

Determining Possible and Necessary Winners under Common Voting Rules Given Partial Orders

Lirong Xia

Vincent Conitzer

Department of Computer Science, Duke University, Durham, NC 27708, USA

LXIA@CS.DUKE.EDU

CONITZER@CS.DUKE.EDU

Abstract

Usually a voting rule or correspondence requires agents to give their preferences as linear orders. However, in some cases it is impractical for an agent to give a linear order over all the alternatives. It has been suggested to let agents submit partial orders instead. Then, given a voting rule, a profile of partial orders, and an alternative (candidate) c , two important questions arise: first, is c guaranteed to win, and second, is it still possible for c to win? These are the *necessary winner* and *possible winner* problems, respectively. Each of these two problems is further divided into two sub-problems: determining whether c is a *unique winner* (that is, c is the only winner), or determining whether c is a *co-winner* (that is, c is in the set of winners).

We consider the setting where the number of alternatives is unbounded and the votes are unweighted. We completely characterize the complexity of possible/necessary winner problems for the following common voting rules: positional scoring rules, Copeland, maximin, Bucklin, ranked pairs, voting trees, and plurality with runoff. More precisely, we prove that the possible winner problems for Copeland, maximin, Bucklin, ranked pairs, and the possible unique winner problem for plurality with runoff are NP-complete; also, we give a sufficient condition on positional scoring rules for the possible winner problem to be NP-complete (Borda satisfies this condition); we give a sufficient condition on voting trees for the possible winner problem to be NP-complete and for the necessary winner problem to be coNP-complete (balanced voting trees satisfy this condition). We also prove that the necessary winner problem for Copeland and ranked pairs, and the necessary co-winner problem for plurality with runoff are coNP-complete. All the hardness results, except the one for possible unique winner w.r.t. plurality with runoff, hold even when the number of undetermined pairs in each vote is no more than a constant. We also present polynomial-time algorithms for the necessary winner problem for positional scoring rules, maximin, and Bucklin. For plurality with runoff, we present polynomial-time algorithms for the possible co-winner problem and necessary unique winner problem.

1. Introduction

In multiagent systems, often, the agents must make a joint decision in spite of the fact that they have different preferences over the alternatives. For example, the agents may have to decide on a joint plan or an allocation of tasks/resources. A general solution to this problem is to have the agents *vote* over the alternatives. That is, each agent i gives a ranking (linear order) \succ_i of all the alternatives; then a *voting rule* takes all of the submitted rankings as input, and based on this produces a chosen alternative (the *winner*), or a set of chosen alternatives. The design of good voting rules has been studied for centuries by the *social choice* community. More recently, computer scientists have become interested in

social choice—motivated in part by applications in multiagent systems, but also by other applications. Hence, a community interested in *computational social choice* has emerged.

In “traditional” social choice, agents are usually required to give a linear order over all the alternatives. However, especially in multiagent systems applications, this is not always practical. For one, sometimes, the set of alternatives is too large. For example, there are generally too many possible joint plans or allocations of tasks/resources for an agent to give a linear order over them. In such settings, agents must use a different *voting language* to represent their preferences; for example, they can use CP-nets (Boutilier, Brafman, Domshlak, Hoos, & Poole, 2004; Lang, 2007; Xia, Lang, & Ying, 2007a, 2007b). However, when an agent uses a CP-net (or a similar language) to represent its preferences, this generally only gives us a partial order over the alternatives. Another issue is that it is not always possible for an agent to compare two alternatives (Pini, Rossi, Venable, & Walsh, 2007). Such incomparabilities also result in a partial order.

In this paper, we study the setting where for each agent, we have a partial order corresponding to that agent’s preferences. We study the following two questions. 1. Is it the case that, for *any* extension of the partial orders to linear orders, alternative c wins? 2. Is it the case that, for *some* extension of the partial orders to linear orders, alternative c wins? These problems are known as the *necessary winner* and *possible winner* problems, respectively (Konczak & Lang, 2005). Depending on the interpretation of “ c wins”, the possible/necessary winner problems are further divided into two sub-problems: one is called the possible/necessary unique winner problem (here “unique” is often omitted when causing no confusion), in which “ c wins” means that c is the only winner of the election; the other is called the possible/necessary co-winner problem, in which “ c wins” means that c is one of the winners. It should be noted that the answer depends on the voting rule used. Previous research has also investigated the setting where there is uncertainty about the *voting rule*; here, a necessary (possible) winner is an alternative that wins for any (some) realization of the rule (Lang, Pini, Rossi, Venable, & Walsh, 2007). In this paper, we will not study this setting; that is, the rule is always fixed.

While these problems are motivated by the above observations on the impracticality of submitting linear orders, they also relate to *preference elicitation* and *manipulation*. In preference elicitation, the idea is that, instead of having each agent report its preferences all at once, we ask them simple queries about their preferences (*e.g.* “Do you prefer a to b ?”), until we have enough information to determine the winner. Preference elicitation has found many applications in multiagent systems, especially in combinatorial auctions (for overviews, see (Parkes, 2006; Sandholm & Boutilier, 2006)) and in voting settings as well (Conitzer & Sandholm, 2002, 2005b; Conitzer, 2009). The problem of deciding whether we can terminate preference elicitation and declare a winner is exactly the necessary winner problem. Manipulation is said to occur when an agent casts a vote that does not correspond to its true preferences, in order to obtain a result that it prefers. By the Gibbard-Satterthwaite Theorem (Gibbard, 1973; Satterthwaite, 1975), for any reasonable voting rule, there are situations where an agent can successfully manipulate the rule. To prevent manipulation, one approach that has been taken in the computational social choice community is to study whether manipulation is (or can be made) computationally hard (Bartholdi, Tovey, & Trick, 1989a; Bartholdi & Orlin, 1991; Elkind & Lipmaa, ; Conitzer, Sandholm, & Lang, 2007; Zuckerman, Procaccia, & Rosenschein, 2009). The fundamental questions that have

been studied here are “Given the other votes, can this coalition of agents cast their votes so that alternative c wins?” (so-called *constructive manipulation*) and “Given the other votes, can this coalition of agents cast their votes so that alternative c does not win?” (so-called *destructive manipulation*). These problems correspond to the possible winner problem and (the complement of) the necessary winner problem, respectively. To be precise, they only correspond to restricted versions of the possible winner problem and (the complement of) the necessary winner problem in which some of the partial orders are linear orders (the nonmanipulators’ votes) and the other partial orders are empty (the manipulators’ votes). However, if there is uncertainty about parts of the nonmanipulators’ votes, or if parts of the manipulators’ votes are already fixed (for example due to preference elicitation), then they can correspond to the general versions of the possible winner problem and (the complement of) the necessary winner problem.

Because of the variety of different interpretations of the possible and necessary winner problems, it is not surprising that there have already been significant studies of these problems. Two main settings have been studied (see (Walsh, 2007) for a good survey). In the first setting, the number of alternatives is bounded, and the votes are weighted. Here, for the Borda, veto, Copeland, maximin, STV, and plurality with runoff rules, the possible winner problem is NP-complete; for the STV and plurality with runoff rules, the necessary winner problem is coNP-complete (Conitzer et al., 2007; Pini et al., 2007; Walsh, 2007). However, in many elections, votes are unweighted (that is, each agent’s vote counts the same). If the votes are unweighted, and the number of alternatives is bounded, then the possible and necessary winner problems can always be solved in polynomial time (assuming the voting rule can be executed in polynomial time) (Conitzer et al., 2007; Walsh, 2007). Hence, the other setting that has been studied is that where the votes are unweighted and the number of alternatives is not bounded; this is the setting that we will study in this paper. In this setting, the possible and necessary winner problems are known to be hard for STV (Bartholdi & Orlin, 1991; Pini et al., 2007; Walsh, 2007). Computing whether an alternative is a possible or necessary Condorcet winner can be done in polynomial time (Konczak & Lang, 2005). However, at the time of the conference version of this work (AAAI-08), for most of the other common rules, there were no prior results (except for the fact that the problems are easy for many of these rules when each partial order is either a linear order or empty, that is, the standard manipulation problem).¹

In this paper, we characterize the complexity of the possible and necessary winner problems for some of the most important other rules—specifically, positional scoring rules, Copeland, maximin, Bucklin, ranked pairs, voting trees, and plurality with runoff. We show that the possible winner problems are NP-complete for all these rules except the possible unique winner problem w.r.t. plurality with runoff. We also show that the necessary winner problems are coNP-complete for the Copeland, ranked pairs, and voting trees, and the necessary co-winner problem is coNP-complete for plurality with runoff. For the remaining cases, we present polynomial-time algorithms.

1. An earlier paper (Konczak & Lang, 2005) studied these problems for positional scoring rules, and claimed that the problems are polynomial-time solvable for these rules; however, there was a subtle mistake in their proofs. We will show that the possible winner problem is in fact NP-complete for these rules. We will also give a correct proof that the necessary winner problem is indeed polynomial-time solvable for these rules.

Since the conference version of this work, a number of new results on the complexity of the unweighted coalitional manipulation problem have been obtained. Specifically, this problem has been shown to be NP-hard for Copeland (Faliszewski, Hemaspaandra, & Schnoor, 2008), maximin and ranked pairs (Xia, Zuckerman, Procaccia, Conitzer, & Rosenschein, 2009), and a specific positional scoring rule (Xia, Conitzer, & Procaccia, 2009). As we mentioned before, this problem is a special case of the possible winner problem studied in this paper (where some partial orders are linear orders and the others are empty); as a result, these results also imply NP-hardness of the possible winner problem for these rules. Still, this subsequent research does not completely imply the NP-hardness results that we prove in this paper for the possible winner problem for these rules, because the hardness results proved in this paper (except the possible unique winner problem for plurality with runoff) hold even when for each partial order, the number of pairs of alternatives for which the order is unknown is a constant. The possible winner problem also reduces to the *swap bribery* problem, in which an interested party can pay voters to swap adjacent alternatives in their rankings, but the price to swap two alternatives depends on both the identity of the alternatives and the identity of the voter. This means that the computational complexity of the possible winner problem w.r.t. a certain voting rule is at least as high as that of the swap bribery problem w.r.t. the same voting rule, in terms of polynomial-time reductions (Elkind, Faliszewski, & Slinko, 2009). The complexity of the possible winner problems have also been studied for bounded parameters such as the number of alternatives, number of voters, and number of unknown pairs in each vote (Betzler, Hemmann, & Niedermeier, 2009).

2. Preliminaries

Let $\mathcal{C} = \{c_1, \dots, c_m\}$ be the set of *alternatives* (or *candidates*). A linear order on \mathcal{C} is a transitive, antisymmetric, and total relation on \mathcal{C} . The set of all linear orders on \mathcal{C} is denoted by $L(\mathcal{C})$. An n -voter profile P on \mathcal{C} consists of n linear orders on \mathcal{C} . That is, $P = (V_1, \dots, V_n)$, where for every $i \leq n$, $V_i \in L(\mathcal{C})$. The set of all profiles on \mathcal{C} is denoted by $P(\mathcal{C})$. In the remainder of the paper, m denotes the number of alternatives and n denotes the number of voters.

A *voting rule* r is a function from the set of all profiles on \mathcal{C} to \mathcal{C} , that is, $r : P(\mathcal{C}) \rightarrow \mathcal{C}$. The following are some common voting rules.

1. *(Positional) scoring rules*: Given a *scoring vector* $\vec{v} = (v(1), \dots, v(m))$, for any vote $V \in L(\mathcal{C})$ and any $c \in \mathcal{C}$, let $s(V, c) = v(j)$, where j is the rank of c in V . For any profile $P = (V_1, \dots, V_n)$, let $s(P, c) = \sum_{i=1}^n s(V_i, c)$. The rule will select $c \in \mathcal{C}$ so that $s(P, c)$ is maximized. Two examples of positional scoring rules are *Borda*, for which the scoring vector is $(m-1, m-2, \dots, 0)$, and *plurality*, for which the scoring vector is $(1, 0, \dots, 0)$.
2. *Copeland*: For any two alternatives c_i and c_j , we can simulate a *pairwise election* between them, by seeing how many votes prefer c_i to c_j , and how many prefer c_j to c_i . Then, an alternative receives one point for each win in a pairwise election.

(Typically, an alternative also receives half a point for each pairwise tie, but this will not matter for our results.) The winner is the alternative who has the highest score.

