Artificial Intelligence 189 (2012) 1–18

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

# On the evaluation of election outcomes under uncertainty

Noam Hazon [a,*], Yonatan Aumann [a], Sarit Kraus [a], Michael Wooldridge [b]

[a] Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
[b] Department of Computer Science, University of Liverpool, Liverpool, United Kingdom

ARTICLE INFO

*Article history:*
Received 7 March 2010
Received in revised form 18 April 2012
Accepted 30 April 2012
Available online 3 May 2012

*Keywords:*
Computational social choice
Voting rules

ABSTRACT

We investigate the extent to which it is possible to compute 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 preferences of the voter, but for each ordering, we know the probability that it does. For the case where the number of candidates is a constant, we are able to give a polynomial time algorithm to compute the probability that a given candidate will win. We present experimental results obtained with an implementation of the algorithm, illustrating how the algorithm's performance in practice is better than its predicted theoretical bound. However, when the number of candidates is not bounded, we prove that the problem becomes #P-hard for the Plurality, *k*-approval, Borda, Copeland, and Bucklin voting rules. We further show that even evaluating if a candidate has *any* chance of winning is NP-complete for the Plurality voting rule in the case where voters may have different weights. With unweighted voters, we give a polynomial algorithm for Plurality, and show that the problem is hard for many other voting rules. Finally, we give a Monte Carlo approximation algorithm for computing the probability of a candidate winning in any settings, with an error that is as small as desired.



## 1. Introduction

*Social choice theory* is concerned with making group decisions in situations where the preferences of participants in the decision-making process may be different [1]. The mechanism by which such a collective decision is made is typically a *voting procedure*. In a voting procedure, participants (voters) express their preferences via votes, and the voting procedure then defines the social outcome chosen as a function of the votes cast. A fundamental issue in the social choice literature is the design of voting procedures that will select a social outcome which reflects the preferences expressed by voters as closely as possible [13]. In recent years, the *computational* aspects of social choice theory have been increasingly studied [20]. From a computational perspective, perhaps the most natural question relating to voting is the following: *Given the preferences/votes of all the voters, is it possible to compute efficiently the winning outcome according to a particular voting rule?* Fortunately, it seems that relatively few widely used voting rules are hard to compute in this sense [5]. Another key computational question that has been studied in the context of voting procedures is that of *manipulation*: the extent to which it is computationally easy or hard for a voter to determine how to vote so as to achieve the best outcome possible for themselves [4,15,19].

As the references above indicate, the computational aspects of voting procedures have been increasingly studied in recent years. However, many computational studies of voting procedures assume *perfect information* about voter preferences

* Corresponding author.
*E-mail addresses:* hazonn@cs.biu.ac.il (N. Hazon), aumann@cs.biu.ac.il (Y. Aumann), sarit@cs.biu.ac.il (S. Kraus), mjw@liv.ac.uk (M. Wooldridge).

0004-3702/$ – see front matter 
http://dx.doi.org/10.1016/j.artint.2012.04.009

or votes.[1] That is, when we compute the social outcome, we are assumed to have complete and correct knowledge of the preferences/votes of all the voters. However, there are important settings in which obtaining the complete preferences/votes of all voters is either not realistic or else not desirable:

- *Communication can be unreliable.* Social choice theory largely ignores the possibility that preferences/votes may be lost in transit. However, if, for example, voting takes place via a communications network such as the Internet, then this assumption is not valid. Data communications networks are inherently unreliable, and in many domains, reliable communication is simply impossible. In such situations, we may need to make social choices without access to the preferences/votes of all voters.
- *Communication can be expensive.* In some situations, the cost associated with gathering complete preferences/votes from voters may be unrealistically high. For example, if a decision is required very quickly, then the time required to gather all preferences/votes in a large system may be unacceptable.

For these reasons, we study the computational aspects of voting rules with an *imperfect information* model of preferences/votes. There are of course many ways in which one could model incomplete information about voter preferences. For example, one model that has been studied in the literature is that of *incomplete* models of preferences [27,30]. Here it is assumed that we have a partial but nevertheless correct model of the preferences of voters. Intuitively, we know how voters rank some of the candidates, but not all of them. One can then ask, for example, whether there is some "completion" of the preferences of voters that would lead to the election of a particular candidate.

In our work, we use a different model of incomplete information; we do not claim this model is superior to that of [27,30], but it provides an interesting alternative framework for modeling voting scenarios with incomplete information. We assume that 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, for example, be obtained from historical voting data, or by sampling. In this setting, the following fundamental question arises: *Given such an incomplete information model of voter preferences, a particular candidate, and a particular voting rule, what is the probability that the given candidate would win using the given voting rule, assuming the given voter preference model?* We refer to this as the EVALUATION problem. The EVALUATION problem has received very little attention to date.[2] Our aim is thus to gain an understanding of the computational properties of this problem; and in particular, classes of problem instances for which EVALUATION is computationally easy, and classes of problem instances for which it is computationally hard.

The remainder of the paper is structured as follows. We first give some background and review some common voting rules in Section 2. We formally define the EVALUATION problem in Definition 1. In Section 3, we first give a polynomial algorithm to solve the evaluation problem if the number of candidates is a constant. While a result in [16] establishes that EVALUATION is NP-hard for several key voting procedures, even under quite stringent assumptions about probability distributions, we show that this result holds only for weighted voting rules with weights that are not bounded by poly($n$). We then experimentally evaluate our algorithm, showing that the actual running time and space are smaller than the asymptotic bound. Therefore, we also test how many voters the polynomial-time algorithm can handle for a given set of candidates. The results demonstrate that even with 6 or 7 candidates, the algorithm can handle more than 100 voters, which suggests that it may be used in many real-world voting scenarios. If the number of candidates is not bounded, the evaluation problem becomes much harder: we show in Section 4 that even for the well-known Plurality, $k$-approval, Borda, Copeland, and Bucklin voting rules, the problem is #P-hard. We then analyze a simpler question, known as the CHANCE-EVALUATION problem (Definition 2). This question simply asks whether a candidate has *any* chance of being the winner (i.e., whether the probability that the candidate will win is greater than 0). Surprisingly, this problem is shown to be NP-complete (in the strong sense) even for the Plurality voting rule, when voters do not all have equal weights. We give a polynomial time algorithm for Plurality when all voters have equal weights, and show that the CHANCE-EVALUATION problem is hard for many other voting rules (including $k$-approval, Borda, Copeland and Bucklin). Finally, we present a Monte Carlo algorithm that is able to approximately answer even the EVALUATION problem where the number of candidates is a parameter, with an error as small as desired. We discuss related work in Section 5. Table 1 summarizes our key results (for comparison, we also include results from [16]).

## 2. Preliminary definitions

An *election* is given by a set of *candidates* (also referred to as *alternatives*) $C = \{c_1, \ldots, c_m\}$ and a set of *voters* $V = \{1, \ldots, n\}$. Each voter $i \in V$ is associated with a *preference order* $R_i$, which is a total order over $C$. A vector $\mathcal{R} = (R_1, \ldots, R_n)$, containing a preference order for each voter, is called a *preference profile*.

A *voting rule* $\mathcal{F}$ is a mapping from the set of all preference profiles to the set of candidates: if $\mathcal{F}(\mathcal{R}) = c$, we say that $c$ *wins* under $\mathcal{F}$ in $\mathcal{R}$. A voting rule is said to be *anonymous* if $\mathcal{F}(\mathcal{R}) = \mathcal{F}(\mathcal{R}')$ for all preference profiles $\mathcal{R}'$ that could be

[1] Not all previous studies have assumed perfect information: we discuss other imperfect information models and the results associated with these below.
[2] The exception we know of is the work of [16]; we discuss the relationship of our work to [16] in Section 5.

**Table 1**
Summary of key results. The parentheses near a complexity class indicates the voting rules for which the results have been proved. Key: all = any (polynomial-time computable) voting rule, $p$ = Plurality, $m$ = Maximin, 1 = Borda, Copeland, Maximin, STV, 2 = $k$-approval, Borda, Copeland, Bucklin.

| Number of candidates | Weights | Chance-Evaluation | Evaluation |
|---|---|---|---|
| Constant | equal | $\mathrm{P}_{(all)}$ | $\mathrm{P}_{(all)}$ |
| | bounded by poly($n$) | $\mathrm{P}_{(all)}$ | $\mathrm{P}_{(all)}$ |
| | otherwise | NP-hard$_{(1)}$ [16] | NP-hard$_{(1)}$ [16] |
| Parameter | equal | $\mathrm{P}_{(p)}$, NP-complete$_{(m,2)}$ | #P-hard$_{(p,2)}$ |
| | bounded by poly($n$) | NP-complete$_{(p,m,2)}$ | #P-hard$_{(p,2)}$ |
| | otherwise | NP-complete$_{(p,m,2)}$ | #P-hard$_{(p,2)}$ |
| Approximation | any | $\mathrm{P}_{(all)}$ | $\mathrm{P}_{(all)}$ |

obtained by permuting the entries of $\mathcal{R}$. In this paper we restrict our attention to anonymous voting rules only. In addition, we only consider voting rules that are polynomial-time computable.

During an election, each voter $i$ submits a preference order $L_i$, representing their "vote". The outcome of the election is then given by $\mathcal{F}(L_1, \ldots, L_n)$. We will assume that the voters are *truthful*, i.e., for each voter $i$, $L_i = R_i$. Of course, in general, voters will not necessarily be truthful, but for the purposes of the present paper we will ignore this possibility.

