Voting with Partial Information: What Questions to Ask?

Ning Ding and Fangzhen Lin
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay Road, Hong Kong

ABSTRACT

Voting is a way to aggregate individual voters' preferences. Traditionally a voter's preference is represented by a total order on the set of candidates. However, sometimes one may not have complete information about a voter's preference, and in this case, can only model a voter's preference as a partial order. Given this framework, there has been work on computing the possible and necessary winners of a (partial) profile. In this paper, we take a step further, look at sets of questions to ask in order to determine the outcome of such a partial profile. Specifically, we call a set of questions a deciding set for a candidate if the outcome of the vote for the candidate is determined no matter how the questions are answered by the voters, and a possible winning (losing) set if there is a way to answer these questions to make the candidate a winner (loser) of the vote. We discuss some interesting properties about these sets of queries, prove some complexity results about them under some well-known voting rules such as plurality and Borda, and consider their application in vote elicitation.

1. INTRODUCTION

Voting is a general way to aggregate preferences when a group of people need to make a common decision but have disagreements on which decision to take. Voting is traditionally studied in game theory and social choice theory. Recently it has attracted much attention in AI for various reasons, see for example the survey [4].

Traditionally, a voter's preference is assumed to be a complete linear order over possible candidates (outcomes, or alternatives). One can easily imagine situations where this assumption is too strong, either because the voter herself cannot rank all of the possibilities linearly or because as an observer, we do not have a complete knowledge about her preferences. In fact, one of the well-known formalisms for representing agents' preferences in AI, called CP-nets [3], assumes agents' preferences are partial-ordered. In the context of voting, there has been work in this direction as well. Given a partial ordering for each voter, Konczak and Lang [11] considered the problem of deciding whether a candidate is a *necessary winner* and *possible winner*. A necessary winner is a candidate who is always a winner in every possible completion of the given partial preference profile, while a possible winner is one who is a winner in some of the completions. The complexities of these two problems under a variety of voting rules, especially the so-called positional scoring rules, have been extensively studied [14; 17; 2; 1]. More recently, Conitzer *et al.* [7] considered a notion of manipulations in voting with partial information.

In this paper, we continue this line of work. Given a voting context consisting of a set of candidates, a set of voters, and for each voter, a partial order on the candidates, we consider in general how much additional information is still needed in order to make a particular candidate a winner or loser under a voting rule. If the candidate is already a necessary winner or a necessary loser, then no additional information is needed. Otherwise, one may want to know which voter is crucial in deciding the outcome, and for that voter what would be the important questions to ask. These are obviously important issues to consider when doing voter preference elicitation, and should have some interesting applications. For instance, in an election, a candidate's team may want to know that given what they already know about a group of people, whether more knowledge about their voting preferences would make any differences to the outcome of the election.

This "additional information or knowledge" can come in many forms. Here we take it to be a set of pair-wise comparison questions [5] of the following form: voter i, which candidate do you prefer, a or b? We then consider sets of these questions that can settle the outcome for a candidate. There are at least two possible approaches here. A cautious approach looks for a set of questions such that no matter how these questions are answered by the voters will determine whether the candidate will win or lose. We call such a set of queries a deciding set for the candidate. This amounts to saying that as far as the candidate is concerned, if a question is not in a deciding set, then this question is irrelevant and can be ignored. It is thus not surprising that there is a unique minimal deciding set regardless of which voting rule to use.

The cautious approach makes sense when we want additional information that can decide the outcome for the candidate in question. If we want additional information that would make the candidate a winner (or a loser), then another notion may be more appropriate. Consider the case when we want a candidate to be a winner. Here we may be interested in a set of questions for which there are answers that would lead to the candidate being a winner (or loser), hoping that when voters are asked about these questions they will either indeed answer them as expected or that we can somehow influence them to answer them that way. We call such a set of questions a possible winning (or losing) set for the candidate. As can be expected, minimal possible winning (or losing) sets may not be unique.

These various sets of questions are useful when doing vote elicitation, especially the one-step vote elicitation where there is only one chance to communicate with the voters. Under this circumstance, if we are to select a set of questions to ask voters about,

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

minimality is obviously a desirable property. To date, much work on vote elicitation is about a dynamic question and answering process where voters are queried one question at a time (e.g. [6; 15; 12; 13; 10]). As we shall see, even in this case, our notions of minimal deciding sets are also relevant.

The rest of the paper is organized as follows. We first review some basic notions of voting with complete and partial information, and the notions of possible and necessary winners [11]. We then define our notions of deciding sets, possible winning sets, and possible losing sets. We then prove some interesting properties about deciding sets, and consider how to compute minimal deciding set under various voting rules. We next do the same for possible winning sets, consider how our notions can be used in vote elicitation, and then conclude the paper.

2. PRELIMINARIES

We assume a finite set $N = \{1, ..., n\}$ for voters (players, or agents), and a finite set O for candidates (outcomes, or alternatives). A *preference ordering* p_i of a voter i is a total (linear) order on O, and a preference profile p is a tuple of preference orderings, one for each voter.

A voting rule (method) f is a function from preferences profiles to non-empty sets of outcomes. For a preference profile p, $f(p)$ is the set of winners. When a single winner is desired, a tie-breaking rule can be used to select the one from $f(p)$. Or f is required to be single-valued. In social choice theory terminology, when $f(p)$ can be a set of outcomes, it is called a social choice correspondence, and when $f(p)$ is always single-valued, it is called a social choice function.

Most of the popular voting rules can be defined using a score vector $(s_1, s_2, \cdots, s_m)$, where m is the number of candidates, and $\forall i < m, s_i \geq s_{i+1}$. Given such a score vector, for each voter i and preference ordering p_i, the kth ranked candidate according to p_i receives the score s_k from the voter. Given a preference profile p, a candidate's score is then the sum of of the scores that she receives from each voter, and the winners are those that have the highest score. Such voting rules are called scoring rules.

For instance, the *plurality* voting rule uses the score vector $(1, 0, \cdots, 0)$, the *veto* rule uses the score vector $(1, \cdots, 1, 0)$, and the *Borda* rule uses the score vector $(m, m-1, \cdots, 1)$.

As mentioned in the introduction, we consider the situation when the preference ordering of a voter may not be total, either because the onlooker who is studying the voting does not have a complete knowledge of the voter's preference or that the voter herself is not certain of her own preferences.