3. *Maximin*: Let $N(c_i, c_j)$ denote the number of votes that rank c_i ahead of c_j . The winner is the alternative c that maximizes $\min\{N(c, c') : c' \in \mathcal{C}, c' \neq c\}$.
4. *Bucklin*: An alternative c 's Bucklin score is the smallest number k such that more than half of the votes rank c among the top k alternatives. The winner is the alternative who has the smallest Bucklin score. (Sometimes, ties are broken by the number of votes that rank an alternative among the top k , but for simplicity we will not consider this tiebreaking rule here.)
5. *Ranked pairs*: This rule first creates an entire ranking of all the alternatives. $N(c_i, c_j)$ is defined as for the maximin rule. In each step, we will consider a pair of alternatives c_i, c_j that we have not previously considered; specifically, we choose the remaining pair with the highest $N(c_i, c_j)$. We then fix the order $c_i > c_j$, unless this contradicts previous orders that we fixed (that is, it violates transitivity). We continue until we have considered all pairs of alternatives (hence we have a full ranking). The alternative at the top of the ranking wins.
6. *Voting trees*: A voting tree is a binary tree with m leaves, where each leaf is associated with an alternative. In each round, there is a pairwise election between an alternative c_i and its sibling c_j : if a majority of voters prefers c_i to c_j , then c_j is eliminated, and c_i is associated with the parent of these two nodes; similarly, if a majority of voters prefers c_j to c_i , then c_i is eliminated, and c_j is associated with the parent of these two nodes. The alternative that is associated with the root of the tree (wins all its rounds) is the winner.
7. *Plurality with runoff*: The rule has two steps. In the first step, all alternatives except the two that are ranked in the top position for most times are eliminated, and the votes transfers to the second round, in which the plurality rule (a.k.a. *majority* rule in case of two alternatives) is used to select the winner.

All of these rules allow for the possibility that multiple alternatives end up tied for the win. Technically, therefore, they are really *voting correspondences*; a correspondence can select more than one winner. (In the remainder of this paper, we will sometimes somewhat inaccurately refer to the above correspondences as rules.) We adopt *parallel-universes tiebreaking* (Conitzer, Rognlie, & Xia, 2009) to define the winning alternatives for the rules that have multiple rounds (i.e., ranked pairs, voting trees, and plurality with runoff). That is, an alternative c is a winner if and only if there exists a way to break ties in all of the steps such that c is the winner. A partial order on \mathcal{C} is a reflexive, transitive, and antisymmetric relation on \mathcal{C} . We say a linear order V extends a partial order O if $O \subseteq V$.

Definition 1 *A linear order V on \mathcal{C} extends a partial order O on \mathcal{C} if for any $i, j \leq m$, $c_i \succ_O c_j \Rightarrow c_i \succ_V c_j$.*

We are now ready to define possible (necessary) winners. We will define them for a correspondence r .

Definition 2 Given a profile of partial orders $P_O = (O_1, \dots, O_n)$ on \mathcal{C} , we say that an alternative $c \in \mathcal{C}$ is: 1. a possible winner if there exists $P = (V_1, \dots, V_n)$ such that each V_i extends O_i , and $r(P) = \{c\}$. 2. a necessary winner if for any $P = (V_1, \dots, V_n)$ such that each V_i extends O_i , $r(P) = \{c\}$. 3. a possible co-winner if there exists $P = (V_1, \dots, V_n)$ such that each V_i extends O_i , and $c \in r(P)$. 4. a necessary co-winner if for any $P = (V_1, \dots, V_n)$ such that each V_i extends O_i , $c \in r(P)$.

Example 1 Let there be three alternatives $\{c_1, c_2, c_3\}$. Three partial orders are illustrated in Figure 1. Let $P_O = (O_1, O_2, O_3)$. c_1 is a possible (co-)winner of P_O w.r.t. plurality, because we can complete O_1 by adding $c_2 \succ c_3$, complete O_2 by adding $c_1 \succ c_2$, and complete O_3 by adding $c_1 \succ c_2$ and $c_1 \succ c_3$; then, c_1 is the only winner. However, c_1 is not a necessary (co-)winner, because we can complete O_1 by adding $c_2 \succ c_3$, complete O_2 by adding $c_2 \succ c_1$, and complete O_3 by adding $c_2 \succ c_1$ and $c_1 \succ c_3$; then, c_2 is the only winner.

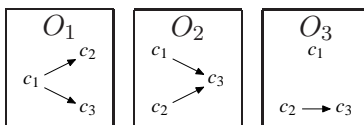


Figure 1: Partial orders.

However, if we let $P'_O = (O_1, O_1, O_2)$, then c_1 is the (only) necessary winner, because c_1 will be ranked first for at least two votes.

Now, we define the computational problems:

Definition 3 Define the problem Possible Winner (PW) w.r.t. voting correspondence r to be: given a profile P_O of partial orders and an alternative c , we are asked whether or not c is a possible winner for P_O w.r.t. r .

Necessary Winner (NW), Possible co-Winner (PcW), and Necessary co-Winner (NcW) are defined similarly.

3. Hardness results

In this section, we will prove that PW (PcW) is NP-complete w.r.t. positional scoring rules, Copeland, maximin, Bucklin, ranked pairs, and voting trees; NW (NcW) is coNP-complete w.r.t. Copeland, ranked pairs, and voting trees; and PW is NP-complete and NcW is coNP-complete w.r.t. plurality with runoff. For positional scoring rules, we will not show that PW is hard for all positional scoring rules—in fact, for plurality, PW is easy; rather, we will give a sufficient condition on a positional scoring rule such that PW is hard. Borda satisfies this condition. Similarly for voting trees, we provide a necessary condition under which the hardness results to hold. Balanced voting trees satisfy this condition. All of these results (except the one for PW w.r.t. plurality with runoff) hold even when the partial orders are “almost” linear orders. That is, the number of undetermined pairs in each partial order is bounded above by a constant.

All the hardness results are proved by reductions from the EXACT 3-COVER (X3C) problem (where we are given a set and a collection of subsets of size 3 of this set, and we

are asked if we can cover all of the elements in the set with nonoverlapping subsets). X3C is known to be strongly NP-complete (Garey & Johnson, 1979), that is, it is NP-complete even if the inputs are encoded in unary. In each proof, the instance that we construct from an X3C instance consists of two parts. The first part is a set of partial orders that encode the X3C instance. The second part is a set of linear orders (so that there are no uncertainties here) whose purpose is, informally stated, to adjust the scores of the alternatives. We denote an X3C instance by $\mathcal{V} = \{v_1, \dots, v_q\}$, $\mathcal{S} = \{S_1, \dots, S_t\}$, where $S_i = \{v_{l(i,1)}, v_{l(i,2)}, v_{l(i,3)}\} \subseteq \mathcal{V}$ for all $i \leq t$, with $1 \leq l(i, 1), l(i, 2), l(i, 3) \leq q$.

First we introduce a notation to represent the set of all pairwise orders in a linear order.

Definition 4 For any set $\{a_1, \dots, a_l\}$, let $O(a_1, \dots, a_l) = \{(a_i, a_j) : i < j\}$.

That is, $O(a_1, \dots, a_l)$ is the set of all pairs consistent with the linear order $a_1 \succ \dots \succ a_l$. For example, $O(a, b, c) = \{(a, b), (b, c), (a, c)\}$.

Usually, a positional scoring rule is defined for a fixed number of alternatives, which means that the number of alternatives is bounded. Then, there exist polynomial-time algorithms for both PW and NW (Walsh, 2007; Conitzer et al., 2007). However, there are positional scoring rules that are defined for any number of alternatives—for example, Borda and plurality. For such positional scoring rules, the number of alternatives is not bounded, and indeed, we will prove that PW is not always easy w.r.t. such rules. In the remainder of the paper, a positional scoring rule r consists of a sequence of scoring vectors $\{\vec{s}_1, \vec{s}_2, \dots\}$ such that for any $i \in \mathbb{N}$, \vec{s}_i is a scoring vector for i alternatives. The next theorem provides a sufficient condition on a positional scoring rule for PW to be NP-complete. (Below, we do not prove membership in (co)NP, because this follows from the fact that, given an extension of the partial orders to linear orders, we can compute the winner(s) in polynomial-time for the rules in this paper. There do exist rules for which computing the winner(s) is NP-hard, for example the Dodgson rule (Bartholdi, Tovey, & Trick, 1989b; Hemaspaandra, Hemaspaandra, & Rothe, 1997) and the Young rule (Rothe, Spakowski, & Vogel, 2003), but we will not study those here.)

Theorem 1 For any positional scoring rule r with scoring vectors $\{\vec{s}_1, \vec{s}_2, \dots\}$, if there exists a polynomial $f(x)$ such that for any $x \in \mathbb{N}$, there exist $x \leq l \leq f(x)$ and $k \leq l - 4$ satisfying the following two conditions:

1. $\vec{s}_l(k) - \vec{s}_l(k+1) = \vec{s}_l(k+1) - \vec{s}_l(k+2) = \vec{s}_l(k+2) - \vec{s}_l(k+3) > 0$,
2. $\vec{s}_l(k+3) - \vec{s}_l(k+4) > 0$,

then PW and PcW are both NP-complete w.r.t. r , even when the number of undetermined pairs in each vote is no more than 4. (To obtain membership in NP, it is assumed that the score vectors can be computed in polynomial time.)

Theorem 1 provides a sufficient condition on positional scoring rules for PW and PcW to be NP-complete. It can be applied to show NP-completeness for Borda:

Corollary 1 PW and PcW are NP-complete w.r.t. Borda, even when the number of undetermined pairs in each vote is no more than 4.

Proof. For any $l \in \mathbb{N}$, the scoring vector \vec{s}_l for Borda is $(l-1, l-2, \dots, 0)$. If we let $f(x) = x$, $l = x$, and $k = l-4$, then the conditions in Theorem 1 are all satisfied, and the claim follows. \square

Theorem 2 *PW and PcW are NP-complete and NW and NcW are coNP-complete w.r.t. Copeland, even when the number of undetermined pairs in each vote is at most 8.*

Theorem 3 *PW and PcW are NP-complete w.r.t. Bucklin, even when the number of undetermined pairs in each vote is at most 16.*

To prove our hardness results for maximin and ranked pairs, we first give two helpful lemmas.

Definition 5 *Given a profile P , the pairwise score difference $D_P(c, c')$ of alternative c and c' is defined to be*

$$D_P(c, c') = |\{V \in P : c \succ_V c'\}| - |\{V \in P : c' \succ_V c\}|$$

The subscript P is omitted when there is no risk of confusion. It follows from the definition that $D(c, c') = -D(c', c)$. We note that although maximin and ranked pairs are based on pairwise scores, it can also be computed by pairwise score differences in the same way, because for any profile P of n votes, and any pair of alternatives (c, c') , we have $D_P(c, c') = 2N_p(c, c') - n$.

We now show that given any pair of alternatives c, c' , there exist two linear orders that increase $D(c, c')$ by two while keeping all other pairwise score differences unchanged. (This lemma has been used previously by others (McGarvey, 1953; Conitzer & Sandholm, 2005a)) We will use this technique in the second (score-adjusting) part of the following reductions.

Lemma 1 *Given any profile P and any two alternatives c, c' , let the remaining alternatives be $\{c_1, \dots, c_{m-2}\}$. Let P' be the profile consisting of P plus the following two votes:*

1. $c \succ c' \succ c_1 \succ \dots \succ c_{m-2}$, and
2. $c_{m-2} \succ \dots \succ c_1 \succ c \succ c'$.

Then $D_{P'}(c, c') = D_P(c, c') + 2$, and for any alternatives p, q such that $\{d, d'\} \neq \{c, c'\}$, $D_{P'}(d, d') = D_P(d, d')$.

This lemma tells us that the pairwise scores can be changed nearly arbitrarily. The following lemma is a direct corollary.

Lemma 2 *(The main theorem in (McGarvey, 1953), also used as Lemma 2 in (Conitzer & Sandholm, 2005a)) Given a profile P and any skew symmetric function $F : \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{N}$ (that is, $F(c_1, c_2) = -F(c_2, c_1)$ for all c_1, c_2), such that for all pairs of alternatives $c, c' \in \mathcal{C}$, $F(c, c') - D_P(c, c')$ are all even (or all odd), then there exists a profile P' such that*

1. $|P'| \leq \frac{1}{2} \sum_{c, c'} (|F(c, c') - D_P(c, c')| + 1)$,

2. $D_{P \cup P'} = F$.

That is, for any skew-symmetric pairwise score difference function F such that $F(c, c')$ has the same parity for any pair of alternatives (c, c') (with $c \neq c'$), we can change the pairwise scores from D_P to F by adding no more than $\frac{1}{2} \sum_{c, c'} (|F(c, c') - D_P(c, c')| + 1)$ votes to P .

Now we are ready to prove the hardness results for maximin and ranked pairs.

Theorem 4 *PW and PcW are NP-complete w.r.t. maximin, even when the number of undetermined pairs in each vote is at most 4.*

Theorem 5 *PW and PcW are NP-complete and NW and NcW are coNP-complete w.r.t. ranked pairs, even when the number of undetermined pairs in each vote is at most 8.*

Next, we consider voting trees. Because a voting tree is defined for a fixed number of alternatives, to study the complexity of the possible/necessary winner problems w.r.t. voting trees, we need to consider an infinite sequence of trees, one for each natural number (representing the number of alternatives).² Therefore, we let a voting tree rule T be composed of an infinite sequence of voting trees $\{T_1, T_2, \dots\}$, where for any $m \in \mathbb{N}$, T_m is a voting tree for m alternatives (that is, T_m is a binary tree that has m leaf nodes, and each leaf is associated with an alternative).

