RC23748 (W0510-105) October 13, 2005
Computer Science

IBM Research Report

Computing Slater Rankings Using Similarities Among Candidates

Vincent Conitzer*
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

*Currently at Carnegie Mellon University, Pittsburgh, PA 15213

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Computing Slater Rankings Using Similarities Among Candidates*

Vincent Conitzer
Carnegie Mellon University
Computer Science Department
Pittsburgh, PA 15213, USA
conitzer@cs.cmu.edu

Abstract

Voting (or *rank aggregation*) is a general method for aggregating the preferences of multiple agents. The *Slater rule* is an important voting rule. It selects a ranking of the candidates to minimize the number of pairs of candidates such that the ranking disagrees with the pairwise majority vote on these two candidates. The use of the Slater rule has been hindered by a lack of techniques to *compute* Slater rankings. In this paper, we show how we can decompose the Slater problem into smaller subproblems if there is a set of *similar* candidates. We show that this technique suffices to compute a Slater ranking in linear time if the pairwise majority graph is hierarchically structured. For the general case, we also give an efficient algorithm for *finding* a set of similar candidates. We provide experimental results that show that this technique significantly (sometimes drastically) speeds up search algorithms. Finally, we also use the technique of similar sets to show that computing an optimal Slater ranking is NP-hard, even in the absence of pairwise ties.

1 Introduction

In multiagent systems with self-interested agents, often the agents need to arrive at a joint decision in spite of different preferences over the available alternatives. *Voting* (or *rank aggregation*) is a general method for doing so. In a rank aggregation setting, each voter ranks all the different alternatives (or *candidates*), and a voting *rule* maps the votes to a single ranking of all the candidates. Rank aggregation has applications outside the space of preference aggregation as well: for example, we can take the rankings that different search engines provide over a set of webpages and produce an aggregate ranking from this. Other applications include collaborative filtering (Pennock, Horvitz, & Giles 2000) and planning among automated agents (Ephrati & Rosenschein 1991; 1993). Hence, voting is a topic of significant and growing interest in the computer science community at large, and the electronic commerce community in particular. Recent work has studied the complexity of executing voting rules (mostly the Kemeny rule, another voting rule) (Cohen, Schapire, & Singer 1999; Dwork *et al.* 2001; Davenport & Kalagnanam 2004; Ailon, Charikar, & Newman 2005); the complexity of manipulating elections (Conitzer & Sandholm 2002a; Conitzer, Lang, & Sandholm 2003; Conitzer & Sandholm 2003); eliciting the votes efficiently (Conitzer & Sandholm 2002b; 2005b); adapting voting theory to the setting where the candidates vote over each other by linking to each other (as in the context of the World Wide Web) (Altman & Tennenholtz 2005b; 2005a); and interpreting common voting rules as maximum likelihood estimators of a "correct" ranking (Conitzer & Sandholm 2005a).

The design of voting rules has been guided by the axiomatic approach: decide on a set of criteria that a good voting rule should satisfy, and determine which (if any) voting rules satisfy all of these criteria. One well-known criterion is that of *independence of irrelevant alternatives*. In its strongest form, this criterion states that the relative ranking of two candidates by a voting rule should not be affected by the presence or absence of other candidates. That is, if a is ranked higher than b by a voting rule that satisfies independence of irrelevant alternatives, then the voting rule will still rank a higher than b after the introduction of another candidate c. Arrow's impossibility result (Arrow 1963) precludes the existence of any reasonable voting rule satisfying this criterion. Intuitively, when independence of irrelevant alternatives is satisfied, whether candidate a or b is preferred should depend only on whether more votes prefer a to b than b to a—that is, the winner of the *pairwise election* should be ranked higher. Unfortunately, as noted by Condorcet (de Caritat (Marquis de Condorcet) 1785), there can be cycles in this relationship: for example, it can be the case that a defeats b, b defeats c, and c defeats a in pairwise elections. If so, no ranking of the candidates will be consistent with the outcomes of all pairwise elections.

The *Slater* voting rule is arguably the most straightforward resolution to this problem: it simply chooses a ranking of the candidates that is inconsistent with the outcomes of as few pairwise elections as possible. Unfortunately, as we will discuss later in this paper, computing a Slater ranking is NP-hard. This suggests that we need a search-based algorithm

*This material is based on work done while the author was visiting IBM's T.J. Watson Research Center in the context of an IBM Ph.D. Fellowship. The author thanks Andrew Davenport and Jayant Kalagnanam for a significant amount of valuable direction and feedback.

to compute Slater rankings.[1]

In this paper, we introduce a powerful *preprocessing* technique that can reduce the size of instances of the Slater problem before the search is started. We say that a subset of the candidates consists of *similar candidates* if for every candidate outside of the subset, all candidates inside the subset achieve the same result in the pairwise election against that candidate. Given a set of similar candidates, we can recursively solve the Slater problem for that subset, and for the original set with the entire subset replaced by a single candidate, to obtain a solution to the original Slater problem. In addition, we also make the following contributions:

- We show that if the results of the pairwise elections have a particular hierarchical structure, the preprocessing technique is sufficient to solve the Slater problem in linear time.
- For the general case, we give a polynomial-time algorithm for *finding* a set of similar candidates (if it exists). This algorithm is based on satisfiability techniques.
- We exhibit the power of the preprocessing technique experimentally.
- We use the concept of a set of similar candidates to give the first straightforward reduction (that is, not a randomized reduction or a derandomization thereof) showing that the Slater problem is NP-hard in the absence of pairwise ties.