Formally a *partial preference ordering* p_i of voter i is a partial order on the set O of candidates: for each $o \in O$, $(o, o) \in p_i$ (reflexivity), if both (o_1, o_2) and (o_2, o_1) are in p_i, then $o_1 = o_2$ (antisymmetry), and if (o_1, o_2) and (o_2, o_3) are in p_i, then $(o_1, o_3) \in p_i$ (transitivity). A *partial preference profile* is then a tuple of partial preference orderings, one for each voter.

Given a partial preference ordering p_i, an *extension* of p_i is a partial preference ordering p_i' such that $p_i \subseteq p_i'$. An extension of p_i that is a total order is called a *completion* of p_i. Similarly, an extension of a partial preference profile p is a partial preference profile p' such that for each i, p_i' is an extension of p_i, and a completion of a partial preference profile p is a preference profile that is an extension of p.

Under a voting rule f, a candidate o is said to be a *necessary winner* of a partial preference profile p, if for all completion p' of p, $o \in f(p')$. If there exists such a completion, then o is said to be a *possible winner* [11]. Furthermore, if o is not a possible winner, then we call o a necessary loser; and if o is not a necessary winner, then we call o a possible loser.

3. DECIDING SETS, POSSIBLE WINNING SETS, AND POSSIBLE LOSING SETS OF QUERIES

As mentioned in the introduction, our interest in this paper is on getting additional information to decide the outcome of a vote. This additional information will be in the form of comparison queries [5] to voters.

DEFINITION 1. *A (comparison) query to voter i is one of the form i:$\{a, b\}$ that asks i to rank candidates a and b.*

When presented with the query i:$\{a, b\}$, the voter i has to answer either "a" (she prefers a over b) or "b" (she prefers b over a).

DEFINITION 2. *An answer to a set Q of questions is a function σ from Q to O such that for any i:$\{a, b\} \in Q$, $\sigma(i$:$\{a, b\}) \in \{a, b\}$.*

Intuitively, if an answer σ maps i:$\{a, b\}$ to "a", then the preference (a, b) ($a \geq b$) is added to voter i's partial preference ordering, and this may entail some new preferences for i, and may even lead to a contradiction. In the following, we require an answer to be consistent with the preferences that the voters already have.

DEFINITION 3. *Let p be a partial preference profile and Q a set of queries. An answer σ to Q is legal under p if for each voter i, the transitive closure of the following set*

$$p_i \cup \{(a, b) \mid i\text{:}\{a, b\} \in Q \wedge \sigma(i\text{:}\{a, b\}) = a\}$$

which we denote by $p_i(\sigma, Q)$, is a partial order on O, the set of candidates. Given a legal answer σ to Q under p, the resulting partial preference profile is then

$$p(\sigma, Q) = (p_1(\sigma, Q), ..., p_n(\sigma, Q)),$$

In the following, unless stated otherwise, we always assume that answers to sets of questions are legal under the given partial preference profile.

We can now define the sets of questions that we are interested in this paper. A deciding set of queries for a candidate o determines the outcome of the vote for o no matter how the queries in the set are answered.

DEFINITION 4. *Let p be a partial preference profile, o a candidate, and f a voting rule. A set Q of queries is a deciding set for o (in p under f) if for every answer σ, o is either a necessary winner or a necessary loser in the new partial profile $\sigma(p, Q)$ under f. Q is a minimal deciding set for o if it is a deciding set and there is no other deciding set Q' such that $Q' \subset Q$.*

Consider the incomplete profile in Table 1. If we take plurality as the voting rule, the minimal deciding set for candidate a is $\{2$:$\{a, b\}, 3$:$\{b, c\}\}$. Firstly, it is a deciding set: if $\sigma(2$:$\{a, b\}) = a$ then a is necessary winner; otherwise if $\sigma(2$:$\{a, b\}) = b$ and $\sigma(3$:$\{b, c\}) = c$, then a is also a necessary winner; and otherwise if $\sigma(2$:$\{a, b\}) = b$ and $\sigma(3$:$\{b, c\}) = b$, then a is a necessary loser.

Next we prove that all its proper subsets are not deciding sets. To prove this we only need to look at its subsets with size one. For $\{2$:$\{a, b\}\}$, a counterexample is when $\sigma(2$:$\{a, b\}) = b$. Given this answer, a is both a possible winner and a possible loser in the new partial preference profile. Similarly for $\{3$:$\{b, c\}\}$, we get a counterexample when $\sigma(3$:$\{b, c\}) = b$.

1	a	$>$	b	$>$	c
2	b	$>$	c		
3	c	$>$	a		

Table 1: Partial preference profile

Notice here that the comparison queries 2:$\{a, c\}$ and 3:$\{a, b\}$ are not in the minimal deciding set.

Sometimes one may also be interested in knowing the ways to make a candidate a winner or a loser in a vote. In this case, one may want to find sets of queries that when answered properly will lead to the candidate being a winner (or loser).

DEFINITION 5. *Let p be a partial preference profile, o a candidate, and f a voting rule. A set Q of queries is a possible winning (losing) set for o (in p under f) if there is an answer σ such that o is a necessary winner (loser) in the new partial profile $\sigma(p, Q)$ under f. Q is a minimal possible winning (losing) set for o if it is a possible winning (losing) set for o, and there is no other possible winning (losing) set Q' for a such that $Q' \subset Q$.*

For the example in Table 1, if we still use plurality as the voting rule, then $Q_1 = \{2{:}\{a, b\}\}$ is a possible winning set for a to win because if we set $\sigma_1(2{:}\{a, b\}) = a$ then in the new partial profile $p(\sigma_1, Q_1)$ as shown in Table 2, a is a necessary winner. Notice that it is not a deciding set. And this possible winning set is obviously minimal because its only proper subset $\emptyset$ is not a possible winning set for a.

The set $Q_2 = \{3{:}\{a, b\}\}$ is also a minimal possible winning set because when $\sigma_2(3{:}\{a, b\}) = a$ as shown in Table 3, then in the new partial profile a is again a necessary winner. From this we can see that there could be multiple minimal possible winning sets for a candidate. Also notice that the query 3:$\{a, b\}$ is not in the minimal deciding set. So a minimal possible winning set may not have any overlap with the minimal deciding set.

