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 voters' 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 win 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.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination, Intelligent agents*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems; J.4 [**Social and Behaviorial Sciences**]: [Economics]

General Terms

Algorithms, Economics, Theory

Keywords

Computational social choice, Voting protocols

1. INTRODUCTION

In many multi-agent systems, it is desirable to have a mechanism which enables the agents within the system to make a collective decision on a given issue. The mechanism by which such a collective decision is made is typically a *voting procedure*. When considering voting procedures from a computational perspective, many interesting theoretical questions arise. Perhaps the most natural question from a computer science 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 in this sense [4]. Perhaps more intriguing are questions related to the complexity of *manipulating* a voting procedure. It can often be computationally infeasible for an agent to compute the most beneficial manipulation [3], implying that while manipulation is possible in theory, it is infeasible in practice. The complexity of manipulation was studied in [3, 2, 6] under the assumption that the number of outcomes is unbounded, while [5, 7] analyzed the complexity of manipulation with a constant number of outcomes. However, most of these results assume *perfect information* about voter preferences, which is surely 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 a 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;

[1]The exception is the work of [5] but their result holds only for weighted voters with weights that are not bounded by $Poly(n)$ as we will show later.

Cite as: Evaluation of Election Outcomes under Uncertainty, Noam Hazon, Yonatan Aumann, Sarit Kraus and Michael Wooldridge, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16.,2008,Estoril,Portugal,pp.959-966.

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 outcomes, by asking people who know them, or by using the history of their former votes on the same issue. The ability to calculate the probability of an outcome winning should then help A to decide whether B has a valid point.

This ability to calculate the probability of an outcome winning might also be useful in 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 subset 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 the result of [5] holds only for weighted voting systems with weights that are not bounded by $Poly(n)$. 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 8): 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 voters have equal weights.

Table 1 summarizes our results. For comparison, we also include results from [5] (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

Underlying our work is the notion of a *social choice domain*. Formally, a social choice domain is a tuple $S = \langle V, W, \Omega, \succ \rangle$ where $V = \{1, \ldots, n\}$ is a non-empty set of *voters* – the electorate; $W = \{w_1, \ldots, w_n\}$ is a non-empty set of *weights*, $w_i \in \mathbb{N}$ 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 $\succ = \{\succ_1, \ldots, \succ_n\}$ is a non-empty set of *preference relations*, $\succ_i \subseteq \Omega \times \Omega$ 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*, 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 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 a 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 only express a first and second choice of candidate, while under the Contingent Vote they must rank all of them.

Number of Candidates	Weights	Chance-Evaluation	Evaluation
constant	equal	$\mathrm{P}_{(p,b,c,m,i,\ldots)}$	$\mathrm{P}_{(p,b,c,m,i,\ldots)}$
	bounded by $Poly(n)$	$\mathrm{P}_{(p,b,c,m,i,\ldots)}$	$\mathrm{P}_{(p,b,c,m,i,\ldots)}$
	otherwise	NP-Hard$_{(b,c,m,i)}$ [5]	NP-Hard$_{(b,c,m,i)}$ [5]
parameter	equal	$\mathrm{P}_{(p)}$	#P-Hard$_{(p,b,c)}$
	bounded by $Poly(n)$	NP-Complete$_{(p)}$	#P-Hard$_{(p,b,c)}$
	otherwise	NP-Complete$_{(p)}$	#P-Hard$_{(p,b,c)}$

Table 1: Summary of results. 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

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 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 margin of victory: largest first, 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 select a candidate randomly among all the tied candidates, or, alternatively, we can 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 imperfect information, we assume 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 for 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 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.

Consider the following illustrative example. Suppose we have 3 candidates, ω_1, ω_2 and ω_3, and 3 voters, V_1, V_2 and V_3. In this example $n = m = k = 3$. Assume that the random tie-breaking method is used. The voters' preferences are summarized in Table 2 with a probability associated to each preference order. The probability that ω_1 is the winner according to Plurality is $\frac{9}{20}/3$, because the only voting scenario where it has a chance to win is when V_1 votes for him and V_3 votes for ω_3 so there is a tie between all the candidates; V_2 always votes for ω_2 (remember that in the plurality protocol every voter votes for its most preferred candidate; the other preferences are not taken into account). The winning probabilities for each candidate under the Plurality, Borda and Copeland voting protocols are summarized in Table 3. Note that ω_3 has the highest probability of winning under Plurality and Borda, but ω_2 has the highest probability of winning under Copeland.

