Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008)

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

Lirong Xia
Department of Computer Science
Duke University
Durham, NC 27708, USA
lxia@cs.duke.edu

Vincent Conitzer
Department of Computer Science
Duke University
Durham, NC 27708, USA
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 profile of partial orders and a 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.

We consider the setting where the number of alternatives is unbounded and the votes are unweighted. We prove that for Copeland, maximin, Bucklin, and ranked pairs, the possible winner problem is NP-complete; also, we give a sufficient condition on scoring rules for the possible winner problem to be NP-complete (Borda satisfies this condition). We also prove that for Copeland and ranked pairs, the necessary winner problem is coNP-complete. All the hardness results 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 scoring rules, maximin, and Bucklin.

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*). 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 *et al.* 1999; 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 *et al.* 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). 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 *et al.* 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; 2005; Conitzer 2007). 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 2005; Conitzer, Sandholm, & Lang 2007; Zuckerman, Procaccia, & Rosenschein 2008). 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, Sandholm, & Lang 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, Sandholm, & Lang 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, for most of the other common rules, there are 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, and ranked pairs. We show that the possible winner problem is NP-complete for all these rules. We show that the necessary winner problem is coNP-complete for the Copeland and ranked pairs rules; for the remaining rules, we give polynomial-time algorithms for this problem.

Preliminaries

Let $\mathcal{C} = \{c_1, \ldots, c_m\}$ 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, $P = (V_1, \ldots, V_n)$, 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, $r : P(\mathcal{C}) \to \mathcal{C}$. The following are some common voting rules.

1. *(Positional) scoring rules*: Given a *scoring vector* $\vec{v} = (v(1), \ldots, v(m))$, 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 $P = (V_1, \ldots, V_n)$, 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 scoring rules are *Borda*, for which the scoring vector is $(m-1, m-2, \ldots, 0)$, and *plurality*, for which the scoring vector is $(1, 0, \ldots, 0)$.
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;

[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.

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.

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

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.

Example 1 *Let there be three alternatives $\{c_1, c_2, c_3\}$. Three partial orders are illustrated in Figure 3. 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.*

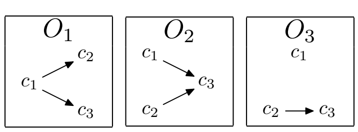

Figure 1: Partial orders.

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

Hardness results

In this section, we will prove that PW is NP-complete w.r.t. scoring rule, Copeland, maximin, Bucklin, and ranked pairs; NW is coNP-complete w.r.t. Copeland and ranked pairs. For scoring rules, we will not show that PW is hard for all scoring rules—in fact, for plurality, PW is easy; rather, we will give a sufficient condition on a scoring rule such that PW is hard. Borda satisfies this condition.

For each hardness result, the proof can be easily modified to show the same result for PcW and NcW (the proofs contain instructions on how they should be modified). All of these results 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 3-cover 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). In each proof, the instance that we construct from a 3-cover instance consists of two parts. The first part is a set of partial orders that encode the 3-cover 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 a 3-cover instance by $V = \{v_1, \ldots, v_q\}, S = \{S_1, \ldots, S_t\}$, where $|S_i| = 3$ and $S_i \subseteq V$ for all $i \leq t$.

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

Definition 4 *For any set $\mathcal{C} = \{c_1, \ldots, c_n\}$, let $O(c_1, \ldots, c_n) = \{(c_i, c_j) : i < j\}$.*

That is, $O(c_1, \ldots, c_n)$ is the set of all pairs consistent with the linear order $c_1 \succ \ldots \succ c_n$. For example, $O(a, b, c) = \{(a, b), (b, c), (a, c)\}$.

Usually, a 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, Sandholm, & Lang 2007). However, there are scoring rules that are defined for any number of alternatives—for example, Borda and plurality. For such 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 scoring rule r consists of a sequence of scoring vectors $\{s_1, s_2, \ldots\}$ such that for any $i \in \mathbb{N}$, s_i is a scoring vector for i alternatives. The next theorem provides a sufficient condition on a 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 a 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 scoring rule r with scoring vectors $\{s_1, s_2, \ldots\}$, 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. $s_l(k) - s_l(k+1) = s_l(k+1) - s_l(k+2) = s_l(k+2) - s_l(k+3) > 0$,

2. $s_l(k+3) - s_l(k+4) > 0$,
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.)

