Incompleteness and Incomparability in Preference Aggregation: Complexity Results

M. S. Pini[1], F. Rossi[1], K. B. Venable[1] and T. Walsh [2]

1 : *Department of Pure and Applied Mathematics, University of Padova, Italy*
E-mail: {mpini,frossi,kvenable}@math.unipd.it
2 : *NICTA and UNSW Sydney, Australia*
E-mail: Toby.Walsh@nicta.com.au

Abstract

We consider how to combine the preferences of multiple agents despite the presence of incompleteness and incomparability in their preference orderings over possible candidates. An agent's preference ordering may be incomplete because, for example, there is an ongoing preference elicitation process. It may also contain incomparability as this is useful, for example, in multi-criteria scenarios. We focus here on the problem of computing the *possible* and *necessary* winners, that is, those candidates which can be, or always are, the most preferred for the agents. Possible and necessary winners are useful in many scenarios including preference elicitation. First, we show that testing possible and necessary winners is in general a computationally intractable problem in the worst case, as is providing a good approximation of such sets. Then, we identify general properties of the preference aggregation function which are sufficient for such sets to be computed in polynomial time. Finally, we show how possible and necessary winners can be used to focus preference elicitation. We show that in general it is computationally intractable in the worst-cae to decide when to terminate elicitation. However, we identify a property of the preference aggregation function that allows us to decide when to terminate elicitation in polynomial time, by focusing on possible and necessary winners.

Key words: preference in multi-agent systems, preference aggregation, computational complexity, uncertainty, elicitation

1. Introduction

We consider a multi-agent setting where each agent specifies their preferences by means of an ordering over the possible candidates. A pair of candidates can be ordered, incomparable, in a tie, or the relationship between them may not yet be specified. Incomparability and incompleteness represent very different concepts. Candidates may be *incomparable* because the agent does not wish very dissimilar candidates to be compared. For example, we might not want to compare a biography with a novel since the criteria along which we judge them are just too different. Candidates can also be incomparable because the agent has multiple criteria to optimize. For example, we might not wish to compare a faster but more expensive laptop with a slower and cheaper one. *Incompleteness*, on the other hand, represents simply an absence of knowledge about the relationship between certain pairs of candidates. Incompleteness arises when we have not fully elicited an agent's preferences or when agents have privacy concerns which prevent them from revealing their complete preference ordering.

Since we wish to aggregate together the agents' preferences into a single preference ordering, we must modify preference aggregation functions to deal with incompleteness. One possibility is to consider all possible ways in which the incomplete preference orders can be consistently completed. In each possible completion, preference aggregation may give different optimal elements (or *winners*). This leads to the idea of *possible winners* (those candidates which are winners in at least one possible completion) and *necessary winners* (those candidates which are winners in all possible completions) [10].

While voting theory has been mainly interested in the fairness of social choice or social welfare functions, there has been recent interest in computational properties of preference aggregation [14, 11, 10, 7]. It has already been noted that the computational complexity of manipulating an election is closely related to that of computing possible winners

[10, 4]. In this paper we start by considering the computational complexity of testing if a given candidate is a necessary or a possible winner. We consider two different dimensions: weighted/unweighted agents and bounded/unbounded number of candidates. Weights are useful in multiagent systems where we have different types of agents and some agent may have more importance than some other agent in the process of taking a decision. We provide the computational complexity of computing the necessary and possible winners in all of the considered scenarios in the worst case. Moreover, we show that it is intractable even to obtain a good approximation of such sets, except when we have unweighted agents and a bounded number of candidates. Then, we identify sufficient conditions that assure tractability. Such conditions concern properties of the preference aggregation function, such as monotonicity and independence to irrelevant alternatives (IIA) [1].

We stress that worst-case complexity results like these are only the beginning to understand the complexity of computing possible and necessary winners. The complexity of these problems in practice may still be easy. For example, a number of recent theoretical results suggest that the closely related problem of manipulating an election may be NP-hard in the worst case but computationally easy in practice [8].

Possible and necessary winners are useful in many scenarios including preference elicitation [3]. In fact, elicitation can be terminated when the set of possible winners coincides with that of the necessary winners [7]. We show that in general deciding when to terminate the elicitation process is a computationally intractable problem. However, if the preference aggregation function is IIA, preference elicitation can be stopped in polynomial time. This can be done by focusing just on the incompleteness concerning those candidates which are possible and necessary winners, allowing us to ignore all other candidates.

The paper is a revised and an extended version of [13, 12, 16].

2. Basic notions

We now introduce some basic notions on which our work is based. First, we give the definitions regarding preference orderings and incomplete profiles. We, then, consider the notions of preference aggregation functions, possible and necessary winners and describe how to represent compactly the set of the results of a preference aggregation function. We conclude by motivating the usefulness of weighted votes in terms of some real-world setting.

2.1. Preferences and incomplete profiles

We assume that each agent's *preferences* (that is, each agent's vote) are specified via a partially specified partial order over the set of possible candidates, that we will denote by Ω. A partially specified partial order is a partial order (PO) (that is, a reflexive, antisymmetric and transitive relation), where some relation between pairs of candidates is unknown. More precisely, given two candidates A and B, an agent can specify one of the following:

- $A < B$, that is, B is preferred to A;
- $A > B$, that is, A is preferred to B;
- $A = B$, that is, A and B are equally preferred;
- $A \sim B$, that is, A and B are incomparable (i.e., neither $A > B$ nor $B > A$);
- $A?B$, that is, the relation between A and B is unknown; this means that it could be any element of $\{=, >, <, \sim\}$.

Example 1. *Given candidates A, B, and C, an agent may state preferences such as $A > B$, $B \sim C$, and $A > C$, or also just $A > B$ and $B \sim C$, that corresponds to preferences $A > B$, $B \sim C$ and $A?C$. However, an agent cannot state preferences such as $A > B$, $B > C$, $C > A$, or also $A > B$, $B > C$, $A \sim C$ since they don't satisfy transitivity and thus they are not POs.* □

Definition 1 (profile). *A profile is a sequence of n partial orders $p_1, \ldots, p_n$ over the candidates, one for each agent $i \in \{1, \ldots, n\}$, describing the preferences of the agents.*

When one or more of these partial orders is partially specified, we have an incomplete profile.

Definition 2 (incomplete profile). *An incomplete profile is a sequence of n partially specified partial orders* $ip_1, \ldots, ip_n$ *over candidates, one for each agent* $i \in \{1, \ldots, n\}$.

Example 2. *Given three agents and three candidates A, B, and C, a possible incomplete profile is ((A > B > C); (A > C, A > B, B?C); (C > A, A?B, B?C)).* □

Definition 3 (partial profile). *A partial profile is profile where agents can be partitioned into two subsets: the agents that completely specify their partial order and the agents that give no information on their preference.*

Definition 4 (completions of a profile). *Given a profile p, the set of the completions of p, denoted by compl(p), is defined as follows:* $compl(p) = \{p' | p'$ *is a profile obtained from p by replacing every ? with an element in* $\{>, <, =\sim\}\}$.

2.2. Social welfare and preference aggregation

Social welfare functions [1] are functions from profiles to partial orders. In what follows we will consider social welfare functions that take polynomial time to apply.

Example 3. *An example of a social welfare function is the Pareto rule [1]. The Pareto social welfare function f works as follows: given a profile p, for any two candidates A and B, if all agents say* $A > B$ *or* $A = B$ *and at least one says* $A > B$ *in p, then* $A > B$ *in* $f(p)$*; if all agents say* $A = B$ *in p, then* $A = B$ *in* $f(p)$*; otherwise,* $A \sim B$ *in* $f(p)$. □

Given a social welfare function, we define a corresponding preference aggregation function as follows.

Definition 5 (preference aggregation function). *Given a social welfare function f, its corresponding preference aggregation function, written* pa_f*, is a function from incomplete profiles to sets of partial orders that maps every incomplete profile* $ip = (ip_1, \ldots, ip_n)$*, into the set* $pa_f(ip) = \{f(p_1), \ldots, f(p_k)\}$*, that will be called the* set of results *of f on profile ip. The set of the results is obtained by applying f to all the completions of ip, say* $p_1, \ldots, p_k$.

