Evaluation of election outcomes under uncertainty

Noam Hazon[1], Yonatan Aumann[1], Sarit Kraus[1], Michael Wooldridge[2]

[1]Department of Computer Science
Bar-Ilan University
Israel
{hazonn,aumann,sarit}@cs.biu.ac.il

[2]Department of Computer Science
University of Liverpool
United Kingdom
mjw@csc.liv.ac.uk

ABSTRACT

We investigate the extent to which it is possible to evaluate the probability of a particular candidate winning an election, given imperfect information about the preferences of the electorate. We assume that for each voter, we have a probability distribution over a set of preference orderings. Thus, for each voter, we have a number of possible preference orderings – we do not know which of these orderings actually represents the voter's preferences, but we know for each one the probability that it does. We give a polynomial algorithm to solve the problem of computing the probability that a given candidate will win when the number of candidates is a constant. However, when the number of candidates is not bounded, we prove that the problem becomes #P-Hard for the Plurality, Borda, and Copeland voting protocols. We further show that even evaluating if a candidate has *any* chance to be a winner is NP-Complete for the Plurality voting protocol, in the weighted voters case. We give a polynomial algorithm for this problem when the voters weights are equal.

1. INTRODUCTION

In many multi-agent environments it is desirable to have a mechanism which enables the agents within a system to make a collective decision on a given issue. The means by which such a collective decision is made is typically a *voting procedure*. A classic, much studied issue in the political science literature is the design of voting procedures that, given the (typically different) preferences of voters within a system, will result in an outcome that will be acceptable to most of the voters, i.e., that will as closely as possible reflect the preferences of voters.

When considering voting procedures from a computational perspective, many interesting theoretical questions arise. Perhaps the most natural question from a computer scientist perspective is: are the voting protocols to select a winning outcome efficiently computable, given all the agents preferences? Fortunately, it seems that relatively few voting protocols are hard to compute [4]. Perhaps more interestingly are questions related to the complexity of *manipulating* a voting procedure. Famously, Gibbard-Satterthwaite showed that, if there are three or more candidates, then in any non-dictatorial voting system, there are situations in which an agent is better off voting strategically (i.e., against its preferences) [10, 12]. This is generally regarded as a very negative result, since it implies that in any realistic circumstance, it is possible for voters to benefit by being "insincere". Here, however, computational complexity is viewed as a *positive* property. The basic issue here is how hard it is for an agent to compute the most beneficial manipulation. This question was studied by [3], [2] and [7], who explored the computational complexity of voting protocols when the number of outcomes is unbounded. [6] and [8] analyzed the computational complexity of manipulating different voting protocols with a constant number of outcomes. Of course, the manipulation of elections is not restricted to voters: manipulation is also possible by election officers – those responsible for organizing an election. [5] and [11] investigated the extent to which voting systems can be manipulated by election officers. However, most of the results mentioned above assume *perfect information* about all the voters preferences, which seems a very unrealistic assumption in real world settings.

In this work, we investigate voting systems under an *imperfect information* model. We assume that what is known about an electorate is the following. For each voter, we have *a probability distribution over a set of preference orderings*. The idea is that although we do not know a voter's preference ordering exactly, we know that it is one of a set of possible orderings (typically a subset of the overall set of possible preference orders), and we have a probability distribution over these. This information may be estimated using historical data. In this setting, the following fundamental question arises: given such an incomplete information model of voter preferences and a particular voting system, how hard is it to compute the probability that a particular candidate will win? To the best of our knowledge, this question is not addressed in the existing literature[1].

The motivation for investigating this question is not merely theoretical interest (which is, of course, by itself a legitimate thing). In many situations, it might be beneficial to try to foresee the probability of an outcome being chosen using only partial knowledge about the other agents preferences, which is modeled by a probability distribution as we have described. One area is the avoidance of strategic voting by coalition of manipulators. Suppose that agent A wants to vote for an outcome which is its most preferred one. Another manipulator agent, B, could try to convince A that its outcome does not have any chance to be the winner so he should directly vote for his second preferred outcome; otherwise this outcome will also lose to A's least preferred candidate. Due to lack of exact knowledge how the other agents will vote, A may be convinced by B. Alternatively, A can estimate the other agent's probabilities to vote for the

[1]The exception is the work of [6] but their result holds only for weighted voters with un-bounded weights as we will show later.

outcomes, by asking people who knows them or by using the history of their former votes on the same issue. The ability to calculate the probability of an outcome to win should help A to decide whether B is right.

This ability to calculate the probability of an outcome winning might also help other domains. For example, (and somewhat more speculatively), consider large multi-agent environments, in which there is a need to keep communication to a minimum. The voting process inevitably requires communication between the election officer and the voters in order to elicit their preferences. However, one way to reduce the communication load is to calculate the probabilities on the agents preferences from their voting history and then calculate the probability of each outcome to win: the winning outcome is then the one which gets the highest probability to be the winner. In this way, we simulate a voting process by choosing the successful outcome without the need to use communication at all. (This method might be extended to a more sophisticated protocol which uses limited communication by asking only a partial set of the voters about their current preferences, although we do not investigate this possibility here.)