1	a	$>$	b	$>$	c
2	a	$>$	b	$>$	c
3	c	$>$	a		

Table 2: $\sigma_1(p, Q_1)$

1	a	$>$	b	$>$	c
2	b	$>$	c		
3	c	$>$	a	$>$	b

Table 3: $\sigma_2(p, Q_2)$

It is easy to see that deciding sets always exist, and if Q is a deciding set, and $Q \subseteq Q'$, then Q' is also a deciding set. Furthermore, if $Q \neq \emptyset$, and Q is a deciding set for o, then Q is both a possible winning set and a possible losing set for o. But the converse is obviously not true in general.

In the following, we consider computing minimal deciding sets under the plurality and Borda rules. The case for the veto voting rule is similar to that of plurality.

4. COMPUTING MINIMAL DECIDING SETS

If Q is a deciding set for candidate o, then for any query q not in Q, as far as the outcome for o is concerned, the answer to q is immaterial, thus can be totally ignored. This suggests that for any voting rule, any partial preference profile, and any candidate, there is a unique minimal deciding set for the candidate. This is indeed the case.

THEOREM 1. *For any voting rule f, partial preference profile p, and candidate o, there is a unique minimal deciding set for o in p under f.*

To prove this theorem, we need the following lemma about partial orders.

LEMMA 1. *Let R be a partial order on S, and $a \neq b$ two elements in S that are not comparable in R. Then there are two total orders R_1 and R_2 such that they both extend R, and are exactly the same except on a and b: for any x and y, $(x, y) \in R_1$ iff $(x, y) \in R_2$ provided $\{x, y\} \neq \{a, b\}$, and $(a, b) \in R_1$ but $(b, a) \in R_2$.*

Proof of Theorem Since the number of voters is finite, there exists a minimal deciding set Q for o. Let Q' be any other deciding set for o. If Q is not a subset of Q', then there is a $q \in Q$ but $q \notin Q'$. Let $Q_0 = Q \setminus \{q\}$. We show that Q_0 is also a deciding set. To show this, suppose σ is an answer to Q_0 under p. We need to show that o is either a necessary winner or a necessary loser in the new partial profile $p(\sigma, Q_0)$. Suppose q is $i{:}\{x, y\}$ for some voter i and candidates $x \neq y$. There are two cases:

1. The answer σ already entails an answer to q, that is, either (x, y) or (y, x) is in $p_i(\sigma, Q_0)$. This basically means that σ is also an answer to Q. Thus o must be either a necessary winner or a necessary loser in the new partial profile $p(\sigma, Q_0)$ as $p(\sigma, Q_0) = p(\sigma, Q)$ and Q is a deciding set.

2. Otherwise, by applying Lemma 1 to the partial order $p_i(\sigma, Q_0)$, we see that there are two answers σ_1 and σ_2 to $Q \cup Q'$ such that σ_1 and σ_2 are the same except on q where we have $\sigma_1(q) = x$ and $\sigma_2(q) = y$. Since $q \notin Q'$, σ_1 and σ_2 are the same answer when restricted to Q'. Since Q' is a deciding set, this means that o must be either a necessary winner in $p(\sigma_1, Q')$ or a necessary loser in $p(\sigma_1, Q')$. Suppose o is a necessary winner in $p(\sigma_1, Q')$. Then o is also a necessary winner in $p(\sigma_2, Q')$ as $p(\sigma_1, Q')$ is the same as $p(\sigma_2, Q')$. It follows then that o must also be a necessary winner in both $p(\sigma_1, Q \cup Q')$ and $p(\sigma_2, Q \cup Q')$. Since Q is also a deciding set, o is also a necessary winner in both $p(\sigma_1, Q)$ and $p(\sigma_2, Q)$. This means that o is a necessary winner in $p(\sigma, Q_0)$. Similarly, if a is a necessary loser in $p(\sigma_1, Q')$, then o is also a necessary loser in $p(\sigma, Q_0)$.

□

From this theorem, we get the following corollary.

COROLLARY 2. *If Q_1 and Q_2 are both deciding sets for a candidate o, then $Q_1 \cap Q_2$ is also a deciding set for o.*

Our next result provides a way to check if a query is in a minimal deciding set.

Suppose S is the set of all comparison queries. Then trivially, S is a deciding set for any candidate in any partial preference profile under any voting rule. Now consider any query $q \in S$, and any given candidate o and partial profile p. Since there is a unique minimal deciding set for o in p, it is clear that q is in the minimal deciding set iff $S \setminus \{q\}$ is not a deciding set.

We thus have the following proposition.

PROPOSITION 1. *Let S be the set of all comparison queries. For any candidate o, and any partial preference profile p, a query $q = i{:}\{a, b\}$ is in the minimal deciding set if and only if there is an answer σ to $S \setminus \{q\}$ such that it can be extended to two answers σ_1 and σ_2 to S such that $\sigma_1(q) = a$, $\sigma_2(q) = b$, and the outcome of o is different in $p(\sigma_1, S)$ and $p(\sigma_2, S)$.*

This proposition will be used in our algorithm for computing the minimal deciding sets under the plurality rule.

4.1 Plurality

For the plurality and the veto rules, computing the minimal deciding set can be done in polynomial time. We show this for the plurality rule.

According to Proposition 1, it suffices to check each query independently whether it is in the minimal deciding set. As Proposition 1 shows, the key is to decide, given a query, whether there are two extensions that differ only on the query but give two different outcomes for the candidate concerned. The basic idea of our algorithm is to reduce this problem to that of computing maximum flow of a network (Harris 1954, see e.g. [16]).

Given a graph (network) (V, E) with two distinguished nodes s, for source, and t, for sink or target, and edges labeled by numbers representing the capacities of the links, a flow is a mapping f from edges to numbers that satisfies the following two conditions:

- for each edge (a, b), $f(a, b)$ is not greater than the capacity of the edge.
- for each node a that is different from the source and the sink, $\sum_{(a,v)\in E} f(a, v) = \sum_{(v,a)\in E} f(v, a)$.

The maximum flow problem is then to determine the maximum value of all possible flows going from source s to sink t, i.e. to maximize the following number among all possible flows f:

$$\sum_{(s,v)\in E} |f(s, v)|.$$

There are several good polynomial time algorithms for computing the maximum value of all flows of a graph (see, e.g. [8]). In the following, we denote by MAX-FLOW(G, s, t) the maximal value of the flows from s to t in a graph G with s as the source and t the sink.