For any $t \in \mathbb{N}$, a voting tree T_m is t -well-spread if there exist t pairs of leaves $(c_1, a_1), \dots, (c_t, a_t)$, such that for any $i \leq t$, c_i and a_i are siblings. We say any such a leaf in a pair is a *rich leaf*. A voting tree is *balanced* if the depths of any two leaves differ at most by one, and the number of leaves whose sibling is not a leaf is at most one.

Example 2 *Two voting trees are illustrated in Figure 2. The voting tree in (a) is 1-well-spread, and c_1 and c_2 are rich leaves; the voting tree in (b) is balanced and 3-well-spread, and all leaves except c_5 are rich leaves.*

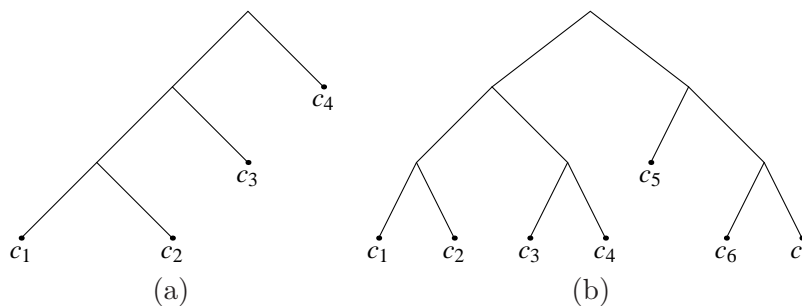


Figure 2: Voting trees.

2. This is similar to the case of positional scoring rules, which are technically defined only for a specific number of alternatives; to study the complexity of social choice problems that involve a growing number of alternatives, it is necessary to associate a score vector with every natural number of alternatives. Rules such as plurality, veto, and Borda do this naturally.

Theorem 6 *For any voting tree rule $T = \{T_1, T_2, \dots\}$, if there exists an alternative c and a polynomial $f(x)$ such that for any $x \in \mathbb{N}$, there exists an $l \in \mathbb{N}$ with $x \leq l \leq f(x)$ such that T_l is x -well-spread, and c is a rich leaf; then, PW and PcW are NP -complete, and NW and NcW are $coNP$ -complete w.r.t. T , even when the number of undetermined pairs in each vote is at most 16.*

From Theorem 6, we immediately obtain the following hardness results for voting tree rules composed of balanced trees, by setting $f(x) = 4x$ (because there will exist some integer y such that $2x \leq 2^y \leq 4x$, so in the balanced tree for 2^y alternatives there will be at least x pairs of siblings).

Corollary 2 *PW and PcW are NP -complete and NW and NcW are $coNP$ -complete w.r.t. the voting tree rule that is composed of balanced binary trees, even when the number of undetermined pairs in each vote is at most 16.*

Finally, we have the following theorems on the complexity of PW and NcW w.r.t. plurality with runoff.

Theorem 7 *PW is NP -complete w.r.t. plurality with runoff.*

Theorem 8 *NcW is $coNP$ -complete w.r.t. plurality with runoff, even when the number of undetermined pairs in each vote is at most 4.*

4. Polynomial time algorithms for possible and necessary winner problems

In this section we present polynomial-time algorithms to compute whether an alternative is a necessary (co-)winner w.r.t. positional scoring rules, maximin, and Bucklin. For all the problems mentioned above, the time complexities are $O(nm^3)$, where m is the number of alternatives and n is the number of votes. For plurality with runoff, we present a $O(m^2n^3(m+n)(\log \log n) \log(m+n))$ algorithm for PcW , and a $O(m^3n^3(m+n)(\log \log n) \log(m+n))$ algorithm for NW . We recall that PW is NP -complete (Theorem 7) and NcW is $coNP$ -complete (Theorem 8), both w.r.t. plurality with runoff.

We note that positional scoring rules, maximin, and Bucklin are all based on some type of scores, so if we can find an extension of the partial orders to linear orders so that the score of c , denoted by $S(c)$, is at most the score of another alternative w , then c is not the (unique) winner in this profile, and hence c is not a necessary winner. So, in the following algorithms for these rules, we check all alternatives $w \neq c$, and try to make $S(c) - S(w)$ as low as possible on a vote-by-vote basis. For each vote O (partial order), there can be two cases. In the first case, $c \not\prec_O w$. In this case, we just consider c and w separately, raising w as high as possible and lower c as low as possible. (This part of the algorithm has already been considered in (Konczak & Lang, 2005).) We will illustrate this method in Example 3. In the second case, $c \succ_O w$. This case is more complicated, and below we show how to minimize $S(c) - S(w)$ for positional scoring rules, maximin, and Bucklin. For plurality with runoff, we convert PcW into a minimum cost flow problem to solve it; this also gives an algorithm for NW , simply by checking whether any other alternative is a possible co-winner.

In this section, the input consists of $\mathcal{C} = \{c, c_1, \dots, c_{m-1}\}$, c (the alternative for which we wish to decide whether or not it is a necessary (co-)winner), a profile P_O of n partial orders, and the voting rule r .

Example 3 A partial order O is illustrated in Figure 3. Since $c_2 \not\succ_O c_5$, we can raise c_5 as high as possible while lowering c_2 as low as possible, as shown in Figure 4.

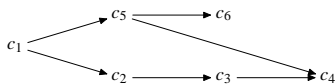


Figure 3: A partial order O .

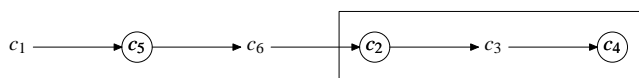


Figure 4: An extension V_1 of O .

We first define some notations that will be used in the algorithms.

Definition 6 Given a partial order O and an alternative c , let $Up_O(c) = \{c' \in \mathcal{C} : c' \succeq_O c\}$ and $Down_O(c) = \{c' \in \mathcal{C} : c \succeq_O c'\}$. Given another alternative w such that $c \succ_O w$, let O 's $c \succ w$ block be defined as follows: $Block_O(c, w) = \{c' \in \mathcal{C} : c \succeq_O c' \succeq_O w\}$.

That is, $Up_O(c)$ is the set of alternatives that are weakly preferred to c in O (including c itself), and $Down_O(c)$ is the set of alternatives that c is weakly preferred to in O (including c itself). If $c \succ_O w$, then $Block_O(c, w)$ is the set of all the alternatives, including c and w , that are ranked between c and w .

Example 4 Let O be the partial order illustrated in Figure 3. We have that $Up_O(c_2) = \{c_1\}$, $Up_O(c_4) = \{c_1, c_2, c_3\}$, $Down_O(c_2) = \{c_2, c_4\}$, $Down_O(c_4) = \emptyset$, and $Block_O(c_2, c_4) = \{c_2, c_3, c_4\}$.

The notion of a block is useful for the following reason. In the algorithm, we want to think about an extension of the partial orders in which w does as well as possible, and c does as poorly as possible. When $c \succ_O w$ in some partial order O , we cannot rank c below w ; but at least it makes sense to have as few alternatives between them as possible. The alternatives in the block are exactly the ones that need to be between them; we will rank the others outside of the block. Then, the question is where to position the block, and we will “slide” the block through the ranking.

Now we are ready to present the algorithms. First we note that computing the Up and $Down$ sets of all alternatives, given a partial order O is equivalent to computing the *transitive closure* of a graph with m vertices, which can be solved by the Floyd-Warshall algorithm (Cormen, Leiserson, Rivest, & Stein, 2001) in time $O(m^3)$.

Algorithm 1 (Computing NW w.r.t. a positional scoring rule)

1. Compute the Up and $Down$ sets for each partial order O .
2. Repeat Steps 3a-c for all $w \neq c$:
- 3a. Let $S(w) = S(c) = 0$.
- 3b. For each partial order O in P ,
 - if $c \not\succeq_O w$, then (following Example 3) the lowest possible position for c is the $m + 1 - |Down_O(c)|$ th position, and the highest possible position for w is the $|Up_O(w)|$ th position, so we add the scores $r(|Up_O(w)|)$ and $r(m + 1 - |Down_O(c)|)$ to $S(w)$ and $S(c)$, respectively;
 - if $c \succ_O w$, then the highest that we can slide O 's $c \succ w$ block (as measured by c 's position, which is at the top of the block) is position $|Up_O(w) \setminus Down_O(c)| + 1$ (if an alternative a is ranked above w in the partial order, then we will place it above c , unless the partial order ranks c above a), and the lowest (as measured by w 's position, which is at the bottom of the block) is position $m - |Down_O(c) \setminus Up_O(w)|$ (if an alternative a is ranked below c in the partial order, then we will place it below w , unless the partial order ranks a above w). Any position between these extremes is also possible. We find the position that minimizes the score of c minus the score of w , then add the scores c and w get for these positions to $S(c)$ and $S(w)$, respectively.
- 3c. If the result is that $S(w) \geq S(c)$, then output that c is not a necessary winner (terminating the algorithm).
4. Output that c is a necessary winner (if we reach this point).

The algorithm for computing NcW is obtained simply by checking whether $S(w) > S(c)$ in Step 4.

Proposition 1 *Algorithm 1 checks whether or not c is a necessary winner for P_O w.r.t. a given positional scoring rule. It runs in time $O(nm^3)$.*

We now move on to the maximin rule. We note that c is not a necessary winner for P_O w.r.t. maximin if and only if there exists a profile of linear orders P extending P_O , and two alternatives w and w' , such that $N(w, d) \geq N(c, w')$ for all alternatives d . Therefore, our algorithm considers all pairs (w, w') , and then checks whether there exists an extension of the input partial order, for which the inequality holds for all alternatives d .

Algorithm 2 (Computing NW w.r.t. maximin)

1. Compute the Up set for each partial order O .
2. Repeat 3a-c for all tuples w, w' , in which c, w, w' are different from each other.
- 3a. Let $S(c, w') = 0$, and for any alternative $d \neq w$, let $S(w, d) = 0$.
- 3b. For each partial order O ,
 - if $c \not\succeq_O w'$, then add $w' \succ c$ to O and raise w as high as possible; for any $d \neq w$, if, in the resulting vote, w is ahead of d (that is, $d \notin Up_O(w)$ and if $c \in Up_O(w)$, $d \notin Up_O(w')$), add 1 to $S(w, d)$.
 - if $c \succ_O w'$, then raise w as high as possible; add 1 to $S(c, w')$; for any $d \neq w$, if, in the resulting vote, w is ahead of d (that is, $d \notin Up_O(w)$), add 1 to $S(w, d)$.
- 3c. Check if for all $d \neq w$, $S(w, d) \geq S(c, w')$; if the answer is yes, then output that c is not a necessary winner (terminating the algorithm).
4. Output that c is a necessary winner.

The algorithm for computing NcW for maximin is similar: the only modification is that in Step 3, we check if for all alternatives $d \neq w$, $S(w, d) > S(c, w')$.

Proposition 2 *Algorithm 2 checks whether or not c is a necessary winner for P_O w.r.t. maximin. It runs in time $O(nm^3)$.*

Now we move on to the Bucklin rule. We note that c is not a necessary winner of P_O w.r.t. Bucklin, if and only if there exists an extension P of P_O and an alternative w , such that either w 's Bucklin score is 1, or there exists $2 \leq k \leq m$, such that w is among the top k for more than $\frac{n}{2}$ votes, and c is among the top $k - 1$ for at most $\frac{n}{2}$ votes. Therefore, like Algorithm 1, the algorithm for Bucklin considers each alternative w , computes the possible positions for the blocks $Block_O(c, w)$, and then checks for all k from 1 to m whether the above condition can be made to hold.

Algorithm 3 (Computing NW w.r.t. Bucklin)

1. Compute the Up and $Down$ sets for each partial order O .
2. Repeat Steps 3a-d for all $w \neq c$:
- 3a. For any $j \leq n$, let $High(j) = Low(j) = Length(j) = 0$. For any $i \leq m$, let $S(i, c) = S(i, w) = U(i) = 0$.
- 3b. For each partial order O_j ,
 - if $c \not\prec_{O_j} w$, then let $Length(j) = 0$, and let $High(j) = |Up_{O_j}(w)|$, $Low(j) = m + 1 - |Down_{O_j}(c)|$;
 - if $c \succ_{O_j} w$, then let $Length(j) = |Block_{O_j}(c, w)|$, $High(j) = |Up_{O_j}(w) \setminus Down_{O_j}(c)| + 1$, $Low(j) = m + 1 - |Down_{O_j}(c)|$.
- 3c. For each $k \leq m$, each $j \leq n$,
 - if $Length(j) = 0$, then add 1 to $S(k, w)$ if $High(j) \leq k$, and add 1 to $S(k - 1, c)$ if $Low(j) \leq k - 1$.
 - If $Length(j) > 0$, then: add 1 to $S(k, w)$ if either $Low(j) + Length(j) - 1 \leq k$, or the following two conditions both hold: $Low(j) \leq k - 1$ and $High(j) + Length(j) - 1 \leq k$. Also, add 1 to $S(k - 1, c)$ if $Low(j) \leq k - 1$, and add 1 to $U(k)$ if $Low(j) > k - 1$ and $High(j) + Length(j) - 1 \leq k$.
- 3d. If $S(1, w) + U(1) > \frac{n}{2}$, or there exists $2 \leq k \leq m$ such that $S(k, w) \geq S(k - 1, c)$, $S(k - 1, c) \leq \frac{n}{2}$, and $S(k, w) + U(k) > \frac{n}{2}$, then output that c is not a necessary winner (terminating the algorithm).
4. Output that c is a necessary winner.