Example 4. *Consider the Pareto social welfare function f described in Example 3. In Figure 1 we show how its corresponding preference aggregation function works. Assume to have three agents and three candidates A, B, and C, and that the agents express their preferences via the incomplete profile* $ip = (ip_1, ip_2, ip_3)$ *shown in the upper part of Figure 1. We consider all the completions of p and we obtain the profiles* p_1 *and* p_2*. Finally, f is applied to both* p_1 *and* p_2*, thus obtaining the set of the result* $pa_f(ip) = \{f(p_1), f(p_2)\}$*. The notion of the combined result will be defined later in the paper.* □

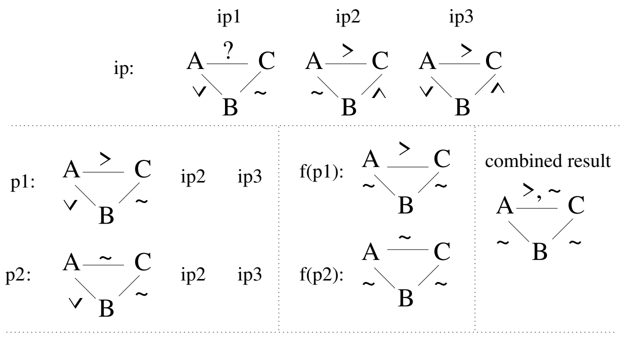

Figure 1: An incomplete profile ip, its completions p_1 and p_2, results $f(p_1)$ and $f(p_2)$, and combined result $cr(f, ip)$.

A social welfare function can satisfy some properties. The first property we consider is independence to irrelevant alternatives (IIA) [1].

Definition 6 (IIA). *A social welfare function f is said to be IIA when, for any pair of candidates A and B, the ordering between A and B in the result depends only on the relation between A and B in the preference orderings of the profile.*

Many preference aggregation functions are IIA (such as the Pareto rule shown in Example 3), and this is a property which is related to the notion of fairness in voting theory [1].

An other interesting property that a social choice function can satisfy is monotonicity.

Definition 7 (monotonicity). *A social welfare function f is monotonic if, for any two profiles p and p' and any two candidates A and B, if passing from p to p', B improves with respect to A in the preference ordering of an agent i and $p_j = p'_j$ for all $j \neq i$, then, passing from $f(p)$ to $f(p')$, B improves with respect to A. A candidate B* improves with respect to *another candidate A if the relationship between A and B does not move left along the following sequence: $>$, ($\sim$ or $=$), $<$*[1].

In words, if B is preferable to A in the result, and we move to a profile where B is not worsened w.r.t. A, then B is still preferable to A in the new result.

Example 5. *The Pareto rule shown in Example 3 is both IIA and monotonic.* □

2.3. Necessary and possible winners

We extend the notions of possible and necessary winners presented in [10] for total orders to the case of partial orders. A necessary winner must be a winner, no matter how incompleteness is resolved in the incomplete profile, while a possible winner is a winner in at least one possible completion of the incomplete profile.

Definition 8 (necessary winner). *Given a social welfare function f and an incomplete profile ip, a necessary winner of f given ip is a candidate which is a maximal element*[2] *in all POs contained in $pa_f(ip)$.*

Definition 9 (possible winner). *Given a social welfare function f and an incomplete profile ip, a possible winner is a candidate which is a maximal element in at least one of the POs contained in $pa_f(ip)$.*

We will write $NW(f, ip)$ and $PW(f, ip)$ for the set of necessary and possible winners of f on profile ip. We will sometimes omit f and/or ip, and just write NW and PW when the two parameters are obvious or irrelevant.

Example 6. *In Example 4, candidates A and B are necessary winners, since they are maximal elements in all POs $f(p_i)$, for all $i = 1, 2$. Candidate C is a possible winner, since it wins in $f(p_2)$. However, C is not a necessary winner, since it does not win in $f(p_1)$.* □

2.4. Combined result

The set of results of a preference aggregation function can be exponentially large. We will therefore also consider a compact representation that is polynomial in size. This may throw away information by compacting together results into a single combined result.

Definition 10 (combined result). *Given a social welfare function f and an incomplete profile ip, the combined result of f on ip, denoted by $cr(f, ip)$, is a graph, whose nodes are the candidates, and whose arcs are labeled by non-empty subsets of $\{<, >, =, \sim\}$. Label l is on the arc between candidates A and B if there exists a PO in $pa_f(ip)$ where A and B are related by l. If an arc is labeled by set $\{<, >, =, \sim\}$, we will say that it is* fully incomplete. *Otherwise, we say that it is* partially incomplete. *The set of labels on the arc between A and B will be called $rel(A, B)$.*

Example 7. *Consider the preference aggregation function and the profile presented in Example 4. The corresponding combined result is shown in the right part of Figure 1. In this combined result, no arc is fully incomplete.* □

[1] For example, B improves with respect to A if it passes from $A \sim B$ to $A < B$ or if it remains the same.

[2] In a partial order, a candidate C is a maximal element if there are no other candidate C' such $C' > C$.

2.5. Weighted votes

As in [4], we will consider both weighted and unweighted agents. A agent with weight an integer k can be viewed as k agents who vote identically. Although human elections are often unweighted, the addition of weights makes voting schemes more general. Weighted voting systems are also used in a number of real-world settings like shareholder meetings and elected assemblies. Weights are useful in multi-agent systems where we have different types of agents. Weights are also interesting from a computational perspective. First, weights can increase computational complexity. Second, the weighted case informs us about the unweighted case when we have probabilistic information about the votes. For instance, if it is NP-hard to compute if the election can be manipulated with weighted votes, then it is NP-hard to compute the probability of a candidate winning when there is uncertainty about how the unweighted votes have been cast [4]. To reduce the impact of ties, we assume that the sum of weights is odd.

3. Complexity of winner determination

We now give worst-case complexity results about computing and testing possible and necessary winners, as well as finding a good approximation of such set of winners.

3.1. Unweighted votes

We first assume to have unweighted agents. In this context, we analyze the complexity of computing and testing possible and necessary winners, and also of finding good approximations of them, both when we have a bounded number of candidates and when we have an unbounded number of candidates.

3.1.1. Bounded number of candidates.

If we have a bounded number of candidates and unweighted votes, the possible and necessary winners can be computed in polynomial time.

Given an incomplete unweighted profile over a bounded number of candidates and a candidate a, we say UNWEIGH TEDBOUNDEDPOSSIBLEWINNER(a) holds iff a is a possible winner of the election, and UNWEIGHTED-BOUN DEDNECESSARYWINNER(a) holds iff a is a necessary winner of the election.

Theorem 1. *Given a candidate a,* `UnweightedBoundedPossibleWinner`(a) *and* `UnweightedBounded NecessaryWinner`(a) *are both polynomial.*

Proof: If the number of candidates is bounded, there are only a polynomial number of effectively different votes. We can thus enumerate and evaluate all different votes in polynomial time. Hence, computing exactly the sets of possible and necessary winners is polynomial[3]. □

3.1.2. Unbounded number of candidates.

Suppose now that the number of candidates is not bounded. In this context, testing necessary and possible winners is NP-hard, even if we restrict ourselves to partially specified but *total* orders. We will consider the following, well known, voting rule.

Definition 11 (Single Transferable Vote). *Single Transferable Vote (STV) is a voting rule where each agent provides a total order on candidates and, initially, an agent's vote is allocated to their most preferred candidate. The* quota *of the election is the minimum number of votes necessary to get elected. If only one candidate is to be elected, then the quota is* $\lfloor n/2 \rfloor + 1$*, where n is the number of agents. If no candidate exceeds the quota, the candidate with the fewest votes is eliminated, and his votes are equally distributed among the second choices of the agents who had selected him as first choice until some candidate reaches the quota.*

[3] A similar argument was made in [4, 6] to show that manipulation of an election by a coalition of agents is polynomial when the number of candidates is bounded.