**Voting rules:** We will now define the main voting rules considered in this paper. Before we begin, a small technical remark. In the definition of a voting rule we gave above (i.e., as a function $\mathcal{F}$ from the set of preference profiles to the set of candidates), we implicitly required that voting rules are deterministic, in the sense that they select exactly one winner. The voting rules we consider in what follows do not quite fit this definition. All of these rules work by assigning scores to candidates on the basis of votes; the winner is then selected from the set of candidates with the highest score by using a *tie-breaking rule*, i.e., a mapping $T : 2^C \to C$ that satisfies $T(S) \in S$. We consider two tie-breaking rules: *random*, where the winner is randomly selected among all the tied candidates; and *lexicographic*, where given a set of tied candidates the winner is the candidate that is maximal with respect to a fixed ordering $\succ$. (Our results can be easily shown to hold for other tie-breaking rules.) Now, if we break ties at random, then the voting rule is not deterministic, and does not quite fit with our formal definition above. However, this observation does not affect our results in any way.

Given a vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ with $\alpha_1 \geqslant \cdots \geqslant \alpha_m$, the *score* $s_\alpha(c)$ of a candidate $c \in C$ under a *positional scoring rule* $F_\alpha$ is given by

$$s_\alpha(c) = \sum_{i \in V} \alpha_{j(i,c)}$$

where $j(i, c)$ is the position in which voter $i$ ranks candidate $c$. Note that many classic voting rules can be represented using this framework. For example:

- *Plurality* is the scoring rule with $\alpha = (1, 0, \ldots, 0)$;
- *Borda* is the scoring rule with $\alpha = (m-1, m-2, \ldots, 1, 0)$; and
- *k-approval* is the scoring rule with $\alpha$ given by $\alpha_1 = \cdots = \alpha_k = 1$, $\alpha_{k+1} = \cdots = \alpha_m = 0$.

The *Bucklin rule* can be viewed as an adaptive version of $k$-approval. We say that $k$, $1 \leqslant k \leqslant m$, is the *Bucklin winning round* if for any $j < k$ no candidate is ranked in top $j$ positions by at least $\lceil n/2 \rceil$ voters, and there exists some candidate that is ranked in top $k$ positions by at least $\lceil n/2 \rceil$ voters. We say that the candidate $c$'s *score in round* $j$ is his $j$-approval score, and his *Bucklin score* $s_B(c)$ is his $k$-approval score, where $k$ is the Bucklin winning round. The *Bucklin winner* is the candidate with the highest Bucklin score. Observe that the Bucklin score of the Bucklin winner is at least $\lceil n/2 \rceil$.

The *Copeland* rule is defined based on the notion of *pairwise elections*. We say that a candidate $c \in C$ beats another candidate $c' \in C$ in a pairwise election if a majority of voters rank $c$ above $c'$. The *Copeland score* $s_C(c)$ of a candidate $c$ is given by the number of pairwise elections that $c$ wins minus the number of pairwise elections that $c$ loses.

**Weighted voters:** Our model can be extended to the situation where not all voters are equally important by assigning an integer *weight*, $w_i$, to each voter $i$. To compute the winner on a profile $(R_1, \ldots, R_n)$ under a voting rule $\mathcal{F}$ given voters' weights $\mathbf{w} = (w_1, \ldots, w_n)$, we apply $\mathcal{F}$ on a modified profile which, for each $i = 1, \ldots, n$, contains $w_i$ copies of $R_i$. When all the weights are equal, we say that the voters are *unweighted*.

**Imperfect information:** To model imperfect information, we assume that we have for each voter at most $l$ 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 1; all the other possible preference orders which are not explicitly given are assumed to have a probability of 0. If all probabilities are 0 or 1 then the scenario is one of *perfect information*, otherwise it is one of *imperfect information*. Consider the following illustrative example. Suppose we have four candidates, $c_1$, $c_2$, $c_3$, and $c_4$, and three voters, $V_1$, $V_2$ and $V_3$.

**Table 2**
An example of our imperfect information model of voter preferences.

| $V_1$ | $V_2$ | $V_3$ |
|---|---|---|
| $\frac{1}{3}$ $(c_1, c_2, c_3, c_4)$ | $\frac{3}{4}$ $(c_4, c_2, c_1, c_3)$ | $\frac{9}{10}$ $(c_4, c_2, c_3, c_1)$ |
| $\frac{1}{2}$ $(c_2, c_1, c_3, c_4)$ | $\frac{1}{4}$ $(c_2, c_1, c_3, c_4)$ | $\frac{1}{10}$ $(c_3, c_1, c_4, c_2)$ |
| $\frac{1}{6}$ $(c_3, c_1, c_2, c_4)$ | | |

**Table 3**
Winning probabilities for each candidate, rounded to 3 decimal places. Bold font represents the highest probability in each voting rule.

| | Plurality | Borda | Copeland |
|---|---|---|---|
| $c_1$ | 0.036 | 0.058 | 0.052 |
| $c_2$ | 0.178 | **0.7** | 0.256 |
| $c_3$ | 0.053 | 0.017 | 0.017 |
| $c_4$ | **0.733** | 0.225 | **0.675** |

The voters' preferences are summarized in Table 2 with a probability associated to each preference order. In this example $n = l = 3$ (i.e., three voters and three possible preference profiles) and $m = 4$ (four candidates).

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 one possible preference profile that we call a *voting scenario*, from which the winner can be calculated using one of the voting rules listed above (Plurality, Borda, ...). The probability of any given voting scenario occurring is simply the product of the probabilities of its preference orders from the different voters.

**Evaluation and Chance-Evaluation:** We are now ready to define the main problems that we study throughout the remainder of this paper.

**Definition 1** *(*Evaluation*).* Given candidates $C$, voters $V$, an imperfect information model of voters' preferences, as described above, a specific candidate $c^* \in C$, and a voting rule $\mathcal{F}$, what is the probability that $c^*$ will be chosen using $\mathcal{F}$?

The answer to this question is the sum of probabilities of all the voting scenarios where $c^*$ wins using $\mathcal{F}$. For example, recall the imperfect information model of preferences shown in Table 2. Assume that random tie-breaking rule is used. The winning probabilities for each candidate under the Plurality, Borda and Copeland voting rules are summarized in Table 3. Note that $c_4$ has the highest probability of winning under Plurality and Copeland, but $c_2$ has the highest probability of winning under Borda.

Note that the complexity of the Evaluation problem is a function of the number of voters ($n$), the number of candidates ($m$), and the number of possible non-zero probability preference orders for each voter ($l$). We also define a related decision problem, which asks a weaker question.

**Definition 2** *(*Chance-Evaluation*).* Given candidates $C$, voters $V$, an imperfect information model of voters' preferences, as described above, a specific candidate $c^* \in C$, and a voting rule $\mathcal{F}$, is the probability that $c^*$ will be chosen using $\mathcal{F}$ greater than zero?

Of course, an answer to the Evaluation problem immediately yields an answer to the Chance-Evaluation problem, so if the former problem is easy, then so is the latter. However, it could in principle be the case that Evaluation is hard while Chance-Evaluation is easy, which suggests that it is worth studying Chance-Evaluation as a problem in its own right.

Note that the Chance-Evaluation problem also seems to be a very natural one. In many cases there will be some candidates that do not have any chance of winning, and a voter might reasonably contemplate which candidates have no chance of winning when deciding how to vote.

In the following sections, we analyze the complexity of the Evaluation and Chance-Evaluation problems in two scenarios: when 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 give a polynomial algorithm for the Evaluation problem under the assumption of a constant number of alternatives. Clearly, this algorithm also answers the Chance-Evaluation problem in polynomial time. We then present some experimental results obtained with this algorithm, evaluating its performance in practice.

### 3.1. The algorithm

The key to the efficiency of our algorithm is the distinction between a voting scenario and a *voting result*. Intuitively, in a voting scenario we know for each voter which preference order he votes for. But to identify a winning candidate, we do not care actually exactly which voter votes for which candidate; we can aggregate the possible voting scenarios into a compact intermediate form, which we refer to as a voting result. For example, suppose we use the Plurality rule. With Plurality, a voting result may be a vector which stores the total number of votes for each candidate. Now suppose that there are three voters and two candidates, $c_1$ and $c_2$, and 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 $c_1$ and one vote for $c_2$.

Voting results are compact representations for voting scenarios, and, in this sense, they have close connections with work on the *compilation complexity* of voting rules [14,35]. Work on the compilation complexity of voting attempts to give bounds on the number of bits required to summarize votes for a particular voting rule, and in addition considers specific compilation functions for summarizing votes. A voting result in our sense can therefore be thought of as a compilation function. We remark on these issues below in a little more detail.

Expressed a little more formally, a voting result can be understood as follows:

**Definition 3.** Given a voting rule, a *voting result* is a succinct way to represent one or more voting scenarios over $i$ voters, $0 \leqslant i \leqslant n$, such that:

1. For $i = n$, the winner can be determined from the voting result over the $n$ voters in polynomial time.
2. A voting result over $i+1$ voters can be generated from combining the voting result for $i$ voters and one additional preference order, in polynomial time.

Of course, this is a rather abstract definition: after we present the algorithm, we describe some concrete representations for voting results for many common voting rules. Let us first describe the algorithm where all the voters' weights are equal. A formal proof of the correctness of this algorithm may be found in Appendix A.