Our algorithm below makes use of the following notations. Given a partial preference profile p, we use $a >_i b$ to stand for $(a, b) \in p_i$. When we add some new preferences $a >_i b, c >_i d$, etc, to p, we mean that we get a new partial preference profile p' such that $p'_j = p_j$ for every $j \neq i$, and p'_i is the transitive closure of $p_i \cup \{(a, b), (c, d), \cdots\}$. When we delete some voters $i_1, i_2, \cdots, i_k$ from p, we mean that we get a new partial preference profile p' such that the set of voters is $V\setminus\{i_1, i_2, \cdots, i_k\}$, and $p'_j = p_j$ for all $j \notin \{i_1, i_2, \cdots, i_k\}$. When we delete some candidates $o_1, o_2, \cdots, o_k$ from p, we mean that we get a new partial profile p' such that the set of candidates is $O' = O\setminus\{o_1, o_2, \cdots, o_k\}$, and p'_j is just p_j constrained to O'.

Algorithm: QueryInMDS($i{:}\{o_1, o_2\}, a, p$)

Input: a query $i{:}\{o_1, o_2\}$, a candidate a and a partial preference profile p.
Output: *yes* or *no* of whether $i{:}\{o_1, o_2\}$ is in the minimal deciding set of a in p.

1. If in p there is a w in $O\setminus\{o_1, o_2\}$ s.t. $w >_i o_1$ or $w >_i o_2$, then return *no*.
2. Else if $a \notin \{o_1, o_2\}$, then we do the following. First, let p^1 be the profile we get by adding $o_1 >_i o_2$ and $o_2 >_i c$ for every $c \notin \{o_1, o_2\}$ to p. If EqualScore$(a, o_2, p^1) =$ *yes*, then return *yes*, else let p^2 be the profile we get by adding $o_2 >_i o_1$ and $o_1 >_i c$ for every $c \neq o_1, o_2$ into p. If EqualScore$(a, o_1, p^2) =$ *yes*, then return *yes*, else return *no*.
3. Else, $a \in \{o_1, o_2\}$. W.L.O.G. let $a = o_1$. The case for $a = o_2$ is exactly the same. Let p^3 be the profile we get by adding $a >_i c$ for all alternative $c \neq a$ to p. If EqualScore$(a, m, p^3) =$ *yes* for some candidate $m \in O, m \neq a$, then return *yes*, else let p^4 be the profile we get by deleting voter i from p. p^4 has one less voter than p. If EqualScore$(a, o_2, p^4) =$ *yes*, then return *yes*, else return *no*.

Algorithm: EqualScore(a, b, p)

Input: candidates a and b and a partial preference profile p.
Output: *yes* or *no* of whether there is a completion p' of p s.t. both a and b receive maximum score in p'.

Let $S_a = \{i \mid \neg\exists w \in O\setminus\{a\}, w >_i a \in p\}$, $S_b = \{i \mid \neg\exists w \in O\setminus\{b\}, w >_i b \in p\}$, $S_i = S_a \cap S_b$, $s_a = |S_a|$, $s_b = |S_b|$, $s_i = |S_i|$.

1. If $|s_a - s_b| \leq s_i$ and $(s_a + s_b - s_i) \bmod 2 = 0$, then let p' be the new profile we get by deleting all the voters in $S_a \cup S_b$ from p, and $\theta = (s_a + s_b - s_i)/2$. If Graph$(p', \theta) =$ *yes*, return *yes*, else return *no*.
2. Else if $|s_a - s_b| \leq s_i$ and $(s_a + s_b - s_i) \bmod 2 \neq 0$, then let $\theta = (s_a + s_b - s_i - 1)/2$. For every $i \in S_a \cup S_b$, let p' be the profile we get by deleting all the voters in $S_a \cup S_b\setminus\{i\}$ and candidates a and b from p. If Graph$(p', \theta) =$ *yes* then return *yes*. If none of these return *yes*, then return *no*.
3. Else we have $|s_a - s_b| > s_i$. W.L.O.G., let $s_b > s_a$. The case for $s_a > s_b$ is exactly the same. Let p' be the profile we get by deleting all the votes in S_a and candidate a from p and set $\theta = s_a$. If Graph$(p', \theta) =$ *yes*, then return *yes*, else return *no*.

Algorithm: Graph(p, θ).

Input: a partial preference profile p and a threshold θ.
Output: *yes* or *no* of whether there is a completion p' of p such that the score of every candidate in $p' \leq \theta$.

Let N be the set of voters and O the set of candidates in p. Let s and t be two new atoms not in $N \cup O$. Construct a flow graph G with $\{s, t\} \cup N \cup O$ as the set of nodes, and the following three layers of edges:

1. For every node in N, an edge from s to it with capacity one.
2. For every node $i \in N$ and every node $o \in O$ s.t. $\neg\exists o' \in O, o' >_i o$, an edge from i to o with capacity one.
3. For every $o \in O$, an edge from it to t with capacity θ.

If MAX-FLOW$(G, s, t) = |N|$, then return *yes*, else return *no*.

In the rest of the section, we prove the correctness of QueryInMDS($i{:}\{o_1, o_2\}, a, p$)

LEMMA 2. *Graph(p, θ) returns yes iff p has a completion with every candidate getting at most score θ under plurality.*

The function Graph(p, θ) is similar to the algorithm for POSSIBLE WINNER for plurality in [2]. Its correctness can be proved similarly. The first layer of edges requires that every voter needs to have a candidate ranked at first place. The second layer constraints the candidates that each voter can choose as the top choice. The third layer requires that each candidate do not get more than θ votes.

LEMMA 3. *EqualScore(a, b, p) returns yes iff there is a completion p_c of p s.t. the scores of a and b in p_c are both the maximal score under plurality.*

Notice that under plurality the only thing that influences the score vector is every voter's top choice. So there is such a completion required in the lemma iff there is an assignment of top choice for every voter that is consistent with p, and has both a and b assigned to the maximal number of voters. In the following we analyze the three cases in the procedure one by one.

