one of the candidates at 1 and the rest of the candidates at 0.

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 [Bartholdi *et al.*, 1989]. Now, [Bartholdi *et al.*, 1989] show 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 [Bartholdi *et al.*, 1989]. Our second contribution is in demonstrating that Maximin and Borda, as well as many families of scoring rules, become hard to manipulate if we allow arbitrary polynomial-time tie-breaking rules; our proof also works for Copeland. One can view these results as a continuation of the line of work suggested in [Conitzer and Sandholm, 2003; Elkind and Lipmaa, 2005], 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 [Conitzer and Sandholm, 2003] and [Elkind and Lipmaa, 2005] (i.e., combining a voting rule with a preround or taking a "hybrid" of the rule with itself or another rule).

The rest of the paper is organized as follows. We introduce the necessary notation and definitions in Section 2. Section 3 discusses the algorithm and the formal model of [Bartholdi *et al.*, 1989]. We describe the algorithms for scoring rules and Maximin under randomized tie-breaking in Section 4, and prove our hardness results in Section 5. Section 6 concludes.

2 Preliminaries

An *election* is specified by a set of candidates C, $|C| = m$, and a set of *voters* $V = \{v_1, \dots, v_n\}$, where each voter v_i is associated with a linear order R_i over the candidates in C; this order is called v_i's *preference order*. We denote the space of all linear orderings over C by $\mathcal{L}(C)$. 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 v_i prefers a to b. The vector $\mathcal{R} = (R_1, \dots, R_n)$, where each R_i is a linear order over C, is called a *preference profile*. 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, V) is a mapping $T = T(\mathcal{R}, S)$ that for any $S \subseteq C$, $S \neq \emptyset$, outputs a candidate $c \in S$. We say that 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 $a \in S \setminus \{c\}$.

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 $T \circ \mathcal{F}(\mathcal{R}) \in \mathcal{F}(\mathcal{R})$.

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

Scoring rules Any vector $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{R}^m$ 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 $\alpha^m = (\alpha_1^m, \dots, \alpha_m^m)$. 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 $\alpha_1^m, \dots, \alpha_m^m$. Two well-known examples of polynomial-time computable families of scoring rules are *Borda*, given by $\alpha^m = (m-1, \dots, 1, 0)$, and k-*approval*, given by $\alpha_i^m = 1$ if $i \leq k$, $\alpha_i^m = 0$ if $i > k$. 1-approval is also known as *Plurality*.

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.

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

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 v_i can successfully *manipulate* an election (C, V) with a preference profile $(R_1, \dots, R_n)$ 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.

An instance of the $\mathcal{F}$-MANIPULATION problem is given by a set of candidates C, a set of voters V, a preference profile $\mathcal{R}$, and the manipulating voter v_i. It is a "yes"-instance if there exists a vote L such that $\mathcal{F}(\mathcal{R}_{-i}, L) \succ_i \mathcal{F}(\mathcal{R})$ and a "no"-instance otherwise.

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

Before we describe the algorithm presented in [Bartholdi *et al.*, 1989], we remark that the definition of successful manipulation given in [Bartholdi *et al.*, 1989] differs from our definition of $\mathcal{F}$-MANIPULATION (which is modeled after the standard social choice definition, see, e.g. [Gibbard, 1973; Satterthwaite, 1975]), even if we assume that $\mathcal{F}$ is a voting rule rather than a voting correspondence. Specifically, in [Bartholdi *et al.*, 1989] it is assumed that the manipulator has a preferred candidate p, and his goal is to make p elected; we will refer to this problem as $\mathcal{F}$-MANIPULATION(p). However, a poly-time algorithm for $\mathcal{F}$-MANIPULATION(p) can be converted into a poly-time algorithm for $\mathcal{F}$-MANIPULATION: we can run $\mathcal{F}$-MANIPULATION(p) on all candidates ranked by the manipulator above the current winner, and pick the best among the candidates for which $\mathcal{F}$-MANIPULATION(p) outputs "yes". Thus, if $\mathcal{F}$-MANIPULATION is hard, $\mathcal{F}$-MANIPULATION(p) is hard, too. Moreover, all of our hardness reductions directly show hardness of both variants of the problem.

The algorithm for $\mathcal{F}$-MANIPULATION(p) proposed in [Bartholdi *et al.*, 1989] assumes that the voting rule assigns scores to all candidates, and the winners are the candidates with the highest scores. Let v be the manipulator, and let p be her preferred candidate. The algorithm places 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 v's vote so that his score does not exceed that of p. This approach works as long as the rule is monotone and we can determine a candidate's final score given his position in v's vote and the identities of the candidates that v ranks above him. It is not hard to show that Plurality and Borda (and, in fact, all scoring rules), as well as Copeland and Maximin have this property.

We can easily modify this algorithm for the setting where the ties are broken adversarially to the manipulator: in that case, when the manipulator fills a position i in his vote, $i > 1$, he needs to ensure that the score of the candidate in that position is strictly less than that of p. A similar approach works for an arbitrary lexicographic ordering $\succ$ over the candidates.