In what follows we consider STV elections in which some total orders, provided by the agents, may be partially specified.

In general, given an incomplete unweighted profile over an unbounded number of candidates and a candidate a, we say UNWEIGHTEDUNBOUNDEDPOSSIBLEWINNER(a) holds iff a is a possible winner of the election.

Theorem 2. *Given a candidate a,* `UnweightedUnboundedPossibleWinner`(a) *is NP-complete.*

Proof: Membership in NP follows by giving a completion of the profile in which a wins. Completeness follows from the result that EFFECTIVE PREFERENCE (that is, determining if a particular candidate can win an election with one unknown vote) for STV is NP-complete [2]. □

Given an incomplete unweighted profile over an unbounded number of candidates and a candidate a, we say UNWEIGHTEDUNBOUNDEDNECESSARYWINNER(a) holds iff a is a necessary winner of the election.

Theorem 3. *Given a candidate a,* `UnweightedUnboundedNecessaryWinner`(a) *is coNP-complete.*

Proof: The complement problem is in NP since we can show membership by giving a completion of the profile in which some b different from a wins.

To show completeness of the complement problem, we give a reduction from EFFECTIVE PREFERENCE with STV in which b appears in first place in at least one vote. This restricted form of EFFECTIVE PREFERENCE is NP-complete [2]. Consider an incomplete profile p in which $n + 1$ votes have been cast, b has at least one first place vote, one vote remains unknown, and we wish to decide if b can win. We construct a new election from p with n new additional votes, and one new candidate a. We put a at the top of each of these new votes, and rank the other candidates in any order within these n votes. We place a in last place in the original $n + 1$ votes, except for one vote where b is in first place (by assumption, one such vote must exist) where we place a in second place and shift all other candidates down. We observe that a will survive till the last round as a has at least n votes and no other candidate can have as many votes till the last round. We also observe that, if b remains in the election, the score given to each candidate by STV remains the same as in the original election, so the candidates are eliminated in the same order up till the point b is eliminated. If b is eliminated before the last round, the second choice vote for a is transferred. Since a now has $n + 1$ votes, a is unbeatable and must win the election. If b survives, on the other hand, to the last round, we can assume a is ranked at the bottom of the unknown vote. All the other candidates but b and a have been eliminated so b has $n + 1$ votes and is unbeatable. Hence, a is not the necessary winner of this new election if and only if b is a possible winner in the Effective Preference election. Thus determining if a is not the necessary winner of this new election is as hard as determining if if b is a possible winner of the Effective Preference election which is known to be NP-complete. □

Given these results, we might wonder if it is easy to compute a reasonable approximation of the sets of possible and necessary winners in the worst-case. Unfortunately this is not the case. The reduction described in the proof of Theorem 3 shows that we cannot easily approximate the set of possible winners within a factor of two. In fact, we can show that we cannot easily approximate the set of possible winners within *any* constant factor.

Theorem 4. *Given an incomplete unweighted profile over an unbounded number of candidates, it is NP-hard to return a superset of its possible winners, PW^*, in which we guarantee $|PW^*| < k|PW|$ for some given positive integer k.*

Proof: We again give a reduction from EFFECTIVE PREFERENCE for STV in which a appears in first place at least in one vote. Consider an incomplete profile p in which $n + 1$ votes have been cast, a has at least one first place vote, one vote remains unknown, and we wish to decide if a can win. We construct a new election from p. We make k copies of p. In the ith copy $c(p)_i$, we subscript each candidate with the integer i. We add n new additional votes, and one new candidate b. We put b at the top of each of these new votes, and rank all the other candidates except a_i in any order within these n votes. The ranking of the candidates a_i is left unknown but beneath b. In each $c(p)_i$, we place b in last place except for one vote where a_i is in first place (by assumption, one such vote must exist) where we place b in second place and shift all other candidates down. Finally, for each candidate in $c(p)_j$ not in $c(p)_i$ except for a_j, we rank them in any order at the bottom of the votes in $c(p)_i$. The ranking of the candidates a_i is again left unknown but beneath b. We observe that b will survive till all but one candidate has been eliminated from one of the $c(p)_i$. We also observe that if a_i remains in the election, then the score given to each candidate by STV remains the same as in the original election so the candidates in $c(p)_i$ are eliminated in the same order up till the point a_i is eliminated. Suppose a cannot win the original election. Then a_i will always be eliminated before the final round. The second choice vote for b is transferred. Since b now has at least $n + 1$ votes, b is unbeatable and must win the election. Suppose, on the other hand, that a can win the original election. Then a_i can survive to be the last remaining candidate in $c(p)_i$. We can assume b is ranked at the bottom of the unknown votes of all the candidates with an index i and above all the candidates with an index j different to i. Thus a_i has $n + 1$ votes. If we have the corresponding ranking in the other unknown votes, a_j for $j \neq i$ will also survive. Since b has only n votes, b will be eliminated. It is now possible for any of the candidates, a_i where $1 \leq i \leq k$ to win depending on how exactly the a_i are ranked in the different votes. Thus the set of possible winners is $\{a_i \mid 1 \leq i \leq k\}$ plus b if a is not a necessary winner in the original election. Hence, if a is a possible winner in the original election, the size of the set of possible winners is greater than or equal to k, whilst if it is not, the set is of size 1. If we know that $|PW^*| < k|PW|$, then $|PW^*| < k$ guarantees that $|PW| = 1$, that b is the necessary winner, and hence that a is not a possible winner in the original election. □

Similarly, we cannot approximate efficiently the set of necessary winners within some fixed ratio.

Theorem 5. *Given an incomplete unweighted profile over an unbounded number of candidates, it is NP-hard to return a subset of its necessary winners, NW^*, in which we guarantee $|NW^*| > \frac{1}{k}|NW|$ whenever $|NW| > 0$ for any given positive integer k.*

Proof: In the reduction used in the proof of Theorem 4, $|NW| = 1$ if a is a possible winner in the original election and 0 otherwise. Suppose a is a possible winner. Then in the new election, $|NW| = 1$. As $|NW^*| > \frac{1}{k}|NW|$, it follows that $|NW^*| = 1$. Thus, the size of NW^* will determine if a is possible winner. □

3.2. Weighted votes

We now show that, with weighted votes, when we have at least a certain number of candidates, it is intractable in the worst-case to to compute the possible and necessary winners, both with an unbounded number of candidates and with a bounded number of candidates.

3.2.1. Bounded number of candidates.

When we have weighted agents and a bounded number of candidates, computing possible and necessary winners is computationally intractable in the worst case. To show this, we will consider as voting rule the cup (also known as knockout or sequential majority voting) rule.

Definition 12 (cup rule). *The cup rule maps profiles into a single candidate, called the winner. The winner is the result of a series of pairwise majority elections between candidates. The cup rule is defined by a binary tree (also called an agenda) with one candidate labelled to each leaf. Each non-leaf is assigned to the winner of the majority election between the candidates labelling the children. The candidate labelling the root is the overall winner.*

In all our proofs, we use a balanced binary tree, but this is not necessary for the results to go through.

Given an incomplete weighted profile over a bounded number of candidates and a candidate a, we say WEIGHTEDBOUNDEDPOSSIBLEWINNER(a) holds iff a is a possible winner of the election.