We therefore analyze the ability to calculate the probability of an outcome to win in various different settings. We first give some background and review some common voting systems in Section 2. We formally define the above mentioned "evaluation" question in Definition 1. In Section 3, we give a polynomial algorithm to answer the evaluation problem if the number of outcomes is a constant number, and we show that [6] result hold only for unbounded weights weighted voters. If the number of candidates is not bounded, the evaluation problem becomes much harder: we show in Section 4 that even for the Plurality, Borda, and Copeland voting protocols the problem is #P-Hard. We then analyze a simpler question, (the CHANCE-EVALUATION problem – Definition 7): can we only distinguish between the case where a candidate has any chance to be the winner from the case where its probability to be the winner is zero? Surprisingly, this problem is shown to be NP-Complete even for the Plurality voting protocol, when not all the voters have equal weights. We also give a polynomial time algorithm when all the voters have equal weights.

Table 1 summarizes our results. For comparison, we also include results from [6] in bold font. (Parentheses near a complexity class indicates the voting protocols for which the results have been proved – for example, p is for Plurality and b is for Borda; an ellipsis indicates that the results hold for a large variety of voting protocols.)

2. PRELIMINARY DEFINITIONS

Underpinning our work is the notion of a *social choice domain*. Formally, a social choice domain is an $(2n+2)$-tuple $S = \langle V, w_1, \ldots, w_n, \Omega, \succ_1, \ldots, \succ_n \rangle$ where $V = \{1, \ldots, n\}$ is a non-empty set of *voters* – the electorate; $w_i \in \mathbb{N}$ is a weight for each $i \in V$, to represent the decision power of a given voter in a voting setting where not all voters are considered equal (rational weights can be converted to integers by multiplying them by all the weights' denominators); $\Omega = \{\omega_1, \ldots, \omega_m\}$ is a non-empty set of *outcomes*, or *candidates* – the things the voters are trying to decide over; and $\succ_i \subseteq \Omega \times \Omega$ is a (strict) *preference relation* over Ω, for each $i \in V$, which is usually private to i.

The *preference aggregation problem* is that of combining the preference relations $\succ_i$ to obtain a *social preference order* $\succ$, and the general problem of social choice theory is to find some way of aggregating preference relations in such a way that certain principles (such as the Pareto condition) are satisfied [1]. Generating the social preference order is commonly done by a *voting system* which specifies the form of the ballot, the set of allowable votes, and the the voting protocol (an algorithm for determining the outcome). We are concerned with settings in which we simply want to select one outcome from Ω, and the voting protocol runs in polynomial time.

We now review some common voting systems in the case of un-weighted votes (i.e., the case where $w_i = 1$ for all i). The evaluation of a voting protocol for weighted votes is done by simply replacing the vote of each voter i with w_i identical un-weighted votes. In general, voting systems can be classified based on their ballot type. In *binary* voting systems a voter either votes or does not vote for a given candidate. In *ranked* voting systems, each voter ranks the candidates in order of preference. We represent this ballot as a vector where the first candidate is the most preferred candidate, the second one is the second preferred candidate and so on. *Condorcet* systems (or *pairwise* systems) are a class of ranked voting systems that meet the *Condorcet criterion*. That is, the candidate who, when compared in turn with each of the other candidates, is preferred over the other candidate is always declared to be the winner, if such candidate exists.

- *Binary voting systems*:
  - *Plurality (aka. first-past-the-post, relative majority, or winner-take-all).* Each voter votes for one candidate, and the candidate that receives the most votes wins (even if it receives less than a majority of votes).
  - *Approval voting.* Voters may vote for as many candidates as they like. The candidate that receives the most approval votes wins.
- *Preferential voting systems*:
  - *Instant-runoff voting (IRV).* The voters rank candidates in order of preference. If no candidate receives an overall majority (more than half of the votes) of first choices, the candidates with fewest votes are eliminated one by one, and ballots cast for those candidates are recounted for the next choice candidate until the winner achieves a majority among remaining candidates.
  - *Contingent Vote (aka. plurality with run-off).* The contingent vote is the same as IRV except that all but the two candidates with most votes are eliminated after the first iteration; therefore there are always only two iterations.
  - *Supplementary Vote.* The Supplementary vote is a variant of Contingent Vote. The difference is only in the ballot type; voters express a first and second choice of candidate only while under the Contingent Vote they must rank all of them.
  - *Borda.* Voters rank candidates in order of preference. Then for each voter, a candidate receives m points if it is the voter's top choice, $m-1$ if it

Candidates number	Weights	Chance-Evaluation	Evaluation
constant	equal	$\mathrm{P}_{(p,b,c,m,i,\ldots)}$	$\mathrm{P}_{(p,b,c,m,i,\ldots)}$
	bounded	$\mathrm{P}_{(p,b,c,m,i,\ldots)}$	$\mathrm{P}_{(p,b,c,m,i,\ldots)}$
	un-bounded	**NP-Hard**$_{(b,c,m,i)}$	**NP-Hard**$_{(b,c,m,i)}$
parameter	equal	$\mathrm{P}_{(p)}$	#P-Hard$_{(p,b,c)}$
	bounded	NP-Complete$_{(p)}$	#P-Hard$_{(p,b,c)}$
	un-bounded	NP-Complete$_{(p)}$	#P-Hard$_{(p,b,c)}$

Table 1: Summary of results. Results in bold are due to [6]. The parentheses near a complexity class indicates the voting protocols for which the results have been proved. Key: p=plurality, b=borda, c=copeland, m=minimax, i=irv, ...=many more voting protocols

is the second choice, ..., 1 if it is the last. The candidate with the most points wins.