V_1	V_2	V_3
$\frac{1}{2}$ $(\omega_1, \omega_2, \omega_3)$	$\frac{1}{4}$ $(\omega_2, \omega_1, \omega_3)$	$\frac{9}{10}$ $(\omega_3, \omega_1, \omega_2)$
$\frac{1}{3}$ $(\omega_3, \omega_1, \omega_2)$	$\frac{3}{4}$ $(\omega_2, \omega_3, \omega_1)$	$\frac{1}{10}$ $(\omega_2, \omega_1, \omega_3)$
$\frac{1}{6}$ $(\omega_2, \omega_1, \omega_3)$		

Table 2: An example of how we represent the imperfect information

	Plurality	Borda	Copeland
ω_1	0.15	0.225	0.1125
ω_2	0.4	0.3625	**0.5875**
ω_3	**0.45**	**0.4125**	0.3

Table 3: Winning probabilities for each candidate. Bold font represents the highest probability in each voting protocol

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 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. This is done by using the possible voting results for $n-1$ voters and their probabilities, which is 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)

```
1: Init T[.,.] ← 0,    T[0⃗, 0] ← 1.
2: for i ← 0 to n − 1 do
3:    for all cells in column i do
4:       r⃗ ← the voting results of the cell's row
5:       for j ← 1 to k do
6:          cur⃗ ← preference order j of voter i + 1
7:          next⃗ ← the voting result from adding cur to r
8:          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 occurring. 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 4(a). Table 4(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 $\frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} \cdot (\frac{1}{3} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{3}{4}) + \frac{1}{2} \cdot (\frac{1}{6} \cdot \frac{1}{4})$.

Table 4: An example of how algorithm 1 builds a table from a given set of preferences

(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

The time complexity of the algorithm is roughly $O(n\times$ number of rows of $T \times k)$, and the space complexity 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 who voted for candidate i.
2. a vector of $[0, n]^{m(m-1)/2}$ which represents the number of voters who preferred the first candidate in each possible pair of candidates.
3. a vector of $[0, n]^{m!}$ which represents the number of voters who voted for each possible preference order permutation.

We now show which method to use for each voting system.

- 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 tighter 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 non-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*;

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

we denote this value 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 who 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, [5] 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 weights that are not bounded by $Poly(n)$.

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 $O(Poly(n)^{m!})$, depending on the specific voting system (as described before). In all the cases it is still in P. □

This result may be understood with reference to the proofs of [5], which uses a NP-Hard reduction from the PARTITION problem. PARTITION is known to have a pseudo-polynomial time dynamic programming solution [8]. The restriction to weights that are bounded by $Poly(n)$ 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 particular voter power that is exponentially larger than another 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 voters have equal weights, but we give a polynomial algorithm for the case when all 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 time polynomial 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 $G=(X+Y,E)$ such that $\forall (x,y) \in E, x \in X$ and $y \in Y$, and $|X|=|Y|=k$, 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 are 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 $G=(X+Y,E)$, with $X=\{x_1,\ldots,x_k\}$ and $Y=\{y_1,\ldots,y_k\}$, for which we wish to count the number of perfect matchings, we construct an instance of the EVALUATION problem such that the probability of the chosen candidate to win is linear in 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 X plus two additional voters x_0 and $\hat{w}$, all with equal weights. The candidates are all the vertices of Y plus two additional candidates y_0 and $\hat{a}$. For every $x \in X$, if $(x,y) \in E$, set the probability that voter x votes for candidate y to be $\frac{1}{k}$. With the remaining probability $(1-\frac{\deg(x)}{k}$, where $\deg(x)$ is the degree of x) voter x votes for y_0. Finally, $\hat{w}$ votes for candidate $\hat{a}$ with probability 1, and x_0 votes for candidate y_0 with probability 1.