Proof of Theorem 1: Given a 3-cover instance, let $q+3 \leq l \leq f(q+3)$ (where q is the number of elements in the 3-cover instance) satisfy the two conditions in the assumption, and let $k \leq l-4$ satisfy $s_l(k) - s_l(k+1) = s_l(k+1) - s_l(k+2) = s_l(k+2) - s_l(k+3) > 0$, and $s_l(k+3) - s_l(k+4) > 0$. We construct the PW instance as follows

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

First part (P_1) of the profile: For any S_i, choose any $B_i \subset \mathcal{C} - (S_i \cup \{w, d\})$ 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

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

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 $P_1 = \{O_{S_1}, \ldots, O_{S_t}\}$.

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) due to space constraints. 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 $i \leq q$, $s_l(Q, c) - s_l(Q, v_i) = 2(s_l(k) - s_l(k+1))$, $s_l(Q, w) - s_l(Q, c) = \frac{q}{3} \times (s_l(k) - s_l(k+4)) - s_l(k+3) + s_l(k+4)$.
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 3-cover instance. Conversely, given a solution to the 3-cover instance, let I be the set of indices of S_i that are included in the 3-cover. 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 3-cover problem.

For possible co-winner, we replace 1. by
1'. For any $i \leq q$, $s(Q, c) - s(Q, v_i) = s_l(k) - s_l(k+1)$, $s(Q, w) - s(Q, c) = \frac{q}{3} \times (s_l(k) - s_l(k+4))$. □

Theorem 1 provides a sufficient condition on 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 s_l for Borda is $(l-1, l-2, \ldots, 0)$. 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. □

The remaining proofs are omitted due to space constraints.

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.*

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.*

Algorithms for NW and NcW

In this section we present polynomial-time algorithms to compute whether an alternative is a necessary (co-)winner for scoring rules, maximin, and Bucklin. The time complexities are $O(nm^2)$, $O(nm^3)$, $O(nm^2)$ respectively, where m is the number of alternatives and n is the number of votes. We note that these rules 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, 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, $c \not\succ_O w$. 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 2. 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 scoring rules, maximin, and Bucklin. In this section, the input consists of $\mathcal{C} = \{c_1, \ldots, c_m\}$, 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 2 *A partial order O is illustrated in Figure 2. 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 3.*

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

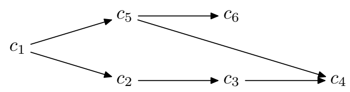

Figure 2: A partial order O.

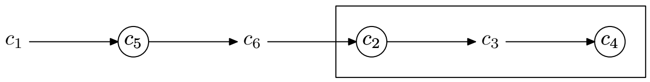

Figure 3: An extension V_1 of O.

Definition 5 *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.

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.

Algorithm 1 (Computing NW w.r.t. a 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 $c \not\succ_O w$, then (following Example 2) 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* $O(nm^2)$.

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 the inequality holds for all alternatives d. (Due to space constraints, we just present the algorithms for maximin and Bucklin, without too much detail about the intuitions for the algorithms.)

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

1. Compute the Up and $Down$ sets for each partial order O.

2. Repeat 3a-c for all tuples c, 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 $c \not\succ_O w$, then raise w as high as possible and lower c as low as possible; if, in the resulting vote, c is ahead of w', add 1 to $S(c, w')$; and for any $d \neq w$, if w is ahead of d, add 1 to $S(w, d)$.

– if $c \succ_O w$, and $c \not\succ_O w'$, then add 0 to $S(c, w')$ and add 1 to $S(w, d)$ for all $d \in \mathcal{C} \setminus (Up_O(w') \cup Up_O(w))$;

– if $c \succ_O w$, and $c \succ_O w'$, then add 1 to $S(c, w')$ and add 1 to $S(w, d)$ for all $d \in \mathcal{C} \setminus Up_O(w)$.

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 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 less than or equal to $\frac{n}{2}$ 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. (The algorithm below is a little more complicated to be more efficient.)

