Approximate Inference of Outcomes in Probabilistic Elections

Batya Kenig*
Paul G. Allen School of Computer Science and Engineering
University of Washington
batyak@cs.washington.edu

Benny Kimelfeld†
Technion Israel Institute of Technology
Haifa 3200003, Israel
bennyk@cs.technion.ac.il

Abstract

We study the complexity of estimating the probability of an outcome in an election over probabilistic votes. The focus is on voting rules expressed as positional scoring rules, and two models of probabilistic voters: the uniform distribution over the completions of a partial voting profile (consisting of a partial ordering of the candidates by each voter), and the Repeated Insertion Model (RIM) over the candidates, including the special case of the Mallows distribution. Past research has established that, while exact inference of the probability of winning is computationally hard (#P-hard), an additive polynomial-time approximation (additive FPRAS) is attained by sampling and averaging. There is often, though, a need for multiplicative approximation guarantees that are crucial for important measures such as conditional probabilities. Unfortunately, a multiplicative approximation of the probability of winning cannot be efficient (under conventional complexity assumptions) since it is already NP-complete to determine whether this probability is nonzero. Contrastingly, we devise multiplicative polynomial-time approximations (multiplicative FPRAS) for the probability of the complement event, namely, losing the election.

Introduction

Various processes require the preferences of different *voters* over *candidates* to be aggregated towards a joint decision; these include political elections, website rankings in search engines, and multiagent systems. In the general approach that has gained the focus of the field of *computational social choice*, every voter provides a ranking (total order) of the candidates, and a *voting rule* maps the collection of rankings, called a *voting profile*, to a set of selected alternatives, namely the *winners* (Brandt et al. 2016). A well studied family of such rules is that of the *positional scoring rules*: a voter contributes a score to each candidate from a shared *scoring vector* according to the position of the candidate in the total order. This family includes the *plurality* rule defined by the scoring vector $(1, 0, \ldots, 0)$, the *veto* rule defined by $(1, \ldots, 1, 0)$, the *k-approval* rule defined by $(1, \ldots, 1, 0, \ldots, 0)$ starting with k ones, the k-veto rule defined by $(1, \ldots, 1, 0, \ldots, 0)$ ending with k zeros, and the Borda rule defined by $(m-1, m-2, \ldots, 0)$.

Yet, rankings of voters may be *uncertain* due to missing or unreliable information. A simple form of uncertainty is incompleteness: voters provide only *partial orders* over the candidates, thereby collectively forming a *partial voting profile*. Another form of uncertainty is *probability*, where we associate with each voter a probability distribution over the complete rankings of the candidates. For partial voting profiles, Konczak and Lang (2005) introduced the notions of *possible* and *necessary* winners: a candidate c is a *possible winner* if there is a completion of (the partial orders in) the partial profile into a complete profile wherein c is a winner, and c is a *necessary winner* if c is a winner in every completion of the partial profile. In the case of probabilistic votes, every candidate c has a *winning probability*—the probability that c is a winner in a random profile (Bachrach, Betzler, and Faliszewski 2010; Hazon et al. 2012).

The above voting framework entails computational problems. In the *possible winner* problem, we are given a partial profile and a candidate c, and wish to determine whether c is a possible winner. Similarly, in the *necessary-winner* problem, the goal is to determine whether c wins in every completion of the partial profile. In the *winning probability* problem, we are given a representation of the corresponding distribution over profiles (which we discuss later in this section), and a candidate c, and the goal is to calculate the probability that c is a winner. The complexity of the first two problems has been studied thoroughly for the family of positional scoring rules. It has been established that the possible winner problem is solvable in polynomial time for the case of plurality and veto, but NP-complete for any other positional scoring rule,[1] under a common assumption that the rule is *pure*, as we formally define later on (Konczak and Lang 2005; Betzler and Dorn 2010; Baumeister and Rothe 2012). On the other hard, the necessary-winner problem can be solved in polynomial time for every positional scoring rule (Xia and Conitzer 2011).

*This work was supported by the Schmidt Family Foundation.
†This work was supported by ISF Grant 5921551 and NSF-BSF Grant 1814152.

[1] Here and throughout the paper, we make the standard assumption that the scores form a coprime set of natural numbers (Betzler and Dorn 2010).

In this paper, we investigate the winning probability problem. We begin by focusing on the probability distribution studied in past research on this problem—the uniform distribution over all the completions of a given partial profile (Hemaspaandra and Hemaspaandra 2007; Betzler and Dorn 2010). This problem generalizes the possible winner problem in the sense that a candidate c is a possible winner if and only if c has a nonzero probability of winning. Hence, computing the winning probability is intractable in the case of a pure positional scoring rule that is neither plurality nor veto. Moreover, even for these two rules, the problem is #P-hard (Bachrach, Betzler, and Faliszewski 2010; Hazon et al. 2012). But this is not the end of the story, as in realistic settings, it is often the case that *approximate* probabilities suffice. An additive *Fully Polynomial-Time Randomized Approximation Scheme* (FPRAS) can be obtained by a simple Monte Carlo estimation (sample and take the ratio of the times in which c is a winner), provided that we are able to efficiently *sample* a random completion of the partial profile (Hazon et al. 2012; Bachrach, Betzler, and Faliszewski 2010).

Yet, when approximating probabilities, one oftentimes seeks estimations with a guarantee stronger than an additive approximation, namely a *multiplicative* (or *relative*) approximation. This is because a multiplicative guarantee allows for approximating *divisions* of probabilities, which is what we need for estimating conditional probabilities. Nevertheless, this brings us back to the hardness of the possible winner problem—a multiplicative FPRAS allows to detect the possible winners in polynomial time (or, more precisely, in the complexity class BPP), since it allows to distinguish between zero and nonzero probabilities. In particular, there is no multiplicative FPRAS for the winning probability for every positional scoring rule other than plurality and veto. So, what multiplicative guarantees can we make? In this work, we seek such error guarantees for the *complement* probability, namely that of *losing* the elections.