The first case is when $|s_a - s_b| \leq s_i$ and $(s_a + s_b - s_i)$ mod $2 = 0$. In this case we can divide $S_a \cup S_b$ into two sets S_1 and S_2 with $|S_1| = |S_2|$, and assign a as top choice for voters in S_1 and b as top choice for voters in S_2. After this a and b will both have $\theta = (s_a + s_b - s_i)/2$ points. We call this assignment τ_1. Notice that p' has all the voters in $N \backslash (S_a \cup S_b)$. If the procedure returns *yes*, it means there is an assignment τ_{-1} of top choice for the voters in $N \backslash (S_a \cup S_b)$ s.t. under τ_{-1} every candidate in $O \backslash \{a, b\}$ does not get more than θ points. The we have that (τ_1, τ_{-1}) is an assignment of top choices for N. This is just an assignment required by the lemma. On the other hand, if the procedure returns *no*, that means even if we assign all votes in $S_a \cup S_b$ to a and b, there is definitely some other candidate whose score exceeds that of a and b. So there is no assignment with the required properties.

The second case is when $|s_a - s_b| < s_i$ and $s_a + s_b - s_i$ mod $2 \neq 0$. In this case, $\theta = (s_a + s_b - s_i - 1)/2$. If our procedure returns *yes*, then let $(S_a \cup S_b) \backslash \{i^*\}$ be the voters that we delete from N to get p' s.t. Graph$(p', \theta) =$ *yes*. Again, we can divide $(S_a \cup S_b) \backslash \{i^*\}$ into S_1 and S_2 with $|S_1| = |S_2|$, and assign a as top choice to voters in S_1 and b to voters in S_2. We call this assignment τ_1. Notice that this assignment have both a and b receiving θ points. Again, because Graph$(p', \theta) =$ *yes*, there is an assignment τ_{-1} of top choices to all voters that remains in p' s.t. the score of every voter in $O \backslash \{a, b\}$ does not exceed θ. Then we have that (τ_1, τ_{-1}) is an assignment of top choices for every voter in N. And this is just the assignment required by the lemma. On the other hand, let us assume there is an assignment τ of top choice for every voter in N such that a and b both get the maximal score among all candidates. If in this assignment a and b both get θ points, then there is exactly one voter $i^* \in S_a \cup S_b$ who is not assigned a or b as top choice. And when we delete $(S_a \cup S_b) \backslash \{i^*\}$ and get p', Graph$(p', \theta) =$ *yes*, and the procedure will return *yes*. If in this assignment a and b both get $\theta' < \theta = (s_a + s_b - s_i - 1)/2$ points, then there is a set of voters $I \subseteq S_a \cup S_b$ who are not assigned a or b as top choice, and $D = (S_a \cup S_b) \backslash I$ is the set of voters who take a or b as top choice. Voters in $N \backslash D$ is assigned top choices such that scores of other candidates doesn't exceed θ'. We can choose an arbitrary voter i^* from I. When p' is obtained by deleting $(S_a \cup S_b) \backslash \{i^*\}$ from p, Graph$(p', \theta) =$ *yes*, because the voters in $p' \subseteq N \backslash D$, and the threshold θ is higher than θ'. So our procedure returns *yes*.

The third case is when $|s_a - s_b| < s_i$. In this case, w.l.o.g. we assume that $s_a < s_b$, and we set $\theta = s_a$. In the algorithm we assign a as top choice to voters in S_a. Let us call this assignment τ_1. Notice that in this case p' has $N \backslash S_a$ as the set of voters. As $|s_a - s_b| < s_i$ and $s_a < s_b$, we have $s_a < s_b - s_i$. So in p' there are more than θ voters who can support b. If our procedure returns *yes*, then Graph$(p', \theta) =$ *yes* and there is an assignment τ_{-1} of top choices to all voters in $N \backslash S_a$ such that the score of every voter in $O \backslash \{a\}$ does not exceed θ. Suppose $R \subseteq S_b \backslash S_a$ is the subset of voters who does not take b as top choice. If b gets $\theta' < \theta$ points in τ_{-1}, we can simply switch the top choice of $\theta - \theta'$ of the voters in R into b, so that b gets exactly θ points. We call this obtained assignment τ'_{-1}. Still, no candidates' score will exceed θ under τ'_{-1}. Here (τ_1, τ_{-1}) is an assignment of top choice for every voter in N required by the lemma. On the other hand, let us assume there is an assignment τ of top choice for every voter in N, s.t. a and b both get the maximal score among all candidates. If for every voter $i \in S_a$, $\tau(i) = a$, then our procedure returns *yes*. If only a subset S'_a of S_a supports a in τ, then our procedure still returns *yes*. The reason is that $N \backslash S_a \subseteq N \backslash S'_a$ and the threshold θ is larger than $|S'_a|$.

THEOREM 3. *QueryInMDS$(i{:}\{o_1, o_2\}, a, p)$ returns yes iff $i{:}\{o_1, o_2\}$ is in the minimal deciding set of a in p under plurality, and it runs in polynomial time.*

Proof of Theorem The number of edges in the flow in procedure Graph is $O(mn)$, and the max flow found by Graph is $O(n)$. So if we use FORD-FULKERSON algorithm in [8] to implement MAX-FLOW, Graph runs in $O(mn^2)$ time. EqualScore calls Graph for at most n times, so the complexity of EqualScore is $O(mn^3)$. QueryInMDS calls EqualScore for $O(m)$ times, so QueryInMDS runs in $O(m^2n^3)$ time.

As under plurality the only thing that influences the score vector is every voter's top choice, according to Proposition 1, $i{:}\{o_1, o_2\}$ is in the minimal deciding set of a iff there are two assignments of top choice for every voter, τ_1 and τ_2, which are consistent with p, satisfying that $\{\tau_1(i), \tau_2(i)\} = \{o_1, o_2\}$, and $\forall j \neq i, \tau_1(j) = \tau_2(j)$ and the outcome for a is different under τ_1 and τ_2. Next we prove the correctness of the algorithm. Firstly the "$\Rightarrow$" part. In step 1, if the procedure does not return *no* then o_1 and o_2 are both legal top choices for i in p. In step 2, if the algorithm returns *yes*, then w.l.o.g. we have EqualScore$(a, o_2, p^1) =$ *yes* so there is an assignment τ_1 assigning the maximal number of votes to both a and o_2 with o_1 the top choice of voter i. This is just the evidence τ_1 in which a is a winner. And we can change i's top choice into o_2 to get τ_2, in which a is a loser. τ_2 and τ_1 differ only on i's vote. So $i{:}\{o_1, o_2\}$ is in the minimal deciding set of a. In step 3, the algorithm returns *yes* in two cases: EqualScore$(a, m, p^3) =$ *yes* for some $m \in O$ or EqualScore$(a, o_2, p^4) =$ *yes*. Here w.l.o.g. we suppose $a = o_1$. If EqualScore$(a, m, p^3) =$ *yes*, then we have an assignment τ_1 where $\tau_1(i) = a$ and a and m both have the maximal score among all candidates. So a wins under τ_1. We can just change voter i's top choice from a into o_2 to get τ_2 under which a loses. So $i{:}\{o_1, o_2\}$ is in the minimal deciding set of a in p. If EqualScore$(a, o_2, p^4) =$ *yes*, then we have an assignment of candidates to every voter except i, such that a and o_2 both have maximal score. This is an incomplete assignment of top choices τ'. Combining τ' with $\tau_1(i) = a$ we get τ_1 in which a wins, while combining it with $\tau_2(i) = o_2$ we get τ_2 in which a loses. So we can conclude $i{:}\{o_1, o_2\}$ is in the minimal deciding set of a.