The basic idea is to use dynamic programming to compute the possible voting results from preference profiles and to calculate the probability of these voting results. The algorithm computes possible voting results from the preferences of $n-1$ voters and their probabilities, which are in turn computed using the voting results from the preferences of $n-2$ voters, and so on. Our algorithm builds a table $T$ in which the rows are possible voting results and the columns represent the voters. We denote by $T[\vec{row}, col]$ the cell in the table at the row which represents the voting result vector $\vec{row}$, and at column $col$. At any stage, the algorithm only stores two columns in memory.

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

Init $T[.,.] \leftarrow 0$, $\quad T[\vec{0}, 0] \leftarrow 1$.
**for** $i \leftarrow 0$ to $n-1$ **do**
  **for all** cells in column $i$ **do**
    $\vec{r} \leftarrow$ the voting results of the cell's row
    **for** $j \leftarrow 1$ to $l$ **do**
      $\vec{cur} \leftarrow$ preference order $j$ of voter $i+1$
      $\vec{next} \leftarrow$ the voting result from adding $\vec{cur}$ to $\vec{r}$
      $T[\vec{next}, i+1] \leftarrow T[\vec{next}, i+1] +$ (probability of $\vec{cur} \times T[\vec{r}, i]$)
    **end for**
  **end for**
**end for**

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 rule. So, we can answer the Evaluation problem from Definition 1 by simply summing for $c^*$ the probabilities of the voting results where it wins. Consider the following small example. Suppose we use the Plurality voting rule with 3 candidates ($c_1, c_2$, and $c_3$) and two 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 for this data. Every row represents a voting result which is a vector such that index $i$ counts the number of votes for candidate $c_i$. The last column shows the probabilities for every possible voting result with voters $V_1$ and $V_2$. Thus, the probability that $c_1$ is the winner, assuming a random tie-breaking method is used, is $\frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} \cdot (\frac{1}{3} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{3}{4}) + \frac{1}{2} \cdot (\frac{1}{6} \cdot \frac{1}{4})$. Note that, in this example, $(0,0,2)$ is not a possible voting result.

The time complexity of the algorithm is $O(n \times$ number of rows of $T \times l)$, and the space complexity is $O($number of rows of $T)$. The specific voting rule determines how to express the possible voting results which in turn determines the number of rows. Clearly, we seek a representation that is as compact as possible. This issue, of how to summarize votes in a compact way, was also studied in [14] and [35], and we reuse some techniques presented in [14,35]. It also seems that their results, showing that the upper bound is tight, can be imported to our settings to show that our ways for the representation of

**Table 4**
An example of how Algorithm 1 builds a table from a given set of preferences.

(a) A set of voters' preferences.

| $V_1$ | $V_2$ |
|---|---|
| $\frac{1}{2}$ $c_1$ | $\frac{1}{4}$ $c_1$ |
| $\frac{1}{3}$ $c_2$ | $\frac{3}{4}$ $c_2$ |
| $\frac{1}{6}$ $c_3$ | |

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

| Voting result $(c_1, c_2, c_3)$ | 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 | $\frac{1}{6} \cdot \frac{3}{4}$ |

voting results are optimal (but we keep it to future work). Now, for many voting rules one of the following methods can be used to express the possible voting results:

1. A vector in $[0, n]^m$ such that index $i$ represents the number of voters who voted for candidate $i$.
2. A vector in $[0, n]^{m(m-1)/2}$ which represents the number of voters who preferred the first candidate in each possible pair of candidates.
3. From [35]: A vector in $[0, n]^{m^2}$ which represents the number of voters who ranked each candidate in each position.
4. From [14]: A vector in $[0, n]^{m \cdot 2^{m-1}}$. For each candidate $i$, let $z_{-i}$ be a possible subset of candidates not containing $i$. The voting result vector represents, for any candidate $i$ and any possible $z_{-i}$, the number of voters who preferred $i$ over any candidate in $z_{-i}$.
5. A vector in $[0, n]^{m!}$ which represents the number of voters who voted for each possible preference order permutation.

We now show which method to use for some voting rules.

- *Plurality*. The first method can be used so the number of rows is $O((n+1)^m)$ and the time complexity is $O((n+1)^m \cdot l)$. However, the actual number of voting results will never be $(n+1)^m$, since the total number of points given by *all* the candidates is $n$. Instead, the actual number of voting results with $n$ voters is exactly the number of ways of splitting 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*; we denote this value by $WC(n, m)$, $WC(n, m) = \binom{n+m-1}{m-1} = \frac{(n+m-1)!}{n!(m-1)!}$. So the running time complexity is $O(l \sum_{i=0}^{n} WC(i, m))$ and the space required is $O(WC(n, m))$.
- *k-approval*. The first method can be used, with a running time complexity of $O(l \sum_{i=0}^{n} WC(i \cdot k, m))$ and a space complexity of $O(WC(n \cdot k, m))$.
- *Borda*. We can use a modified version of the first method – a vector in $[0, (m-1)n]^m$ which represents the total score for each candidate. Thus, the time and space complexity are $O(l \sum_{i=0}^{n} WC(i \cdot \frac{m(m-1)}{2}, m))$ and $O(WC(n \cdot \frac{m(m-1)}{2}, m))$, respectively. We can also use the second method, since the Borda score of a candidate $c$ equals to the number of times in which $c$ is preferred to another candidate $c'$, for each $c' \in C$.
- *Bucklin*. The third method can be used so the time and space complexity are $O(l \sum_{i=0}^{n} WC(i \cdot m, m))$ and $O(WC(n \cdot m, m))$, respectively.
- *Copeland*. The second method can be used, so the number of rows is $O((n+1)^{m(m-1)/2})$ and the time complexity is $O((n+1)^{m(m-1)/2} l)$. This method can be used for many other Condorcet-consistent rules, e.g., Maximin, Ranked pairs, Voting trees, etc. (For an extensive discussion of voting rules, we refer the reader to [1].)
- *STV* (see a definition in [1]). Recall that STV performs in successive rounds: at each round, the candidate with the lowest plurality score gets eliminated, and its votes are transferred to the next preferred candidate in each vote, until there is a majority winner. We can thus use the fourth method, since the candidate that gets eliminated in each round is exactly the one which has the lowest number of voters that prefer it over all the remaining candidates that have not been eliminated yet. Therefore, the number of rows is $O((n+1)^{m \cdot 2^{m-1}})$ and the time complexity is $O((n+1)^{m \cdot 2^{m-1}} l)$. If $m \leqslant 4$, the last method shall be used to calculate the number of scores for each candidate from the preference orders.

When we move to the weighted voters case, Conitzer and Sandholm [16] expressed the EVALUATION problem as the following decision problem: Given a number $r$, $0 \leqslant r \leqslant 1$, is the probability of $c^*$ winning greater than $r$? They showed that this problem is NP-hard for Borda, Copeland, Minimax and STV, even for extremely restricted probability distributions. We show that their results hold only for weights that are not bounded by poly$(n)$.

**Claim 4.** *For a constant number of candidates, the* Evaluation *problem is in P even for weighted voters, when the weights are in* 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 at most $O(\text{poly}(n)^m)$, $O(\text{poly}(n)^{m(m-1)/2})$, $O(\text{poly}(n)^{m^2})$, $O(\text{poly}(n)^{m\cdot 2^{m-1}})$ or $O(\text{poly}(n)^{m!})$, depending on the specific voting rule (as described before). In all the cases it is still in P. □

This result may be understood with reference to the proofs of [16], which uses a reduction from the Partition problem. Partition is known to have a pseudo-polynomial time dynamic programming solution [24].

### *3.2. Experiments*

In the preceding section, we gave analytical results for the case where the number of candidates is a constant. We showed that the complexity of our algorithm grows exponentially with the number of candidates. As with many other problems that have worst-case exponential running time, it is interesting to ask whether we do indeed see worst case performance in practice. Our hypothesis was that in our problem, the actual number of voting results that the algorithm stores is much smaller than the asymptotic bound in most cases (and hence the required memory and time are smaller too). In particular, we investigated the effect of the probabilistic structure of the imperfect information on the number of stored voting results. Additionally, we tested the effect of the parameter $l$ on the actual number of voting results. In this section we present experimental results obtained with an implementation of the algorithm for the Plurality rule with unweighted voters, which validate our hypothesis. We also found it interesting to check how many voters the polynomial-time algorithm can reasonably handle in practice, for a given number of candidates.

Our implementation (written in C++) ran on a 64-bit Linux PC, with 8 GB of RAM. The large amount of main memory was needed for the algorithm to store the table of the voting results. This table was implemented using Judy array [6], a complex but very fast associative array data structure for storing and retrieving values. We chose to use this data structure since it typically requires much less memory than a conventional hash table. We measured the algorithm's performance by counting the total number of cells that were produced during run-time (in contrast to measuring time, which would depend on the actual testing hardware). In most cases we ran 15 iterations and took the average. In cases where the running time proved to be very high, we took the average of only 5 iterations.

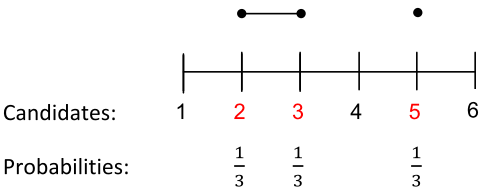
As an input, the algorithm takes an imperfect information matrix. Unlike in other experimental work in social choice which generates random *preferences* or random voters' *weights* (see [33] for example), we need to randomly generate *probabilities* over possible preferences. As noted before, in this work we assume that some knowledge on the preferences can be derived, and only $l$ preference orders have a non-zero probability for each voter. Therefore, the impartial culture assumption [12], which is a model of an electorate in which all preference orders are equally likely, cannot be used. We considered two methods for selecting $l$ candidates for each voter (Plurality needs only the top choice candidate) and generating the probabilities, using uniform and normal distributions. In the first method, $l$ candidates were randomly chosen for each voter and the probability that she will vote for each one of them was set to $1/l$. The second method defined an arbitrary fixed order over the candidates. It then randomly chose one candidate to be the mean of the normal distribution, for each voter. The other $l-1$ candidates were chosen by their proximity to the mean candidate. The probability that each voter will vote for each one of the candidates was set according to the normal distribution, with the selected mean and a variance of 1. Fig. 1 demonstrates the difference between these two methods.