- *Condorcet systems*:
  - *Copeland (aka. Tournament).* The winner is the candidate that wins the most pairwise contests (in a pairwise contest, a candidate wins if it is preferred over the other candidate by more than half of the voters). The score for every candidate is 1 point when it wins, -1 when it loses and 0 if the pairwise contest ends with a draw. The candidate with the most points wins.
  - *Minimax.* If no candidate is undefeated, the candidate that is defeated by the fewest votes in its worst defeat wins.
  - *Ranked pairs.* Tally the vote count comparing each pair of candidates. Sort the pairs, by the largest margin of victory first to smallest last. Then create a directed *majority graph*, where the nodes are the candidates, and an edge (ω, ω') means that ω would beat ω' in a pairwise simple majority ballot. The graph is built by starting with the pair with the largest number of winning votes, and adding one pair in turn to the graph as long as they do not create a cycle (which would create an ambiguity). The completed graph shows the winner- the node with in-degree of zero.

For breaking ties, we consider two alternatives. We can choose one candidate randomly among all the tied candidates, alternatively, we simply select the first candidate according to a pre-defined lexicographic order. Our results can be easily extended to hold for other tie-breaking methods.

Now, a voter will not usually know the preferences of the other individual voters – but he may know the *probability* that a voter will vote for a specific candidate, or the probability that he will prefer one candidate over another. If all probabilities are 0 or 1 then the scenario is one of *perfect information*, otherwise it is one of *imperfect information*. To model the imperfect information, it is assumed that we have for each voter at most k possible preference orders, which are permutations over the available alternatives. Each such order is associated with a non-zero probability that this voter will choose to vote it, and the sum of probabilities of the given preference orders is one; all the other possible preference orders which are not explicitly given are assumed to have a probability of zero.

We consider the case where the voters choices are independent. If we collect from each voter just one preference order (from the ones that are associated with him) we get a voting scenario, from which the winner can be calculated using one of the voting protocols listed above (Plurality, Borda, ...). The probability of any given voting scenario occurring is simply the multiplication of the probabilities of its preference orders from the different voters. We are now ready to define our main problem.

Definition 1. *[EVALUATION] Given a social choice domain, an imperfect information model of voters' preferences, as described above, and a specific candidate, ω^*, what is the probability that ω^* will be chosen?*

The answer to this question is the sum of probabilities of all the voting scenarios where ω^* wins. Note that the complexity of this problem is a function of the number of voters (n), the number of outcomes (m), and the number of possible non-zero probability preference orders for each voter (k). In the following sections, we analyze the complexity of the problem in two main different scenarios: where the number of candidates is bounded by a constant, and when it is not bounded.

3. CONSTANT NUMBER OF CANDIDATES

In many real-world scenarios the number of alternatives is small and can be bounded by a constant. For example, if a group of agents want to decide on a full hour to meet in a given day, the number of alternatives is always 24. In this section we will show a polynomial algorithm for the EVALUATION problem under the assumption of a constant number of alternatives[2].

The key to the efficiency of our algorithm is the distinction between a voting scenario to a *voting result*. In a voting scenario we know for each voter which preference order he votes for. But to identify a winning candidate, we actually do not care which voter votes for each candidate; rather, we are concerned with what the total number of votes are. That is a voting result. Many voting scenarios may lead to the same voting result. For example, suppose we use the Plurality protocol with three voters and two candidates, ω_1 and ω_2. Suppose also that all the voters do not have a probability of 1 to vote for one of the candidates. Thus, there are three voting scenarios with the same voting result of two votes for ω_1 and one vote for ω_2. After we present

[2]We thank Efrat Manisterski for her contribution in developing this algorithm

the algorithm, we describe different ways to represent voting results for many common voting protocols.

Let us first describe the algorithm where all the voters weights are equal. We use a dynamic programming approach to enumerate all the possible voting results of a given voting protocol for n voters and calculate their probability. It is done by using the possible voting results for $n-1$ voters and their probabilities, which in turn done by using the voting results of $n-2$ voters, and so on. Our algorithm builds a table where the rows are all the possible voting results for n voters and the columns represent the voters. We denote by $T[\vec{i}, j]$ the cell in the table at the row which represents the voting result vector $\vec{i}$, and at column j. In any stage, the algorithm only requires memory to hold 2 columns.