Then we prove the "$\Leftarrow$" part of the theorem. As we argued in the previous paragraph, if $i{:}\{o_1, o_2\}$ is in the minimal deciding set then there are two assignments of top choice τ_1 and τ_2 for every voter as we described. Suppose a wins under τ_1 and loses under τ_2. We record the score of a candidate c under τ_1 and τ_2 as $s_1(c)$ and $s_2(c)$, and the maximal score among all candidates under τ_1 and τ_2 as $\overline{s_1}$ and $\overline{s_2}$. Notice that we always have $|s_1(a) - s_2(a)| \leq 1, |\overline{s_1} - \overline{s_2}| \leq 1$ and $s_1(a) = \overline{s_1}$ and $s_2(a) < \overline{s_2}$.

If QueryInMDS$(i{:}\{o_1, o_2\})$ = *yes* then first and foremost o_1 and

o_2 are both possible top choices of i. Suppose not, the score vector under τ_1 and τ_2 will be the same, resulting in same outcomes for a under τ_1 and τ_2. So our algorithm does not get into case 1. That leaves use with two cases. If $a \notin \{o_1, o_2\}$, then $s_1(a) = s_2(a)$, so $\overline{s_2} - \overline{s_1} = 1$, and $\overline{s_1} = s_1(a)$. As $\overline{s_2} > \overline{s_1}$ and only o_2 has a higher score in τ_2 than in τ_1, so we have that $s_1(o_2) = \overline{s_1} = s_1(a)$ and $s_2(o_2) = \overline{s_2}$. Notice that o_2 and a both have maximal score among all candidates under τ_1. So EqualScore(a, o_2, p^2) = *yes* and our algorithm will return *yes* in step 2.

If $a \in \{o_1, o_2\}$, then w.l.o.g. we suppose $a = o_1$. Similarly, $s_1(a) = s_2(a) + 1$ and $s_2(o_2) = s_1(o_2) + 1$. Also, $|\overline{s_1} - \overline{s_2}| \leq 1$ and $s_1(a) = \overline{s_1}$ and $s_2(a) < \overline{s_2}$. If a has unique maximal score in τ_1, then we have $s_1(o_2) = s_1(a) - 1$ because only o_2 has a higher score under τ_2 than under τ_1. So EqualScore(a, o_2, p^4) = *yes* and our algorithm will return *yes*. If some other candidate m also has maximal score under τ_1, then it will also return *yes* because EqualScore(a, m, p^3) = *yes* for candidate m. □

As there are only polynomial such queries, computing the minimal deciding set also takes only polynomial time.

4.2 Borda and other scoring voting rules

On the other hand, under the Borda voting rule and other scoring rules, computing minimal deciding sets is NP-complete.

The Borda voting rule uses the score vector $W = \{m, m-1, \cdots, 1\}$, where m is the number of candidates. Given a partial preference profile p, and a candidate o, it is known that checking if o is a possible winner in p under Borda is an NP-complete problem [17]. It came as no surprise that checking whether a query q is in the minimal deciding set is also an NP-complete problem.

THEOREM 4. *The problem of checking if a query q is in the minimal deciding set for a candidate o in a partial profile p under the Borda voting rule is NP-complete.*

PROOF. The problem is in NP follows from Proposition 1 as checking whether an answer σ to $S\backslash\{q\}$ is legal in p, whether it can be extended to two different answers to q such that the outcomes of o in the two extensions are different under the Borda rule can all be done in polynomial time, where S is the set of all queries.

The problem is NP-hard because o is a possible winner iff either o is a necessary winner or the minimal deciding set for o is not empty. Notice that checking if o is a necessary winner under Borda can be done in polynomial time [11]. □

In fact, the proof of this theorem gives a more general result:

THEOREM 5. *For any polynomial time voting rule under which the possible winner problem is NP-complete and the necessary winner problem is in P, the problem of checking if a query is in the minimal deciding set is NP-complete.*

It is known that except for plurality and veto rules, all scoring rules have the property in the above theorem [17; 2; 1]. We thus have the following corollary.

COROLLARY 6. *For any scoring voting rule that is different from the plurality and the veto rules, checking if a query is in the minimal deciding set is NP-complete.*

5. COMPUTING POSSIBLE WINNING SETS

We have seen that there may be multiple minimal possible winning sets. This makes the problem of computing these sets harder.

PROPOSITION 2. *A candidate o is a possible winner in a partial preference profile p iff the set of all queries is a possible winning set for a. A candidate o is a possible loser iff she is not a necessary winner iff the set of all queries is a possible losing set for a.*

Thus just like Corollary 6, we have the following result.

THEOREM 7. *For any scoring voting rule that is different from the plurality and the veto rules, checking if a set of queries is a possible winning set is NP-complete.*

We do not at present know the complexity of computing a minimal possible winning set. Our guess is that it is Π_2^P-complete, same as the complexity of computing minimal models (circumscription) in propositional logic [9].

Neither do we know the exact complexity of checking if a set of queries is a possible losing set for a candidate. While Proposition 2 implies that checking if the set of all queries is a possible losing set for a candidate is in P, the problem seems to be harder in general as it requires checking whether an answer has enough information to conclude that the candidate is a necessary loser, which is a coNP-complete problem for voting rules such as Borda.

We now show that for the plurality voting rule, deciding whether a query set is a minimal possible winning set is in P.