Consider a particular set of votes cast by the voters. Voters x_0 and $\hat{w}$ have no choice, so consider the choices made by

voters in X. Each such set of choices naturally corresponds to a matching, M, between X and Y:

$$M = \{(x, y) \in X \times Y : x \text{ voted for } y\}$$

(note that if x voted for y_0 then this pair is not included in M). We show that $\hat{a}$ wins the election iff M is a perfect matching.

Suppose that M is a perfect matching, then all candidates in Y get exactly one vote (from the voters in X) as do $\hat{a}$ and y_0 (from $\hat{w}$ and x_0, respectively). Thus, all candidates obtain the same score, and $\hat{a}$ wins by lexicographic order. Conversely, suppose that M is not a perfect matching. Then, either there is a candidate $y \in Y$ that gets more than one vote, or else there is a voter $x \in X$ that voted for y_0 (in addition to the vote y_0 surely received from x_0). In either case, there is a candidates that got more than one vote, while $\hat{a}$ received only one vote (from $\hat{w}$). Hence, $\hat{a}$ does not win the election.

The probability that the voters of X elect any specific perfect matching is k^{-k}. Thus

$$\Pr[\hat{a} \text{ wins the election}] = k^{-k} \cdot \text{PM}(G)$$

where $\text{PM}(G)$ denotes the number of perfect matchings in G. Hence, the answer to the EVALUATION problem also gives us one for the number of perfect matchings.

The proof for random tie-breaking is essentially identical, only that in the case of an exact matching $\hat{a}$ does not necessarily win, but only wins with probability $\frac{1}{k+2}$. Hence, in this case $\Pr[\hat{a} \text{ wins the election}] = \frac{k^{-k}}{k+2} \cdot \text{PM}(G)$. The rest of the proof remains the same. □

We now turn to the Borda and Copeland protocols. We start with a simple lemma, the proof of which is trivial.

LEMMA 5. *Let V be a set of voters, each with an individual preference order over a set of candidates. Suppose that all orders are different, and that for each preference order of any voter v, there exists another voter v' with the exact opposite preference order. Then:*

- *In the Borda protocol all candidates get the exact same score (which is also the average score).*
- *In the Copeland protocol, all pairwise contests are tied, for a total 0 score for all candidates.*

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

PROOF. Let $G = (X + Y, E)$ be a bipartite graph, with $X = \{x_1, \ldots, x_k\}$ and $Y = \{y_1, \ldots, y_k\}$, for which we wish to count the number of perfect matchings. We construct an instance of the EVALUATION problem as follows. There are $2(k + 1)$ voters composed of two subsets: X^+ and W, with $k + 1$ voters in each. The set X^+ consists of the set X plus one additional voter x_0. The set W consists of $k + 1$ voters $w_0, \ldots, w_k$. All voters have equal weights. There are $k + 2$ candidates: $C = \{c_0, \ldots, c_k\}$ and one "special" candidate $\hat{a}$. We build the EVALUATION instance in such a way that every perfect matching in G corresponds to a voting choice in which for every voter $x_i \in X^+$, there is a voter $w_j \in W$ with the exact reverse preference order. In this case, by Lemma 5 all candidates have the same score, and $\hat{a}$ wins by lexicographic order. Furthermore, the EVALUATION instance is constructed such that $\hat{a}$ only wins in votings that correspond to perfect matchings in G. The details follow.