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 $c \not\succ_{O_j} w$, 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 3 and 4. Due to space constraints, we omit the details of this change.

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

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. The following table summarizes our results. These results hold whether or not the alternative must be the unique winner, or merely a co-winner.

	Possible Winner	**Necessary Winner**
scoring	NP-complete	$O(nm^2)$
Copeland	NP-complete	coNP-complete
maximin	NP-complete	$O(nm^3)$
Bucklin	NP-complete	$O(nm^2)$
ranked pairs	NP-complete	coNP-complete

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. In the full version of this paper, we prove that the possible and necessary winner problems are NP-complete and coNP-complete for STV even when the partial orders must correspond to CP-nets. Moreover, we also give a way of embedding any partial order into a CP-net of polynomial size (by introducing exponentially many new alternatives), and use this to show that all of the hardness results in this paper extend to the setting where preferences can be represented either as a CP-net or as a special kind of linear order.

Another approach is to approximate the sets of possible/necessary winners; this has been studied previously for STV (Pini *et al.* 2007). Some of the results in this paper can be extended to show hardness of approximation.

Acknowledgements

We thank Toby Walsh and anonymous reviewers 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.

References

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

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

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

Boutilier, C.; Brafman, R.; Hoos, H.; and Poole, D. 1999. Reasoning with conditional *ceteris paribus* statements. In *UAI'99*, 71–80. Morgan Kaufmann.

Conitzer, V., and Sandholm, T. 2002. Vote elicitation: Complexity and strategy-proofness. In *AAAI'02*, 392–397.

Conitzer, V., and Sandholm, T. 2005. Communication complexity of common voting rules. In *EC'05*, 78–87.

Conitzer, V.; Sandholm, T.; and Lang, J. 2007. When are elections with few candidates hard to manipulate? *JACM* 54(3):Article 14, 1–33.

Conitzer, V. 2007. Eliciting single-peaked preferences using comparison queries. In *AAMAS'07*, 408–415.

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

Gibbard, A. 1973. Manipulation of voting schemes: a general result. *Econometrica* 41:587–602.

Hemaspaandra, E.; Hemaspaandra, L. A.; and Rothe, J. 1997. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *JACM* 44(6):806–825.

Konczak, K., and Lang, J. 2005. Voting procedures with incomplete preferences. In *Multidisciplinary Workshop on Advances in Preference Handling*.

Lang, J.; Pini, M. S.; Rossi, F.; Venable, K. B.; and Walsh, T. 2007. Winner determination in sequential majority voting. In *IJCAI'07*.

Lang, J. 2007. Vote and aggregation in combinatorial domains with structured preferences. In *IJCAI'07*, 1366–1371.

Parkes, D. 2006. Iterative combinatorial auctions. In Cramton, P.; Shoham, Y.; and Steinberg, R., eds., *Combinatorial Auctions*. MIT Press. chapter 3.

Pini, M. S.; Rossi, F.; Venable, K. B.; and Walsh, T. 2007. Incompleteness and incomparability in preference aggregation. In *IJCAI'07*.

Rothe, J.; Spakowski, H.; and Vogel, J. 2003. Exact complexity of the winner problem for Young elections. In *Theory of Computing Systems*, volume 36(4). Springer-Verlag. 375–386.

Sandholm, T., and Boutilier, C. 2006. Preference elicitation in combinatorial auctions. In Cramton, P.; Shoham, Y.; and Steinberg, R., eds., *Combinatorial Auctions*. chapter 10, 233–263.

Satterthwaite, M. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *J. Economic Theory* 10:187–217.

Walsh, T. 2007. Uncertainty in preference elicitation and aggregation. In *AAAI'07*, 3–8.

Xia, L.; Lang, J.; and Ying, M. 2007a. Sequential voting rules and multiple elections paradoxes. In *TARK'07*.

Xia, L.; Lang, J.; and Ying, M. 2007b. Strongly decomposable voting rules on multiattribute domains. In *AAAI'07*.

Zuckerman, M.; Procaccia, A. D.; and Rosenschein, J. S. 2008. Algorithms for the coalitional manipulation problem. In *SODA'08*.