In the first set of experiments, we tested the effect of the random methods that we used to generate the voters' preferences. In these experiments we fixed $l$ to be 3, and we evaluated the effect of the two methods on the running time (in terms of number of generated cells) for 5 and 6 candidates and 20–100 voters. The results are shown in Fig. 2.

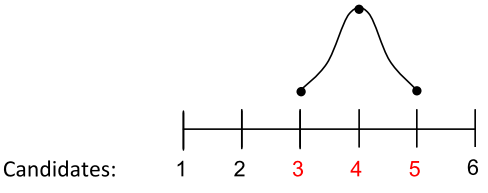
Clearly, using the uniform distribution to generate the preferences results in more ways of splitting the total number of votes among the candidates. Thus, increasing the number of voting results yields a higher running time. The second method simulates a more realistic scenario, the "single-peaked preference" principle [11]. In this case, there is some predetermined linear ordering of the candidate set. Every voter has some most preferred point on that line, and his dislike for a candidate grows larger as the candidate goes further away from that point. Similarly, in our case every voter has some special place that we believe has the highest probability to be selected, and the probability that the voter will vote for a candidate decreases as the candidate goes further away from that spot. In this case the votes are less scattered among the candidates, and thus the number of voting results is lower, yielding a lower running time.

We also use these settings to demonstrate how the ratio of the actual number of voting results to the theoretical number behaves. Since we use Plurality, the theoretical bound was computed using the *WC* function (as described above). The results are summarized in Fig. 3. As there are more voting scenarios which lead to the same voting result, the gap between the theoretical bound and the actual number of voting results increases. Thus, this ratio is lower when there are more candidates or when a normal distribution rather than a uniform distribution is used to generate the imperfect information. On the other hand, the number of voters does not have a significant effect on this ratio.

In the second set of experiments, we investigated the effect of $l$ on the actual number of voting results. Although the algorithm's running time is (asymptotically) linear in $l$, it was interesting to check if $l$ has the same effect on the actual




(a) uniform distribution.




(b) normal distribution.

**Fig. 1.** An example of how to generate random probabilities where $m = 6$, $l = 3$.

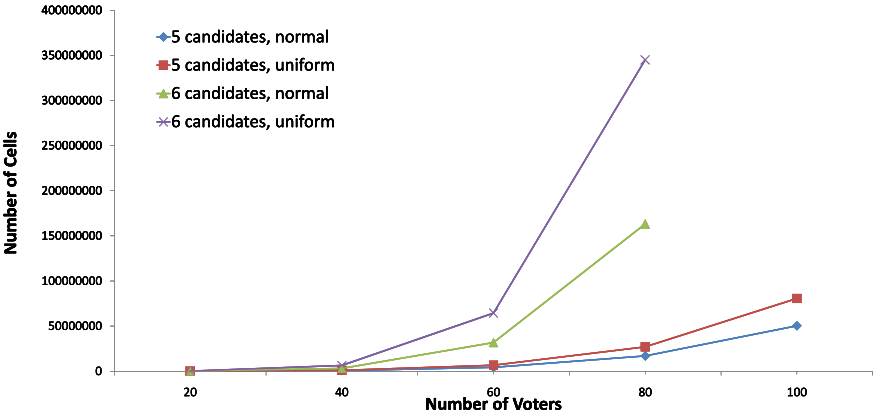



**Fig. 2.** Results of the first set of experiments.

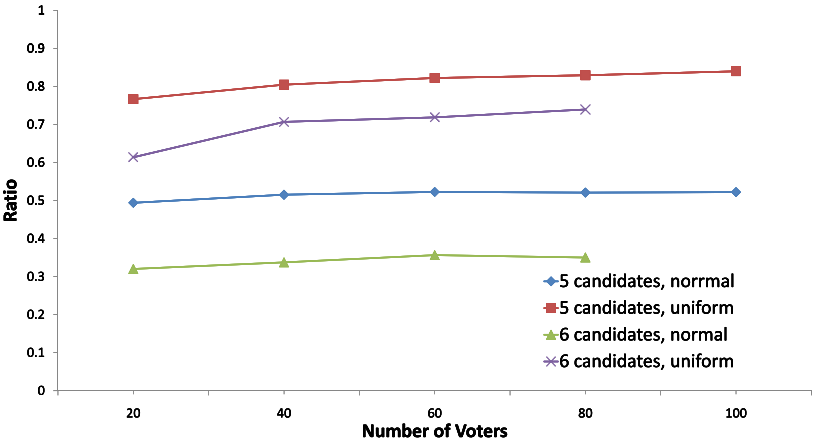



**Fig. 3.** Ratio of actual to theoretical number of voting results for increasing numbers of voters.

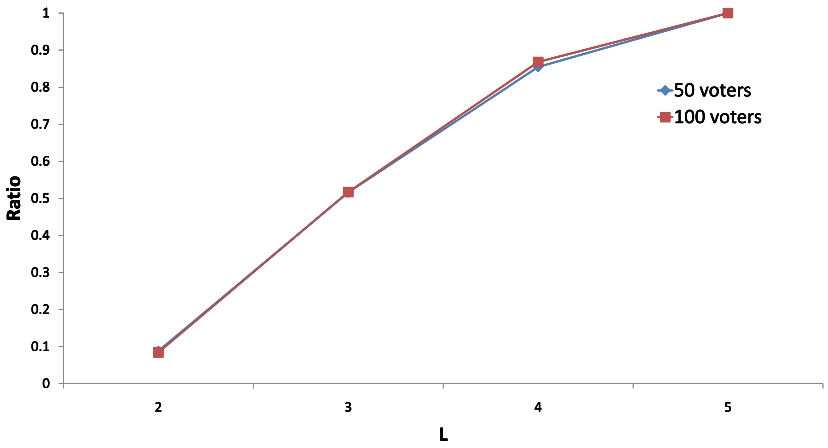



**Fig. 4.** Ratio of actual to theoretical number of voting results for increasing number of non-zero probability preference orders for each voter ($l$).

**Table 5**
Extreme results. Fractions are rounded to 3 decimal places.

| # of candidates | # of voters | Theoretical # of voting results | Actual # of voting results | Ratio | Time (s) | Total # of cells |
|---|---|---|---|---|---|---|
| 4 | 1100 | 223,045,351 | 47,331,609.2 | 0.212 | 111,183.4 | 13,122,678,458.0 |
| 5 | 400 | 1,093,567,501 | 93,506,124.2 | 0.086 | 338,200.4 | 7,640,484,607.0 |
| 6 | 140 | 498,187,404 | 18,146,578.2 | 0.036 | 4756 | 442,092,341.2 |
| 7 | 100 | 1,705,904,746 | 22,381,578.8 | 0.013 | 3792 | 346,504,543.2 |

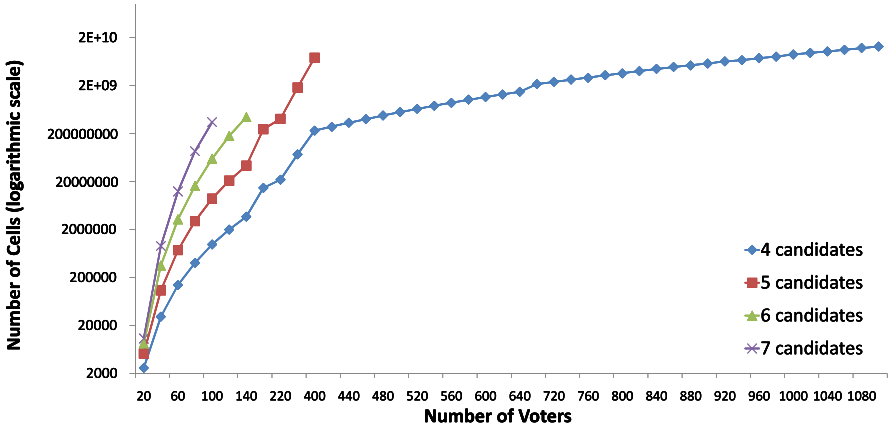



**Fig. 5.** Results of the last set of experiments.

number of voting results. In these experiments, we fixed the number of candidates to 5, and used the normal distribution to generate preferences. We measured the ratio of the actual number of voting results to the theoretical bound (computed using the *WC* function) for 50 and 100 voters and $l$ between 2 and 5. The results are shown in Fig. 4. Indeed, as $l$ increases, the ratio of actual to theoretical number of voting results increases in the same manner. As in the previous experiment, this ratio is not affected by the number of voters. Note that since $m = 5$, if $l = 5$ too, every possible voting result may happen, thus the ratio is 1.