Algorithm 1 VotingResult(table T, preference orders for each voter)

```
Init T[.,.] ← 0,    T[0⃗, 0] ← 1.
for i ← 0 to n − 1 do
    for all cells in column i do
        r⃗ ← the voting results of the cell's row
        for j ← 1 to k do
            cur⃗ ← preference order j of voter i + 1
            next⃗ ← the voting result from adding cur to r
            T[next⃗, i+1] ← T[next⃗, i+1] + ( probability of cur ×
            T[r⃗, i])
```

When the algorithm terminates, each cell in the last column contains the probability of that cell's row voting result to occur. We can identify the winner for each voting result according to the specific voting protocol. So, we can answer the EVALUATION problem from definition 1 by simply summing for ω^* the probabilities of the voting results where it wins. Consider the following small example. Suppose we use the plurality voting protocol with 3 candidates, ω_1, ω_2 and ω_3 and 2 voters, V_1 and V_2. The voters preferences are summarized in table 1(a). Table 1(b) shows the table T that is built by the algorithm. Every row represents a voting result which is a vector such that index i counts the number of votes for candidate ω_i. The last column shows the probabilities for every possible voting result with voters V_1 and V_2. Thus, the probability that ω_1 is the winner, assuming a random tie-breaking method is used, is $\frac{1}{2}\cdot\frac{1}{4}+\frac{1}{2}\cdot(\frac{1}{3}\cdot\frac{1}{4}+\frac{1}{2}\cdot\frac{3}{4})+\frac{1}{2}\cdot(\frac{1}{6}\cdot\frac{1}{4})$.

The algorithm running time complexity is roughly $O(n\times$ number of rows of $T\times k)$, and the required memory is $O(2\times$ number of rows of $T)$. The specific voting system determines how to express the possible voting results which in turn determines the number of rows. For many voting systems one of the following three methods can be used to express the possible voting results:

1. a vector of $[0,n]^m$ such that index i represents the number of voters which voted for candidate i.
2. a vector of $[0,n]^{m(m-1)/2}$ which represents the number of voters which preferred the first candidate in each possible pair of candidates.
3. a vector of $[0,n]^{m!}$ which represents the number of voters which voted for each possible preference order permutation.

We will now show which method to use in each voting systems.

(a) A set of voters preferences

V_1	V_2
$\frac{1}{2}$ ω_1	$\frac{1}{4}$ ω_1
$\frac{1}{3}$ ω_2	$\frac{3}{4}$ ω_2
$\frac{1}{6}$ ω_3	

(b) The corresponding table T, that is built by the algorithm

Voting result $(\omega_1,\omega_2,\omega_3)$	0	1	2
0,0,0	1	0	0
1,0,0	0	$\frac{1}{2}$	0
0,1,0	0	$\frac{1}{3}$	0
0,0,1	0	$\frac{1}{6}$	0
2,0,0	0	0	$\frac{1}{2}\cdot\frac{1}{4}$
1,1,0	0	0	$\frac{1}{3}\cdot\frac{1}{4}+\frac{1}{2}\cdot\frac{3}{4}$
1,0,1	0	0	$\frac{1}{6}\cdot\frac{1}{4}$
0,2,0	0	0	$\frac{1}{3}\cdot\frac{3}{4}$
0,1,1	0	0	$\frac{1}{6}\cdot\frac{3}{4}$
0,0,2	0	0	0

Table 2: An example of how algorithm 1 builds a table from a given set of preferences

- *Binary voting systems*:
  - *Plurality.* The first method can be used so the number of rows is n^m and the time complexity is $O(n^{m+1}k)$, but we can give a more precise bound. The actual number of voting results with n voters is exactly the number of options to split the integer number n to exactly m not negative integers, such that their sum is equal to n. Two sums which differ in the order of their summands are considered to be different compositions. This is called a *weak composition of n with exactly m parts*, denote it by $WC(n,m)$. So the running time complexity is $O(k\sum_{i=1}^{n} WC(i,m))$ and the space required is $O(WC(n-1,m)+WC(n,m))$.
  - *Approval voting.* The first method can be used.
- *Preferential voting systems*:
  - *IRV and Contingent Vote*: the third method can be used so the number of rows is $n^{m!}$ and the time complexity is $O(n^{m!+1}k)$. Again, the more precise bound is $O(k\sum_{i=1}^{n} WC(i,m!))$
  - *Supplementary Vote*: because every voter expresses a first and second choice of candidate only, we can use a modified version of the first method – a vector of $[0,n]^{m^2}$ such that each index counts the number of voters which voted for a specific ordered pair of candidates. The number of rows is n^{m^2}, and a precise time bound is $O(k\sum_{i=1}^{n} WC(i,m^2))$
  - *Borda*: If $(mn)^m < n^{m!}$ we shall use a modified version of the first method- a vector of $[0,mn]^m$ which represents the total score for each candidate. A more precise time bound is $O(k\sum_{i=1}^{n} WC(i*m,m))$. If not, we can use the third method and calculate the number of scores for each candidate from the preference orders.
- *Condorcet systems*: (*Copeland*, *Minimax*, *ranked pairs*). The second method can be used, so the number of rows is $n^{m(m-1)/2}$.

When we move to the weighted voters case, [6] expressed the EVALUATION problem as the following decision problem: given a number r, $0 \leq r \leq 1$, is the probability of ω^* winning greater than r? They showed that Borda, Copeland, Minimax and IRV are NP-hard to evaluate even for extremely restricted probability distributions. We show that their results hold only for un-bounded weights.

CLAIM 2. *The EVALUATION problem is in P even for weighted voters, when the weights are in $O(Poly(n))$*

PROOF. Our dynamic programming approach (algorithm 1) can be easily extended to work with weighted voters. Actually, the only thing that has to be changed is the range of possible voting results which determines the number of rows in the table. The number of rows will now become $O(Poly(n)^m)$, $O(Poly(n)^{m(m-1)/2})$ or $O(Poly(n)^{m!})$, depends on the specific voting system (as described before). In all the cases it is still in P. □

This result is understood given the proofs of [6]. They use a reduction from the PARTITION problem, which is known to be hard only for un-bounded numbers [9]. The restriction to bounded weights in our case however, seems to be a very natural and realistic assumption. It seems unlikely that there exist meaningful real world scenarios in which one gives a voter power that is exponentially larger than any other voter's power.