4 Randomized Tie-Breaking Rules

In this section, we consider 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 v 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 v strongly prefers a, but is essentially indifferent between b and c, then the answer is probably positive, but if v strongly dislikes c and slightly prefers a to b, the answer is likely to be negative (of course, this also depends on v'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 $S \subseteq C$. Given a voting correspondence $\mathcal{F}$ and an election (C, V) with a preference profile $\mathcal{R}$, we say that a vote L is *optimal* for a voter $v_i \in V$ with a utility function $u_i : C \to \mathbb{N}$ with respect to $\mathcal{F}$ combined with the randomized tie-breaking rule if $u_i(\mathcal{F}(\mathcal{R}_{-i}, L)) \geq u_i(\mathcal{F}(\mathcal{R}_{-i}), L')$ for all $L' \in \mathcal{L}(C)$. We say that v_i has a *successful manipulation* if his optimal vote L satisfies $u_i(\mathcal{F}(\mathcal{R}_{-i}, L)) > u_i(\mathcal{F}(\mathcal{R}))$. In the rest of this section, we will explore the complexity of finding an optimal vote with respect to scoring rules and Maximin.

4.1 Scoring rules

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

Theorem 4.1 *For any election $E = (C, V)$ with $|C| = m$, any voter $v \in V$ with a utility function $u : C \to \mathbb{N}$, and any scoring vector $\alpha = (\alpha_1, \ldots, \alpha_m)$, we can find in polynomial time an optimal vote for v with respect to the scoring rule $\mathcal{F}_\alpha$ combined with the randomized tie-breaking rule.*

Proof Fix a voter $v \in V$ 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 v 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 argue that if L maximizes v's utility, then either L is conservative or it can be chosen so that $\mathcal{F}_\alpha$ has a unique winner under $(\mathcal{R}', L)$. Indeed, suppose that this is not the case, i.e., any vote L that maximizes v'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 v'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 $c_i + \alpha_j > T$, 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_i 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 v 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 winners' set. 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_i, we check if $u(c_i) > u(S)$; if this is not the case, we terminate, as all subsequent candidates have even lower utility. Otherwise, we check if we can swap c_i with another candidate that is currently not in S and receives $T - s_i$ points from L (so that c_i gets T points in the resulting vote). If this is the case, we update L by performing the swap and set $S = S \cup \{c_i\}$. 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_i 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.

The following lemma shows that $\hat{S} \setminus S$ is empty.

Lemma 4.2 *We have $\hat{S} \setminus S = \emptyset$.*

Proof Suppose $\hat{S} \setminus S \neq \emptyset$, 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 $u(\hat{S} \setminus \{c_i\}) > u(\hat{S})$, a contradiction to 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 to 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 4.3 *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 to 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 4.2: 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 4.2 and Lemma 4.3, 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 to 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.

4.2 Maximin

For Maximin with randomized tie-breaking, we have not been able to design an efficient algorithm for finding an optimal manipulation in the general utility model. However, this problem admits a poly-time algorithm if the manipulator's utility function has a special structure. Specifically, recall that in the model of [Bartholdi *et al.*, 1989] 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 poly-time algorithm for finding an optimal manipulation under Maximin combined with the randomized tie-breaking rule.

Theorem 4.4 *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 sketch We construct a directed graph whose vertices are candidates, and there is an edge from c_i to c_j if c_i is c_j's most dangerous opponent. Let $s_M(c)$ denote the Maximin score of c. Under the utility function u, our goal is to make p one of the winners and minimize the number of candidates tied with p. This can only be achieved if $s(p) \geq \max_i s(c_i) - 1$; assume that this is indeed the case. We say that a vertex c is *purple* if $s(c) = s(p) + 1$, and *red* if $s(c) = s(p)$. Then the manipulator's goal can be reformulated as follows: order the vertices of the graph so that each purple vertex, and as many red vertices as possible, have an incoming edge from a predecessor. This can be achieved by purely graph-theoretic means, but the algorithm is not straightforward; in particular, it may have to contract and then expand cycles.

5 Hardness results

In this section, we consider deterministic polynomial-time tie-breaking rules, We will first present a specific tie-breaking rule T, and then show that manipulating the composition of this rule with Borda, Copeland or Maximin is NP-hard.

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 s variables $x_1, \ldots, x_s$, there are exactly $\ell = \binom{2s}{3}$ 3-literal clauses that can be formed from these variables (this includes clauses of the form $x_1 \wedge \overline{x}_1 \wedge x_2$). Ordering the literals as $x_1 < \overline{x}_1 < \cdots < x_s < \overline{x}_s$ 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 s 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 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.