For ease of notation we denote $i \oplus j = (i + j) mod (k + 1)$. Define the following set of orderings over the candidate set. For each $i = 0, \ldots, k$ let $s_i = (c_i, c_{i\oplus 1}, \ldots, c_{i\oplus k}, \hat{a})$, and denote by $(s_i)^R$ the reverse order to s_i. For each $(x_j, y_i) \in E$ (an edge in G), there is a probability of $1/k$ that voter x_j vote for order s_i. With the remaining probability $(1 - \frac{\deg(x_j)}{k})$ voter x_j votes for order s_0. Voter x_0 votes for s_0 with probability 1. For voters in W, voter w_j votes for order $(s_j)^R$ with probability 1. Note that, in particular, $\hat{a}$ is last in all votes of X^+ and first in all votes of W. See Figure 1 for example of how to build an instance from a given bipartite graph where $k = 3$.

Consider a set of orders chosen by the voters. Only the voters of X have any choice, so consider their votes. Each such set of choices naturally corresponds to a matching, M, between X and Y:

$$M = \{(x_i, y_j) \in X \times Y : x_i \text{ voted } s_j\}$$

We show that for lexicographic order tie-breaking, $\hat{a}$ wins the election iff M is a perfect matching in G.

Suppose that M is a perfect matching in G. Then, each s_i is voted exactly once, by the voters in X^+. However, each $(s_i)^R$ is also voted exactly once, by the voters of W. Hence, each voted order has the exact opposite order also voted for, and by Lemma 5, $\hat{a}$ wins by lexicographic order.

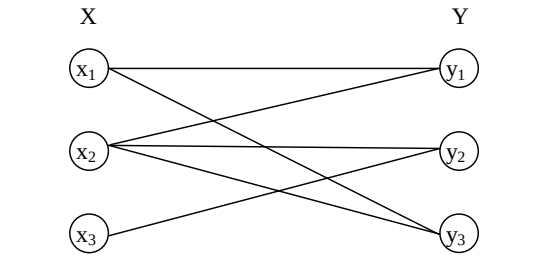
Conversely, suppose that M is not a perfect matching. Denote by α the average total score of the candidates. Since α is an *average*, it is independent of the actual choices made by the voters. Consider M. Since M is not a perfect matching, there exists at least one order s_i that is not voted by any voter of X^+. W.l.o.g. assume that this is s_k. Note that in all orders s_i with $i \neq k$ candidate c_k appears after candidate c_{k-1}. Hence, the total score that c_{k-1} gets from voters of X^+ must be higher than the total score they give c_k. The voters of W, on the other hand, in total give all candidates of C the exact same score (since the construction of the s_i's is symmetric). Hence, c_{k-1} gets a higher total score than c_k, and, in particular, it is not the case that all candidates get an identical total score. Thus, there must be a candidate c_{i_0} that gets a total score β strictly greater than the average α. On the other hand, the score of $\hat{a}$ is always the same (being always last in votes of X^+ and first in votes of W). Hence, its score is always identical to the one it gets in a perfect matching, namely α. Hence, $\hat{a}$ does not win the elections.

The probability that the voters of X elect any specific perfect matching is k^{-k}. Thus, $\Pr[\hat{a} \text{ wins the election}] = k^{-k} \cdot \text{PM}(G)$. Hence, the answer to the EVALUATION problem also gives us one for the number of perfect matchings.

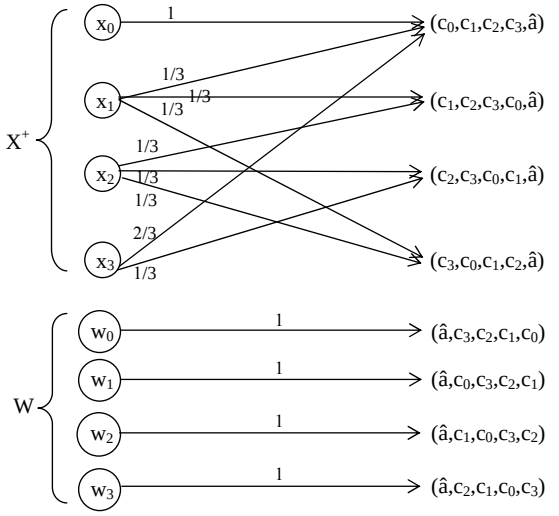
The proof for random tie-breaking (instead of lexicographic), is essentially identical, as in the previous proof. □

THEOREM 7. *If n and m are not constant, the EVALUATION problem is #P-Hard for the Copeland voting protocol.*

PROOF. The proof is very similar to that of the Borda protocol, and uses the exact same construction. Following that proof, we show that also for the Copeland protocol, $\hat{a}$ can win iff M (as defined in the Borda proof) is a perfect matching. Indeed, if M is a perfect matching, then as shown above, for each vote for a given preference order there is a vote for the exact reverse order. Thus, the conditions of



(a) Bipartite graph example, $k = 3$



(b) The corresponding instance for the EVALUATION algorithm

Figure 1: Reduction of Permanent to EVALUATION problem used in proof of Theorems 6 and 7

Lemma 5 hold, and all candidates get an identical 0 score. Hence, $\hat{a}$ can win (either by lexicographic order or by random choice, depending on the protocol).