2 Definitions

We use C to denote the set of candidates. We say that candidate a defeats candidate b in their *pairwise election*, denoted by $a \rightarrow b$, if the number of votes ranking a above b is greater than the number of votes ranking b above a. The Slater rule is defined as follows: find a ranking $\succ$ on the candidates that minimizes the number of ordered pairs (a, b) such that $a \succ b$ but b defeats a in their pairwise election. (Equivalently, we want to maximize the number of pairs of candidates for which $\succ$ *is* consistent with the result of the pairwise election—we will refer to this number as the *Slater score*.) We will refer to the problem of computing a Slater ranking as the Slater problem. An instance of the Slater problem can be represented by a "pairwise election" graph whose vertices are the candidates, and which has a directed edge from a to b if and only if a defeats b in their pairwise election. The goal, then, is to minimize the number of edges that must be flipped in order to make the graph acyclic.

Most elections do not have any ties in pairwise elections. For example, if the number of votes is odd, there is no possibility of a pairwise tie. (We note that in many real-world elections, the number of voters is intentionally made odd to prevent ties.) Hence, we will restrict our attention to elections without pairwise ties (in which case the pairwise election graph becomes a tournament graph). For our positive results, this is merely for simplicity—they can easily be extended to deal with ties as well. Our one negative result, the NP-hardness of computing Slater rankings, is made stronger by this restriction (in fact, without the restriction the hardness has effectively been known for a long time).

[1]Another approach is to look for rankings that are *approximately* optimal in the Slater sense. Of course, this is not entirely satisfactory as it is effectively changing the voting rule.

3 Sets of similar candidates

We are now ready to give the formal definition of a set of similar candidates.

Definition 1 *We say that a subset $S \subseteq C$ consists of* similar candidates *if for any $s_1, s_2 \in S$, for any $c \in C - S$, $s_1 \rightarrow c$ if and only if $s_2 \rightarrow c$ (and hence $c \rightarrow s_1$ if and only if $c \rightarrow s_2$).*

We emphasize that in this definition, it is *not* required that *every* vote prefers s_1 to c if and only if that vote prefers s_2 to c. Rather, the condition only needs to hold on the aggregated pairwise election graph, and hence it is robust to a few voters who do not perceive the candidates as similar.

There are a few trivial sets of similar candidates: 1. the set of all candidates, and 2. any set of at most one candidate. We will be interested in nontrivial sets of similar candidates, because, as will become clear shortly, the trivial sets have no algorithmic use.

The following is the key observation of this paper:

Theorem 1 *If S consists of similar candidates, then there exists a Slater ranking $\succ$ in which the candidates in S form a* (contiguous) block *(that is, there do not exist $s_1, s_2 \in S$ and $c \in C - S$ such that $s_1 \succ c \succ s_2$).*

Proof: Consider any ranking $\succ_1$ of the candidates in which the candidates in S are split into $k > 1$ blocks; we will show how to transform this ranking into another ranking $\succ_2$ with the properties that:

- the candidates in S are split into $k - 1$ blocks in $\succ_2$, and
- the Slater score of $\succ_2$ is at least as high as that of $\succ_1$.

By applying this transformation repeatedly, we can transform the original ranking into an ranking in which the candidates in S form a single block, and that has at least as high a Slater score as the original ranking.

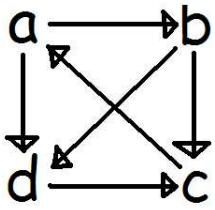
Consider a subsequence of $\succ_1$ consisting of two blocks of candidates in S, $\{s_i^1\}$ and $\{s_i^2\}$, and a block of candidates in $C - S$ that divides them, $\{c_i\}$: $s_1^1 \succ_1 s_2^1 \succ_1 \ldots \succ_1 s_{l_1}^1 \succ_1 c_1 \succ_1 c_2 \succ_1 \ldots \succ_1 c_l \succ_1 s_1^2 \succ_1 s_2^2 \succ_1 \ldots \succ_1 s_{l_2}^2$. Because S consists of similar candidates, a given candidate c_i has the same relationship in the pairwise election graph to *every* s_i^j. Hence, one of the following two cases must apply:

1. For at least half of the candidates c_i, for every s_i^j, $c_i \rightarrow s_i^j$
2. For at least half of the candidates c_i, for every s_i^j, $s_i^j \rightarrow c_i$.