5.1 Plurality

Algorithm: PossibleWinningSet(Q, p, a):

Input: A query set Q, a partial profile p and a candidate a, and we assume that $\forall i$:$\{b, c\} \in Q, (b, c) \notin p_i, (c, b) \notin p_i$.
Output: *yes* or *no* of whether Q is a possible winning set of a in p.

1. For every voter i, let G_i be the undirected graph with O as nodes and $\{(b, c) \mid i$:$\{b, c\} \in Q\}$ as edges and S_i be the set of all strongly connected components of G_i. For a strongly connected component u, we use $V(u)$ to represent the set of vertices of u. For every voter i, for every strongly connected component $u \in S_i$ s.t. $a \in V(u)$ and $\neg\exists o \in O, o >_i a$, add $a >_i c$ to p for every $c \neq a$ in $V(u)$. Set s_a = the minimal score of a in p.

2. $\forall i \in N, U_i = \{u \in S_i \mid \forall o \in V(u), \neg\exists w \in O, w >_i o\}$, $U = U_1 \cup \cdots \cup U_n$

3. Let O be the set of candidates in p and U as defined. Let s and t be two new atoms not in $U \cup O$. Construct a graph G with $\{s, t\} \cup U \cup O\backslash\{a\}$ as set of nodes, and the following three layers of edges:

   (a) for every node in U an edge from s to it with capacity one.

   (b) for every node $u \in U$ and every candidate o s.t. $o \in O\backslash\{a\}$ and $o \in V(u)$, an edge from u to o with capacity one.

   (c) for every $o \in O\backslash\{a\}$ an edge from it to t with capacity s_a.

   If MAX-FLOW$(G, s, t) = |U|$, then return *yes*, else return *no*.

THEOREM 8. *PossibleWinningSet(Q, p, a) returns yes iff Q is a possible winning set of a in p, and it runs in polynomial time.*

As proved in [11], a is a necessary winner in p *iff* the minimal score of a is higher than the maximal score of any other candidate $c \in O$.

Intuitively, our algorithm tries to maximize the *min score* of a and see whether the *max score* of other candidates can be less than the min score of a under some answer of Q. The detailed proof of the correctness of the algorithm is omitted due to page limit.

The graph constructed in this algorithm has $O(mn)$ edges, and the flow found by the algorithm is $O(mn)$. So if we use FORD-FULKERSON alogrithm to implement MAX-FLOW, the flow calculation takes $O(m^2n^2)$ time. Deciding the strongly connected components costs $O(m^2n)$ time . So PossibleWinningSet runs in $O(m^2n^2)$ time.

To determine whether Q is a minimal possible winning set, we just need $|Q| + 1$ calls of the above algorithm. So determining whether a query set is a minimal possible winning set under plurality is also in P.

6. VOTE ELICITATION

Our notion of deciding sets generalizes the notions of necessary and possible winners [11]. It is also closely related to work on vote elicitation (e.g. [6; 15]). Generally speaking, vote elicitation is about getting information about voters' preferences. It can take many forms. For instance, we can ask an undecided voter which candidate she will place the first or just ask her to give a complete ranking of the candidates. Given that we have assumed in this paper to consider pair-wise comparison queries, we consider vote elicitation by asking voters pair-wise comparison queries as well.

Besides the types of questions to ask, another important consideration is whether these questions are asked one at a time. Our notion of minimal deciding sets is obviously suitable for vote elicitation when we can ask a set of comparison questions. One can easily imagine situations when this is necessary. For instance, given the time and resource constraints, there may be only one chance to communicate with voters. In other cases, we may have the luxury of asking questions one at a time. This means that we can ask one question first, and depending on the answer given to that question, decide which question to ask next. To distinguish these two different ways of doing vote elicitation, we call the latter dynamic vote elicitation and former one-step vote elicitation.

For one-step vote elicitation, if we are interested in a particular candidate, then the obvious way is to compute the minimal deciding set for this candidate, and send out the questions in the minimal deciding set to respective voters. If we want to find out the outcomes for a set of candidates, we will need to compute the minimal deciding set for each of the candidates, and send out the union of all the minimal deciding sets to the voters.

To date, much current work on vote elicitation has been on what we call dynamic vote elicitation. Conitzer and Sandholm [6] introduced what they called (fine) elicitation trees for modeling this dynamic process where questions can be asked one at a time. These elicitation trees are very general, without any restriction on the type of questions that can be asked. We can adapt them here in our framework as follows.

A query tree is just a tree of queries with two possible answers.

DEFINITION 6. *A query tree is one where each non-leaf node is labeled by a query of the form i:$\{x, y\}$ and has two children, one labeled as the x branch child, and the other the y branch child.*

If T is a query tree, and p a partial preference profile, then we can associate a partial preference profile with each node in T as follows:

- if n is the root, then its partial preference profile is p;
- inductively, if n is a non-leaf node labeled by the query i: $\{x, y\}$, and has partial preference profile p', then the partial preference profile for its x branch child is the transitive closure of $p' \cup \{x >_i y\}$, and for its y branch is the transitive closure of $p' \cup \{y >_i x\}$.

An elicitation tree for a candidate a is then a query tree that has consistent partial profile for very node in the tree, and can decide the outcome for a in every leaf node under a given voting rule.

DEFINITION 7. *Given a voting rule f, and a partial preference profile p, an elicitation tree for a candidate a (in p under f) is a query tree with the following properties:*

- *for each node in the tree, its partial preference profile is consistent;*
- *if l is a leaf, and p_l is the partial preference profile of l, then a is either a necessary winner or a necessary loser in p_l under the voting rule f.*

The following proposition summarizes some simple properties of elicitation trees.

PROPOSITION 3. *Suppose T is an elicitation tree for a in profile p under the voting rule f. Then*

- *if n is a non-leaf node, i:$\{x, y\}$ the query in n, and p_n the profile of n, then neither $x <_i y$ nor $y <_i x$ is in p_n;*
- *if n_1 is an ancestor of n_2, and q_i, $i = 1, 2$, the query in n_i, then $q_1 \neq q_2$;*
- *if Q is the set of queries in the nodes of T, then Q is a deciding set for a in p under f.*
- *if P is a path from root to a leaf n where a is a necessary winner (loser), and Q the set of queries in the nodes of P, then Q is a winning (losing) set of queries for a in p under f.*

This proposition shows that any elicitation tree for a candidate must include all queries from the minimal deciding set for the candidate, and any path from the root to a winning (losing) leaf must include all queries from one of the minimal winning (losing) query sets for the candidate.