4. THE NUMBER OF CANDIDATES AS A PARAMETER

If we cannot bound the number of candidates, then EVALUATION becomes much harder. In this section, we show that EVALUATION for Borda, Copeland and even for Plurality is #P-Hard in this case. We also define and analyze a seemingly much weaker question for the Plurality voting protocol. Surprisingly, we show that even this problem is hard to compute when not all the voters have equal weights, but we give a polynomial algorithm for the case when all the voters have equal weights.

4.1 The Evaluation problem

Sometimes, the number of candidates cannot be assumed to be a constant, but is necessarily a parameter of the problem. For example, if a group of agents wants to choose one of them as a leader, $m = n$ and thus is not a constant. There are some special cases where the number of voters is a constant and so a naive algorithm, which simply evaluates all possible options and runs in a polynomial time of $O(m^n)$ will suffice. In most cases this is probably not going to happen. Unfortunately, if both the number of voters, n, and the number of candidates, m, are given as parameters, the problem is #P-Hard even for the Plurality, Borda and Copeland voting protocols.

All our #P-Hard reductions will be from a well known #P-Complete problem – a calculation of the permanent of a 01-matrix, or counting the number of perfect matching for a bipartite graph.

DEFINITION 3. *Denote by S_n the set of all permutations of the numbers $1, 2, \ldots, n$. The* permanent *of an n-by-n matrix $A = (a_{i,j})$ is defined as*

$$perm(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

For a bipartite graph $G = (V_1 + V_2, E)$ such that $\forall (i,j) \in E, i \in V_1$ and $j \in V_2$, and $|V_1| = |V_2| = k$, a perfect matching *is a set of edges such that no two edges share a common vertex and every vertex is incident to exactly one edge. The permanent of G's adjacency matrix in fact counts the number of perfect matchings for G.*

We now ready to show the proof for the Plurality voting protocol.

THEOREM 4. *If n and m are not constant, the EVALUATION problem is #P-Hard for the Plurality voting protocol.*

PROOF. Given a bipartite graph G, we build the instance to the EVALUATION problem such that the probability of the chosen candidate to win depends on the number of perfect matchings. We first consider the case where the tie-breaking method is to select the first candidate according to a pre-defined lexicographic order. The voters are all the vertices of V_1 plus three additional voters r_0, z_0 and z_1, and they all have equal weights. The candidates are all the vertices of V_2 plus two additional candidates r and z. For every $i \in V_1$, if $(i,j) \in E$, set the probability that voter i votes for candidate j to $\frac{1}{k}$. Denote the out-degree for every $i \in V_1$ by $o(i)$, and set the probability that voter i votes for candidate r to $1 - \frac{o(i)}{k}$ (in order to have 1 as the sum of the preferences probabilities). Finally, set voter r_0 to vote with probability 1 to candidate r, and voters z_0 and z_1 to vote with probability 1 to z. We ask the EVALUATION algorithm to calculate the probability of z to be the winner. Each perfect matching for G represents a voting scenario where z wins. That is because every candidate $j \in V_2$ has exactly one voter who votes for him, r also has only one voter, r_0, who votes for him, and z has two voters, z_0 and z_1 who vote for him. The probability of every such voting scenario is $(\frac{1}{k})^k$. Conversely, every voting scenario where z wins defines a perfect matching. z has always two votes from z_0 and z_1. Due to the tie-breaking method, if any other candidate has two or more votes z is not the winner. So in every voting scenario where z wins, all the other candidates have one or zero votes. But if one candidate has no votes than one of the other candidates has two votes (according to the pigeon hole principle) in contrast to the fact that z is the winner. So every other candidate has exactly one vote, which corresponds to an edge from the voter $i \in V_1$ to the candidate $j \in V_2$, and this is a perfect matching. Hence, given the probability that z is the winner from the EVALUATION algorithm, dividing it by $(\frac{1}{k})^k$ yields the number of perfect matching. □

The proof can be extended to work with random tie-breaking as well. All we have to do is to add $k+1$ voters such that voter k_j votes for candidate $j \in V_2$ with probability one, and the remaining voter votes for candidate r with probability 1 too. Then in each perfect matching all the candidates are tied so z has a probability of $(\frac{1}{k})^k/(k+2)$ to be the winner in each such scenario, and the proof proceeds in the same way, except that now the answer from the EVALUATION algorithm has to be divided by $(\frac{1}{k})^k/(k+2)$. As for the Borda voting protocol, the reduction is a little more involved.