It follows from our results that there is clearly a gap between the theoretically predicted running time and the actual one. Therefore, in the last set of experiments we tested how many voters the polynomial-time algorithm can handle in practice, for a given number of candidates. We set $l$ to its minimum value, 2, and we used the normal distribution to generate the preferences. We then tested for 4 to 7 candidates how many voters the algorithm can handle. We would perhaps have been able to process more voters if we used a hard disk as a virtual memory, but the I/O overheads would result in a much higher running time, and we wanted to test our algorithm with reasonable limits – therefore, the algorithm used only main memory, and the "extreme" results that are shown in Table 5 were achieved just before the algorithm ran out of space. The complete picture is shown in Fig. 5. Note that the $y$-axis is shown in logarithmic scale.

The results show that the actual running time (in terms of generated cells) heavily depends on the number of candidates, as expected. We also see that the algorithm can handle a practical number of voters, even with 6 or 7 candidates. For

example, the Israeli parliament (the Knesset) has 120 voters, and the United States Senate has 100 voters. Table 5 shows again the difference between the theoretical upper bound on the number of voting results (computed using *WC* function), to the actual number.

## 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 the EVALUATION problem for $k$-approval, Borda, Copeland, Bucklin and even Plurality is #P-hard in this case. We also analyze the seemingly weaker question of CHANCE-EVALUATION. Surprisingly, we show that even this problem is hard when voters do not all have equal weights under Plurality. We show that with equal weights, CHANCE-EVALUATION is still hard under $k$-approval, Borda, Copeland, Bucklin and Maximin. However, for the Plurality rule where all voters have equal weights, we are able to give a polynomial time algorithm.

### *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 it 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. Of course, the assumption of a constant number of voters is rather restrictive, and unfortunately, as we will see, 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, $k$-approval, Borda, Copeland and Bucklin voting rules.

All our #P-hardness reductions will be from a well known #P-complete problem, PERMANENT, which is to calculate the permanent of a $\{0, 1\}$-matrix (or, equivalently, to count the number of perfect matchings for a bipartite graph).

**Definition 5.** Denote by $S_n$ the set of all permutations of the numbers $1, 2, \ldots, n$. The *permanent* of an $n \times n$ matrix $A = (a_{i,j})$ is defined as

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

**Definition 6.** For a bipartite graph $G = (X \cup Y, E)$ such that $\forall (x, y) \in E,\ x \in X$ and $y \in Y$, and $|X| = |Y| = r$, 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 any bipartite graph's adjacency matrix in fact counts the number of perfect matchings for the graph. We are now ready to show the proof for the Plurality voting rule.

**Theorem 7.** *If n and m are not constant, the* EVALUATION *problem is #P-hard for the Plurality voting rule.*

**Proof.** Given a bipartite graph $G = (X \cup Y, E)$, with $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_r\}$, 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 a function of (and only of) the number of perfect matchings in $G$. The voters are all the vertices of $X$ plus two additional voters $x_0$ and $w$, all with equal weights. The candidates are all the vertices of $Y$ plus two additional candidates $y_0$ and $a$. We first consider the case where the tie-breaking rule is lexicographic, and $a$ has the top priority. For every $x \in X$, if $(x, y) \in E$, set the probability that voter $x$ votes for candidate $y$ to be $\frac{1}{r}$. With the remaining probability $(1 - \frac{\deg(x)}{r}$, where $\deg(x)$ is the degree of $x$) voter $x$ votes for $y_0$. Finally, $w$ votes for candidate $a$ 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 $w$ have no choice, so consider the choices made by voters in $X$. Each such set of choices naturally corresponds to a collection of $r$ edges, $M$, between $X$ and $Y$:

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

(note that if $x$ voted for $y_0$ then this pair is not included in $M$). We show that $a$ 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 $a$ and $y_0$ (from $w$ and $x_0$, respectively). Thus, all candidates obtain the same score, and $a$ wins by the tie-breaking rule. 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 candidate that got more than one vote, while $a$ received only one vote (from $w$). Hence, $a$ does not win the election.

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

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

where 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 $a$ does not necessarily win, but only wins with probability $\frac{1}{r+2}$. Hence, in this case $\Pr[a \text{ wins the election}] = \frac{r^{-r}}{r+2} \cdot \text{PM}(G)$. The rest of the proof remains the same. □

As for $k$-approval, we only need to slightly modify the reduction used in the proof for Plurality. Recall that in $k$-approval only the $k$ first candidates get scores, so we don't care what their order is, or the order of the other $m-k$ candidates: we only care whether a candidate appears in the top $k$ positions or not.

**Theorem 8.** *If $n$ and $m$ are not constant, the* EVALUATION *problem is #P-hard for the $k$-approval voting rule, for every fixed $k$.*

**Proof.** Given a bipartite graph $G = (X \cup Y, E)$, with $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_r\}$, for which we wish to count the number of perfect matchings, we construct almost the same instance of the EVALUATION problem as in the proof of Theorem 7. The set of voters is the same, but we add a set of dummy candidates $D = \{d^i_{y_j}\} \cup \{d^i_a\}$, where $1 \leqslant i \leqslant k-1$, $0 \leqslant j \leqslant r$. For every $x \in X$, if $(x, y) \in E$, set the probability that voter $x$ gives one point to candidates $y, d^1_y, \ldots, d^{k-1}_y$ to be $\frac{1}{r}$. With the remaining probability $(1 - \frac{\deg(x)}{r}$, where $\deg(x)$ is the degree of $x$) voter $x$ gives one point to $y_0, d^1_{y_0}, \ldots, d^{k-1}_{y_0}$. Finally, $w$ gives one point to candidates $a, d^1_a, \ldots, d^{k-1}_a$ with probability 1, and $x_0$ gives one point to candidates $y_0, d^1_{y_0}, \ldots, d^{k-1}_{y_0}$ with probability 1.

Now, each candidate $d^i_{y_j}$ and $d^i_a$ gets the same number of points as $y_j$ and $a$, respectively. Therefore, the rest of the proof is essentially identical to that for the Plurality rule. □

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

**Lemma 9.** *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 10.** *If $n$ and $m$ are not constant, the* EVALUATION *problem is #P-hard for the Borda voting rule.*

**Proof.** Let $G = (X \cup Y, E)$ be a bipartite graph, with $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_r\}$, for which we wish to count the number of perfect matchings. We construct an instance of the EVALUATION problem as follows. There are $2(r+1)$ voters composed of two subsets: $X^+$ and $W$, with $r+1$ voters in each. The set $X^+$ consists of the set $X$ plus one additional voter $x_0$. The set $W$ consists of $r+1$ voters $w_0, \ldots, w_r$. All voters have equal weights. There are $r+2$ candidates: $C = \{c_0, \ldots, c_r\}$ and one "special" candidate $a$. We build the EVALUATION instance in such a way that every perfect matching in $G$ corresponds to a voting scenario 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 9 all candidates have the same score, and $a$ wins by lexicographic tie-breaking rule. Furthermore, the EVALUATION instance is constructed so that $a$ only wins in voting scenarios that correspond to perfect matchings in $G$. The details follow.

For ease of notation we denote $i \oplus j = (i+j) mod (r+1)$. Define the following set of orderings over the candidate set. For each $i = 0, \ldots, r$ let $s_i = (c_i, c_{i\oplus 1}, \ldots, c_{i\oplus r}, a)$, 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 $1/r$ that voter $x_j$ votes for order $s_i$. With the remaining probability $(1 - \frac{\deg(x_j)}{r})$ 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, $a$ is last in all votes of $X^+$ and first in all votes of $W$. See Fig. 6 for an example of how to build an instance from a given bipartite graph where $r = 3$.

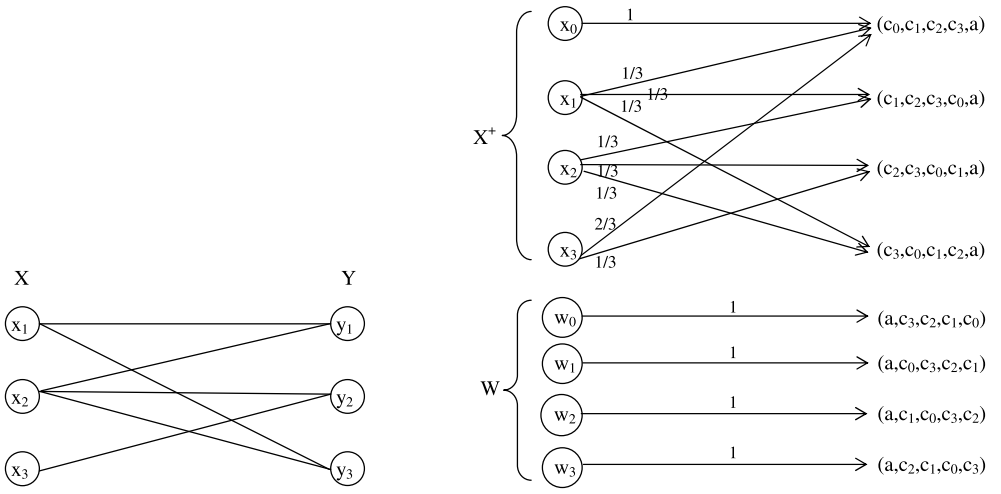
Consider a set of orders chosen by the voters. Only the voters of $X$ have any choice, so consider their votes. Each such set of choices naturally corresponds to a collection of $r$ edges, $M$, between $X$ and $Y$:

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

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

Suppose that $M$ is a perfect matching in $G$. Then, each $s_i$ gets exactly one vote from the voters in $X^+$. However, each $(s_i)^R$ also receives exactly one vote, from the voters of $W$. Hence, for each preference order that received a vote, the exact opposite order was also voted for. In this case, by Lemma 9, $a$ wins by lexicographic tie-breaking rule.