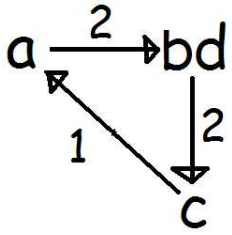
In case 1, we can replace the subsequence by the subsequence $c_1 \succ_2 c_2 \succ_2 \ldots \succ_2 c_l \succ_2 s_1^1 \succ_2 s_2^1 \succ_2 \ldots \succ_2 s_{l_1}^1 \succ_1 s_1^2 \succ_2 s_2^2 \succ_2 \ldots \succ_2 s_{l_2}^2$ to join the blocks without any loss to the Slater score of the ranking. Similarly, in case 2, we can replace the subsequence by the subsequence $s_1^1 \succ_2 s_2^1 \succ_2 \ldots \succ_2 s_{l_1}^1 \succ_1 s_1^2 \succ_2 s_2^2 \succ_2 \ldots \succ_2 s_{l_2}^2 \succ_2 c_1 \succ_2 c_2 \succ_2 \ldots \succ_2 c_l$ to join the blocks without any loss to the Slater score of the ranking. ■

Hence, if we know that S consists of similar candidates, then when we try to compute a Slater ranking, we can without loss of generality restrict our attention to rankings in which all the candidates in S form a (contiguous) block. The optimal internal ranking of the candidates in S within the block is independent of the rest of the ranking, and can be computed recursively.[2] Because of this, we can think of S as a single "super-candidate" with weight $|S|$. Ranking S above a candidate c such that $s \to c$ for all $s \in S$, or below a candidate c such that $c \to s$ for all $s \in S$, will increase the Slater score by $|S|$.

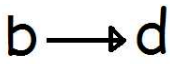
For example, consider the following pairwise election graph:



In this graph, $\{b, d\}$ is a set of similar candidates. Thus, we recursively solve the instance in which b and d are aggregated into a single candidate:



Some of the edges now represent multiple edges in the original graph; this is indicated by the weights on these edges. It is easy to see that the optimal Slater ranking for this graph is $a \succ bd \succ c$. In addition, we need to solve the Slater problem internally for the set of similar candidates:



The optimal Slater ranking for this graph is (of course) $b \succ d$. Thus, the solution to the original problem is $a \succ b \succ d \succ c$.

It is possible to have multiple disjoint sets S_i, each consisting of similar candidates. In this case, we can aggregate each one of them into a single super-candidate. The following lemma will clarify how to compute the Slater scores for such pairs of super-candidates:

[2]Note that if S is a trivial set of similar candidates, there is little use to this: if it is a set of at most one candidate, then the statement that that candidate will form a block by itself is vacuous, and if it is the set of all candidates, we need to recurse on the set of all candidates.

Lemma 1 *If S_1 and S_2 are disjoint sets of similar candidates, then for any $s_1, s_1' \in S_1$ and any $s_2, s_2' \in S_2$, $s_1 \to s_2$ if and only if $s_1' \to s_2'$. (That is, the same relationship holds in the pairwise election graph for any pair of candidates in $S_1 \times S_2$.) Hence, ranking super-candidate S_1 above super-candidate S_2 such that $s_1 \to s_2$ for all $s_1 \in S_1, s_2 \in S_2$, or below a super-candidate S_2 such that $s_2 \to s_1$ for all $s_1 \in S_1, s_2 \in S_2$, will increase the Slater score by $|S_1| \cdot |S_2|$.*

Proof: Because S_1 consists of similar candidates, $s_1 \to s_2$ if and only if $s_1' \to s_2$. And, because S_2 consists of similar candidates, $s_1' \to s_2$ if and only if $s_1' \to s_2'$. ∎

Similar sets that overlap cannot be simultaneously turned into super-candidates. However, the following lemma shows that turning one of them into a super-candidate will (in an informal sense) preserve the structure of the other set:

Lemma 2 *If S_1 and S_2 each consist of similar candidates, and $S_1 \cap S_2$ is nonempty, then $S_1 \cup S_2$ consists of similar candidates.*

Proof: Let $s \in S_1 \cap S_2$. For any $s', s'' \in S_1 \cup S_2$ and any $c \in C - (S_1 \cup S_2)$, we have that $s' \to c$ if and only if $s \to c$, and $s \to c$ if and only if $s'' \to c$. ∎

4 Hierarchical pairwise election graphs can be solved in linear time

In this section, we show that if the pairwise election graph has a certain hierarchical structure, then the Slater problem can be solved efficiently using the techniques from Section 3.

Definition 2 *A* valid candidate tree *is a tree with the following properties:*

- *The leaves are each labeled with a candidate, with each candidate appearing exactly once.*
- *For every internal vertex v, there is a tournament graph $\to_v$ over its children such that for any two distinct children $w_1 \neq w_2$ of v, for any descendants d_1 of w_1 and d_2 of w_2, $d_1 \to d_2$ if and only if $w_1 \to_v w_2$.*

Put alternatively, to find out the direction of the edge between any two candidates in the pairwise election graph, we can simply compare the vertices directly below their least common ancestor. There is always a trivial valid candidate tree, which simply connects every candidate directly to the root node R and uses the pairwise election graph $\to$ as the graph $\to_R$. This tree does not give us any insight. Instead, we will be interested in trees whose vertices have small degree (that is, each vertex has only a few children).

Figure 1 shows an example candidate tree, and Figure 2 shows the graph of pairwise elections that this tree corresponds to.

The following observation will allow us to use the structure of the tree to solve the Slater problem efficiently:

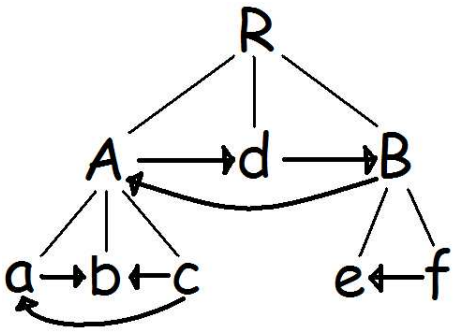


Figure 1: A valid candidate tree.