THEOREM 5. *If n and m are not constant, the EVALUATION problem is #P-Hard for the Borda voting protocol.*

PROOF. Given a bipartite graph G, we build the instance of the EVALUATION problem as follows. We use $2k+2$ voters; k voters $v_0, v_2, \ldots, v_{k-1}$ which correspond to the k vertices of V_1 and $k+2$ additional voters, v_k and $z_0, z_1, \ldots, z_k$. They all have equal weights. There are $k+4$ candidates; k candidates $c_0, c_1, \ldots, c_{k-1}$ and four additional candidates, c_k, g_1, g_2 and z. We define two groups of possible preference orders, each one with $k+1$ items. The first group, P_v, contains for each $j \in V_2$ the vector $(c_j, c_0, \ldots, g_2, \ldots, c_k, z, g_1)$ such that c_j in the first index and c_0 till c_k following in an ascending order, where g_2 replaces c_j in index $j+1$. z and g_1 are located in the end. This preference order vector corresponds to node $j \in V_2$. P_v also contains the vector $(c_k, c_0, \ldots, c_{k-1}, g_2, z, g_1)$. The second group, P_z, contains for each $j \in V_2$ and for $j = k$ the vector $(z, c_k, .., g_1, .., c_0, g_2, c_j)$ with candidate z in the first index, candidates c_k till c_0 following in a descending order, where g_1 replaces c_j in index $k+1-j$, ending with g_2 and c_j. For every $i \in V_1$, if $(i,j) \in E$, set the probability that voter i votes for the order from P_v which corresponds to node j to $\frac{1}{k}$. As before, denote the out-degree for every $i \in V_1$ by $o(i)$, and set the probability that voter i votes for the order from P_v which starts with candidate c_k to $1 - \frac{o(i)}{k}$ (in order to have 1 as the sum of the preferences probabilities). For every $0 \leq i \leq k$, set voter z_i to vote with probability 1 to the order from P_z which ends with candidate c_i. Finally, set voter v_k to vote with probability 1 for the order in P_v which starts with candidate c_k. See figure 1 for example of how to build an instance from a given bipartite graph where $k = 3$. We ask the EVALUATION algorithm to calculate the probability of z to be the winner.

Every perfect matching for G determines a voting scenario. In such scenario each voter $v_0, v_2, \ldots, v_{k-1}$, which represents a node in V_1, vote for exactly one preference order from the group P_v, which represents a node in V_2. The order which does not correspond to a node in V_2 is the one that starts with candidate c_k and it gets its only vote from voter v_k. So the number of points for each candidate $c_i \in \{c_0, c_1, \ldots, c_k\}$ is $(k+1)(k+3-i)+(i+1)$, $(k+6)(k+1)/2$ points for g_2, $k+1$ points for g_1 and $2(k+1)$ points for z so far. The other voters vote always the same and they give for each candidate $c_i \in \{c_0, c_1, \ldots, c_k\}$ $(k+1)(k+i)-(k-1+i)$ points, $2(k+1)$ points for g_2, $(k+6)(k+1)/2$ points for g_1 and $(k+4)(k+1)$ points for z. Overall, candidates $c_0, c_1, \ldots, c_k$ gets k^2+7k+5 points each, g_1 gets $0.5k^2+4.5k+4$ points, g_2 gets $0.5k^2+5.5k+5$ points and z gets k^2+7k+6 points. So z is the winner for every voting scenario which determined by a perfect matching, and the probability of such voting scenario is $(\frac{1}{k})^k$. Conversely, in every voting scenario which does not correspond to a perfect matching for G, there is at least one order from the group P_v which gets more that one vote. Every candidate $c_i \in \{c_0, c_1, \ldots, c_k\}$ has one preference order from P_v when he is located at the first place, which gives $k+4$ points, and in all the other preference orders in P_v he is located at the same place which gives the same number of points. All the preference orders from P_z always give the same number of points. So no matter how the other votes are scattered among the orders of P_v, the candidate in the first place of the order which gets more than one vote, gets at least k^2+7k+6 points, while z's score does not change. According to the tie-breaking method which selects the first candidate from a lexicographic order, z will not be the winner. Hence, given the probability that z is the winner from the EVALUATION algorithm, dividing by $(\frac{1}{k})^k$ yields the number of perfect matchings. □

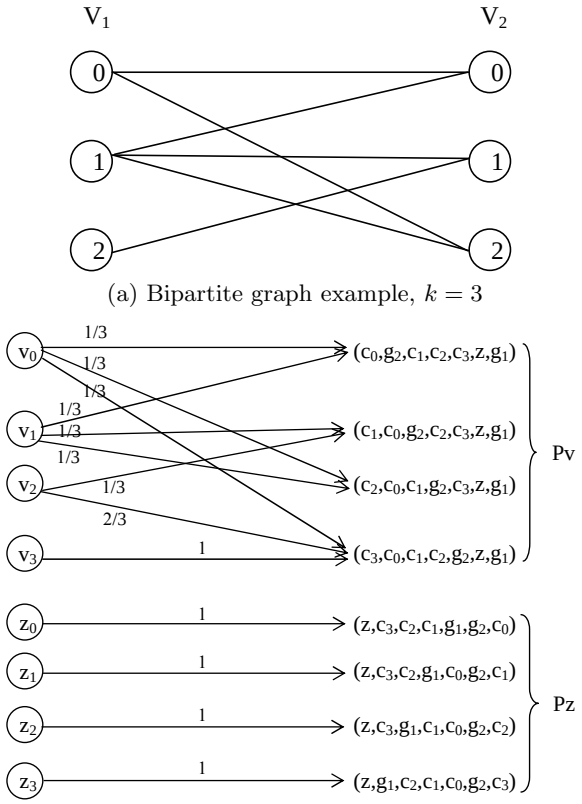


Figure 1: Reduction of Permanent to EVALUATION problem with Borda protocol

This proof can also be extended to work with random tie-breaking as well. All we have to do is to remove candidate g_1 from the preference orders of P_v and to replace it with g_2 in the preference orders of P_z. Now, in each perfect matching, all the candidates are tied with z. z has a probability of $(\frac{1}{k})^k/(k+3)$ to be the winner in each such scenario, and the proof proceeds in the same way, except that now the answer from the EVALUATION algorithm has to be divided by $(\frac{1}{k})^k/(k+3)$.