The algorithm for computing NcW is obtained by a slight change of Steps 3d as follows.

- 3d'. If there exists $0 \leq l \leq U(1)$ such that $S(1, w) + l > \frac{n}{2} \geq S(1, c) + l$, or there exists $2 \leq k \leq m$ and $l \leq U(k)$ such that $S(k, w) + l > \frac{n}{2} \geq S(k, c) + l$, then output that c is not a necessary co-winner (terminating the algorithm).

Proposition 3 *Algorithm 3 checks whether or not c is a necessary winner for P_O w.r.t. Bucklin. It runs in time $O(nm^3)$.*

Finally, we consider the possible co-winner problem for plurality with runoff. We will show that this problem can be solved in polynomial time. From this, it also follows that the necessary (unique) winner problem can be solved in polynomial time (by simply checking whether any other alternative is a possible co-winner). In contrast, we have already shown that for plurality with runoff, the possible unique winner problem is NP-complete (Theorem 7) and the necessary co-winner problem is coNP-complete (Theorem 8).

Our algorithm for determining whether c is a possible co-winner is based on the following key observation: c is a possible co-winner for P_O w.r.t. plurality with runoff if and only if there exists an extension of P_O , denoted by P^* , and an alternative $d \neq c$, such that 1. c is preferred to d in at least half of votes (linear orders) in P^* , and 2. c and d can enter the runoff, that is, the plurality scores of c and d are at least as high as the plurality score of any other alternative. That is, let l_1 (l_2) be the plurality score of c (d) and $l_{min} = \min\{l_1, l_2\}$; then, the plurality score of any other alternative c' ($c' \neq c, c' \neq d$) is at most l_{min} .

For any $i^* \leq m - 1$, we let α_{i^*} be the number of partial orders $O \in P_O$ such that $c_{i^*} \succ_O c$. For any $O \in P_O$, we let $Top(O)$ denote the set of alternatives for which there exists at least one extension of O such that that alternative is in the top position. Based on the above observations, we will consider all possibilities for l_1 , l_2 , and d (we will use i^* to denote possibilities for the index of d), and solve a minimum cost flow problem instance for each possibility. Specifically, for any $l_1, l_2 \leq n$ and $i^* \leq m - 1$ (with $\alpha_{i^*} \leq n/2$), we define a minimum cost flow problem F_{l_1, l_2, i^*} as follows (illustrated in Figure 5, in which $i^* = 1$).

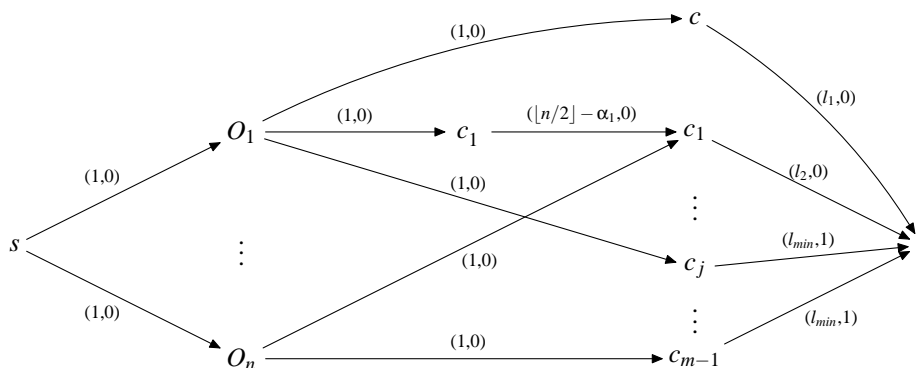


Figure 5: The minimum cost flow problem $F_{l_1, l_2, 1}$.

Vertices: $s, O_1, \dots, O_n, c'_{i^*}, c, c_1, \dots, c_{m-1}, t$.

Edges: we have the following four types of edges.

- **Edges from s to $\{O_1, \dots, O_n\}$:** for any $i \leq n$, there is an edge (s, O_i) with capacity 1 and cost 0.
- **Edges from $\{O_1, \dots, O_n\}$ to $\{c'_{i^*}, c, c_1, \dots, c_{m-1}\}$:** we have
 - * for any $j \leq n$ and any $d \in \mathcal{C}$ such that $d \neq c_{i^*}$, there is an edge (O_j, d) with capacity 1 and cost 0 if $d \in Top(O_j)$;
 - * for any $j \leq n$, there is an edge (O_j, c_{i^*}) with capacity 1 and cost 0 if $c_{i^*} \in Top(O)$ and $c_{i^*} \succ_{O_j} c$;
 - * for any $j \leq n$, there is an edge (O_j, c'_{i^*}) with capacity 1 and cost 0 if $c_{i^*} \in Top(O)$ and $c_{i^*} \not\succeq_{O_j} c$.
- **Edge from c'_{i^*} to c_{i^*} :** there is an edge (c'_{i^*}, c_{i^*}) with capacity $[n/2] - \alpha_{i^*}$ and weight 0.
- **Edges from $\{c, c_1, \dots, c_{m-1}\}$ to t :** we have
 - * an edge (c, t) with capacity l_1 and cost 0;
 - * an edge (c_{i^*}, t) with capacity l_2 and cost 0;
 - * for any $i \leq m - 1$ with $i \neq i^*$, an edge (c_i, t) with capacity l_{min} and cost 1 (we recall that $l_{min} = \min\{l_1, l_2\}$).

Required flow: n .

Next, we prove that c is a PcW for P_O w.r.t. plurality with runoff if and only if there exist $l_1, l_2 \leq n$ and $i^* \leq m - 1$ such that F_{l_1, l_2, i^*} has a solution in which the cost is $n - l_1 - l_2$ (we note that there is never a solution to F_{l_1, l_2, i^*} whose cost is strictly less than $n - l_1 - l_2$).

Because all parameters in F_{l_1, l_2, i^*} are integers, if there is a solution to F_{l_1, l_2, i^*} , then there must also exist an integer solution. First, we show how to convert an integer solution of F_{l_1, l_2, i^*} to a solution of the PcW problem w.r.t. plurality with runoff. Let f be an integer solution to F_{l_1, l_2, i^*} , that is, $f : \text{Vertices} \times \text{Vertices} \rightarrow \mathbb{Z}$. We construct an extension $P^* = (V_1^*, \dots, V_n^*)$ of P_O as follows:

- for any $j \leq n$, if $f(O_j, c_{i^*}^*) = 1$ then we let V_j^* be an extension of O_j in which $c_{i^*}^*$ is ranked in the top position;
- for any $j \leq n$ and any $d \in \mathcal{C}$, if $f(O_j, d) = 1$ then we let O_j^* be an extension of O_j in which d is ranked in the top position, and c is ranked as high as possible.

Because the cost of f is $n - l_1 - l_2$, the plurality score of c is l_1 and the plurality score of $c_{i^*}^*$ is l_2 , while the plurality score of c_i (with $i \neq i^*$) is at most l_{\min} . Therefore, c and $c_{i^*}^*$ enter the runoff together in at least one parallel universe. Now, the capacity constraint on the edge $(c_{i^*}^*, c_{i^*}^*)$ ensures that c will win the runoff: the reason is that if we rank $c_{i^*}^*$ first in a vote in which we could have ranked c ahead of $c_{i^*}^*$, then it will contribute 1 to the flow on this edge. Moreover, the capacity of the edge $(c_{i^*}^*, c_{i^*}^*)$ is $\lfloor n/2 \rfloor - \alpha_{i^*}$, which means that $c_{i^*}^* \succ c$ in at most $\alpha_{i^*} + (\lfloor n/2 \rfloor - \alpha_{i^*}) \leq n/2$ votes of P^* . Hence, c is a co-winner for P^* .

Conversely, if there exists an extension P^* of P such that c is a co-winner of P^* , then there exists a $c_{i^*}^*$ such that in some parallel universe, $\{c, c_{i^*}^*\}$ enter the runoff, and c wins this runoff. Let l_1, l_2 be the plurality scores of $c, c_{i^*}^*$. Then, this extension can be converted to a solution to F_{l_1, l_2, i^*} (we omit the details because they are entirely similar to the details for the other direction).

Therefore, the following algorithm solves PcW w.r.t. plurality with runoff.

Algorithm 4 (Computing PcW w.r.t. plurality with runoff)

1. For any $O \in P_O$, compute $Top(O)$ and $Up_O(c)$. For any $i \leq m - 1$, let $\alpha_i = \#\{O \in P_O : c_i \in Up_O(c)\}$.
2. Repeat Steps 2a-b for all $i \leq m - 1$ and $l_1, l_2 \leq n$:
 - 2a. Construct the min cost flow problem $F_{l_1, l_2, i}$.
 - 2b. Solve $F_{l_1, l_2, i}$ by the double scaling algorithm (Ahuja, Goldberg, Orlin, & Tarjan, 1992). If the minimum cost is $n - l_1 - l_2$, then output that c is a possible co-winner. Terminate the algorithm.
3. Output that c is not a possible co-winner.

The runtime of step 1 is $O(nm^3)$. The runtime of the double scaling algorithm is $O(NM(\log \log U) \log(NC))$, where N is the number of vertices, M is the number of edges, U is the maximum capacity, and C is the maximum cost. In step 2b, $N = O(m + n)$, $M = O(mn)$, $U = O(n)$, and $C = 1$. Therefore, the runtime of step 2b is $O(mn(m + n)(\log \log n) \log(m + n))$. It follows that the runtime of Algorithm 4 is $O(m^2n^3(m + n)(\log \log n) \log(m + n))$.

Proposition 4 *Algorithm 4 checks whether or not c is a possible co-winner for P_O w.r.t. plurality with runoff. It runs in time $O(m^2n^3(m + n)(\log \log n) \log(m + n))$.*

We note that c is a necessary unique winner if and only if no other alternative is a possible co-winner. Therefore, we naturally obtain an algorithm for NW, simply by using Algorithm 4 to check if any alternative other than c is a possible co-winner; this procedure has a runtime of $O(m^3n^3(m+n)(\log \log n) \log(m+n))$.

Proposition 5 *Algorithm 4 can be used to check whether or not c is a necessary unique winner for P_O w.r.t. plurality with runoff, in time $O(m^3n^3(m+n)(\log \log n) \log(m+n))$.*

5. Conclusion

We considered the following problem: given a set of alternatives, a voting rule, and a set of partial orders, which alternatives are possible/necessary winners? That is, which alternatives would win for some/any extension of the partial orders? We considered the case where the votes are not weighted and the number of alternatives is not bounded. Table 1 summarizes our results. These results hold whether or not the alternative must be the unique winner, or merely a co-winner, unless specifically mentioned.

	Possible Winner	Necessary Winner
STV	NP-complete (Bartholdi & Orlin, 1991)	coNP-complete (Bartholdi & Orlin, 1991)
Plurality	P ³	P ³
Veto	P ⁴	P ⁴
Pos. scoring (incl. Borda)	NP-complete ⁵	$O(nm^3)$
Copeland	NP-complete ⁵	coNP-complete ⁵
Maximin	NP-complete ⁵	$O(nm^3)$
Bucklin	NP-complete ⁵	$O(nm^3)$
Ranked pairs	NP-complete ⁵	coNP-complete ⁵
Voting trees (incl. balanced trees)	NP-complete ⁵	coNP-complete ⁵
Plu. w/ runoff	NP-complete (unique winner) $O(m^2n^3(m+n)(\log \log n) \log(m+n))$ (co-winner)	$O(m^3n^3(m+n)(\log \log n) \log(m+n))$ (unique winner) coNP-complete (co-winner) ⁵

Table 1: Summary of complexity of possible/necessary winner problems w.r.t. common voting rules.

In this paper, there was no restriction on the partial orders. However, if the reason that we have partial orders is that preferences are submitted as CP-nets, this introduces additional structure on the partial orders; that is, not all partial orders correspond to a CP-net. Hence, while our positive results would still apply, it is not immediately obvious that our negative results would still apply.

Another approach is to approximate the sets of possible/necessary winners; this has been studied previously for STV (Pini et al., 2007).

3. Easy to prove; also follows from the bribery algorithm by Faliszewski (Faliszewski, 2008).

4. Easy to prove.

5. Hardness results hold even when the number of unknown pairs in each partial order is no more than a constant.

Acknowledgements

We thank Jerome Lang, Toby Walsh, the anonymous reviewers for AAAI-08, and all participants of the Dagstuhl Seminar 07431: *Computational Issues in Social Choice* for helpful discussions and comments. Lirong Xia is supported by a James B. Duke Fellowship and Vincent Conitzer is supported by an Alfred P. Sloan Research Fellowship. This work is supported by NSF under award number IIS-0812113.