Theorem 6. *Given a candidate a,* `WeightedBoundedPossibleWinner`(a) *is NP-complete when there are* 3 *or more candidates.*

Proof: Membership in NP follows by giving a completion of the profile in which a wins. We give a reduction from the number partitioning problem. Consider the cup where A plays B and the winner thus plays C. We have a bag of integers, k_i with sum $2k$ and we wish to decide if they can be partitioned into two bags, each with sum k. We will show that we can build an election where we can complete the incomplete weighted profile so that C wins (i.e., C is a possible winner) iff such a partition exists. We suppose the following votes are given: 1 vote for $C > B > A$ of weight

1, 1 vote $C > A > B$ of weight $2k-1$, and 1 vote $B > C > A$ of weight $2k-1$. Hence, C is ahead of A by $4k-1$ votes, C is ahead of B by 1 vote and B is ahead of A by 1 vote. For eack k_i in the bag of integers, we have an incomplete vote of weight $2k_i$ in which $A > C$ is fixed, but the rest of the vote is incomplete. We are sure A beats C in the final result by 1 vote whatever completion takes place. We now show that the incomplete weighted profile can be completed so that B beats A and C beats B (an thus C is a possible winner) iff there is a partition of size k.

Assume that such a partition exists, and that votes in one partition have $A > C > B$ and the votes in the other have $B > A > C$. Thus, B beats A overall and C beats B. Thus C is the winner.

On the other hand, suppose there is a way to complete the preferences so that C wins. This can only happen if B beats A and C then beats B. In fact, if A beats B in the first round, A will beat C in the second round, since we have shown before that A beats C overall, and then A will be the final winner. For C to beat B, at least half the weight of incomplete votes must rank C above B. Similarly, for B to beat A, at least half the weight of incomplete votes must rank B above A. Since all votes rank A above C, B cannot be both above A and below C. Thus precisely half the weight of incomplete votes ranks B above A and half ranks C above B. Hence, we have a partition of equal weight. Therefore, we can complete the incomplete profile so that C wins iff there is a partition of size k. □

Given an incomplete weighted profile over a bounded number of candidates and a candidate a, we say WEIGHTED BOUNDEDNECESSARYWINNER(a) holds iff a is a necessary winner of the election.

Theorem 69 in [5] shows that constructive manipulation (i.e., a manipulation done to make somebody win) made by a coalition of agents with weighted votes is polynomial. Thus, it follows immediately that, if we have an partial weighted profile, it is polynomial to determine if a is a necessary winner. It is, in fact, sufficient to test if the manipulation for any other candidate is successful. This is polynomial, since we are assuming a bounded number of candidates. However, this is a restricted case since incompleteness occurs only in a specific form. The following theorem shows that, in general, WEIGHTEDBOUNDEDNECESSARYWINNER(a) is a computationally intractable problem.

Theorem 7. *Given a candidate a,* `WeightedBoundedNecessaryWinner`(a) *is coNP-complete when there are* 4 *or more candidates.*

Proof:

The complement problem is in NP since we can show membership by giving a completion of the profile in which some b different from a wins.

To show that with 4 or more candidates it is NP-hard to decide if a is not a necessary winner we consider the cup rule and we perform a reduction from the number partitioning problem.

We have a bag of positive integers, k_i with sum $2k$ and we wish to decide if they can be partitioned into two bags, each with sum k. We will build an incomplete profile such that a is not a necessary winner if and only if there is such a partition.

We construct an incomplete profile where the following votes are given: 1 vote for $b > a > d > c$ of weight 1, 1 vote $b > a > c > d$ of weight $2k-1$, and 1 vote $a > d > b > c$ of weight $2k-1$. For the first number, k_1 in the bag of integers, we have a vote for $a > d > c > b$ of weight $2k_1$. For each other number, k_i where $i > 1$, we have an incomplete vote of weight $2k_i$ in which $c > b$ is fixed, but the rest of the vote is incomplete. We are sure that c beats b in the final result by 1 vote whatever completion takes place. Similarly, we are also sure that a beats c, and a beats d.

Thus, the only winners in the considered cup are a or b. If we complete preferences so that in all incomplete votes we have $a > b$, then a will win overall. We now show that there is a completion that makes b win iff there is a partition of equal weight.

Suppose there is such a partition and that the complete votes in one partition have $d > c > b$ and the complete votes in the other have $c > b > d$. Thus, d beats c overall, b beats d. We suppose also that enough of the incomplete votes are completed with $b > a$ for b to beat a. Thus b is the overall winner.

On the other hand, suppose there is a way to complete the preferences so that b wins. This can only happen if d beats c, b then beats d and b finally beats a. If c beats d in the first round, c will beat b in the second round and then go out to a. For b to beat d, at least half the weight of incomplete votes must rank b above d. Similarly, for d to beat c, at least half the weight of incomplete votes must rank d above c. Since all votes rank c above b, d cannot be both above c and below b. Thus precisely half the weight of incomplete votes ranks d above c and half ranks b above d. Hence, we have a partition of equal weight. Therefore, b can win iff there is a partition of equal weight. Thus, a is not a necessary winner iff there is a partition of size k. □

3.2.2. Unbounded number of candidates.

Theorems 6 and 7 show that testing possible and necessary winners is NP-hard when we have a bounded number of candidates. These worst-case complexity results continue to hold when we have an unbounded number of candidates.

Given an incomplete weighted profile over an unbounded number of candidates and a candidate a, we say WEIGHTEDUNBOUNDEDPOSSIBLEWINNER(a) holds iff a is a possible winner of the election.

Theorem 8. *Given a candidate a,* `WeightedUnboundedPossibleWinner`(a) *is NP-complete when there are* 3 *or more candidates.*

Proof: A witness which can be checked in polynomial time is again a completion of the profile in which a wins. NP-hardness holds using the same argument as Theorem 6.

□

Given an incomplete weighted profile over an unbounded number of candidates and a candidate a, we say WEIGHTEDUNBOUNDEDNECESSARYWINNER holds iff a is a necessary winner of the election.

Theorem 9. *Given a candidate a,* `WeightedUnboundedNecessaryWinner`(a) *is coNP-complete when there are* 4 *or more candidates.*

Proof: A witness for the complement problem which can be checked in polynomial time is again a completion of the profile in which a does not win. NP-hardness holds using the same argument as Theorem 7. □

If we have weighted votes, then, as in the case with unweighted votes, we cannot approximate efficiently the set of possible and necessary winners within some fixed ratio.

Theorem 10. *Given an incomplete weighted profile over an unbounded number of candidates, it is NP-hard to return a superset of its possible winners, PW^*, in which we guarantee $|PW^*| < k|PW|$ for some given positive integer k.*

Proof: It follows from Theorem 4. If it is NP-hard to find a certain approximation of the possible winners when we have unweighted votes, then it is also NP-hard to find this approximation when we have weighted votes. To show this we can use the same proof used to prove Theorem 4 assuming that all the weights are equal to one. □

Theorem 11. *Given an incomplete weighted profile over an unbounded number of candidates, it is NP-hard to return a subset of its necessary winners, NW^*, in which we guarantee $|NW^*| > \frac{1}{k}|NW|$ whenever $|NW| > 0$ for any given positive integer k.*

Proof: It follows from Theorem 5 for the same reasoning performed in the proof of Theorem 10, i.e., if it is NP-hard to find a certain approximation of the necessary winners when we have unweighted votes, then it is also NP-hard to find this approximation when we have weighted votes. To show this we can use the same proof used to prove Theorem 5 assuming that all the weights are equal to one. □

3.3. Summary of the complexity results

All our complexity results are summarized in Table 1. It is possible to see that the only case, where it is computationally easy in the worst-case to find possible and necessary winners, is when we have unweighted votes and a bounded number of candidates. Such a case is of interest to social choice [1] where human elections are analyzed: all the agents have the same importance, thus agents are unweighted, and they vote on a bounded number of candidates. Being able to find easily the set of possible and necessary winners is useful, as we will see in the following, for example to decide when to terminate a preference elicitation process [3]. In fact, when the set of possible winners coincides with the set of necessary winners, elicitation can be stopped, thus avoiding posing useless questions to agents.