THEOREM 6. *If n and m are not constant, the EVALUATION problem is #P-Hard for the Copeland voting protocol.*

PROOF. Given a bipartite graph G, we use the same reduction structure from the previous proof, except that we change the place of g_2 in every order in P_z. Instead of being at index $k+2$, it is now in the second place, right after z, while the other candidates are shifted one place down. In every voting scenario that corresponds to a perfect matching, every candidate from $c_i \in \{c_0, c_1, \ldots, c_k\}$ and g_1 are tied with all the other candidates, so their Copeland score is 0. g_2 tied with all the candidates but z, which he loses to, so its Copeland score is -1. z ties with all the candidates but g_2, which he beats, so its Copeland score is 1 and he is the winner. Conversely, in every voting scenario which

does not correspond to a perfect matching for G, there is at least one order from the group P_v which gets more than one vote. Every candidate $c_i \in \{c_0, c_1, \ldots, c_k\}$ has one preference order from P_v which locates him at the first place, thus prefers him on all the other candidates, and in all the other preference orders in P_v he is located at the same place above the other candidates. All the preference orders from P_z always locate him below the same candidates. So no matter how the other votes are scattered among the orders of P_v, the candidate in the first place of the order which gets more than one vote wins at least one candidate while he keeps being tied with the other ones. However, z number of wins and ties does not change. According to the tie-breaking method which selects the first candidate from a lexicographic order z will not be the winner. Hence, given the probability that z is the winner from the EVALUATION algorithm, dividing it by $(\frac{1}{k})^k$ yields the number of perfect matching. □

Again, this proof can also be extended to work with random tie-breaking as well. All we have to do is to remove candidate g_2 from all the orders, so all the candidates have a Copeland score of 0, and they are all tied with z in every perfect matching for G. Now, z has a probability of $(\frac{1}{k})^k/(k+3)$ to be the winner in each such scenario, and the proof proceeds in the same way, except that now the answer from the EVALUATION algorithm has to be divided by $(\frac{1}{k})^k/(k+3)$.

Note that all our proofs use equal weights for the voters, so the results hold for the weighted voters case with unbounded or bounded weights too.

4.2 Chance-Evaluation problem

Our original definition of the EVALUATION problem yields a problem that is hard to compute for some common voting protocols. Now we thus define a related problem with a weaker question.

Definition 7. *[CHANCE-EVALUATION] Given a social choice domain, an imperfect information model of voters' preferences, as described above, and a specific candidate, ω^*, is the probability that it will be chosen greater than zero?*

This question seems to be very natural one. In many cases there are some candidates which do not have any chance of winning. Every voter will probably want to know which candidates do not have any chance to win regardless of his vote, in order to deliberate between candidates which have at least one voting scenario where they win. Surprisingly, this question is hard even for the simplest voting protocol – Plurality – when not all the voters have equal weights.

Theorem 8. *If n and m are not constant, the CHANCE-EVALUATION problem is NP-Complete for the Plurality voting protocol when not all the voters have equal weights.*

Proof. The problem is clearly in NP – given one voting scenario where ω^* wins, we can calculate its probability of occurring and check that indeed ω^* is the winner in a polynomial time. The NP-Hard reduction is from the NP-Complete *BIN-PACKING* problem: given a finite set U of items, an integer size $s(u)$ for each $u \in U$, a positive integer bin capacity B and a positive integer k, is there a partition of U into disjoint sets $U_1, U_2, \ldots, U_k$ such that the sum of the sizes of the items in each U_i is B or less? The instance for the CHANCE-EVALUATION problem is as follows. Every item is represented by a voter, where the item size is the voter's weight. We add another voter, v_z with the weight $B+1$. Every bin is represented by a candidate, and we add another candidate z. v_z has a probability of 1 to vote to z, and all the other voters have an equal probability to vote for each one of the remaining candidates. We look for the possibility of z to be a winner. Note that every voting scenario is correspond to a packing and vice versa; a voter with weight x which votes for candidate y is like placing item with size x in bin y. One item can not be in more than one bin and every voter can not vote for more than one candidate. Now suppose the tie-breaking method is to select the first candidate according to a pre-defined lexicographic order (the proof can be extended to work with a random tie-breaking method as well). z is the winner if and only if all the other candidates gets B or less votes. So there is a packing if and only if there is a voting scenario where z is the winner. □

This problem is NP-Complete in the strong sense [9], meaning that even if the weights are bounded by $Poly(n)$ the problem remains hard (unlike the case with the constant number of candidates, as shown before). Fortunately, if all the voters have equal weights the problem can be solved in polynomial time.