Appendix: Proofs

Proof of Theorem 1: (hardness of PW/PcW w.r.t. positional scoring rules)

Given an X3C instance, let $q + 3 \leq l \leq f(q + 3)$ (where q is the number of elements in the X3C instance) satisfy the two conditions in the assumption, and let $k \leq l - 4$ satisfy $\bar{s}_l(k) - \bar{s}_l(k+1) = \bar{s}_l(k+1) - \bar{s}_l(k+2) = \bar{s}_l(k+2) - \bar{s}_l(k+3) > 0$, and $\bar{s}_l(k+3) - \bar{s}_l(k+4) > 0$. We construct the PW instance as follows

Alternatives: $\mathcal{C} = \{c, w, d, v_1, \dots, v_q\} \cup A$, where c, w, d and $A = \{a_1, \dots, a_{l-3-q}\}$ are auxiliary alternatives.

First part (P_1) of the profile: For any S_i , choose any $B_i \subset \mathcal{C} \setminus (S_i \cup \{w, d\})$ with $|B_i| = k - 1$. Let $O(B_i, w, S_i, d, Others)$ be some linear order that agrees with $B_i \succ w \succ S_i \succ d \succ Others$. Let us define

$$O_{S_i} = O(B_i, w, S_i, d, Others) \setminus [\{w\} \times (S_i \cup \{d\})]$$

That is, O_{S_i} is a partial order that agrees with $B_i \succ w \succ S_i \succ d \succ Others$, except that the pairwise relations between (w, S_i) and (w, d) are not determined (and these are the only 4 undetermined relations). Let $P_1 = \{O_{S_1}, \dots, O_{S_t}\}$.

Second part (P_2) of the profile: We give the properties that we need P_2 to satisfy; we omit the details of how to construct P_2 (in polynomial time). We recall that all votes in P_2 are linear orders. Let $P'_1 = \{O(B_i, w, S_i, d, Others) : i \leq t\}$. That is, P'_1 ($|P'_1| = t$) is an extension of P_1 (in fact, these are the linear orders that we started with before removing some of the comparisons). P_2 is a set of linear orders such that the following holds for $Q = P'_1 \cup P_2$:

1. For any $i \leq q$, $\bar{s}_l(Q, c) - \bar{s}_l(Q, v_i) = 2(\bar{s}_l(k) - \bar{s}_l(k+1))$, $\bar{s}_l(Q, w) - \bar{s}_l(Q, c) = \frac{q}{3} \times (\bar{s}_l(k) - \bar{s}_l(k+4)) - \bar{s}_l(k+3) + \bar{s}_l(k+4)$.
2. For any $i \leq q$, the scores of v_i and w, c are higher than those of the other alternatives in any extension of $P_1 \cup P_2$.
3. P_2 's size is polynomial in $t + q$.

Given such a P_2 , c is a possible winner if and only if there exists an extension P_1^* of P_1 such that w is ranked lower than c at least $\frac{q}{3}$ times, in order for the total score of w to be lower than the total score of c . Meanwhile, for any $j \leq q$, v_j should not be ranked higher than w more than once in P_1^* , because otherwise the total score of v_j will be higher than or equal to the total score of c . Given a solution to this, let I be the set of subscripts of votes in P_1^* for which w is ranked lower than c ; then, $S_I = \{S_i : i \in I\}$ is a solution to the X3C instance. Conversely, given a solution to the X3C instance, let I be the set of indices

of S_i that are included in the X3C. Then, a solution to the possible winner instance can be obtained by ranking c ahead of w exactly in the votes with subscripts in I . That is, c is a possible winner if and only if there exists a solution to the X3C problem.

For possible co-winner, we replace 1. by

1'. For any $i \leq q$, $s(Q, c) - s(Q, v_i) = \vec{s}_i(k) - \vec{s}_i(k+1)$, $s(Q, w) - s(Q, c) = \frac{q}{3} \times (\vec{s}_i(k) - \vec{s}_i(k+4))$.

□

Proof of Theorem 2: (hardness of PW/PcW/NW/NcW w.r.t. Copeland)

We first prove the PW and NcW parts, in one reduction. Given any X3C instance, we construct a profile of $8t + q + 1$ alternatives, as follows:

Alternatives: $\{c, w, d\} \cup \mathcal{V} \cup A \cup B$, where $A = \{a_1, \dots, a_{t-2}\}$, $B = \{b_1, \dots, b_{7t}\}$.

First part P_1 of the profile: for each $i \leq t$, we obtain a partial order by starting with $O(\mathcal{V} \setminus S_i, d, S_i, w, c, B^c, A)$ —a linear order that is consistent with $\mathcal{V} \setminus S_i \succ d \succ S_i \succ w \succ c \succ B^c \succ A$ —and then removing the ordering relationships in $(\{d\} \cup S_i) \times \{w, c\}$. (B^c means that the alternatives in B are ranked in a cyclic way. For the first occurrence, the order is $b_1 \succ \dots \succ b_{7t}$; for the second occurrence, the order is $b_{7t} \succ b_1 \succ \dots \succ b_{7t-1}$; and so on. We will also use this notation in all the remaining votes. Notice there are $7t$ votes in total, so that each order appears exactly once.)

Second part P_2 of the profile:

1. $m - \frac{2q}{3} + 1$ votes $w \succ c \succ d \succ \mathcal{V} \succ B^c \succ A$
2. $\frac{q}{3} - 2$ votes $w \succ c \succ d \succ \mathcal{V} \succ B^c \succ A$
3. $\frac{q}{3} - 2$ votes $w \succ d \succ c \succ \mathcal{V} \succ B^c \succ A$
4. 2 votes $c \succ w \succ d \succ \mathcal{V} \succ B^c \succ A$
5. 2 votes $d \succ c \succ \mathcal{V} \succ w \succ B^c \succ A$
6. $\frac{1}{2}(5t - 1)$ votes $w \succ A \succ c \succ B^c \succ \mathcal{V} \succ d$
7. $\frac{1}{2}(5t - 1)$ votes $B^c \succ \mathcal{V} \succ w \succ d \succ A \succ c$

It is easy to check that the number of undetermined pairs in each vote is no more than 8. W.l.o.g., we can always assume t is odd and $t \geq q$ because if not, then we can add $t + 2q + 1$ copies of S_1 and rename them. In this case there is an odd number of sets, the problem size is polynomial in that of the old one, and the new X3C instance has a feasible solution if and only if the old one does.

Let P_1' denote the profile that extends P_1 such that in each vote d and S_i are ranked higher than w and c , namely, $P_1' = \{O(\mathcal{V} \setminus S_i, d, S_i, w, c, B, A) : i \leq t\}$. We have the following observations about each pairwise election:

1. w always defeats c, d, B, A , and $N_{P_1' \cup P_2}(v_i, w) = 3$ for each $i \leq q$.
2. c always defeats \mathcal{V}, B , always loses to A , and $N_{P_1' \cup P_2}(d, c) = \frac{2q}{3} - 1$.

3. B always defeats d, S, A , and due to its cyclic order in the profile, b_j always defeats $b_{j+1}, \dots, b_{j+\frac{1}{2}(7t-1)}$, and always loses to the other alternatives in B .

So in $P'_1 \cup P_2$, the total number of pairwise elections won by each alternative is:

1. w wins $|B| + |A| + 2 = 8t$,
2. c wins $|\mathcal{V}| + |B| = 8t$,
3. d , any $v \in \mathcal{V}$, and any $a \in A$ wins at most $8t + q + 1 - 7t = t + q + 1$, because they all lose to B ,
4. any $b \in B$ wins at most $\frac{1}{2}(|B| - 1) + |A| + |\mathcal{V}| + 1 = \frac{1}{2}(9t + 2q - 3)$ pairwise elections.

We have assumed that $t \geq q$, which means in $P'_1 \cup P_2$, the winners are $\{w, c\}$. In order for c to be the unique winner, the only possibility is for c to win the pairwise election against d by putting $c \succ d$ in at least $\frac{q}{3}$ votes in P_1 . However, when we put c ahead of d in a vote corresponding to S_i , the pairwise score between w and v increases by 2 for all $v \in S_i$. Moreover, if $w \succ v$ for some $v \in \mathcal{V}$ at least twice in an extension P^* of P_1 , then $N_{P^* \cup P_2}(v, w) \leq -1$, which means w defeats v in their pairwise election. In this case, w would win $8m + 1$ pairwise elections, so c cannot be a unique winner. Therefore, c is a possible unique winner if and only if there exists an extension P^* of P_1 such that $c \succ d$ in exactly $\frac{q}{3}$ votes in P^* , and the corresponding S_i do not overlap, that is, they constitute an exact cover of \mathcal{V} . This means that PW has a solution if and only if the X3C problem has a solution. So PW is NP-complete.

It is now easy to see that NcW is coNP-complete, because in the above reduction, w would always be a co-winner if c is not the unique winner. For PcW and NW, we just need to modify the above reduction slightly: let $|A| = t - 1$ and keep the rest unchanged. Then w will initially win $8t + 1$ pairwise elections, and c is a possible co-winner (w is not the necessary unique winner) if and only if there exists a feasible solution to the X3C problem. \square

Proof of Theorem 3: (hardness of PW/PcW w.r.t. Bucklin)

First, we give a reduction from X3C to PW. Given any X3C instance, we construct a profile of $3q + 4$ alternatives as follows:

Alternatives: $W \cup D \cup \mathcal{V} \cup \{c, w\}$, where $W = \{w_1, \dots, w_{q+1}\}$, $D = \{d_1, \dots, d_{q+1}\}$.

First part P_1 of the profile: for each $i \leq t$, we construct a partial order by first taking a linear order consistent with $w_1 \succ \dots \succ w_{q+1} \succ S_i \succ c \succ \mathcal{V} \setminus S_i \succ D$ —we will call this order $O(w_1, \dots, w_{q+1}, S_i, c, \mathcal{V} \setminus S_i, D)$ —and subsequently removing the ordering relations in $\{w_{q-2}, w_{q-1}, w_q, w_{q+1}\} \times (\{c\} \cup S_i)$.

Second part P_2 of the profile:

1. t votes $\mathcal{V} \succ c \succ \text{Others}$,
2. $\frac{q}{3} - 1$ votes $\mathcal{V} \succ w \succ c \succ \text{Others}$,
3. $\frac{q}{3} + 2$ votes $D \succ w_1 \succ \text{Others}$.

It is easy to check that the number of undetermined pairs in each vote is no more than 16. Notice $|P_1 \cup P_2| = 2t + \frac{2q}{3} + 1$, and w_1 is in the first $q + 2$ positions in $t + \frac{q}{3} + 2$ votes. So in order for c to win, $c \succ w_{q-2}$ must hold in at least $\frac{q}{3}$ votes in the extension of P_1 . However, whenever we put c ahead of w_{q-2} in a vote, then we are forced to put the alternatives in the S_i corresponding to that vote in the first q positions. If some $v \in \mathcal{V}$ makes it into the first q positions at least twice in an extension of P_1 , then overall it will be in the first q positions in at least $t + \frac{q}{3} + 1$ votes, which means c will not be the winner.

If there exists a feasible solution to the X3C problem, then we can put c ahead of w_{q-2} in the votes corresponding to this solution, so that we obtain an extension P^* of P_1 such that c is in the first $q + 1$ positions in $\frac{q}{3}$ votes, while for any $v \in \mathcal{V}$, v is in the first q (and, in fact, the first $q + 1$) positions just once. As a result, c is the unique winner of the profile $P^* \cup P_2$, because no other alternative is in the first $q + 1$ positions in at least $t + \frac{q}{3}$ votes. Conversely, if c is the unique winner in some profile $P^* \cup P_2$, then P^* will correspond to a feasible solution to the X3C problem. Therefore, PW w.r.t. Bucklin is NP-complete.

For PcW, we just need to modify the reduction slightly, by changing the last $\frac{q}{3} + 1$ votes from $D \succ w_1 \succ \text{others}$ to $d_1 \succ \dots \succ d_q \succ w_1 \succ \text{others}$. In this case, the Bucklin score of w_1 is $q + 1$, which means c can at best hope to be a co-winner. As a result, PcW is also NP-complete. \square

Proof of Theorem 4: (hardness of PW/PcW w.r.t. maximin)

We first consider the case of PW. For any given X3C instance, we construct a profile over $q + 3$ alternatives as follows:

Alternatives: $\mathcal{V} \cup \{c, w, w'\}$.

First part P_1 of the profile: for each $i \leq t$, we obtain a partial order by starting with a linear order consistent with $w \succ S_i \succ c \succ \mathcal{V} \setminus S_i \succ w'$ —we will refer to this linear order as $O(w, S_i, c, \mathcal{V} \setminus S_i, w')$ —and subsequently we obtain a partial order by removing the orderings in $\{w\} \times (\{c\} \cup S_i)$.