Conversely, suppose that M is not a perfect matching. Then, there exists at least one order s_i that is not voted by any voter of X^+. W.l.o.g. assume that this is s_k. In all orders s_i with $i \neq k$ candidate c_{k-1} appears before candidate c_k. In all orders $(s_i)^R$ with $i \neq (k-1)$ candidate c_{k-1} appears immediately after c_k, and in $(s_{k-1})^R$ it appears before candidate c_k. Hence, for any other candidate c_j, if c_k wins the pairwise contest with c_j, so does c_{k-1}. In addition, c_{k-1} wins c_k. Hence, in total, c_{k-1} must win strictly more pairwise contests than c_k. Hence, it cannot be the case that all candidates score exactly 0. Thus, since the average total score is necessarily 0, there must be at least one candidate that scores more than 0. On the other hand, $\hat{a}$ ties all pairwise contests (it is first in all votes by W and last in all those by X^+), for a total of 0. Thus, $\hat{a}$ cannot win the elections. The rest of the proof is identical to that for the Borda protocol. □

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 8. *[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 a 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 voters have equal weights.

Theorem 9. *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 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 for 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 corresponds to a packing and vice versa; a voter with weight x which votes for candidate y is like placing an 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 get 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 [8], 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 voters have equal weights the problem can be solved in polynomial time.

Theorem 10. *Even if n and m are not constant, the CHANCE-EVALUATION problem is in P for the Plurality voting protocol where all voters have equal weights.*

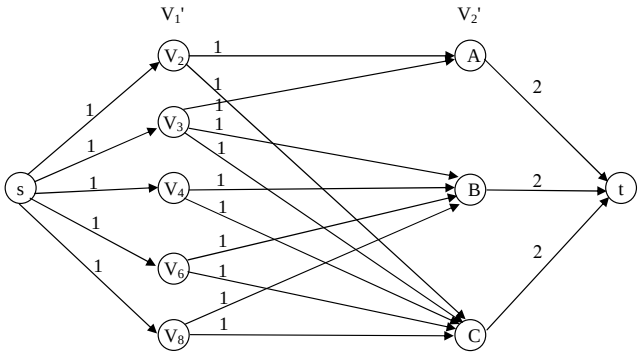
Proof. We give a polynomial time algorithm to answer the CHANCE-EVALUATION problem, assuming a random tie-breaking is used although the algorithm can be extended to work with the second tie-breaking method that was mentioned above as well. The idea is very similar to the technique in [9, p.176]. Let ω^* be the candidate for whom we

are trying to determine whether they have any chance of winning. Count the number of voters that vote for ω^* with non-zero probability, and 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 has 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 how the algorithm builds a flow network from the set of preferences in Figure 2(a). In this example we seek a voting scenario where candidate D wins. We remove voters V_1, V_5 and V_7 which have a non-zero probability of voting 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. CONCLUSIONS AND FUTURE WORK

In many multi-agent systems, it is desirable to use voting protocols to aggregate the preferences of different agents. 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, this perfect information assumption is sometimes not realistic, and what we know instead is only the probability that each voter has a certain preference profile. 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 and 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 winning in many voting systems, no matter whether or not voter weights are equal. However, the second case is #P-Hard to compute, as we proved for Plurality, Borda and Copeland voting protocols. Even to check whether a candidate has any chance to win with the Plurality voting protocol is NP-Complete when not all voter weights are equal. For the case when they are equal, we gave a polynomial time algorithm for computing 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] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, 2002.

[6] 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)*, 2003.

[7] 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.

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

[9] D. B. West, editor. *Introduction to Graph Theory.* Prentice Hall, 2 edition, 2001.