Theorem 9. *Even if n and m are not constant, the CHANCE-EVALUATION problem is in P for the Plurality voting protocol where all the voters have equal weights.*

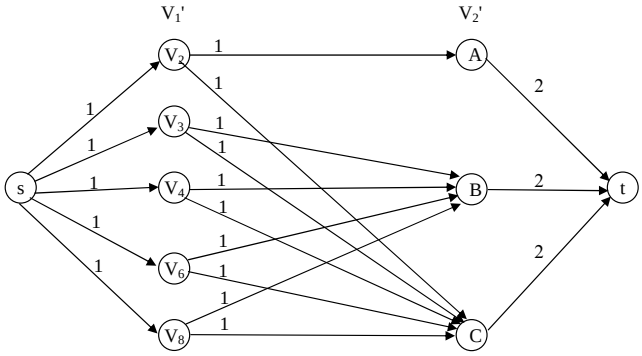
Proof. We give a polynomial time algorithm to answer the CHANCE-EVALUATION problem, assuming a random tie-breaking is used although the algorithm can be extended to work with the second tie-breaking method that was mentioned above as well. First, count the number of voters which have a non-zero probability to vote for ω^*, the candidate that we seek for a voting scenario where it wins, denote this number by k. Then build a flow network $G = (V, E)$ which contains a bipartite graph $G' = (V1' + V2', E')$ and two additional nodes s and t, $V = V1' \cup V2' \cup \{s, t\}$. $V1'$ has a node for every voter which have a zero probability to vote for ω^*, and $V2'$ has a node for every candidate but ω^*. For every $i \in V_1'$, if voter i has a non-zero probability to vote for candidate j then $(i, j) \in E'$. In E, s has an edge with capacity 1 to all the nodes of $V1'$, t has an edge with capacity k from all the nodes of $V2'$, and if $(i, j) \in E'$, $(i, j) \in E$ too, with capacity 1. Now find a maximum flow and check that every edge from s to a node of $V1'$ has a residual capacity of zero. If such flow exists, it represents a voting scenario where ω^* gets k votes and all the other candidates get k or less votes so the algorithm returns "yes". If not, then in every voting scenario, ω^* can get at most k votes and there is at least one candidate who get more than k votes so the algorithm returns "no". The construction of the flow network and all the stages of the algorithm can be done in polynomial time, so the CHANCE-EVALUATION problem for Plurality is in P where all the voters have equal weights. □

Figure 2 shows an example of how the algorithm builds a flow network from a given set of preferences, which shown in figure 2(a). In this example we seek for a voting scenario where candidate D wins. We remove voters V_1, V_5 and V_7 which have a non-zero probability to vote for D, and build

a flow network as described in figure 2(b) to find a voting scenario where all the other candidates receive no more than 2 votes.

V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
$\frac{1}{4}$ A	$\frac{1}{2}$ A	$\frac{1}{3}$ A	$\frac{1}{3}$ B	$\frac{1}{3}$ A	$\frac{1}{3}$ B	$\frac{1}{3}$ B	$\frac{1}{3}$ B
$\frac{1}{2}$ B	$\frac{1}{2}$ C	$\frac{1}{2}$ B	$\frac{2}{3}$ C	$\frac{2}{3}$ D	$\frac{2}{3}$ C	$\frac{2}{3}$ D	$\frac{2}{3}$ C
$\frac{1}{4}$ D		$\frac{1}{6}$ C					

(a) A set of preferences



(b) The corresponding flow network for candidate D

Figure 2: An example of how to build a flow network from a given set of preferences

5. CONCLUSION AND FUTURE WORK

In many multi-agent environments, it is desirable to use voting protocols to aggregate the agents different preferences. If all the agents preference orders are perfectly known, then for any practical voting protocol it is computationally easy to calculate which candidate will win. However, sometimes only the probability to choose each one of the available ballots is known. In this work, we investigated the problem of computing the probability that a candidate will win an election, given this imperfect information scenario. We showed an important distinction between the case where the number of candidates is a constant to the case where it is not bounded. In the first case, our algorithm, which runs in polynomial time, can compute the probability of a candidate to win in many voting systems no matter whether or not voter weights are equal, unless their weights are un-bounded. However, the second case is much harder to compute, as we proved for Plurality, Borda and Copeland voting protocols. Even to check whenever a candidate has any chance to win with the Plurality voting protocol is NP-Complete when not all the voters weights are equal. When they are equal, we gave a polynomial time algorithm which can foresee if a candidate has any chance to win using the Plurality protocol.

For future work, we would like to extend our current analysis to more voting protocols. We would also like to improve our results for the current voting protocols: where we prove that the problem is #P-Hard it would be useful to have an approximation algorithm (or to prove that one cannot be found); even where the problem is in P, our algorithm may have an impractically large running time. Using heuristics may yield more efficient algorithms which yield the correct answer for most of the cases.

6. REFERENCES

[1] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare Volume 1.* Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 2002.

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

[3] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.

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

[5] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. How hard is it to control an election? *Mathematical and Computer Modelling*, 16:27–40, 1992.

[6] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, 2002.

[7] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 781–788, 2003.

[8] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):1–33, June 2007.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of* NP*-Completeness*. W. H. Freeman: New York, 1979.

[10] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.

[11] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-2005)*, pages 95–101, July 2005.

[12] M. A. Satterthwaite. Strategy-proofness and arrows conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.