Conversely, suppose that $M$ is not a perfect matching. Denote by $\alpha$ the average total score of the candidates, $\alpha = (r+1)^2$. Since $\alpha$ 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 does note receive any vote from $X^+$. W.l.o.g. assume that this is $s_r$. Note




(a) Bipartite graph example, $r = 3$.

(b) The corresponding instance for the EVALUATION algorithm.

**Fig. 6.** Reduction from PERMANENT to EVALUATION problem used in the proof of Theorems 10 and 11.

that in all orders $s_i$ with $i \neq r$ candidate $c_r$ appears after candidate $c_{r-1}$. Hence, the total score that $c_{r-1}$ gets from voters of $X^+$ must be higher than the total score they give $c_r$. 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, $c_{r-1}$ gets a higher total score than $c_r$, 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 $\beta$ strictly greater than the average $\alpha$. On the other hand, the score of $a$ 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 $\alpha$. Hence, $a$ does not win the elections.

The probability that the voters of $X$ elect any specific perfect matching is $r^{-r}$. Thus, $\Pr[a \text{ wins the election}] = r^{-r} \cdot \text{PM}(G)$. 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 proof for Plurality. □

**Theorem 11.** *If n and m are not constant, the* EVALUATION *problem is #P-hard for the Copeland voting rule.*

**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, $a$ 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 Lemma 9 hold, and all candidates get an identical score of zero. Hence, $a$ can win (either by lexicographic 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 for by any voter of $X^+$. W.l.o.g. assume that this is $s_r$. In all orders $s_i$ with $i \neq r$, candidate $c_{r-1}$ appears before candidate $c_r$. In all orders $(s_i)^R$ with $i \neq r-1$, candidate $c_{r-1}$ appears immediately after $c_r$, and in $(s_{r-1})^R$ it appears before candidate $c_r$. Hence, for any other candidate $c_j$, if $c_r$ wins the pairwise contest with $c_j$, so does $c_{r-1}$. In addition, $c_{r-1}$ beats $c_r$. Hence, in total, $c_{r-1}$ must win strictly more pairwise contests than $c_r$. 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, $a$ 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, $a$ cannot win the elections. The rest of the proof is identical to that for the Borda rule. □

As for Bucklin, we use a slightly different construction. This proof does not assume the use of any specific tie-breaking rule.

**Theorem 12.** *If n and m are not constant, the* EVALUATION *problem is #P-hard for the Bucklin voting rule.*

**Proof.** Let $G = (X \cup Y, E)$ be a bipartite graph, with $X = \{x_1, \ldots, x_r\}$ and $Y = \{y_1, \ldots, y_r\}$, for which we wish to count the number of perfect matchings. We construct an instance of the EVALUATION problem as follows. There are $2(r+1)$ voters, thus the Bucklin score of the Bucklin winner will be at least $r+2$. The set of voters is composed of two subsets: $X^+$ and $W$, with $r+1$ voters in each. The set $X^+$ consists of the set $X$ plus one additional voter $x_0$. The set $W$ consists of $r+1$ voters $w_0, \ldots, w_r$. All voters have equal weights. There are $r+1$ regular candidates: $C = \{c_0, \ldots, c_r\}$ and one "special" candidate $a$. Additionally, there are $(r+1)^2$ dummy candidates: $D = \{d^i_{y_j}\} \cup \{d^j_a\}$, where $1 \leqslant i \leqslant r$, $0 \leqslant j \leqslant r$.

We build the EVALUATION instance in such a way that every perfect matching in $G$ corresponds to a voting scenario in which the Bucklin wining round is $r+2$ and candidate $a$ wins. Furthermore, the EVALUATION instance is constructed so that in other voting scenarios the Bucklin winning round is strictly less than $r+2$ and one of the candidates from $C$ wins. The details follow.

For ease of notation we denote $i \oplus j = (i+j) mod (r+1)$. In our construction, the Bucklin wining round is always less than or equals $r+2$, thus we show only the first $r+2$ candidates in each preference order (other candidates may be placed arbitrarily). For each $i = 0, \ldots, r$, let $s_i = (c_i, d^1_{y_i}, \ldots, d^r_{y_i}, a)$, and $t_i = (a, c_i, c_{i\oplus 1}, \ldots, c_{i\oplus(r-1)}, d^i_a)$. For each $(x_j, y_i) \in E$ (an edge in $G$), there is a probability of $1/r$ that voter $x_j$ votes for order $s_i$. With the remaining probability $(1 - \frac{\deg(x_j)}{r})$ 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 $t_j$ with probability 1. Note that any order $s_i$ gives one point to candidate $c \in C$ in the first round, and every order $t_i$ gives one point to $c \in C$ on each round $j$, $2 \leqslant j \leqslant r+1$.

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 collection of $r$ edges, $M$, between $X$ and $Y$:

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

We show that $a$ wins the election iff $M$ is a perfect matching in $G$.

Suppose that $M$ is a perfect matching in $G$. Then, each $s_i$ gets exactly one vote from the voters in $X^+$. For every $j$, $1 \leqslant j \leqslant r+1$, the score of every dummy candidate $d \in D$ in round $j$ is less than or equals 1. The score of every candidate $c \in C$ in round $j$ is $j$, and the score of $a$ in round $j$ is $r+1$. Since no candidate has more than $r+2$ points, every $j$, $1 \leqslant j \leqslant r+1$, is not the Bucklin winning round. On the other hand, in round $r+2$ the score of $a$ is $2(r+1)$ while no other candidate has more than $r+1$ points. Therefore the Bucklin winning round is $r+2$ and $a$ is the (unique) winner.

Conversely, suppose that $M$ is not a perfect matching. Then, there exists at least one order $s_i$ that is voted for more than one time by voters of $X^+$. Therefore, there is at least one candidate $c \in C$ with a score of at least $r+2$ in round $r+1$. Then, the Bucklin winning round is less than or equals $r+1$. On the other hand, $a$'s score in every round $j$, $1 \leqslant j \leqslant r+1$, is exactly $r+1$. Thus, $a$ cannot win the elections.

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

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 is computationally hard for some common voting rules. Surprisingly, the weaker question, CHANCE-EVALUATION, is hard even for the simplest voting rule – Plurality – when voters do not all have equal weights.

**Theorem 13.** *If $n$ and $m$ are not constant, the* CHANCE-EVALUATION *problem is NP-complete for the Plurality voting rule when the voters do not all have equal weights.*

**Proof.** The problem is clearly in NP – given one voting scenario where $c^*$ wins, we can check that indeed $c^*$ is the winner in polynomial time. The NP-hardness 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 cannot be in more than one bin and every voter cannot vote for more than one candidate. Now suppose the tie-breaking rule is lexicographic and $z$ is the minimal candidate with respect to the ordering. $z$ 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. The proof for random tie-breaking is similar, only that the weight of $v_z$ is set to $B$. □

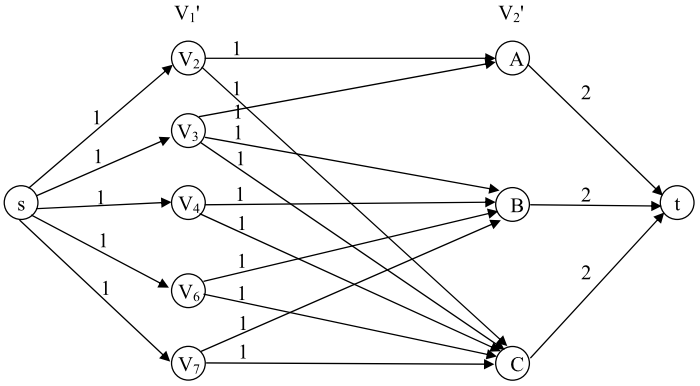
This problem is NP-complete in the strong sense [24], meaning that even if the weights are bounded by poly($n$) the problem remains hard (unlike the case with the constant number of candidates, as shown before).

Fortunately, for Plurality, if all voters have equal weights the problem can be solved in polynomial time.

**Theorem 14.** *Even if $n$ and $m$ are not constant, the* CHANCE-EVALUATION *problem is in P for the Plurality voting rule where all voters have equal weights.*

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

(a) A set of preferences.




(b) The corresponding flow network for candidate $D$.

**Fig. 7.** An example of how to build a flow network from a given set of preferences.

**Proof.** We give a polynomial time algorithm to answer the CHANCE-EVALUATION problem, assuming random tie-breaking is used. The idea is very similar to the technique in [34, p. 176], and we also refer to [22,21] for a different use of network flows techniques in the context of voting problems. Let $c^*$ be the candidate for whom we are trying to determine whether it has any chance of winning. Count the number of voters that vote for $c^*$ with non-zero probability, and denote this number by $b$. Then build a flow network $G = (\mathcal{V}, E)$ which contains a bipartite graph $G' = (V1' \cup V2', E')$ and two additional nodes $s$ and $t$, $\mathcal{V} = V1' \cup V2' \cup \{s, t\}$. $V1'$ has a node for every voter which has a zero probability to vote for $c^*$, and $V2'$ has a node for every candidate but $c^*$. 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 $b$ 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 a flow exists, it represents a voting scenario where $c^*$ gets $b$ votes and all the other candidates get $b$ or less votes so the algorithm returns "yes". If not, then in every voting scenario, $c^*$ can get at most $b$ votes and there is at least one candidate who gets more than $b$ 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.