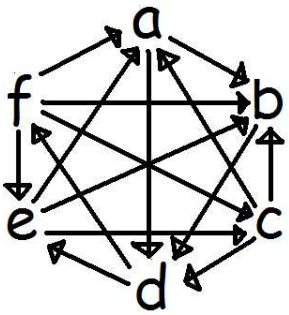


Figure 2: The pairwise election graph corresponding to the valid candidate tree.

Lemma 3 *For any vertex v in a valid candidate tree, the set D_v of candidates that are descendants of v constitutes a set of similar candidates.*

Proof: For any $d_1, d_2 \in D_v$ and $c \in C - D_v$, the least common ancestor of d_1 and c, or of d_2 and c, must be a (strict) ancestor of v. Hence, this least common ancestor must be the same in both cases, and moreover, so must the child of that ancestor from which d_1 and d_2 (and v) descend. Hence $d_1 \rightarrow c$ if and only if $d_2 \rightarrow c$. ∎

Hence, we can solve the Slater problem using the following very simple algorithm:

1. For every child of the root R, generate a super-candidate with weight equal to the number of candidates that descend from it.
2. Solve the Slater problem for the graph $\rightarrow_R$ over these super-candidates (using any algorithm).
3. Solve the Slater problem recursively for each subtree rooted at a child of the root R.
4. Order the candidates, first according to the ranking of the super-candidates that they are in, and then according to the recursive solutions.

Step 2 may be computationally expensive, depending on the number of super-candidates. However, if the degree of each vertex is small, then so is the number of super-candidates in this step. In particular, if the degree is bounded by a constant, then step 2 can be performed in constant time, and the running time of the entire algorithm is linear.

It is easy to see that the algorithm produces the Slater ranking $f \succ e \succ c \succ a \succ b \succ d$ on the example given above.

5 An algorithm for *detecting* sets of similar candidates

In general, we do not know in advance whether there is a nontrivial set of similar candidates in a pairwise election graph. Rather, we need an algorithm that will take as input a pairwise election graph, and discover a nontrivial set of similar candidates if it exists. In this section, we present such an algorithm. The algorithm relies on transforming the problem of detecting a set of similar candidates into a Horn satisfiability problem.

Specifically, for every candidate c we generate a variable $In(c)$ which indicates whether the candidate is in the set of similar candidates. Then, for every ordered triplet of candidates $c_1, c_2, c_3 \in C$, if either $c_1 \rightarrow c_3$ and $c_3 \rightarrow c_2$, or $c_2 \rightarrow c_3$ and $c_3 \rightarrow c_1$, then we generate the clause $In(c_1) \wedge In(c_2) \Rightarrow In(c_3)$ (or, equivalently, $(\neg In(c_1) \vee \neg In(c_2) \vee In(c_3))$.

For example, the instance described in Section 3 produces the following clauses: $In(a) \wedge In(b) \Rightarrow In(c)$, $In(a) \wedge In(c) \Rightarrow In(b) \wedge In(d)$, $In(a) \wedge In(d) \Rightarrow In(b) \wedge In(c)$, $In(b) \wedge In(c) \Rightarrow In(a) \wedge In(d)$, $In(c) \wedge In(d) \Rightarrow In(a)$.

Theorem 2 *A setting of the variables $In(c)$ satisfies all the clauses if and only if $S = \{c \in C : In(c) =$*true*$\}$ consists of similar candidates.*

Proof: First, suppose that the setting satisfies all the clauses. For any $s_1, s_2 \in S$ and $c \in C - S$, if there were a clause $In(s_1) \wedge In(s_2) \Rightarrow In(c)$, it would not be satisfied. It follows that this clause was not generated, and hence either $s_1 \rightarrow c$ and $s_2 \rightarrow c$, or $c \rightarrow s_1$ and $c \rightarrow s_2$. Hence S consists of similar candidates.

Next, suppose that the setting does not satisfy all the clauses. Then, there must be some unsatisfied clause $In(c_1) \wedge In(c_2) \Rightarrow In(c_3)$, which means that $c_1, c_2 \in S$ and $c_3 \notin S$. Because the clause was generated, either $c_1 \rightarrow c_3$ and $c_3 \rightarrow c_2$, or $c_2 \rightarrow c_3$ and $c_3 \rightarrow c_1$, and hence S does not consist of similar candidates. ∎

There are some settings of the variables $In(c)$ that always satisfy all the clauses: setting everything to *true*, and setting at most one variable to *true*. These settings correspond exactly to the trivial sets of similar candidates discussed earlier. Hence, our goal is to find a satisfying setting of the variables in which at least two, but not all, variables are set to *true*. In the example above, the only such solution is to set $In(b)$ and $In(d)$ to *true* and $In(a)$ and $In(c)$ to *false*, corresponding to the set of similar candidates that we used in Section 3. Finding a nontrivial solution can be done in polynomial time with the following simple algorithm: for a given pair of candidates $c_1, c_2 \in C$, set $In(c_1)$ and $In(c_2)$ to *true*, and then follow the implications $\Rightarrow$ in the clauses. If this process terminates without setting all the $In(c)$ variables to *true*, we

have found a nontrivial set of similar candidates. Otherwise, restart with a different pair of candidates, until we have tried every pair of candidates. The algorithm can be improved by keeping track of the initial pairs of candidates for which we have failed to find a similar set, so that when another initial pair leads to one of these pairs being set to *true*, we can fail immediately and continue to the next pair.