Instead of focusing on a given candidate, we can also define elicitation trees for deciding the outcome of the vote for all candidates.

DEFINITION 8. *Given a partial preference profile p, and a voting rule f, a query tree is an elicitation tree in p under f if it satisfies the following conditions:*

- *for each node in the tree, its partial preference profile is consistent;*
- *if l is a leaf, and p_l is the partial preference profile of l, then for every candidate a, a is either a necessary winner or a necessary loser in p_l under the voting rule f.*

Similarly, it is easy to see that an elicitation tree must include all queries in the minimal deciding set of every candidate.

The results by Conitzer and Sandholm [6] show that constructing elicitation trees is hard for most voting rules. Recently there

has been work on finding good heuristics for guiding the elicitation process, and for estimating the election outcome (e.g. [12; 13; 10]).

Notice that the size of an elicitation tree is in general exponential in the number of voters and candidates. However, there is no need to explicitly construct an elicitation tree. It all comes down to deciding which query to ask in each situation, that is, a policy function that maps a state, which is a partial preference profile, to a query. One such policy function is to select a random consistent query to ask at any given state. In fact, this policy function was used as the baseline strategy against which the proposed heuristics from [12; 13; 10] are compared. An obviously more accurate one is to randomly select a query from the minimal deciding set. A drawback is that except for plurality and veto rules, computing minimal deciding set is intractable.

In summary, our various notions of deciding sets are closely related to vote elicitation. It is clear that minimal deciding sets are what we need to compute when considering one-step elicitation. They can used for dynamic vote elicitation as well. However, there is much to be done here, not the least is the need for various notions of "optimal" elicitation trees or policies as it is apparent that there cannot be one that can serve all purposes: one could be looking for an elicitation tree that has the smallest height, one with the smallest number of nodes, or one with the most even distribution of queries among the voters.

7. CONCLUDING REMARKS

We have considered sets of pair-wise comparison queries to ask the voters about in order to determine the outcome of an election with partial information.

A set of queries is a deciding set for a candidate if the outcome of the election for the candidate can be determined no matter how the queries in the set are answered. One fundamental property about this notion is that among these sets, there is a unique minimal one. Thus as far as a candidate is concerned, a comparison between two candidates is irrelevant to her if the associated query is not in her minimal deciding set. Computationally we have shown that the minimal deciding set can be computed in polynomial time for the plurality and veto rules, and is NP-complete to compute for other scoring rules. We will study complexity for other voting rules in our future work.

On the other hand, a possible winning (losing) query set for a candidate is one that has an answer that will lead to the candidate being a necessary winner (loser). For a manipulator, these sets may be of more interest as they could tell her how to influence the voters to make the candidate a winner or a loser. We have shown that for plurality and veto rules, a minimal possible winning set can be computed in polynomial time. We believe that the same is true for computing a minimal possible losing set as well. For scoring voting rules such as Borda, the problem is again NP-hard for checking if a set of queries is a possible winning set.

We have looked at how our various notions of deciding sets can be used for vote elicitation. Towards this end, we distinguish vote elicitation between those that can have just one chance to communicate with the voters and those that have unlimited number of chances. We call the former one-step vote elicitation and the latter dynamic vote elicitation. Our notion of minimal deciding sets can be applied directly to one-step vote elicitation. It is also relevant in dynamic vote elicitation. But there is much to be explored here.

8. REFERENCES

[1] Dorothea Baumeister and Jörg Rothe. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. In *ECAI-10*, pages 1019–1020, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[2] Nadja Betzler and Dorn Britta. Towards a dichotomy for the possible winner problem in elections based on scording rule. *Journal of Computer and System Sciences*, 76:812–836, 2010.

[3] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Cp-nets: A tool for representing and reasoning with conditional *Ceteris Paribus* preference statements. *JAIR*, 21:135–191, 2004.

[4] Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet. A short introduction to computational social choice. In Jan van Leeuwen, Giuseppe Italiano, Wiebe van der Hoek, Christoph Meinel, Harald Sack, and František Plášil, editors, *SOFSEM 2007: Theory and Practice of Computer Science*, volume 4362 of *Lecture Notes in Computer Science*, pages 51–69. Springer Berlin / Heidelberg, 2007.

[5] Vincent Conitzer. Eliciting single-peaked preferences using comparison queries. *JAIR*, 35:161–191, 2009.

[6] Vincent Conitzer and Tuomas Sandholm. Vote elicitation: Complexity and strategy-proofness. In *AAAI-02*, pages 392–397. AAAI, 2002.

[7] Vincent Conitzer, Toby Walsh, and Lirong Xia. Dominating manipulations in voting with partial information. In *IJCAI-2011 Workshop on Social Choice and Artificial Intelligence*, 2011.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*, chapter 26. MIT Press, 2001.

[9] Thomas Eiter and Georg Gottlob. Propositional circumscription and extended closed world reasoning are Π_2^p-complete. *Theoretical Computer Science*, 114:231–245, 1993.

[10] Meir Kalech, Sarit Kraus, Gal A. Kaminka, and Claudia V. Goldman. Practical voting rules with partial information. *Autonomous Agents and Multi-Agent Systems*, 22(1):151–182, January 2011.

[11] Kathrin Konczak and Jérôme Lang. Voting procedures with incomplete preferences. In *Proc. IJCAI- 05 Multidisciplinary Workshop on Advances in Preference Handling*, 2005.

[12] Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *IJCAI-11*, pages 287–293, 2011.

[13] Tyler Lu and Craig Boutilier. Vote elicitation with probabilistic preference models: empirical estimation and cost tradeoffs. In *Proceedings of the Second international conference on Algorithmic decision theory*, ADT'11, pages 135–149, Berlin, Heidelberg, 2011. Springer-Verlag.

[14] M.S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Incompleteness and incomparability in preference aggregation. IJCAI-07, pages 1464–1469, 2007.

[15] Ariel D. Procaccia. *Computational Voting Theory: Of the Agents, By the Agents, For the Agents.* PhD thesis, The Hebrew University of Jerusalem, Sep. 2008.

[16] Alexander Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.

[17] Lirong Xia and Vincent Conitzer. Determining possible and necessary winners under common voting rules given partial orders. *JAIR*, 41:25–67, May 2011.