The algorithm for lexicographic tie-breaking is similar. Let $Top(c^*)$ be the set of all candidates that are more favored than $c^*$ according to the lexicographic tie-breaking. For every edge $(v, t) \in E$, such that $v$ corresponds to a candidate in $Top(c^*)$, set the capacity to $b - 1$. The rest of the construction remains the same. If the required flow exists, it represents a voting scenario where $c^*$ gets $b$ votes, all the candidates that are more favored than $c^*$ get $b - 1$ or less votes, and all the other candidates get $b$ or less votes so the algorithm returns "yes". If not, then in every voting scenario, $c^*$ can get at most $b$ votes and there is at least one candidate which is more favored than $c^*$ who gets more than $b - 1$ votes, or there is other candidate who gets more than $b$ votes. Thus, the algorithm returns "no". □

Fig. 7 shows how the algorithm builds a flow network from the set of preferences in Fig. 7(a). In this example we seek a voting scenario where candidate $D$ has a chance to win, and we use random tie-breaking. We remove voters $V_1$ and $V_5$ which have a non-zero probability of voting for $D$, and build a flow network as described in Fig. 7(b). In this example, a possible maximal flow is to assign 1 to all the outgoing edges of $s$, to the edges $(V_2, A), (V_3, B), (V_4, B), (V_6, C), (V_7, C)$ and $(A, t)$, and to assign 2 to the edges from $B$ and $C$ to $t$. Therefore, $D$ has a chance to win; if $V_1$ and $V_5$ vote for $D$, $V_2$ votes for $A$, $V_3$ and $V_4$ vote for $B$, and $V_6$ and $V_7$ vote for $C$.

For other voting rules, we get NP-hardness results as a corollaries of [36]. This paper considers the possible-winner problem, where it was assumed that for each voter we have a correct but incomplete model of their preference order. The input to our problem is different; we have for each voter a collection of complete preference orders, with associated probabilities. Nevertheless, since the CHANCE-EVALUATION problem ignores the exact values of the probabilities, if the partial orders considered in possible-winner problem have a polynomial number of extensions, then possible-winner becomes a

**Table 6**
Number of iterations as a function of $\epsilon$ and $\alpha$.

| $\alpha$ | $\epsilon$ | $t$ |
|---|---|---|
| 0.05 | 0.05 | 271 |
| 0.05 | 0.01 | 6764 |
| 0.05 | 0.001 | 676,386 |
| 0.01 | 0.05 | 542 |
| 0.01 | 0.01 | 13,530 |
| 0.01 | 0.001 | 1,352,974 |
| 0.001 | 0.05 | 955 |
| 0.001 | 0.01 | 23,874 |
| 0.001 | 0.001 | 2,387,384 |
| 0.0001 | 0.05 | 1384 |
| 0.0001 | 0.01 | 34,578 |
| 0.0001 | 0.001 | 3,457,771 |

subproblem of Chance-Evaluation. It turns out that for many voting rules the possible-winner problem is NP-complete even if the number of undetermined pairs is a constant, which implies that the number of extensions is bounded. We obtain:

**Proposition 15.** *If n and m are not constant, the* Chance-Evaluation *problem is NP-complete for k-approval, Borda, Copeland, Bucklin, and Maximin voting rules, even if the voters are unweighted.*[3]

### 4.3. Monte Carlo approximation

Computing the exact answer for Evaluation and Chance-Evaluation problems seems to be hard in many cases. However, we can utilize the underlying probabilities to achieve an approximate solution even for the Evaluation problem. The idea is to use a statistical approach, in which we sample according to the given probabilities to estimate the true winning probability.

The algorithm is as follows. For each voter, we sample one preference order according to the given distribution, thus obtaining a voting scenario. Since the voters' choices are independent, this process is equivalent to sampling one voting scenario according to the voting scenarios distribution (which we do not know). We then calculate the winner from this voting scenario using the given voting rule, and repeat the whole process $t$ times. Given a specific candidate, $c^*$, we are interested in his winning probability (denote it by $p$). This probability is approximately the number of sampled voting scenarios where $c^*$ wins divided by $t$ (denote this ratio by $\hat{p}$).

From the perspective of $c^*$, each iteration has two possible outcomes: where he wins, or when another candidate wins. The winning probability of $c^*$, $p$, is the same in each iteration, and the iterations are statistically independent. Therefore, the distribution of $p$ is a binomial distribution, and the maximum likelihood estimator for $p$ is $\hat{p}$. This estimator is also known to be unbiased for the binomial distribution. We can build a binomial confidence interval which relies on approximating the binomial distribution with a normal distribution,[4] by the following formula:

$$P\left(\hat{p} - \sqrt{\frac{\hat{p}(1-\hat{p})}{t}} Z_{1-\frac{\alpha}{2}} \leqslant p \leqslant \hat{p} + \sqrt{\frac{\hat{p}(1-\hat{p})}{t}} Z_{1-\frac{\alpha}{2}}\right) = 1-\alpha \tag{1}$$

where $Z_{1-\frac{\alpha}{2}}$ is the $1-\frac{\alpha}{2}$ percentile of a standard normal distribution, and $\alpha$ is our chosen probability of error. For bounding the distance from the true winning probability, we require that given an $\epsilon$,

$$|p - \hat{p}| \leqslant \epsilon \tag{2}$$

Combining (1) and (2) above we get that the number of required iterations is:

$$t \geqslant \left(\frac{\sqrt{\hat{p}(1-\hat{p})} Z_{1-\frac{\alpha}{2}}}{\epsilon}\right)^2 \tag{3}$$

i.e., the winning probability $\hat{p}$ that we have found after such $t$ iterations is, with probability $1-\alpha$, at most $\epsilon$ away from the true probability $p$. Table 6 shows the required number of iterations ($t$) as a function of $\epsilon$ and $\alpha$, assuming that $\hat{p}(1-\hat{p})$ is maximal, i.e. $\hat{p} = 0.5$.

[3] This result holds for Ranked Pairs and Voting trees too, but we did not discuss these rules in our paper.
[4] We note that one can get more explicit bounds by using Chernoff bounds (see, e.g., [2]). However, more samples are required.

## 5. Related work

As we noted in the introduction, the computational aspects of voting have recently received a great deal of attention, leading to the emergence of a new research area, known as *computational social choice* [20].

Within the computational social choice community, much research has considered the complexity of *manipulating* voting procedures – of deciding how an agent can optimally cast a vote in the furtherance of its own preferences. The complexity of manipulation has been studied both under the assumption that the number of candidates is unbounded [4,3,18], and under the assumption of a constant number of candidates [16,19]. However, most computational studies of voting assume *perfect information* about voter preferences or votes, which, as discussed in the introduction, may be an unrealistic assumption in real world settings.

The limiting assumption of perfect information has driven researchers to look for a different, more realistic model. In [27], and later in [30], it was assumed that for each voter we have a correct but *incomplete* model of each voter's preference relation. For this incomplete information setting, [27,30] considered questions such as whether there was some completion of the incompletely known preferences that would make a particular desired candidate a winner. This question, known as the possible-winner problem, is particularly important as it also has a strong connection to preference elicitation [17] and manipulation [27]. This imperfect information model was further investigated under sequential majority voting [28,29], under all scoring rules [8,9,7], and under other common voting rules [36]. The complexity of the possible-winner problem has also been studied for cases where certain parameters are fixed (e.g., the number of candidates, number of voters, and total number of undetermined candidate pairs [10]). However, this model cannot utilize any prior knowledge of the voters' preferences. It is "pessimistic" in nature, as it assumes that the missing data is completely unknown, and thus it ignores any probabilistic estimation on the voters' preferences that could be learned from their voting history. Of course, if very little is known about the voters' preferences, this model may be more appropriate than the probabilistic model we use in this paper.[5] Moreover, our Chance-Evaluation problem ignores the exact values of the probabilities, so it is very close to the settings of the possible-winner problem. The connections between these problems were illustrated in Section 4.2, where we get many hardness results as corollaries of [36]. A recent paper [2] studies the computational complexity of the counting version of the possible-winner problem. They prove #P-hardness results for Plurality and Veto, and provide a randomized approximation algorithm for all voting rules that are polynomial-time computable. Their algorithm may be used to find the probability with which a designated candidate wins, assuming that the completion of the incompletely preferences are chosen uniformly at random.

The previous work that is most closely related to our concerns is [16], which uses a probability distribution over the votes to capture imperfect information. The results of [16] are derived with a restricted model of probability distributions. A key result of [16] is that, if manipulation for some voting rule is hard when complete information is provided, then it is also hard to even evaluate a candidate's winning probability with this protocol when there is uncertainty about the votes. However, as we demonstrated in Section 3, the hardness of the uncertain case holds only for weights that are not bounded by poly($n$), where $n$ is the number of voters. Conitzer and Sandholm [16] consider the unweighted voters case, but only with a probability distribution that allows for perfect correlations among the voters (which actually simulates weights for the voters). This is also the case in [32], which proves some results regarding the connection between the incomplete preferences settings to the settings where we have a probability distribution over the votes, but uses perfectly correlated votes. Probabilities are also used to model imperfect information in [26] and [31], but these papers assume that what is known about an electorate is only the probability that any given candidate will beat another. These papers use this data to investigate the extent to which it is possible to rig the agenda of an election or competition so as to favor a particular candidate.