When we use this algorithm for finding similar sets to help us compute a Slater ranking, after finding a similar set, we need to compute a Slater ranking both on the instance consisting of the similar set only, and on the reduced version of the original instance where the similar set has been replaced by a single super-candidate. Thus, we will be interested in finding similar sets on these instances as well. It is natural to ask whether some of the computation that we did to find a similar set in the original instance can be reused to find similar sets in the two new instances. It turns out that, in the second new instance, this is indeed possible:

Lemma 4 *Suppose that, in the process of detecting a similar set, we failed with the pair of initial candidates $c_1, c_2 \in C$, before discovering that $S \subseteq C$ is a similar set. Then, in the reduced instance where S is replaced by a single super-candidate c_S,*

1. *we will also fail on initial pair c_1, c_2 if $c_1, c_2 \notin S$;*
2. *we will also fail on initial pair c_1, c_S if $c_1 \notin S, c_2 \in S$.*

Proof: Suppose c_1, c_2 (or, in the second case, c_1, c_S) belong to a nontrivial set S' of similar candidates in the reduced instance. Then, consider the same set in the original instance (if $c_S \in S'$, replace it with all members of S); call this set S''. S'' does not include all candidates because S' does not include all candidates in the reduced instance. Moreover, S'' is a set of similar candidates, for the following reasons. Take any $s_1, s_2 \in S''$ and $c \in C - S''$; we must show that $s_1 \to c$ if and only if $s_2 \to c$. If $s_1 \notin S$ and $s_2 \notin S$, then this follows from the fact that S' consists of similar candidates (even if $c \in S$, because in that case $s_i \to c$ if and only if $s_i \to c_S$ in the reduced instance). If exactly one of s_1 and s_2 is in S (without loss of generality, say $s_1 \in S$), then it must be that $c_S \in S' \Leftrightarrow S \subseteq S''$, so that $c \notin S$. Hence, $s_1 \to c$ if and only if $c_s \to c$, and because S' consists of similar candidates this is true if and only if $s_2 \to c$. Finally, if both s_1 and s_2 are in S, then $c \notin S$ because $c_S \in S' \Leftrightarrow S \subseteq S''$, hence $s_1 \to c$ if and only if $s_2 \to c$ because S consists of similar candidates. But if S'' consists of similar candidates, then we would not have failed on the initial pair c_1, c_2 in the original instance. Hence we have the desired contradiction. ■

For the first new instance consisting of S only, such reuse of computation is not possible, because we cannot have failed on a pair of candidates within S (since they were in fact in a similar set). We only know that we will fail on the pair starting with which we found S, because this pair will lead to all candidates in S being included in the similar set.

6 Experimental results

In this section, we experimentally evaluate the use of the techniques described above as a preprocessing technique that serves to reduce the sizes of the instances before a search algorithm is called. We compare two algorithms: a straightforward search technique, and the preprocessing technique combined with the same search technique. The straightforward search technique decides, at every search node, whether a given edge in the graph should be consistent or inconsistent with the final ranking, and subsequently propagates the effect of this decision to other edges (*e.g.* by transitivity, if it has been decided that edges (a, b) and (b, c) will both be consistent with the final ranking, then by transitivity so must the edge (b, c)). As an admissible pruning heuristic, we use the total number of edges for which it has been decided that the final ranking will be inconsistent with them.

The preprocessing technique uses the algorithm described in Section 5 to search for a set of similar candidates. If it finds one, it recursively solves the subproblems as described in Section 3; otherwise, the search algorithm is called to solve the remaining irreducible problem instance. Each data point in the experiments is an average of 30 instances (the same instances for both algorithms).

In the first set of experiments, instances are generated as follows. Every candidate and every voter draws a random position in $[0, 1]$ (this can be thought of as their stance on one issue) and voters rank candidates by proximity to their own position. The results are in Figure 3:

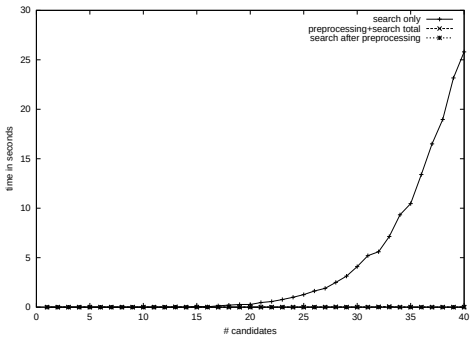


Figure 3: Results with 1 issue, 191 voters, 30 instances per data point.

On these instances, even straightforward search scales reasonably well, but when the preprocessing technique is added, all the instances solve immediately. This is not surprising: the voters' preferences in this domain are *single-peaked*, and it is well-known that for single-peaked preferences, there are no cycles in the pairwise election graph (*e.g.* (Mas-Colell, Whinston, & Green 1995)), so that the final ranking can be read off directly from the graph. Given this, any k candidates in contiguous positions in the final ranking always form a set of similar candidates, so that the preprocessing technique can solve the instances entirely. (No time is spent in search after the preprocessing technique.)

Of course, we do not want to examine only trivial instances. In the next experiment (Figure 4), the candidates and voters draw random positions in $[0,1] \times [0,1]$; in this two-dimensional setup the voters' preferences are no longer single-peaked.