If instead we relax one of the conditions above, that is, if we allow for weighted agents, or if we allow for an unbounded number of candidates, then the complexity of testing if a candidate is a possible or a necessary winner becomes an intractable problem in the worst-case. Weights are typical in Artificial Intelligence (AI) applications. For

	Weighted+Bounded	Unweighted+Bounded	Weighted+Unbounded	Unweighted+Unbounded
PW	NP-complete (Th. 6)	P (Th. 1)	NP-complete (Th. 8)	NP-complete (Th. 2)
NW	coNP-compete (Th. 7)	P (Th. 1)	coNP-complete (Th. 9)	coNP-complete (Th. 3)
PW*	?	P (Th. 1)	NP-hard (Th. 10)	NP-hard (Th. 4)
NW*	?	P (Th. 1)	NP-hard (Th. 11)	NP-hard (Th. 5)

Table 1: Complexity results of computing/testing possible and necessary winners, and of finding an upper approximation of possible winners and a lower approximation of necessary winners.

example, they are useful in multi-agent systems where there are different types of agents. Moreover, in AI, differently from human elections, it is very common to have an unbounded number of alternatives to consider when making a decision. The hardness results that we have shown in these settings can be useful as a barrier to manipulation. Gibbard-Satterthwaite's theorem [9, 15] states that, under some reasonable conditions, it is always possible to manipulate an election, i.e., it is always possible that some agent reveals unsincerily his preferences to obtain a better result. Our complexity results show that manipulation is computationally intractable in the worst-case. In fact, since it is intractable to know if a candidate is a possible winner, a coalition of agents may have little incentive to complete their votes unsincerely to make him win. Moreover, since it is intractable to find out if a candidate is a necessary winner, a coalition of agents may have little incentive to complete their votes insincerely to make him loose.

In the cases considered above (that is, unbounded number of candidates and weighted/ unweighted agents) we have shown that, it is intractable in the worst-case not only to test if a candidate is a possible or a necessary winner, but also to find good approximations of such sets of winners. In particular, we have shown that computing an upper approximation of the possible winners and a lower approximation of the necessary winners is an intractable problem in the worst case. We conjecture that such a computation might be easy when we consider a bounded number of candidates (and weighted votes). We plan to investigate such a scenario in the future.

4. Tractable cases

We now determine sufficient conditions on the preference aggregation function that make it polynomial to compute the sets of possible and necessary winners. We recall that such a computation is polynomial when votes are unweighted and there is a bounded number of candidates, while in all the other cases it is intractable in the worst-case. We will show that in all these cases, if we require that the social welfare function associated to the preference aggregation function is IIA and monotone, it is polynomial to compute the possible and necessary winners.

4.1. Unweighted votes

When we have unweighted votes, testing possible and necessary winner, as well as finding good approximations of such sets, is only intractable in the worst-case when we have an unbounded number of candidates. We show that, in this case, if the preference aggregation function is IIA and monotone, if we start from an approximation of the combined result, that can be computed in polynomial time, then also possible and necessary winners can be computed in polynomial time.

When instead we have a bounded number of candidates, we have shown in the previous section that is polynomial to compute possible and necessary winners in general. Thus, trivially, it continues to remain polynomial if we require that the preference aggregation function satisfies such properties.

4.1.1. Unbounded number of candidates.

The problem of computing the combined result is in general computationally intractable in the worst-case. However, we identify some restrictions which allow us to compute a useful approximation of the combined result in polynomial time.

Theorem 12. *Given an incomplete profile, determining if a label is in the combined result is NP-complete.*

Proof: A polynomial witness is a completion of the incomplete profile. To show NP-hardness, we notice that by performing a polynomial number of calls to this problem we can determine if a given candidate is a possible winner. It is in fact sufficient to check if in the combined results, on each edge connecting this candidate to another candidate at least one of the the two relations "greater than" and "incomparable" is present. However, from Theorem 2 we know that testing if a candidate is a possible winner is NP-hard for STV. □

From this result we immediately get the following corollary.

Corollary 12.1. *Given an incomplete profile ip and a social welfare function f, computing the combined result $cr(f, ip)$ is NP-hard.*

We now identify some properties of preference aggregation functions which allow us to compute an upper approximation of the combined result in polynomial time. An upper approximation of a combined result cr is a graph cr^* with the same arcs and the same nodes as cr, and where the set of labels of every arc between a pair of candidates in cr^* is a superset of the set of labels of the same arc in cr. We recall that the set of labels of an arc between A and B in the combined result is called $rel(A, B)$.

We will now give a characterization of this approximation of the combined result, that will allow us to compute it in polynomial time.

Theorem 13. *Given an incomplete profile ip and a social welfare function f that is IIA and monotonic, it is possible to compute in polynomial time an approximation of $cr(f, ip)$, say $cr^*(f, ip)$ such that, given two candidates A and B, $rel^*(A, B) \supseteq rel(A, B)$. Moreover, if $rel^*(A, B) = \{<, >, \sim, =\}$, then either $rel^*(A, B) = rel(A, B)$ or $rel^*(A, B) - rel(A, B) \subseteq \{\sim, =\}$.*

Proof: Since f is IIA, to compute the set $rel(A, B)$, we just need to ask each agent their preference over the pair A and B, and then use f to compute all possible results between A and B. However, if agents have incompleteness between A and B, f has to consider all the possible completions, which is exponential in the number of such agents. However, since f is monotonic, we don't need to consider all possible completions for all agents with incompleteness between A and B, but just two completions: the completion where $A?B$ is replaced with $A < B$ and the completion where $A?B$ is replaced with $A > B$. Notice that, since f is IIA, we don't need to consider complete profiles, but just the part of the profile concerning A and B. This means that transitivity issues do not arise. Function f will return a result for each of these two completions, say AxB and AyB, where $x, y \in \{<, >, =, \sim\}$. Since f is monotonic, the results of all the other completions will necessarily be between x and y in the ordering $>$, ($\sim$ or $=$), $<$. By taking all such relations, we obtain $rel^*(A, B)$, that is a superset of $rel(A, B)$. In fact, monotonicity of f assures that, if we consider profile where $A < B$ and we get a certain result, then considering profiles where A is in a better position w.r.t. B (that is, $A > B$, $A = B$, or $A \sim B$), will give an equal or better situation for A in the result. Thus, $cr^*(f, ip)$ is an upper approximation of the combined result. Moreover, if $rel(A, B) = \{<, >, \sim, =\}$, then there is a completion where $>$ is in the result, that is, the completion where we replace every $A?B$ with $A > B$, and there is a completion where $<$ is the result, that is, the completion where we replace every $A?B$ with $A < B$. Hence, the only labels that can be added from $rel(A, B)$ to $rel^*(A, B)$ are the elements $=$ and $\sim$. □

Example 8. *Consider the Lex rule [1], in which agents are totally ordered and, given any two candidates A and B, the relation between A and B in the result is the relation given by the first agent in the order that does not declare a tie between A and B. Consider the following incomplete profile $((A > C, B > C, A?B); (A > B > C))$. Then $rel^*(A, B) = \{<, =, \sim, >\}$, whereas $rel(A, B) = \{<, \sim, >\}$.* □

We will now show how to determine the possible and necessary winners, given $cr^*(f, ip)$. Consider the arc between a candidate A and a candidate C in $cr^*(f, ip)$. Then, if this arc has the label $A < C$, then A is not a necessary winner, since there is a candidate C which is better than A in some result. If this arc *only* has the label $A < C$, then A is not a possible winner since we must have $A < C$ in all results. Moreover, consider all the arcs between A and every other candidate C. Then, if no such arc has label which includes $A < C$, then A is a necessary winner. Notice, however, that in general, even if there are no arcs connecting A to every other candidate C with the unique label $A < C$, A could not be a possible winner. A could be better than some candidates in every completion, but there might be no completion where it is better than all of them. We will show that this is not the case if f is IIA and monotonic.

We now define Algorithm 1, which, given $cr^*(f, ip)$, computes NW and PW, in polynomial time.