Second part P_2 of the profile: according to Lemma 1, a set of votes such that the pairwise score differences of $\{O(w, S_i, c, \mathcal{V} \setminus S_i, w') : i \leq t\} \cup P_2$ satisfy:

1. $D(w, c) = t + \frac{2q}{3} - 2$, $D(w, v_i) = t + 2$ for all $i \leq q$, $D(w', w) = D(v_1, w') = t + 4$,
 $D(w', c) = t - 2$.
2. $D(l, r) \leq 1$ for all other pairwise scores not defined above.

It is easy to check that the number of undetermined pairs in each vote is no more than 4. Lemma 2 implies that $|P_2| \leq (t + \frac{q}{3} - 1) \times (q + 3)^2$, which is polynomial in $(q + t)$.

We note that the minimum pairwise score difference of w is $D(w, w') = -t - 4$; the minimum pairwise score difference of w' is also $-t - 4 = D(w', v_1)$. Therefore, the only way for c to win is to decrease $D(w, c)$ by raising c higher than w in at least $\frac{q}{3}$ votes in an extension of P_1 . However, each time that we decrease $D(w, c)$ by 2 due to adding $c \succ w$ to $O_{S_i} \in P_1$, for any $v \in S_i$, $D(w, v)$ is also decreased by two. Notice that because

$D(w', c) = t - 2$, decreasing $D(w, c)$ to less than $t - 2$ would not raise the minimum pairwise score difference of c . But if $D(w, v_i)$ is decreased by 4 or more for some $i \leq q$, then the minimum pairwise score of v_j is at least $-t + 2$, which means that in this case c cannot be the winner. So, if there exists a profile P^* extending P_1 such that c wins in $P^* \cup P_2$, then the sets S_i in the votes in P^* such that $c \succ w$ cannot overlap, and because there must be at least $q/3$ of these votes, the corresponding subsets S_i constitute a feasible solution to the X3C problem. Conversely, for each feasible solution of the X3C problem instance, we can find a P^* extending P_1 such that c is the unique winner of the profile $P^* \cup P_2$ w.r.t. the maximin rule. Therefore PW is NP-complete.

For PcW, we just need to modify the above reduction slightly: we replace the condition $D(w, v_i) = t + 2$ by $D(w, v_i) = t$ when constructing P_2 . In this case, if some v_i is covered at least twice, then its minimum pairwise score will be at least $-t + 4$, which means that c cannot be a co-winner. If there exists an exact cover by 3-sets, then there is an extension such that the maximin pairwise score is $-t + 2$, which is reached by c and all alternatives in \mathcal{V} . Therefore PcW is NP-complete. \square

Proof of Theorem 5: (hardness of PW/PcW w.r.t. ranked pairs)

We first prove the hardness of PW and NcW in one reduction. We reduce an arbitrary X3C instance to the following instance.

Alternatives: $\mathcal{V} \cup \{c, a, b\}$.

First part P_1 of the profile: for each $i \leq t$, we obtain a partial order by starting with a linear order consistent with $a \succ c \succ S_i \succ b \succ Others$ —we will refer to this linear order as $O(a, c, S_i, b, Others)$ —and subsequently we obtain a partial order by removing the orderings in $(\{a, c\} \times (S_i \cup \{b\}))$. The partial order obtained in this way is denoted by O_i .

Second part P_2 of the profile: according to Lemma 1,³ a set of votes such that the pairwise score differences of $\{O(a, c, S_i, b, Others) : i \leq t\} \cup P_2$ satisfy:

1. For all $i \leq q$, $D(c, b) = D(w, a) = D(w, v_i) = 3t + \frac{2q}{3}$.
2. $D(a, c) = t + \frac{2q}{3}$, $D(c, w) = t + \frac{2q}{3} - 2$, $D(v_i, c) = t + \frac{2q}{3} - 6$, $D(b, a) = t + 2$.
3. $D(l, r) = 0$ in all other cases.

It is easy to check that the number of undetermined pairs in each vote is no more than 8. By Lemma 2, there exists a profile P_2 satisfying the above conditions and $|P_2| \leq (2t + \frac{q}{3}) \times (q + 4)^2$, polynomial in $(q + t)$.

In the output (a linear order over \mathcal{C}) of any extension of $P_1 \cup P_2$, we must have that $c \succ b$, $w \succ a$, and $w \succ v_i$ (for any $i \leq q$). We note that the only way for c to be the unique winner is to lock $b \succ a$ before $a \succ c$. That is, $D(b, a)$ must be at least $t + 2 + \frac{2q}{3}$. However, whenever we let $b \succ a$ in an extension O_i , we are forcing $S_i \succ c$. Let P'_1 be an extension of P_1 such that c is the unique winner for the profile $P'_1 \cup P_2$ (or, equivalently, such that w is not a co-winner for the profile $P'_1 \cup P_2$). We note that if there exists $i \leq q$ such that $v_i \succ c$

3. We can assume without loss of generality that t is an even number, so that the lemma can be applied.

in at least two votes in P'_1 , then $D(v_i, c) \geq t + \frac{2q}{3} - 6 + 4 = t + \frac{2q}{3} - 2 = D(c, w)$, which means that w is a co-winner (by locking $v_i \succ c$ before $c \succ w$). Therefore, in P'_1 , we must have that $b \succ a$ in exactly $\frac{q}{3}$ votes of P'_1 , and for all $i \leq q$, $v_i \succ c$ in exactly one vote of P'_1 . This naturally corresponds to a solution to the X3C instance.

On the other hand, if the X3C problem instance has a solution $\{S_{i_1}, \dots, S_{i_{q/3}}\}$, then let P'_1 be the profile obtained from P_1 by letting $b \succ a$ in O_{i_j} , and letting all other votes be $a \succ c \succ S_i \succ b \succ \text{Others}$ (where $i \neq i_j$ for any $j \leq q/3$). It follows that c is the unique winner under this profile (and hence, w is not a co-winner). Therefore, PW is NP-complete and NcW is coNP-complete under ranked pairs.

For PcW and NW, we slightly modify the above reduction by letting $D(b, a) = t$ and $D(v_i, c) = t + \frac{2q}{3} - 4$. □

Proof of Theorem 6: (hardness of PW/PcW/NW/NcW w.r.t. voting trees)

Let j_2, j_3, \dots be the index of the voting trees such that for any $z \in \mathbb{N}$ ($z \geq 2$), T_{j_z} is $2(z+1)$ -well-spread, c is a rich leaf, and $j_z \leq f(2(z+1))$.

It is easy to verify that PW and PcW are in NP, and that NW and NcW are in coNP. We next prove the hardness results in a single reduction. Given an X3C instance $\{c_1, \dots, c_q\}, S_1, \dots, S_t$, we construct the PW instance as follows.

Alternatives: Let \mathcal{C} be the leaves of T_{j_q} , where $\mathcal{C} = \{c, d, w\} \cup \mathcal{V} \cup A \cup E$, and $A = \{a_1, \dots, a_q\}$, $E = \{e_1, \dots, e_{m_q-2q-3}\}$, and m_q is the number of leaves in T_{j_q} . Let the tree be such that $\{c, d\} \cup \mathcal{V} \cup A$ are rich leaves in a subtree whose root is a child of the root of T_{j_q} (because T_{j_q} is $2(q+1)$ -well-spread, this is always possible); d is the sibling of c ; the only common ancestor of c and w is the root; and for any $1 \leq i \leq q$, v_i and a_i are each other's sibling. The positions of $\{c, d, w\} \cup \mathcal{V} \cup A$ are illustrated in Figure 6. E is the set of all other alternatives in T_{j_q} . For any $i \leq t$, if $S_i = \{v_{l(i,1)}, v_{l(i,2)}, v_{l(i,3)}\}$, then we let $A_i = \{a_{l(i,1)}, a_{l(i,2)}, a_{l(i,3)}\}$ —that is, A_i consists of the siblings of the subset S_i from the X3C problem.

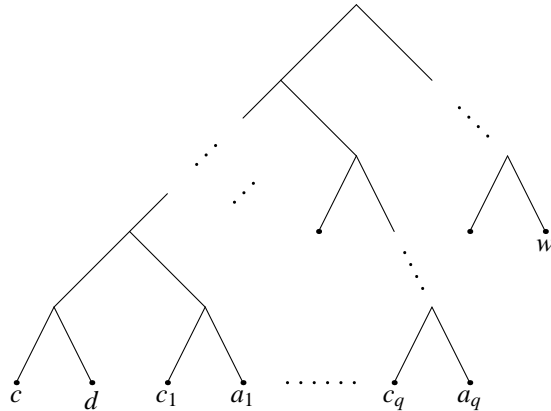


Figure 6: Positions of the alternatives in T_{j_q} .

First part (P_1) of the profile: for each $i \leq t$, we take a linear order that is consistent with $d \succ A_i \succ S_i \succ c \succ Others$ —that is, $O(d, A_i, S_i, c, Others)$ —and subsequently turn it into a partial order by removing $(\{d\} \cup A_i) \times (S_i \cup \{c\})$.

Second part P_2 of the profile: according to Lemma 1, a set of votes (linear orders) such that the pairwise score differences of $\{O(d, A_i, S_i, c, Others) : i \leq t\} \cup P_2$ satisfy:

1. $D(c, d) = -2q/3 + 1$, $D(c, w) = 2q + 1$.
2. For any $i \leq q$, $D(a_i, v_i) = 3$, $D(v_i, c) = D(c, a_i) = D(c, w) = 2q + 1$.
3. For any $c' \in \mathcal{X}$ (with $c' \neq c$), $D(w, c') = 2q + 1$.
4. For any $i, i' \leq q$ (with $i \neq i'$), $D(v_i, a_{i'}) = 2q + 1$.
5. For any $x \in \mathcal{C} \setminus E, e \in E$, $D(x, e) = 2q + 1$.

We note that the only way for c to win is to beat d in the first round, and not to meet any of $\{v_1, \dots, v_q\}$ in later rounds, which can only happen if every v_i is beaten by the corresponding a_i in the first round. It follows that in an extension of P_1 that makes c win, c must be ranked higher than d at least $q/3$ times. However, if we rank c higher than d in such a vote, then all the alternatives in that vote's S_i must be ranked ahead of all the alternatives in that vote's A_i . In order for every a_i to defeat the corresponding v_i , for any $i \leq q$, v_i can be ranked higher than a_i at most once in the extension of P_1 . So, if there exists a profile P^* extending P_1 such that c wins in $P^* \cup P_2$, then the sets S_i in the votes in P^* such that $c \succ d$ constitute a feasible solution to the X3C problem instance, and, conversely, for each feasible solution to the X3C problem instance, we can find a P^* extending P_1 such that c is the unique winner of the profile $P^* \cup P_2$ w.r.t. T_{i_j} . Therefore, PW and PcW are NP-complete.

We also note that if c is not the unique winner, then w is always the unique winner. Therefore, NW and NcW are coNP-complete. \square

Proof of Theorem 7: (hardness of PW w.r.t. plurality with runoff)

Membership in NP is easy to check. We now prove NP-hardness by showing a reduction from an arbitrary X3C instance. For any X3C instance, we construct a PW instance as follows.

Alternatives: $\{c, d, e\} \cup \mathcal{S}_V \cup E$, where $\mathcal{S}_V = \{s_1, \dots, s_t\}$ and $E = \{e_1, \dots, e_{(q+4)^2(t+4)^4}\}$.

First part P_1 of the profile: $P_1 = P_1^1 \cup P_1^2$, where P_1^1 and P_1^2 are defined as follows.

- P_1^1 : for each $i \leq q$, we start from a linear order $d \succ \mathcal{S}_V \succ c \succ Others$, and subsequently obtain a partial order O_i by removing $\mathcal{C} \times \{s_j : c_i \in S_j\}$. That is, we remove a minimum set of constraints such that any alternative in $\{s_j : c_i \in S_j\}$ can be ranked in the top position in at least one extension of O_i . Let $P_1^1 = \{O_i : i \leq q\}$.
- P_1^2 : for each $j \leq t$, we starts from a linear order $d \succ e \succ c \succ Others$, and subsequently obtain a partial order Q_j^1 by removing $[\{d\} \times \{e\}] \cup [\mathcal{C} \times \{s_j\}]$. That is, in any extension of Q_j^1 , only d, e , and s_j can be ranked in the top position. We let $Q_j^2 = Q_j^1$, and $P_1^2 = \{Q_j^1 : j \leq t\} \cup \{Q_j^2 : j \leq t\}$.

Second part P_2 of the profile: $P_2 = P_2^1 \cup P_2^2$, where P_2^1 and P_2^2 are defined as follows.

- P_2^1 : a set of $q(t + 7/3) + 8$ votes, in which c is ranked in the top position $q + 4$ times, d is ranked in the top position $q + 2$ times, e is ranked in the top position $q/3 + 2$ times, and for any $j \leq t$, s_j is ranked in the top position q times.
- P_2^2 : we first obtain, according to Lemma 1, a profile \hat{P}_2^2 such that the pairwise score differences of $\{q \text{ copies of } O(d, \mathcal{S}_V, c, \text{Others})\} \cup \{2t \text{ copies of } O(d, e, c, \text{Others})\} \cup P_2^1 \cup \hat{P}_2^2$ satisfy the following conditions.
 1. $D(d, c) = D(e, c) = 1$;
 2. for all $j \leq t$, $D(c, s_j) = 1$.