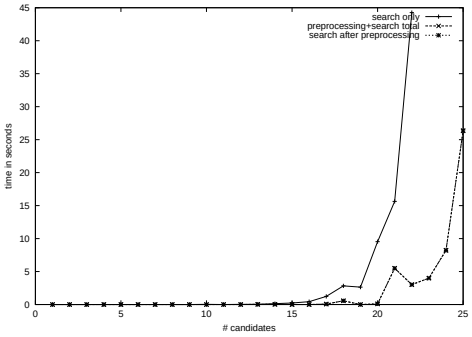


Figure 4: Results with 2 issues, 191 voters, 30 instances per data point.

These instances are much harder to solve, but adding the preprocessing technique significantly speeds up the search. We note that essentially no time is spent in the preprocessing stage (the "preprocessing + search total" and "search after preprocessing" curves are essentially identical), hence the benefits of preprocessing effectively come for free.

We also looked at the effects of changing the number of votes to a small number. Figure 5 shows the results with only 3 votes:

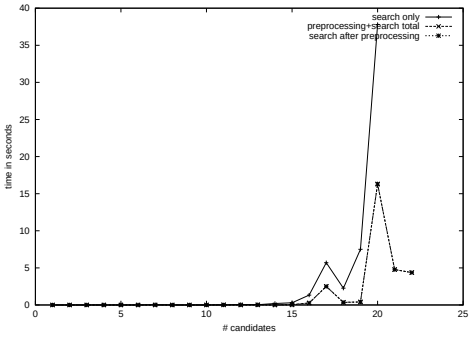


Figure 5: Results with 2 issues, 3 voters, 30 instances per data point.

We experimented with introducing additional structure on the set of candidates. In the next experiment, there are 5 *parties* that draw random positions in $[0,1] \times [0,1]$; each candidate randomly chooses a party, and then takes a position that is the average of the party's position and another random point in $[0,1] \times [0,1]$. The results did not change significantly, as shown in Figure 6.

We also experimented with having the voters and candidates draw positions on an even larger number of issues (10 issues). Perhaps surprisingly, here the preprocessing technique once again solved all instances immediately (Figure 7).

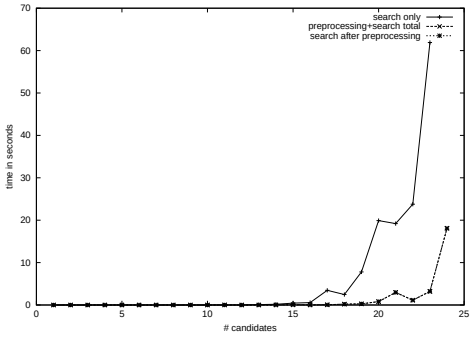


Figure 6: Results with 2 issues, 5 parties, 191 voters, 30 instances per data point.

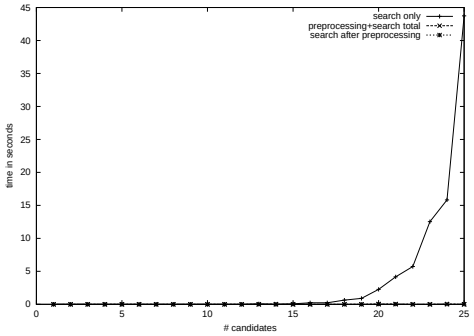


Figure 7: Results with 10 issues, 191 voters, 30 instances per data point.

7 NP-hardness of the Slater problem

In this section, we use the technique of sets of similar candidates in an entirely different manner: we show that it is useful in demonstrating the hardness of the Slater problem when there are no pairwise ties. In the case where pairwise ties between candidates are possible, the hardness of the Slater problem follows from the hardness of the Minimum Feedback Edge Set problem. However, as we have already pointed out, most elections do not have pairwise ties (for example, if the number of votes is odd, then there cannot be any pairwise ties). So, how hard is the problem when there are no ties? This problem is equivalent to the Minimum Feedback Edge Set problem on tournament graphs, and was conjectured to be NP-hard as early as 1992 (Bang-Jensen & Thomassen 1992). The conjecture remained unproven until 2005, when a randomized reduction was given (Ailon, Charikar, & Newman 2005). A later derandomization of this proof finally proved the conjecture completely (Alon 2005). Interestingly, the observations about sets of similar candidates made above allow us to give a more direct proof of this result (which does not rely on derandomizing a randomized reduction). This proof is given below.

Theorem 3 *The* Slater *problem is NP-hard (even in the absence of pairwise ties).*

Proof: We reduce from the NP-complete Maximum Satisfiability (MAXSAT) problem. We show how to reduce an arbitrary MAXSAT instance, given by a set of clauses K over a set of variables V, and a target number t_1 of clauses to satisfy, to an instance of the Slater problem and a target score t_2, such that there is an ranking with Slater score at least t_2 if and only if there is a solution to the MAXSAT instance (that satisfies at least t_1 clauses). Let M be a sufficiently large number ($M > 6|K||V| + |K|^2$). For every variable $v \in V$, let there be the following 6 super-candidates, each of size M (that is, representing M individual candidates): $C_v = \{a_v, +_v, -_v, b_v, d_v, e_v\}$.[3] Let the individual candidates that any given single super-candidate represents constitute an acyclic pairwise election graph, so that we can order them perfectly and obtain a Slater score of $M(M-1)/2$. Let the super-candidates have the following relationships to each other in the aggregated graph:

- Fix some order $>$ over the variables (e.g. $x_1 > x_2 > \ldots > x_{|V|}$). Then, for any two super-candidates $c_v \in C_v, c_{v'} \in C_{v'}$ with $v > v'$, $c_v \to c_{v'}$.
- For any $v \in V$, for any $c_v \in \{a_v, +_v, -_v\}$ and $c'_v \in \{b_v, d_v, e_v\}$, $c_v \to c'_v$.
- For any $v \in V$, $a_v \to +_v, +_v \to -_v, -_v \to a_v$; $b_v \to d_v, b_v \to e_v, d_v \to e_v$.