Algorithm 1: Computing NW and PW

Input: $cr^*(f, ip)$, where f: IIA and monotonic preference aggregation function and ip: incomplete profile;
Output: P, N: sets of candidates;
$P \leftarrow \Omega$;
$N \leftarrow \Omega$;
foreach $A \in \Omega$ **do**
  if $\exists\, C \in \Omega$ *such that* $\{<\} \subseteq rel^*(A, C)$ **then**
    $N \leftarrow N - A$;
  if $\exists\, C \in \Omega$ *such that* $\{<\} = rel^*(A, C)$ **then** $P \leftarrow P - A$;
return P, N;

Theorem 14. *Given a preference aggregation function f which is IIA and monotonic, and an incomplete unweighted profile ip over an unbounded number of candidates, and its approximate combined result $cr^*(f, ip)$, Algorithm 1 terminates in $O(m^2)$ time, where $m = |\Omega|$, returning $N = NW$ and $P = PW$.*

Proof: Algorithm 1 considers, in the worst case, each arc exactly once, thus runs in $O(m^2)$ time.

N=NW. By construction of $cr^*(f, ip)$, $<\notin rel^*(A, C)$ iff $<\notin rel(A, C)$. By Algorithm 1, $A \in N$ iff $\forall C, <\notin rel\{A, C\}$, and this implies that there is no result in which there exists an candidate C that beats A. Thus, $A \in NW$. On the contrary, $A \in NW$ iff $A \not< C, \forall C \in \Omega$, for all results, from which, $A \in N$.

$P = PW$. A candidate A is in PW if there is no other candidate which beats it in all results. Thus, there cannot exist any other candidate C such that $<$ is the only label in $rel\{A, C\}$ and, thus by construction, also in $rel^*\{A, C\}$. Thus, $PW \subseteq P$. To show the other inclusion we consider $A \in P$ and we construct a completion of ip such that A wins in its result. First, let us point out that for any candidate A, $A \in P$ iff $\nexists C \in \Omega$, $rel^*(A, C) = \{<\}$. If $\forall C \in \Omega$, $<\notin rel^*(A, C)$, then A is never beaten by any other candidate C and A is NW and, thus, a PW. Secondly, let us consider the case in which A is such that whenever $<\in rel^*(A, C)$, either $\sim$ or $>$ (or both) are also in $rel^*(A, C)$ and let us denote with X such set of candidates. Then for every candidate in $C \in X$ we choose $>$ whenever available and $\sim$ otherwise. This corresponds to replacing $A?C$ with $A > C$ in the incomplete profile. Such choice on AC arcs cannot cause a transitivity inconsistency and thus can be completed to a result in which A is a winner. Finally, let us consider the case in which there is at least a C such that $rel^*(A, C) = \{<, =\}$. If for every other candidate C', $rel^*(A, C')$ contains exactly one label from the set: $\{>, \sim, =\}$ then we can safely set AC to $=$ since there is, for sure, a result with that labeling. Moreover, in such a result A is a winner. Assume, instead, that there is at least a candidate C' such that $|rel^*(A, C') > 1|$. This means that there is at least an agent which has not declared his preference on AC' and that such preference cannot be induced by transitivity closure. We replace $A?C'$ with $A > C'$ everywhere in the profile, we perform the transitive closure of all the modified partially specified POs, and we apply f. We will prove that such transitive closure does not force label $<$ on AC. After the procedure, due to monotonicity, $rel(A, C')$ will contain exactly one label from the set: $\{>, \sim, =\}$. Let us assume that, after the procedure, $A = C'$ and let us now consider $rel(C', C)$. Had it been $rel^*(C', C) = \{<\}$ from the start, this would have forced $rel(A, C) = \{<\}$. However, this is not possible since $A \in P$. This allows us to conclude that $(rel^*(C', C) \cap \{>, \sim, =\}) \neq \emptyset$ and any of such additional labels together with $A = C'$ can never force $A < C$. Clearly, if $A > C'$ or $A \sim C'$, there is no labeling of $C'C$ which can force $A < C$. It should be noticed that any available choice on $C'C$ can always be made safely due to the fact that the function is IIA and that the transitive closure of the profiles has already ruled out inconsistent choices. By iterating the procedure until every ? in the incomplete profile is replaced, we can construct a result of the function in which A is a winner. □

Example 9. *We have already shown that the Pareto rule described in Example 3 is IIA and monotonic. Another social welfare function which is both IIA and monotonic is the Lex rule, described in Example 6.* □

f: IIA + monotone	**Weighted+Bounded**	**Unweighted+Bounded**	**Weighted+Unbounded**	**Unweighted+Unbounded**
PW	P (Th. 17)	P (Th. 15)	P (Th. 16)	P (Th. 14)
NW	P (Th. 17)	P (Th.15)	P (Th. 16)	P (Th. 14)

Table 2: Complexity results of computing possible and necessary winners, given an IIA and monotone preference aggregation function.

4.2. Bounded number of candidates

In the previous section we have shown that when we have an unbounded number of candidates and unweighted votes, it is polynomial to compute possible and necessary winners, thus it is also polynomial if we require that the preference aggregation function is IIA and monotonic.

Theorem 15. *Given a candidate a, if the preference aggregation function is IIA and monotonic, then* `UnweightedBoundedPossibleWinner`(a) *and* `UnweightedBoundedNecessaryWinner`(a) *are both polynomial.*

Proof: It follows directly from Theorem 1. □

4.3. Weighted votes

In the presence of weighted agents, if we assume that the social welfare function is IIA and monotonic, then we can use the same procedure shown above to find the approximate combined result and Algorithm 1 to find possible and necessary winners, since they do not depend on the fact that the agents are weighted or not, but only on IIA and monotonicity.

4.3.1. Unbounded number of candidates.

Theorem 14 states that, when the social welfare function is IIA and monotone, computing possible and necessary winners for unweighted agents and an unbounded number of candidates is polynomial. The same holds also for weighted agents and an unbounded number of candidates.

Theorem 16. *Given a preference aggregation function f which is IIA and monotonic, and an incomplete weighted profile ip over an unbounded number of candidates, and its approximate combined result $cr^*(f, ip)$, it is polynomial to compute the set of possible and necessary winners.*

Proof: It follows from the procedure used to compute $cr^*(f, ip)$, that holds both with and without weighted agents, and by Theorem 14 that depends only by $cr^*(f, ip)$. □

4.3.2. Bounded number of candidates.

From Theorem 16 it follows immediately that, in presence of weighted agents, computing possible and necessary winners continues to be polynomial also when we have a bounded number of candidates.

Theorem 17. *Given a preference aggregation function f which is IIA and monotonic, and an incomplete weighted profile ip over a bounded number of candidates, and its approximate combined result $cr^*(f, ip)$, it is polynomial to compute the set of possible and necessary winners.*

Proof: It follows immediately from Theorem 16. If computing possible and necessary winners is polynomial when we have an unbounded number of candidates, it continues to be polynomial also when we have a bounded number of candidates. □

4.4. Summary

Table 2 summarizes the complexity results of this section. In particular, when we require that the preference aggregation function is IIA and monotone the computation of possible and necessary winners is always polynomial.

5. Elicitation

One use of necessary and possible winners is in eliciting preferences [3]. Preference elicitation is the process of asking queries to agents in order to determine their preferences over candidates. Eliciting preferences takes time and effort. We therefore want to stop elicitation as soon as one candidate has enough support that he must win regardless of any missing preferences. Preference elicitation can be stopped when $NW = PW$, since we have enough information to declare the winners. At the beginning, NW is empty and PW contains all candidates. As preferences are declared, NW grows and PW shrinks. At each step, a candidate in PW can either pass to NW or become a loser. When PW is larger than NW, we can use these two sets to guide preference elicitation and avoid useless work.

In this section we show that deciding when we can stop eliciting preferences is in general computationally intractable in the worst-case but it is polynomial when we have unweighted agents and we require an IIA preference aggregation function.

5.1. Complexity of terminating elicitation

We consider two decision problems. If we elicit complete votes from each agent (e.g. How do you rank all the candidates?), `CoarseElicitationOver` is true iff the winner is determined irrespective of how the remaining agents vote. On the other hand, if we elicit just individual preferences (e.g. Do you prefer Bush to Gore?), `FineElicitationOver` is true iff the winner is determined irrespective of how the undeclared preferences are revealed. Note that in both cases, the missing preferences are assumed to be transitive.