## 6. Conclusions and future work

In many situations, it is desirable to use voting rules to aggregate the preferences of different agents in order to make a social choice. If the preference orders of all agents are perfectly known, then for any practical voting rule 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 order. In this work, we investigated the problem of computing the probability that a candidate will win an election, given such an imperfect information model. 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 rules, no matter whether or not voter weights are equal. However, the second case is #P-hard to compute, as we proved for Plurality, $k$-approval, Borda, Copeland and Bucklin voting rules. Even to check whether a candidate has any chance of winning with the Plurality voting rule is NP-complete when voter weights are not all equal. For the case when they are equal, it remains NP-complete for $k$-approval, Borda, Copeland, Bucklin and Maximin rules, but we gave a polynomial time algorithm for computing if a candidate has any chance to win using the Plurality protocol. We also

[5] In addition, this incomplete information model uses a more succinct representation than our probabilistic model. However, if the number of candidates is a constant, then any succinct representation is equal to our representation in terms of time and space complexity.

gave a simple Monte Carlo algorithm that is able to approximately compute the probability of a candidate winning in any setting, with an error as small as desired.

For future work, we would like to extend our current analysis to more voting rules, including multi-winner protocols. Even with the current protocols that we have considered there are still some open question. For example, the complexity of Evaluation with unweighted voters under the Maximin voting rule, and the Chance-Evaluation with unweighted voters under the STV rule. Another extension to consider is to define and analyze a general imperfect knowledge model, which combines the model of [27] – incomplete preferences – along with our model of probabilistic estimate of the voters' preferences. We would also like to improve our results for the current voting rules: where we prove that the problem is #P-hard or NP-complete it would be useful to have an approximation algorithm (or to prove that one cannot be found). There is also another promising direction to investigate, which is the use of the parameterized complexity paradigm [23] to analyze our problem. We have already demonstrated that, if we bound one of our parameters, (the number of candidates), then the Evaluation problem becomes easy to solve (and thus our problem belongs to XP). It will be interesting to check if other restrictions might also help, for example, if the number of different probability distributions is bounded by a constant.

## Acknowledgements

We gratefully acknowledge the detailed and helpful comments of the anonymous referees, which have enabled us to considerably improve this paper. We thank Efrat Manisterski for her help in developing the algorithm in Section 3. This work was supported in part by the Israel Ministry of Science and Technology, grant 3-6797. This paper subsumes an earlier conference paper [25].

## Appendix A. Correctness proof for Algorithm 1

**Theorem 16.** *Given an imperfect information voting domain (consisting of a set of candidates $C$, a finite set of voters $V$, a set of preferences profiles, and a probability distribution over this set of preference profiles for each voter), Algorithm 1 enumerates all the possible voting scenarios in polynomial time, when the number of candidates is considered a constant.*

**Proof.** Let $\mathsf{VR}^i$ be the set of voting results that the algorithm generates in iteration $i$. We prove that the algorithm enumerates all the relevant voting scenarios, by induction on the number of voters. If there is only one voter, the algorithm will generate at most $l$ voting results from the voter's preferences. Clearly, these are all possible voting scenarios for this voter. Otherwise, consider the $n$-th iteration. Every voting scenario of $n$ voters consist of a voting scenario of $n-1$ voters plus one preference order of the $n$-th voter. By the inductive hypothesis, all the possible voting scenarios of $n-1$ voters are summarized by $\mathsf{VR}^{n-1}$. Thus, combining every voting results from $\mathsf{VR}^{n-1}$ with every preference order of the $n$-th voter generates $\mathsf{VR}^n$, which summarizes all the voting scenarios of $n$ voters, as required.

As for the running time, the total number of voting results is polynomial in $n$ (since $m$ is a constant), and for each voting result there are $O(l)$ operations for generating other voting results. Generating voting results takes polynomial time, by definition. Thus, the algorithm's running time is polynomial in $n$ and $l$. □

## References

[1] K.J. Arrow, A.K. Sen, K. Suzumura (Eds.), Handbook of Social Choice and Welfare, vol. 1, Elsevier, Amsterdam, 2002.
[2] Y. Bachrach, N. Betzler, P. Faliszewski, Probabilistic possible winner determination, in: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010), 2010, pp. 697–702.
[3] J.J. Bartholdi, J. Orlin, Single transferable vote resists strategic voting, Social Choice and Welfare 8 (4) (1991) 341–354.
[4] J.J. Bartholdi, C.A. Tovey, M.A. Trick, The computational difficulty of manipulating an election, Social Choice and Welfare 6 (1989) 227–241.
[5] J.J. Bartholdi, C.A. Tovey, M.A. Trick, Voting schemes for which it can be difficult to tell who won the election, Social Choice and Welfare 6 (1989) 157–165.
[6] D. Baskins, Judy IV, http://judy.sourceforge.net/, 2001.
[7] D. Baumeister, J. Rothe, Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules, Lisbon, Portugal, 2010, pp. 1019–1020.
[8] N. Betzler, B. Dorn, Towards a dichotomy of finding possible winners in elections based on scoring rules, in: Proceedings of the Thirty-Fourth International Symposium on Mathematical Foundations of Computer Science (MFCS-2009), in: Lecture Notes in Computer Science (LNCS), vol. 5734, Springer-Verlag, 2009, pp. 124–136.
[9] N. Betzler, B. Dorn, Towards a dichotomy for the possible winner problem in elections based on scoring rules, Journal of Computer and System Sciences 76 (8) (2010) 812–836.
[10] N. Betzler, S. Hemmann, R. Niedermeier, A multivariate complexity analysis of determining possible winners given incomplete votes, in: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-2009), 2009, pp. 53–58.
[11] D. Black, On the rationale of group decision-making, Journal of Political Economy 56 (1) (1948) 23–34.
[12] D. Black (Ed.), The Theory of Committees and Elections, Cambridge University Press, Cambridge, 1958.
[13] S.J. Brams, P.C. Fishburn, Voting procedures, in: K.J. Arrow, A.K. Sen, K. Suzumura (Eds.), Handbook of Social Choice and Welfare, vol. 1, Elsevier, Amsterdam, 2002, Chapter 4.
[14] Y. Chevaleyre, J. Lang, N. Maudet, G. Ravilly-Abadie, Compiling the votes of a subelectorate, in: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-2009), 2009, pp. 97–102.
[15] V. Conitzer, Computational aspects of preference aggregation, PhD thesis, Department of Computer Science, Carnegie Mellon University, 2006.
[16] V. Conitzer, T. Sandholm, Complexity of manipulating elections with few candidates, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002), 2002, pp. 314–319.

[17] V. Conitzer, T. Sandholm, Vote elicitation: complexity and strategy-proofness, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002), 2002, pp. 392–397.
[18] V. Conitzer, T. Sandholm, Universal voting protocol tweaks to make manipulation hard, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003), 2003, pp. 781–788.
[19] V. Conitzer, T. Sandholm, J. Lang, When are elections with few candidates hard to manipulate?, Journal of the ACM 54 (3) (2007) 1–33.
[20] U. Endriss, J. Lang (Eds.), Proceedings of the First International Workshop on Computational Social Choice Theory 2006 (COMSOC-2006), ILLC, University of Amsterdam, Amsterdam, The Netherlands, 2006.
[21] P. Faliszewski, Nonuniform bribery, in: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008), 2008, pp. 1569–1572.
[22] P. Faliszewski, E. Hemaspaandra, L.A. Hemaspaandra, J. Rothe, Llull and Copeland voting broadly resist bribery and control, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-2007), 2007, pp. 724–730.
[23] J. Flum, M. Grohe (Eds.), Parameterized Complexity Theory, Springer, 2006.
[24] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.
[25] N. Hazon, Y. Aumann, S. Kraus, M. Wooldridge, Evaluation of election outcomes under uncertainty, in: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008), 2008, pp. 959–966.
[26] N. Hazon, P.E. Dunne, S. Kraus, M. Wooldridge, How to rig elections and competitions, in: Proceedings of the Second International Workshop on Computational Social Choice (COMSOC-2008), 2008, pp. 301–312.
[27] K. Konczak, J. Lang, Voting procedures with incomplete preferences, in: Proceedings of the Multidisciplinary IJCAI-05 Workshop on Advances in Preference Handling (M-PREF), 2005.
[28] J. Lang, M.S. Pini, F. Rossi, D. Salvagnin, K.B. Venable, T. Walsh, Winner determination in voting trees with incomplete preferences and weighted votes, Autonomous Agents and Multi-Agent Systems 25 (1) (2012) 130–157.
[29] M.S. Pini, F. Rossi, K.B. Venable, T. Walsh, Dealing with incomplete agents preferences and an uncertain agenda in group decision making via sequential majority voting, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference (KR-2008), 2008, pp. 571–578.
[30] M.S. Pini, F. Rossi, K.B. Venable, T. Walsh, Incompleteness and incomparability in preference aggregation: complexity results, Artificial Intelligence 175 (7–8) (2011) 1272–1289.
[31] T. Vu, A. Altman, Y. Shoham, On the complexity of schedule control problems for knockout tournaments, in: Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2009), 2009, pp. 225–232.
[32] T. Walsh, Uncertainty in preference elicitation and aggregation, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-2007), 2007, pp. 3–8.
[33] T. Walsh, Where are the really hard manipulation problems? The phase transition in manipulating the veto rule, in: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-2009), 2009, pp. 324–329.
[34] D.B. West (Ed.), Introduction to Graph Theory, 2nd edition, Prentice Hall, 2001.
[35] L. Xia, V. Conitzer, Compilation complexity of common voting rules, in: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010), 2010, pp. 915–920.
[36] L. Xia, V. Conitzer, Determining possible and necessary winners under common voting rules given partial orders, Journal of Artificial Intelligence Research 41 (2011) 25–67.