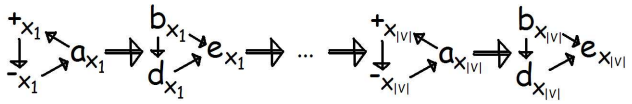


Figure 8: Illustration of part of the reduction.

Additionally, for every clause $k \in K$, let there be a single candidate (not a super-candidate) c_k, with the following relationships to the candidates corresponding to variables. Assume without loss of generality that opposite literals ($+v$ and $-v$) never occur in the same clause. Then,

- If $+v \in k$, then $+_v \to c_k, d_v \to c_k, e_v \to c_k$ and $c_k \to a_v, c_k \to -_v, c_k \to b_v$.
- If $-v \in k$, then $-_v \to c_k, d_v \to c_k, e_v \to c_k$ and $c_k \to a_v, c_k \to +_v, c_k \to b_v$.
- If $\{+v, -v\} \cap k = \emptyset$, then $b_v \to c_k, d_v \to c_k, e_v \to c_k$ and $c_k \to a_v, c_k \to +_v, c_k \to -_v$.

The relationships among the c_k are irrelevant. Finally, let the target Slater score be $t_2 = 6|V|M(M-1)/2 + 36M^2|V|(|V|-1)/2 + 14M^2|V| + t_1 M$.

We now make some observations about the Slater problem instance that we have constructed. First, by Theorem 1, we can restrict our attention to rankings in which the individual candidates in any given super-candidate form a (contiguous) block. Recall that within such a block, we can order the individual candidates to obtain a Slater score of $M(M-1)/2$, which will give us a total of $6|V|M(M-1)/2$ points. Now, if our ranking of two super-candidates is consistent with the pairwise election graph, according to Lemma 1 this will increase the Slater score by M^2. By contrast, the total number of Slater points that we can obtain from *all* the edges in the pairwise election graph that involve a candidate c_k corresponding to a clause is at most $|K| \cdot 6|V|M + |K|^2 < 6|K||V|M + |K|^2 M < M^2$. Hence, it is never worth it to sacrifice an agreement on an edge involving two super-candidates to obtain a better result with regard to the remaining candidates, and therefore we can initially restrict our attention to the super-candidates only as these are our primary concern. It is clear that for $v > v'$ we should rank all the candidates in C_v above all those in $C_{v'}$. Doing this for all variables will increase the Slater score by $36M^2|V|(|V|-1)/2$. Moreover, it is clear that for every v we should rank all the candidates in $\{a_v, +_v, -_v\}$ above all those in $\{b_v, d_v, e_v\}$, and $b_v \succ d_v \succ e_v$. Doing this for all variables will increase the Slater score by $(9M^2 + 3M^2)|V| = 12M^2|V|$. Finally, for every v, any one of the rankings $+_v \succ -_v \succ a_v$, $-_v \succ a_v \succ +_v$, and $a_v \succ +_v \succ -_v$ are equally good, leaving us a choice. Choosing one of these for all variables increases the Slater score by another $2M^2|V|$.

Now, as a secondary concern, we can analyze edges involving the c_k. Agreement on an edge between a c_k and one of the super-candidates will increase the Slater score by M. By contrast, the total number of Slater points that we can obtain from *all* the edges in the pairwise election graph that involve only candidates c_k is at most $|K|(|K|-1)/2 < |K|^2 < M$. Hence, it is never worth it to sacrifice an agreement on an edge involving a super-candidate to obtain a better result with regard to the edges involving only candidates c_k, and hence we can restrict our attention to edges involving a super-candidate. (In fact, the edges involving only candidates c_k will turn out to have such a minimal effect on the total score that we need not consider them at all.) Now, we note that whether a candidate c_k is ranked before *all* the candidates in C_v or after *all* of them makes no difference to the total score, because three of these candidates will have an edge into c_k, and three of them will have have an edge out of c_k. Nevertheless, a candidate c_k could be ranked *among* the candidates C_v for (at most) one $v \in V$. Because d_v and e_v always have edges into c_k and are always ranked last among the candidates in C_v, ranking c_k after at least two of the candidates in C_v will never make a positive contribution to the Slater score. Hence, there are only two possibilities to increase the Slater score (by exactly M) for a given c_k: either rank c_k directly after some $+_v$ such that $+v \in k$ and $+_v$ is ranked first among the C_v, or rank c_k directly after some $-_v$ such that $-v \in k$ and $-_v$ is ranked first among the C_v. Of course, for each variable v, we can rank at most one of $+_v$ and $-_v$ first. (We can also rank a_v first, but this will never help us.) Now we can see that this corresponds to the MAXSAT problem: say that we set v to *true* if $+_v$ is ranked first, and to *false* if $-_v$ is ranked first. Then, we can obtain an additional M points for a candidate c_k if and only if clause k is satisfied, and hence we can increase the Slater score by an additional $t_1 M$ points if and only if we can set the variables in such a way as to satisfy at least t_1 clauses. ∎

[3]The letter c is skipped only to avoid confusion with the use of c as an arbitrary candidate.