From Lemma 1, we have that $|\hat{P}_2^2| \leq (q + 4)^2(t + 4)^4$. Next, we obtain P_2^2 from \hat{P}_2^2 by raising an alternative in E to the top position in each vote, in such a way that no two votes in P_2^2 rank the same alternative in the top position.

We make the following observations about the profile $\{q \text{ copies of } O(d, \mathcal{S}_V, c, \text{Others})\} \cup \{2t \text{ copies of } O(d, e, c, \text{Others})\} \cup P_2^1 \cup \hat{P}_2^2$:

- $D(d, c) = D(e, c) = 1$, and for all $j \leq t$, $D(c, s_j) = 1$;
- $Plu(c) = q + 4$, $Plu(d) = 2t + 2q + 2$, $Plu(e) = q/3 + 2$; for any $j \leq t$, $Plu(s_j) = q$; for any $e' \in E$, $Plu(e') \leq 1$.

We also note that in any extension of $P_1 \cup P_2$, $Plu(c) = q + 4$.

If the X3C instance has a solution $S_{j_1}, \dots, S_{j_{q/3}}$, then we construct a solution to the PW instance as follows.

- For any $i \leq q$, let $V_i = [s_{j_l} \succ d \succ \mathcal{S}_V \setminus \{s_{j_l}\} \succ c \succ \text{Others}]$, where j_l is such that $c_i \in S_{j_l}$; we note that V_i extends O_i ;
- for any $l \leq q/3$, let $V_{j_l}^1 = V_{j_l}^2 = [e \succ d \succ c \succ \text{Others}]$; we note that $V_{j_l}^1$ ($V_{j_l}^2$) extends $Q_{j_l}^1$ ($Q_{j_l}^2$);
- for any $j \leq t$ (with $j \neq j_l$ for any $l \leq q/3$), let $V_j^1 = V_j^2 = [s_j \succ d \succ e \succ c \succ \text{Others}]$; we note that V_j^1 (V_j^2) extends Q_j^1 (Q_j^2);
- then, we use these votes to extend the partial orders in P_1 : let $P_1^* = \{V_i : i \leq q\} \cup \{V_j^1, V_j^2 : j \leq t\}$.

In $P_1^* \cup P_2$, we have $Plu(c) = q + 4$, $Plu(d) = Plu(e) = q + 2$; for any $l \leq q/3$, $Plu(s_{j_l}) = q + 3$; for any $j \neq j_l$ ($l = 1, \dots, q/3$), $Plu(s_j) = q + 2$; and for any $e' \in E$, $Plu(e') \leq 1$. Also, we have that for any $l \leq q/3$, $D(c, s_{j_l}) = 1$. It follows that the pairs that enter the runoff (in some parallel universe) are $(c, s_{j_1}), \dots, (c, s_{j_{q/3}})$, and c wins each of these pairwise elections. Therefore, c is the unique winner for $P_1^* \cup P_2$.

Next, we show how to convert a solution to the PW instance to a solution to the X3C instance. Let $P_1^* = P_1^{1*} \cup P_1^{2*}$ be an extension of P_1 such that c is the unique winner for $P_1^* \cup P_2$, where $P_1^{1*} = \{V_i : i \leq q\}$ extends P_1^1 , and $P_1^{2*} = \{V_j^1 : j \leq t\} \cup \{V_j^2 : j \leq t\}$ extends P_1^2 . We make the following sequence of claims.

Claim 1: Neither d nor e can enter the runoff, which means that the only pairs that could potentially still enter the runoff are the (c, s_j) , for some $j \leq t$.

Proof: If d or e entered the runoff in some parallel universe, then it would defeat c in the runoff (unless c is not even in the runoff, in which case c also does not win in this parallel universe), contradicting that c is the unique winner. \square

Claim 2: For any $j \leq t$, $Plu_{P_1^*}(s_j) \leq 3$.

Proof: If this does not hold, then we let j^* be an index of s such that $Plu_{P_1^*}(s_{j^*})$ is maximized. It follows that $Plu_{P_1^*}(s_{j^*}) \geq 1$, because $Plu_{P_1^*}(s_{j^*}) \leq 3$. However, by putting s_{j^*} in the top position in a partial order in P_1^2 , we are forcing $D(c, s_{j^*})$ to be reduced by 2, which means that s_{j^*} beats c in their pairwise election. Moreover, because, by Claim 1, one of the s_j must enter the runoff, and because s_{j^*} has the maximum plurality score among the s_j alternatives, in one of the parallel universes, s_{j^*} must be in the runoff. Hence, c cannot win in this parallel universe, contradicting that c is the unique winner. \square

Claim 3: $Plu_{P_1^*}(d) = 0$, $Plu_{P_1^*}(e) \leq 2q/3$.

Proof: It follows from Claim 2 that for any $j \leq t$, $Plu_{P_1^* \cup P_2}(s_j) \leq q + 3$. Therefore, from Claim 1 we must have that $Plu_{P_1^* \cup P_2}(d) \leq q + 2$ and $Plu_{P_1^* \cup P_2}(e) \leq q + 2$. The claim follows. \square

Claim 4: For any $j \leq t$, if $Plu_{P_1^2}(s_j) \geq 1$, then $Plu_{P_1^*}(s_j) \leq 2$.

Proof: This follows from the proof for Claim 2: if $Plu_{P_1^2}(s_j) \geq 1$, and s_j enters the runoff in some parallel universe, then c cannot win in that parallel universe. Therefore, s_j cannot be in the runoff; but because, by Claim 2, for any j' , $Plu_{P_1^*}(s_{j'}) \leq 3$, and by Claim 1, one of the $s_{j'}$ must be in the runoff, it follows that we must have that $Plu_{P_1^*}(s_j) \leq 2$. \square

Claim 5: Let $X_1 = \{s_j : Plu_{P_1^*}(s_j) > 0, Plu_{P_1^2}(s_j) = 0\}$, and $X_2 = \{s_j : Plu_{P_1^*}(s_j) = 0, Plu_{P_1^2}(s_j) > 0\}$. We have $X_1 \cup X_2 = \mathcal{S}_V$ and $|X_1| = q/3$.

Proof: Let $x_1 = |X_1|$, $x_2 = |X_2|$, and $x_3 = t - x_1 - x_2$. We recall that for any $O \in P_1^1$, the top-ranked alternative of any extension of O must be either d or an element in \mathcal{S}_V ; for any $Q \in P_1^2$, the top-ranked alternative of any extension of Q must be either e or an element in \mathcal{S}_V . We now use this to obtain two inequalities.

First, in order for c to be the unique winner, d cannot be in the top position in any vote in P_1^1 . Therefore, all q of the top positions in P_1^1 must be taken by alternatives in \mathcal{S}_V . A given alternative in X_1 can take at most 3 of these top positions; alternatives in X_2 can take none of these top positions; and a given alternative in $\mathcal{S}_V \setminus (X_1 \cup X_2)$ can take at most 1 of these top positions, by Claim 4. It follows that $3x_1 + x_3 \geq q$.

Now, we apply a similar analysis to P_1^2 . In order for c to be the unique winner, e cannot be in the top position in more than $2q/3$ of the votes in P_1^2 , leaving at least $2t - 2q/3$ top positions to be filled. Alternatives in X_1 can take none of these top positions; a given alternative in X_2 can take at most 2 of these top positions, by Claim 4; and a given alternative in $\mathcal{S}_V \setminus (X_1 \cup X_2)$ can take at most 1 of these top positions, by Claim 4. It follows that $2x_2 + x_3 \geq 2t - 2q/3$.

By substituting the q in the second inequality by the q in the first inequality, we obtain $2x_1 + 2x_2 + \frac{5}{3}x_3 \geq 2t$. But, we recall that $x_1 + x_2 + x_3 = t$. Therefore, $x_3 = 0$, $x_1 + x_2 = t$. Moreover, $x_1 = q/3$ then follows from the first inequality. \square

Based on all these claims, we can now construct a solution to the X3C instance. Let $X_1 = \{s_{j_1}, \dots, s_{j_{q/3}}\}$. From Claim 2, Claim 5, and the fact that every top position in P_1^1

must be occupied by one of the alternatives in X_1 , it follows that $S_{j_1}, \dots, S_{j_{q/3}}$ is a solution to the X3C instance. Therefore, PW w.r.t. plurality with runoff is NP-complete to compute. \square

Proof of Theorem 8: (hardness of NcW w.r.t. plurality with runoff)

Membership in coNP is easy to check. We now prove coNP-hardness by showing a reduction from an arbitrary X3C instance. For any X3C instance, we construct a NcW instance as follows. For any profile P , and any alternative c' , we let $Plu_P(c')$ denote the plurality score of c' in P , that is, the number of times that c' is ranked in the top position in the votes of P .

Alternatives: $\{c, d\} \cup \mathcal{V} \cup E$, where $E = \{e_1, \dots, e_{t(q+2)^3}\}$.

First part P_1 of the profile: for each $i \leq t$, we obtain a partial order by starting with a linear order consistent with $d \succ S_i \succ c \succ Others$, and subsequently we obtain a partial order by removing the orderings in $(\{d\} \cup S_i) \times \{c\}$. The partial order obtained in this way is denoted by O_i .

Second part P_2 of the profile: $P_2 = P_2^1 \cup P_2^2$, where P_2^1 and P_2^2 are defined as follows.

- P_2^1 : a set of $t(q+1) + q/3$ votes, such that c is ranked in the top position $t+1$ times; d is ranked in the top position $q/3 - 1$ times; and for any $i \leq q$, v_i is ranked in the top position t times.
- P_2^2 : we first obtain, according to Lemma 1, a profile \hat{P}_2^2 such that the pairwise score differences of $\{O(d, S_j, c, Others) : j \leq t\} \cup P_2^1 \cup \hat{P}_2^2$ satisfy the following conditions.
 1. $D(c, d) = 2t + 1$;
 2. for all $i \leq q$, $D(v_i, c) = 3$.

From Lemma 1, we have that $|\hat{P}_2^2| \leq t(q+2)^3$. Next, we obtain P_2^2 from \hat{P}_2^2 by raising an alternative in E to the top position in each vote, in such a way that no two votes in P_2^2 rank the same alternative in the top position.

We make the following observations about $\{O(d, S_i, c, Others) : i \leq t\} \cup P_2$:

- $D(c, d) = 2t + 1$, and for all $i \leq q$, $D(v_i, c) = 3$;
- $Plu(c) = t + 1$, $Plu(d) = t - 1 + q/3$; for any $i \leq q$, $Plu(v_i) = t$; for any $e \in E$, $Plu(e) \leq 1$.

It follows from the above observation that in any extension of $P_1 \cup P_2$, c must enter the runoff; also, in any extension, c beats d in the pairwise election. Let $P_1^* \cup P_2$ (where P_1^* is an extension of P_1) be a profile in which c is not a co-winner. We must have that d does not enter the runoff, which means that $Plu_{P_1^* \cup P_2}(d) \leq t - 1$. It follows that $c \succ d$ in at least $q/3$ votes in P_1^* . However, by ranking $c \succ d$ in a partial order O_i , we are forcing $c \succ S_i$. Then, the pairs of alternatives that enter the runoff (in parallel universes) are $(c, v_1), \dots, (c, v_q)$. Since c loses in any of these pairwise elections in the runoff (because, by assumption, c is not a co-winner), we must have, for any v_j , that $c \succ v_j$ in at most one vote in P_1^* . Hence,

a solution to the NcW instance naturally corresponds to a solution to the X3C instance. Conversely, it is easy to see that any solution to the X3C instance corresponds to a solution to the NcW instance. This proves the hardness of NcW w.r.t. plurality with runoff. \square

Proof of Proposition 1: (algorithm for NW/NcW w.r.t. positional scoring rules)

To check if there exists an extension P of P_O such that $s(P, w) \geq s(P, c)$, for any $O \in P_O$, we maximize $s(V_O, w) - s(V_O, c)$ over all extensions V_O of O .

We note that for any extension V_O of O , $s(V_O, w) \leq r(|Up_O(w)|)$ (because w cannot be ranked in a position higher than the $|Up_O(w)|$ th position) and $s(V_O, c) \geq r(m + 1 - |Down_O(c)|)$ (because c cannot be ranked in a position lower than the $(m+1-|Down_O(c)|)$ th position). These two bounds can be achieved if $c \not\prec_O w$: for any $d \in \mathcal{C} \setminus (\{w\} \cup Up_O(w))$, we add $w \succ d$ to O ; and for any $d \in \mathcal{C} \setminus (\{c\} \cup Down_O(c))$, we add $d \succ c$ to O . We obtain a partial order O' this way, and we let V_O be an (arbitrary) linear extension of O' . It follows that $s(V_O, w) - s(V_O, c) = r(|Up_O(w)|) - r(m + 1 - |Down_O(c)|)$.