Definition 13 (`CoarseElicitationOver`) **.** *Input: a partial profile. Output: true iff only one candidate can win irrespective of how the remaining agents vote.*

Definition 14 (`FineElicitationOver`) **.** *Input: an incomplete profile. Output: true iff only one candidate can win irrespective of how the incomplete profile is completed.*

`CoarseElicitationOver` and `FineElicitationOver` are in coNP as a polynomial witness for elicitation not being over are two completions of the profile in which different candidates win. Since `CoarseElicitation Over` is a special case of `FineElicitationOver`, it is polynomial to see that if `FineElicitationOver` is polynomial then `CoarseElicitationOver` is too. Similarly, if `CoarseElicitationOver` is coNP-complete then `FineElicitationOver` is too. However, as we show later, these implications do not necessarily reverse. For example, there are voting rules where `CoarseElicitationOver` is polynomial but `FineElicitationOver` is coNP-complete. Our analysis of the complexity of terminating preference elicitation considers, as in the previous sections, two different dimensions: weighted or unweighted agents, and a bounded or unbounded number of candidates.

5.1.1. Unweighted agents.

We now show that, when we have unweighted agents, computing `CoarseElicitationOver` and `FineElicitationOver` are both polynomial.

Given an incomplete profile over a bounded number of candidates, we say UNWEIGHTEDBOUNDEDCOARSEELICITATIONOVER holds iff `CoarseElicitationOver` holds for this election and UNWEIGHTEDBOUNDEDFINEELICITATIONOVER holds iff `FineElicitationOver` holds for this election.

Theorem 18. `UnweightedBoundedFineElicitationOver` *and* `UnweightedBoundedFineElicitationOver` *are polynomial.*

Proof: As we have noticed at the beginning of the Section 3.1, if the number of candidates is bounded, there are only a polynomial number of different votes. We can thus enumerate and evaluate all different votes in polynomial time. Hence computing `CoarseElicitationOver` and `FineElicitationOver` are both polynomial. □

5.1.2. Weighted votes.

We now show that there are voting rules where `CoarseElicitationOver` is polynomial but `FineElicitationOver` is intractable.

Given an incomplete weighted profile over a bounded number of candidates, we say WEIGHTEDBOUNDEDCOARSEELICITATIONOVER holds iff `CoarseElicitationOver` holds for this election and WEIGHTEDBOUNDEDFINEELICITATIONOVER holds iff `FineElicitationOver` holds for this election.

Theorem 19. *For the cup rule,*

- `WeightedBoundedFineElicitationOver` *is coNP-complete when there are 4 or more candidates,*
- `WeightedBoundedCoarseElicitationOver` *is polynomial irrespective of the number of candidates.*

Proof: Theorem 69 in [5] shows that manipulation of the cup rule by a coalition of agents with weighted votes is polynomial. It follows immediately that Coarse Elicitation Over is polynomial too.

To show that `FineElicitationOver` is NP-hard with 4 candidates, we can use a reasoning similar to the one in the proof of Theorem 7, since deciding when `FineElicitationOver` coincides with the problem of determining if there is a necessary winner. For clarity, we give the formal proof in detail.

To show that `FineElicitationOver` is NP-hard with 4 candidates, consider the cup in which A plays B, the winner then plays C, and the winner of this match goes forward to the final match against D. We will reduce number partitioning to deciding if elicitation is over for this cup rule given a particular incomplete profile. Suppose we have a bag of integers, k_i with sum $2k$ and we wish to decide if they can be partitioned into two bags, each with sum k. We construct an incomplete profile in which the following weighted votes are completely fixed: 1 vote for $C > D > B > A$ of weight 1, 1 vote $C > D > A > B$ of weight 2k - 1, and 1 vote $D > B > C > A$ of weight 2k - 1. For the first number, k_1 in the bag of integers, we have a fixed vote for $D > B > A > C$ of weight $2k_1$. For each other number, k_i where $i > 1$, we have an incomplete vote of weight $2k_i$ in which $A > C$ is fixed but the rest of the vote is unspecified. We are sure A beats C in the final result by 1 vote whatever happens. Similarly, we are also sure that D beats A, and D beats B. Thus, the only winners of the cup rule are D or C. If in all the incomplete votes we have $D > C$, then D will win overall. We now show that C can win iff there is a partition of equal weight. Suppose there is such a partition and that the incomplete votes corresponding to one partition have $B > A > C$ whilst the incomplete votes corresponding to the other partition have $A > B > C$. Thus, B beats A overall, and C beats B. We suppose also that enough of the incomplete votes have $C > D$ for C to beat D. Hence C is the winner of the cup rule and D does not win. On the other hand, suppose C wins. This can only happen if B beats A, C then beats B and C finally beats D. If A beats B in the first round, A will beat C in the second round and then go out to D. For C to beat B, at least half the weight of incomplete votes must rank C above B. Similarly, for B to beat A, at least half the weight of incomplete votes must rank B above A. Since all votes rank A above C, B cannot be both above A and below C. Thus precisely half the weight of incomplete votes ranks B above A and half ranks C above B. Hence, we have a partition of equal weight. Therefore, both C and D can win iff there is a partition of equal weight. That is, elicitation is not over iff there is a partition of equal weight. □

Since for the cup rule, on weighted votes, `FineElicitationOver` is coNP-complete, when there are 4 or more candidates, then in general, when we have a voting rule on weighted votes with 4 or more candidates, deciding when to terminate elicitation is coNP-complete.

The result shown in Theorem 19 suggests that, if preferences are being combined with the cup rule, we might prefer eliciting whole votes from agents as opposed to individual preferences since we can then easily decide when to stop. Computational complexity can thus motivate the choice of an elicitation strategy. Note that for the cup rule with just 3 or fewer candidates, it is polynomial to decide if elicitation is over.

Theorem 20. *For the cup rule,* `WeightedBoundedFineElicitationOver` *and* `WeightedBoundedCoarseElicitationOver` *are both polynomial with 3 or fewer candidates.*

Proof: For 2 candidates, the cup rule degenerates to the majority rule, and `FineElicitationOver` degenerates to `CoarseElicitationOver`. In this case, elicitation can be terminated iff a majority in weight of votes prefer one candidate. For 3 candidates, without loss of generality, we consider the cup in which A plays B, the winner then plays C. Suppose we have an incomplete profile over these three candidates. For A to win, they must beat B and C in pairwise elections. We do not care about the ordering between B and C since if A wins, B and C do not meet. Thus, we complete the profile placing A above B and C wherever possible, and ordering B and C as we wish. To see if B can win, we complete the profile in an analogous fashion. Finally, for C to win they must beat the winner of A and B. We therefore consider two completions of the profile: one in which C is placed above A, and A above B wherever possible, and the second in which C is placed above B, and B above A wherever possible. In total, we have just four completions to consider. These can be tested in polynomial time. Eliciting preferences can be terminated iff the same candidate wins in each case. Given a partial vote, we complete the profile in a similar way to test `CoarseElicitationOver`. □

5.1.3. Uncertainty about votes

Many of our results so far have considered weighted votes. One reason to consider weighted votes is that they inform us about unweighted votes when we have uncertainty about the votes cast. Given a probability distribution over the votes, a number $r \in [0, 1]$ and a candidate, EVALUATION is the problem of deciding if the probability of the candidate winning is strictly greater than r [4].

Theorem 21. `FineElicitationOver` *is NP-hard for a voting rule on weighted votes implies* `Evaluation` *with the same rule on unweighted votes is also NP-hard.*

Proof: We reduce `FineElicitationOver` to Evaluation. Each agent of weight k is replaced by k agents of weight 1 whose votes are perfectly correlated. We then construct a joint probability distribution over the votes so that each completion is drawn with the correct frequency. We set r = 0. `Evaluation` then decides `FineElicitation-Over`. □

Note that the reduction can take on board any restriction on the voting rule or election. For example, if `FineElicitationOver` is NP-hard for weighted votes with 3 or more candidates then `Evaluation` is NP-hard for unweighted votes with 3 or more candidates. In a similar fashion, we can show that if `CoarseElicitationOver` on weighted votes is NP-hard then `Evaluation` is also. However, this is a weaker result as it has a more specific hypothesis. There are voting rules like the cup rule for which `CoarseElicitationOver` is polynomial but `FineElicitationOver` is NP-hard. As simple corollary of this theorem is that we can conclude that `Evaluation` for the cup rule is NP-hard.