In all cases, our argument proceeds as follows. Given an instance $\mathcal{C}$ of 3-SAT with s variables, we create an election with $m = \ell + 2s + 4$ candidates, where $\ell = \binom{2s}{3}$. The manipulator v ranks c_m first, followed by c_{m-1}, followed by all the remaining candidates. The voters' preferences are set so that if v votes truthfully, c_{m-1} wins, so v can only benefit from the manipulation if he can make c_m the winner. Moreover, the preferences of the non-manipulators are selected so that (a) the set of candidates in $\{c_1, \ldots, c_\ell\}$ with the maximum score encodes $\mathcal{C}$, and the manipulator cannot change this without jeopardizing c_m's chances of winning; (b) for each pair $(c_{\ell+2i-1}, c_{\ell+2i})$, $i = 1, \ldots, s$, the manipulator can ensure that exactly one of these candidates has the maximum score, i.e., v can select an arbitrary truth assignment; (c) v can ensure that c_m has the maximum score. We remark that constructing an election with these properties is not trivial; in particular, the constructions for Borda, Maximin and Copeland are very different. It follows that any beneficial manipulation corresponds to a satisfying assignment and vice versa. Thus, we obtain the following result.

Theorem 5.1 $T \circ \mathcal{F}$-MANIPULATION *is* NP*-hard for* $\mathcal{F} \in \{$ Borda*,* Maximin*,* Copeland$^\alpha$*,* $\alpha \in [0, 1]\}$.

The tie-breaking rule T has the attractive property that to compute the winner, we only need to know the set of tied candidates S. Theorem 5.1 can be generalized to other families of scoring rules, including k-approval, where k is polynomially related to m (i.e., $m = \text{poly}(k)$), using a modified version of T which still has this property. However, the combination of any such tie-breaking rule and Plurality can be shown to be easy to manipulate. Therefore, to prove an analogue of Theorem 5.1 for Plurality, we use a tie-breaking rule that depends on the votes themselves, and not just on S.

Theorem 5.2 *There exists a tie-breaking rule* T' *such that* $T' \circ$ Plurality-MANIPULATION *is* NP*-complete.*

6 Conclusions and Future Work

We have explored the complexity of manipulating many common voting rules under randomized tie-breaking as well as under arbitrary polynomial-time tie-breaking procedures. Our results for randomized tie-breaking are far from complete, and a natural research direction is to extend them to other voting rules, such as Copeland or Bucklin, as well as to the Maximin rule with general utilities.

Acknowledgments This research was supported by National Research Foundation (Singapore) under grant 2009-08 (Elkind), by NTU SUG (Elkind), by SINGA graduate fellowship (Obraztsova). and by the President of RF grant for leading scientific schools support NSh-5282.2010.1.

References

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

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

[Conitzer and Sandholm, 2003] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *IJCAI'03*, pp. 781–788, 2003.

[Conitzer *et al.*, 2007] V. Conitzer, T. Sandholm, and J. Lang, When ere elections with few candidates hard to manipulate?. *J. ACM*, 54:1–33, 2007.

[Davies *et al.*, 2010] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. An empirical study of Borda manipulation. In *COMSOC'10*, pp. 91–102, 2010.

[Desmedt and Elkind, 2010] Y. Desmedt, E. Elkind. Equilibria of plurality voting with abstentions. In EC'10, pp. 347–356, 2010.

[Elkind and Lipmaa, 2005] E. Elkind, H. Lipmaa, Hybrid voting protocols and hardness of manipulation. In *ISAAC'05*, 2005.

[Faliszewski *et al.*, 2008] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: ties matter. In *AAMAS'08*, 2008.

[Faliszewski and Procaccia, 2010] P. Faliszewski, A. Procaccia. AI's war on manipulation: are we winning? In *AI Magazine*, 31(4):53–64, 2010.

[Friedgut *et al.*, 2008] E. Friedgut, G. Kalai, and N. Nisan. Elections can be manipulated often. In *FOCS'08*, pp. 243–249, 2008.

[Garey and Johnson, 1979] M. R. Garey, D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness, 1979.

[Gibbard, 1973] A. F. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41:597–601, 1973.

[Isaksson *et al.*, 2010] M. Isaksson, G. Kindler, and E. Mossel. The geometry of manipulation—a quantitative proof of the Gibbard–Satterthwaite theorem. In *FOCS'10*, pp. 319–328, 2010.

[McGarvey, 1953] D. C. McGarvey. A Theorem on the Construction of Voting Paradoxes. *Econometrica*, 21(4):608–610, 1953.

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

[Satterthwaite, 1975] M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[Walsh, 2009] T. Walsh. Where are the really hard manipulation problems? The phase transition in manipulating the Veto rule. In *IJCAI'09*, pp. 324–329, 2009.

[Xia and Conitzer, 2008] L. Xia and V. Conitzer, A sufficient condition for voting rules to be frequently manipulable. In *EC'08*, pp. 99–108, 2008.

[Xia *et al.*, 2010] L. Xia, V. Conitzer, A. Procaccia. A scheduling approach to coalitional manipulation. In *EC'10*, 2010.

[Xia *et al.*, 2009] L. Xia, M. Zuckerman, A. Procaccia, V. Conitzer, and J. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *IJCAI'09*, pp. 348–352, 2009.