8 Extension to Kemeny rule

The *Kemeny* rule (Kemeny 1959) is another voting rule that is similar to the Slater rule. The Kemeny rule, instead of minimizing the number of pairwise elections that the final ranking disagrees with, tries to minimize the total *weight* of such pairwise elections—where the weight of a pairwise election is the number of votes by which its winner defeated its loser.

The techniques presented in this paper can be extended to apply to the Kemeny rule as well. However, to apply to the Kemeny rule, the definition of a set of similar candidates must be modified to state that for any fixed candidate outside the set, all candidates inside the set must receive exactly the same number of votes in the pairwise election against that candidate (rather than merely obtain the same result). This modified definition is much less likely to apply than the original version.

9 Conclusions

Voting (or rank aggregation) is a fundamental problem in systems of multiple self-interested agents that need to coordinate their actions. One voting rule, the *Slater* rule, is perhaps the most natural method to resolve Condorcet paradoxes (in which most voters prefer a to b, most voters prefer b to c, and most voters prefer c to a): it determines a ranking that minimizes the number of pairs of candidates for which the ranking is inconsistent with the outcome of the pairwise election. Unfortunately, computing a Slater ranking is NP-hard.

In this paper, we defined the concept of a set of similar candidates, and showed that any set of similar candidates is contiguous in some Slater ranking. This yields a powerful preprocessing technique for computing Slater rankings: we can solve the Slater problem on the set of similar candidates, and subsequently replace the set of similar candidates in the original instance by a single super-candidate. We showed that this technique suffices to compute a Slater ranking in linear time if the pairwise majority graph is hierarchically structured. In general, we need to be able to *discover* a set of similar candidates for this technique to be useful. We gave a polynomial-time algorithm for doing so based on techniques from satisfiability. We evaluated these techniques experimentally. On highly structured instances, the preprocessing technique solves the problem immediately. On others, it does not solve the entire problem, but still reduces the search time significantly. Moreover, the time spent on the preprocessing technique (including the search for sets of similar candidates) is insignificant relative to the time spent in search, so that the use of the preprocessing technique essentially comes for free. Finally, we gave a completely different use of the concept of similar candidates: we used it to help us show the NP-hardness of the Slater problem when there are no pairwise ties. This proof is arguably more direct than the only other known proof (which itself was discovered only recently after the problem had been open for over a decade).

References

Ailon, N.; Charikar, M.; and Newman, A. 2005. Aggregating inconsistent information: Ranking and clustering. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*.

Alon, N. 2005. Ranking tournaments. Draft.

Altman, A., and Tennenholtz, M. 2005a. On the axiomatic foundations of ranking systems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*.

Altman, A., and Tennenholtz, M. 2005b. Ranking systems: The PageRank axioms. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*.

Arrow, K. 1963. *Social choice and individual values*. New Haven: Cowles Foundation, 2nd edition. 1st edition 1951.

Bang-Jensen, J., and Thomassen, C. 1992. A polynomial algorithm for the 2-path problem for semicomplete digraphs. *SIAM Journal of Discrete Mathematics* 5(3):366–376.

Cohen, W.; Schapire, R.; and Singer, Y. 1999. Learning to order things. *Journal of Artificial Intelligence Research* 10:213–270.

Conitzer, V., and Sandholm, T. 2002a. Complexity of manipulating elections with few candidates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 314–319.

Conitzer, V., and Sandholm, T. 2002b. Vote elicitation: Complexity and strategy-proofness. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 392–397.

Conitzer, V., and Sandholm, T. 2003. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 781–788.

Conitzer, V., and Sandholm, T. 2005a. Common voting rules as maximum likelihood estimators. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 145–152.

Conitzer, V., and Sandholm, T. 2005b. Communication complexity of common voting rules. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, 78–87.

Conitzer, V.; Lang, J.; and Sandholm, T. 2003. How many candidates are needed to make elections hard to manipulate? In *Theoretical Aspects of Rationality and Knowledge (TARK)*, 201–214.

Davenport, A., and Kalagnanam, J. 2004. A computational study of the Kemeny rule for preference aggregation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 697–702.

de Caritat (Marquis de Condorcet), M. J. A. N. 1785. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. Paris: L'Imprimerie Royale.

Dwork, C.; Kumar, R.; Naor, M.; and Sivakumar, D. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th World Wide Web Conference*, 613–622.

Ephrati, E., and Rosenschein, J. S. 1991. The Clarke tax as a consensus mechanism among automated agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 173–178.

Ephrati, E., and Rosenschein, J. S. 1993. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 423–429.

Kemeny, J. 1959. Mathematics without numbers. In *Daedalus*, volume 88. 571–591.

Mas-Colell, A.; Whinston, M.; and Green, J. R. 1995. *Microeconomic Theory*. Oxford University Press.

Pennock, D. M.; Horvitz, E.; and Giles, C. L. 2000. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 729–734.