However, if $c \succ_O w$, there may not exist V_O in which $s(V_O, w) = r(|Up_O(w)|)$ and $s(V_O, c) = r(m + 1 - |Down_O(c)|)$ hold simultaneously. We note that in any V_O^* that maximizes $s(V_O, w) - s(V_O, c)$, the only alternatives between c and w must be those in $Block_O(c, w)$. Therefore, for any $d \succ_O w$ and $c \not\prec_O d$, we must have that $d \succ_{V_O^*} c$; and for any $c \succ_O d$ and $d \not\prec_O w$, we must have that $w \succ_{V_O^*} d$. It follows that $s(V_O, w) - s(V_O, c) \leq \max_l (r(l + |Block_O(c, w)| - 1) - r(l))$, where l ranges between $|Up_O(w) \setminus Down_O(c)| + 1$ and $m - |Down_O(c) \setminus Up_O(w)|$. Let V'_O be an extension of O restricted to $\mathcal{C} \setminus Block_O(c, w)$ in which $Up_O(w) \setminus Down_O(c)$ is ranked at the top and $Down_O(c) \setminus Up_O(w)$ is ranked at the bottom. For any $d \in \mathcal{C} \setminus (Up_O(w) \cup Down_O(c))$ and any $d' \in Block_O(c, w)$, we must have $d \not\prec_O d'$ and $d' \not\prec_O d$. Therefore, for any $|Up_O(w) \setminus Down_O(c)| + 1 \leq l \leq m - |Down_O(c) \setminus Up_O(w)|$, we can put $Block_O(c, w)$ between the $(l - 1)$ th position and l th position in V'_O , to obtain a linear order that extends O .

This proves the correctness of Step 3b, which computes $\max_{V_O} (s(V_O, w) - s(V_O, c))$. It follows that the algorithm correctly checks whether or not c is a possible winner.

The complexity of computing the Up set and $Down$ set of any alternative is $O(m^3)$, so the time complexity for Step 1 is $O(nm^3)$. The time complexity for Step 3b is nm . Hence, Algorithm 1 runs in time $O(nm^3)$. \square

Proof of Proposition 2: (algorithm for NW/NcW w.r.t. maximin)

The function $S(x, y)$ computed in the algorithm is the number of times x is preferred to y in an extension of P_O . For any partial order O , we let V_O be the extension computed in Step 3b. Let $g(V, d) = D_V(w, d) - D_V(c, w')$. We next prove that for any $d \neq w$ and any extension V'_O of O , $g(V_O, d) \geq g(V'_O, d)$. If $c \not\prec_O w'$ and $c \succ_{V'_O} w'$, then $g(V'_O, d) \leq 0 \leq g(V_O, d)$ (because $D_{V'_O}(c, w') = -2$). If $c \not\prec_O w'$ and $w' \succ_{V'_O} c$, then $D_{V'_O}(c, w') = D_{V_O}(c, w')$. We note that V_O is obtained by raising w as high as possible in O while $w' \succ c$, which means that $D_{V'_O}(w, d) \leq D_{V_O}(w, d)$. It follows that $g(V_O, d) \geq g(V'_O, d)$. Similarly, if $c \succ_O w'$ we also have that $D_{V'_O}(w, d) \leq D_{V_O}(w, d)$.

Therefore, for any extension P of P_O and any $d \neq w$, $2(S(w, d) - S(c, w')) = D_P(w, d) - D_P(c, w') \leq \sum_{O \in P_O} g(V_O, d)$, and when P is the profile computed in Step 3b, the inequality becomes an equality. It follows that the algorithm is correct.

The time complexity of Step 1 is $O(nm^3)$; the time complexity of Step 3b is $O(nm)$; the time complexity of Step 3c is $O(m)$. Therefore, the algorithm runs in time $O(nm^3)$. \square

Proof of Proposition 3: (algorithm for NW/NcW w.r.t. Bucklin)

Similarly to the case of positional scoring rules, for Bucklin, if $c \not\succeq_O w$, then we can simply rank c as low as possible and w as high as possible; on the other hand, if $c \succ_O w$, then we can without loss of generality place as few alternatives between c and w as possible, but the question is where to place this block. The algorithm will consider a particular k , and try to make it so that w is among the top k for more than half the votes, and c is among the top $k - 1$ for at most half the votes. For a particular vote with $c \succ_O w$, depending on where the block is placed, either (1) c is among the top $k - 1$ and w is among the top k ; or, (2) c is among the top $k - 1$ and w is not among the top k ; or, (3) c is not among the top $k - 1$ and w is not among the top k . However, not all three of these possibilities may exist for a particular vote. The algorithm will never choose (2) unless that is the only option, so that the only difficult case is when a decision must be made between (1) and (3).

In the algorithm, if $c \not\succeq_{O_j} w$, then $High(j)$ ($Low(j)$) is the highest (lowest) position that w (c) reaches in an extension of O_j . If $c \succ_{O_j} w$, then $High(j)$ ($Low(j)$) is the highest (lowest) position of c given that c and w are ranked as close to each other as possible, and $Length(j)$ is the size of $Block_{O_j}(c, w)$. For any $i \leq m$, $d \in \{c, w\}$, $S(i, d)$ is the minimum number of times that d is ranked in the top i positions, where the minimum is taken over all extensions of P_O that are consistent with the observations in the previous paragraph (specifically, option (2) is never chosen unless there is no other choice). $U(k)$ is the number of partial orders for which there exists an extension in which c is in the top $k - 1$ positions and w is in the top k positions, as well as an extension in which c is not in the top $k - 1$ positions and w is not in the top k positions (that is, we have a choice between (1) and (3)).

For any $k \leq m$, and any $j \leq n$, we consider how to extend O_j .

- If $c \not\succeq_{O_j} w$, then the positions of c and w are already determined by our previous observations (w is ranked as high as possible and c is ranked as low as possible).
- If $c \succ_{O_j} w$ and $High(j) \leq k$, then c cannot be ranked in the top $k - 1$ positions and w cannot be ranked in the top k positions; therefore, we add 0 to $S(k - 1, c)$ and $S(k, w)$.
- If $c \succ_{O_j} w$, $High(j) < k$ and $Low(j) + Length(j) - 1 > k$, then c can be ranked in the top $k - 1$ positions, but w cannot be ranked in the top k positions. There are two sub-cases: 1. if $Low(j) \geq k$, then we rank c in the $Low(j)$ th position, and henceforth add 0 to both $S(k - 1, c)$ and $S(k, w)$; 2. if $Low(j) < k$, then c is inevitably ranked in the top $k - 1$ positions, and w cannot be ranked in the top k positions, which means that we add 1 to $S(k - 1, c)$ and we add 0 to $S(k, w)$.
- The final case is where $c \succ_{O_j} w$, $High(j) < k$ and $Low(j) + Length(j) - 1 \leq k$. Again, there are two subcases: 1. if $Low(j) < k$, then it means that c must be ranked in the top $k - 1$ positions. Therefore we rank w in the top k positions, and add 1 to both $S(k - 1, c)$ and $S(k, w)$; 2. if $Low(j) \geq k$, then it means that we have three options for an extension of O_j , corresponding to the cases (1), (2), (3) above:

- (1) c 's position is in the top $k - 1$ and w 's position is in the top k .
- (2) c 's position is in the top $k - 1$ and w 's position is not in the top k (if $Length(i) > 2$).

(3) c 's position is not in the top $k - 1$ and w 's position is not in the top k .

As we already discussed, option (2) is suboptimal. So, we add 0 to both $S(k - 1, c)$ and $S(k, w)$, and add 1 to $U(k)$.

The only remaining decision is for how many of the votes corresponding to the number $U(k)$ to choose option (1) (as opposed to option (3)). This corresponds to Step 3d of the algorithm, where it checks whether there exists a way of choosing the number of these that choose (1) in such a way that c is not the winner.

The time complexity of Step 1 is $O(nm^3)$; the time complexity of Step 3b is $O(nm)$; the time complexity of Step 3d is $O(m)$. Therefore, the algorithm runs in time $O(nm^3)$. \square

References

- Ahuja, R. K., Goldberg, A. V., Orlin, J. B., & Tarjan, R. E. (1992). Finding minimum-cost flows by double scaling. *Mathematical Programming*, 53(3), 243–266.
- Bartholdi, III, J., & Orlin, J. (1991). Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4), 341–354.
- Bartholdi, III, J., Tovey, C., & Trick, M. (1989a). The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3), 227–241.
- Bartholdi, III, J., Tovey, C., & Trick, M. (1989b). Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6, 157–165.
- Betzler, N., Hemmann, S., & Niedermeier, R. (2009). A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 53–58, Pasadena, CA, USA.
- Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2004). CP-nets: a tool for representing and reasoning with conditional ceteris paribus statements. *Journal of Artificial Intelligence Research*, 21, 135–191.
- Conitzer, V. (2009). Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research*, 35, 161–191.
- Conitzer, V., Rognlie, M., & Xia, L. (2009). Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, Pasadena, CA, USA.
- Conitzer, V., & Sandholm, T. (2002). Vote elicitation: Complexity and strategy-proofness. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 392–397, Edmonton, AB, Canada.
- Conitzer, V., & Sandholm, T. (2005a). Common voting rules as maximum likelihood estimators. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 145–152, Edinburgh, UK.
- Conitzer, V., & Sandholm, T. (2005b). Communication complexity of common voting rules. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pp. 78–87, Vancouver, BC, Canada.

- Conitzer, V., Sandholm, T., & Lang, J. (2007). When are elections with few candidates hard to manipulate?. *Journal of the ACM*, 54(3), Article 14, 1–33. Early versions in AAAI-02 and TARK-03.
- Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2001). *Introduction to Algorithms* (Second edition). MIT Press.
- Elkind, E., Faliszewski, P., & Slinko, A. (2009). Swap bribery. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory*.
- Elkind, E., & Lipmaa, H. Hybrid voting protocols and hardness of manipulation. In *Annual International Symposium on Algorithms and Computation (ISAAC)*, 3827 of *Lecture Notes in Computer Science*, pp. 206–215, Sanya, Hainan, China.
- Faliszewski, P. (2008). Nonuniform bribery. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1569–1572, Estoril, Portugal.
- Faliszewski, P., Hemaspaandra, E., & Schnoor, H. (2008). Copeland voting: ties matter. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 983–990, Estoril, Portugal.
- Garey, M., & Johnson, D. (1979). *Computers and Intractability*. W. H. Freeman and Company.
- Gibbard, A. (1973). Manipulation of voting schemes: a general result. *Econometrica*, 41, 587–602.
- Hemaspaandra, E., Hemaspaandra, L. A., & Rothe, J. (1997). Exact analysis of Dodgson elections: Lewis Carroll’s 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6), 806–825.
- Konczak, K., & Lang, J. (2005). Voting procedures with incomplete preferences. In *Multi-disciplinary Workshop on Advances in Preference Handling*.
- Lang, J. (2007). Vote and aggregation in combinatorial domains with structured preferences. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1366–1371, Hyderabad, India.
- Lang, J., Pini, M. S., Rossi, F., Venable, K. B., & Walsh, T. (2007). Winner determination in sequential majority voting. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India.
- McGarvey, D. C. (1953). A theorem on the construction of voting paradoxes. *Econometrica*, 21(4), 608–610.
- Parkes, D. (2006). Iterative combinatorial auctions. In Cramton, P., Shoham, Y., & Steinberg, R. (Eds.), *Combinatorial Auctions*, chap. 2, pp. 41–77. MIT Press.
- Pini, M. S., Rossi, F., Venable, K. B., & Walsh, T. (2007). Incompleteness and incomparability in preference aggregation. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India.
- Rothe, J., Spakowski, H., & Vogel, J. (2003). Exact complexity of the winner problem for Young elections. In *Theory of Computing Systems*, Vol. 36(4), pp. 375–386. Springer-Verlag.

- Sandholm, T., & Boutilier, C. (2006). Preference elicitation in combinatorial auctions. In Cramton, P., Shoham, Y., & Steinberg, R. (Eds.), *Combinatorial Auctions*, chap. 10, pp. 233–263. MIT Press.
- Satterthwaite, M. (1975). Strategy-proofness and Arrow’s conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10, 187–217.
- Walsh, T. (2007). Uncertainty in preference elicitation and aggregation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 3–8, Vancouver, BC, Canada.
- Xia, L., Conitzer, V., & Procaccia, A. D. (2009). A scheduling approach to coalitional manipulation. Working paper.
- Xia, L., Lang, J., & Ying, M. (2007a). Sequential voting rules and multiple elections paradoxes. In *Theoretical Aspects of Rationality and Knowledge (TARK)*, Brussels, Belgium.
- Xia, L., Lang, J., & Ying, M. (2007b). Strongly decomposable voting rules on multiattribute domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Vancouver, BC, Canada.
- Xia, L., Zuckerman, M., Procaccia, A. D., Conitzer, V., & Rosenschein, J. (2009). Complexity of unweighted coalitional manipulation under some common voting rules. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, Pasadena, CA, USA.
- Zuckerman, M., Procaccia, A. D., & Rosenschein, J. S. (2009). Algorithms for the coalitional manipulation problem. *Artificial Intelligence*, 173(2), 392–412.