Corollary 21.1. `Evaluation` *for the cup rule with 4 or more candidates is NP-hard.*

Proof: It follows from Theorem 19 and Theorem 21. □.

5.2. A tractable case

We now identify a property of the preference aggregation function that allows us to terminate the elicitation process in polynomial time. We focus on the more general elicitation problem, i.e., on `FineElicitationOver`, that is the problem of deciding when to terminate the elicitation process if we elicit only parts of the agents' preferences.

We focus on unweighted votes. Moreover, we consider an unbounded number of candidates, since we know that, when we have unweighted votes and a bounded number of candidates the problem of terminating elicitation is already polynomial. In this context we show that, if the preference aggregation function is IIA, then we can compute in polynomial time the set of winners of the elicitation process.

Theorem 22. *If f is IIA, then determining the set of winners for f in a preference elicitation is polynomial in the number of agents and candidates.*

Proof: We recall that preference elicitation can be stopped when $NW = PW$, since we have enough information to declare the winners. At the beginning, NW is empty and PW contains all candidates. As preferences are declared, NW grows and PW shrinks. At each step, a candidate in PW can either pass to NW or become a loser. When PW is larger than NW, we can use these two sets to guide preference elicitation. If the preference aggregation function is IIA, then to determine if a candidate $A \in PW - NW$ is a loser or a necessary winner, it is enough to ask agents to declare their preferences over all pairs involving A and another candidate, say B, in PW. Moreover, IIA allows us to consider just one profile when computing the relations between A and B in the result, and guarantees that the result is a precise relation, that is, either <, or >, or =, or ∼. To determine all the winners, we thus need to know the relations between A and B for all $A \in PW - NW$ and $B \in PW$. There are examples where all such pairs must be considered. Algorithm 2, in $O(|PW|^2)$ steps eliminates enough incompleteness to determine the winners. At each step, the algorithm asks each agent to express their preferences on a pair of candidates (via procedure $ask(A, B)$) and aggregates such preferences via function f. Since we assume that all the social welfare functions that we consider are polynomially computable, it follows that the whole computation is polynomial in the number of agents and candidates. □

If we use the results of the previous sections, under certain conditions, we know how to compute efficiently the necessary winners and the possible winners. Thus the output of Algorithm 1 can be given as the input to Algorithm 2.

Algorithm 2: Winner determination

```
Input: PW, NW: sets of candidates; f: preference aggregation function;
Output: W: set of candidates;
wins: bool;
P ← PW; N ← NW;
while P ≠ N do
    choose A ∈ P − N;
    wins ← true; P_A ← P − {A};
    repeat
        choose B ∈ P_A;
        if ∃ an agent such that A?B then
            ask(A,B);
            compute f(A,B);
        if f(A, B) = (A > B) then
            P ← P − {B};
        if f(A, B) = (A < B) then
            P ← P − {A}; wins ← false;
        P_A ← P_A − {B};
    until f(A, B) = (A < B) or P_A = ∅ ;
    if wins = true then
        N ← N ∪ {A};
W ← N;
return W;
```

6. Related work

In [10] preference aggregation functions for combining incomplete total orders are considered. Compared to our work, we permit both incompleteness and incomparability, while they allow only for incompleteness. Second, they consider social choice functions which return the (non-empty) set of winners. Instead, we consider social welfare functions which return a complete partial order. Social welfare functions give a finer grained view of the result. Third, they consider specific voting rules like the Borda procedure whilst we have focused on general properties that ensure tractability.

We have shown that with weighted votes and an unbounded number of candidates, testing possible winners for STV is NP-hard. Thus, we can conclude that in general testing possible winners of an election is NP-hard. However, it should be noticed that for many rules used in practice including some positional scoring rules [10], deciding if a candidate is a possible winner is polynomial. The complexity of computing possible winners is related to the complexity of manipulating an election [10]. For instance, it is NP-complete to determine for the Borda, Copeland, Maximin and STV rules if a coalition can cast weighted votes to ensure a given winner [4]. It follows therefore that with weighted votes, deciding if a candidate is a possible winner is NP-hard for these rules.

Conitzer and Sandholm prove that `CoarseElicitationOver` and `FineElicitationOver` are coNP-complete for STV when votes are unweighted and the number of candidates is unbounded [7]. On the other hand, `CoarseElicitationOver` and `FineElicitationOver` are polynomial for the plurality, Borda, veto and Copeland rules with any number of candidates [7]. With weighted votes, deciding if elicitation is over can be intractable even when the number of candidates is small. For example, Conitzer and Sandholm prove that `CoarseElicitationOver` and `FineElicitationOver` are coNPcomplete for STV when votes are weighted and there are just 4 (or more) candidates [7]. However, `CoarseElicitationOver` and `FineElicitationOver` are polynomial for the plurality, Borda, veto and Copeland with weighted votes and any number of candidates [7]. We have shown that, if we have weighted votes and a bounded number of candidates, there are voting rules where `CoarseElicitationOver` is polynomial but `FineElicitationOver` is intractable.

7. Future work

Some of our results use IIA which is a strong assumption. However, we use it just to show that intractability is not inevitable on incomplete profiles. We plan to find other cases where IIA can be relaxed. We also plan to consider the addition of constraints to agents' preferences. This means that preference aggregation must take into account the feasibility of the candidates. Thus possible and necessary winners must now be feasible. It is also important to consider compact knowledge representation formalisms to express agents' preferences, such as CP-nets and soft constraints. Possible and necessary winners should then be defined directly from such compact representations, and preference elicitation should concern statements allowed in the representation language. Finally, a possibility or a probability distribution over the completions of an incomplete preference relation can be used to provide additional information when computing possible and necessary winners.

References

[1] K. J. Arrow, A. K. Sen, and K. Suzumara. *Handbook of Social Choice and Welfare.* North-Holland, Elsevier, 2002.

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

[3] L. Chen and P. Pu. Survey of preference elicitation methods. Technical Report IC/200467, Swiss Federal Institute of Technology in Lausanne (EPFL), 2004.

[4] Conitzer and T. Sandholm. Complexity of manipulating an election with few candidates. In *Proceedings of AAAI-02*, pages 314–319, 2002.

[5] V. Conitzer. Computational aspects of preference aggregation. In *PhD Thesis*. Computer Science Department, Carnagie Mellon University, 2006.

[6] V. Conitzer, J. Lang, and T. Sandholm. How many candidates are required to make an election hard to manipulate? In *Proceedings of TARK-03*, pages 201–214, 2003.

[7] V. Conitzer and T. Sandholm. Vote elicitation: complexity and strategy-proofness. In *Proceedings of AAAI-02*, pages 392–397, 2002.

[8] Vincent Conitzer and Tuomas Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *AAAI*. AAAI Press, 2006.

[9] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(3):587–601, 1973.

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

[11] J. Lang. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence*, 42(1):37–71, 2004.

[12] M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Incompleteness and incomparability in preference aggregation. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1464–1469. AAAI Press, 2007.

[13] M. S. Pini, F. Rossi, K. Brent Venable, and T. Walsh. Computing possible and necessary winners from incomplete partially-ordered preferences. In *Proceeding of ECAI-06*, pages 767–768, 2006.

[14] F. Rossi, K. B. Venable, and T. Walsh. mCP nets: representing and reasoning with preferences of multiple agents. In *AAAI-2004*, pages 322–327, 2004.

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

[16] T. Walsh. Complexity of terminating preference elicitation. In *AAMAS'08*, 2008.