For example, consider an election scenario where a promising candidate c receives the word that she is likely to lose the election. Equipped with this information, she wishes to identify strong candidates she should support in the race. These are the candidates d for which the probability $\Pr(d \text{ loses} | c \text{ loses}) = \Pr(c \text{ and } d \text{ lose})/\Pr(c \text{ loses})$ is relatively low. Now, suppose that a political party wishes to secure a representative among the winners of an election. Towards that, it wishes to determine whether it is worthwhile to invest in candidate c_1 or candidate c_2. So, it might be interested in finding the ratio $\Pr(c_1 \text{ loses})/\Pr(c_2 \text{ loses})$. To approximate both of these ratios, multiplicative approximations are required, while additive guarantees do not suffice.

The main contribution of this paper is a multiplicative FPRAS for the probability of *losing*. We do so for a large class of positional scoring rules, namely the *almost-constant* positional scoring rules, that is, the scoring rules that have the same value in every position, except for a constant number of positions. This class includes plurality, k-approval, veto and k-veto, as well as the rule $(2, 1, 1, \ldots, 1, 0)$ that received a considerable attention (Baumeister, Roos, and Rothe 2011). Note that we do not encounter the computational hardness of distinguishing between zero and nonzero, as this is the tractable necessary-winner problem. To this end, we adapt the Karp-Luby-Madras approximation algorithm (Karp, Luby, and Madras 1989), and the main theoretical challenges we face is in establishing (provable approximations of) the requirements that this algorithm makes on a specific use case. To the best of our knowledge, this paper presents the first multiplicative approximation algorithm in the context of elections over probabilistic voters.

Our FPRAS relies on the ability to sample a complete profile from the uniform distribution over the completions of a given partial profile. Since we assume that the voters rank candidates independently, this problem reduces to the one of sampling a completion of a partial order uniformly at random. The theoretically most efficient algorithms known to date sample completions of a partial order along a rapidly mixing Markov Chain (Sinclair and Jerrum 1989). The worst case mixing time is $\Theta(n^3 \log n)$ for an n-element partial order (Karzanov and Khachiyan 1991; Bubley and Dyer 1999). This mixing time leads to an FPRAS for counting the completions of a partial order that scales as $\epsilon^{-2} n^5$ for an n-element partial order, and relative error ϵ (Huber 2006; Banks et al. 2010). In contrast, exact counting of the completions of a partial order is $\#P$-complete (Brightwell and Winkler 1991). While the worst-case mixing times of the Markov chain are practically prohibitive, recent empirical results have shown that the MCMC approach can be made to scale well in practice (Talvitie, Niinimäki, and Koivisto 2017), and techniques for improving the mixing time are an active area of research (Talvitie et al. 2018a; 2018b).

Finally, we consider another model of probabilistic votes, namely the Repeated Insertion Model (RIM) (Doignon, Pekeč, and Regenwetter 2004), which generalizes other popular probabilistic models such as Mallows (Mallows 1957), generalized Mallows (Fligner and Verducci 1986), and the multistage ranking model (Fligner and Verducci 1988). More precisely, we assume that every voter is represented as an independent RIM model over the set of candidates. We show that, although the voters are not associated with any partial order, the hardness of the possible winners (i.e., determine whether the winning probability is nonzero) still holds. On the positive side, we show that our multiplicative FPRAS can be adjusted to RIM voters.

Due to a lack of space, most of the proofs are omitted from the paper, and will appear in the full version.

Preliminaries

We begin with basic notation and terminology that we use throughout the paper.

Elections and Voting Rules

We use the standard formal modelling of an election, where the winners are determined by preferences of voters over candidates (Brandt et al. 2016). Formally, we have a set $\mathcal{C} = \{c_1, \ldots, c_m\}$ of *candidates* and a set of n *voters*. A *voting profile* $\mathbf{T}$ is a sequence $(\succ_1, \ldots, \succ_n)$ of total (linear) orders over $\mathcal{C}$, where each $\succ_i$ stands for the ranking of the

candidates by the ith voter. A *voting rule* r is a function that maps a given voting profile $\mathbf{T}$ into a nonempty set $r(\mathbf{T})$ of *winners*. A candidate c is a *winner* if $c \in r(\mathbf{T})$.

A large and extensively studied class of voting rules is that of the *positional scoring rules*, where voters score candidates according to the position in the total order. Formally, a *scoring vector* over a set of m candidates is a sequence $\vec{\alpha} = (\alpha_m, \ldots, \alpha_1)$ of m natural numbers such that $\alpha_m \geq \cdots \geq \alpha_1$. Whenever the ith voter positions a candidate c in the jth place according to the linear order $\succ_j$, it contributes the portion $s_i(c) = \alpha_j$ to the total score of c. The winners are the candidates c with a maximum total score $\sum_{i=1}^{n} s_i(c)$. A *positional scoring rule* r is a function that associates a scoring vector with each number m of candidates. Computationally, we assume that r is represented as a polynomial-time computable function that computes, for all m and $j \leq m$, the score $r(j, m)$. In particular, the numbers $\alpha_m \geq \cdots \geq \alpha_1$ are represented by $r(1, m) \geq \cdots \geq r(m, m)$. We denote by $r(\cdot, m)$ the sequence $(r(1, m), \ldots, r(m, m))$.

Well studied instances of positional scoring rules include the *plurality* rule $(1, 0, \ldots, 0)$, where the winners are the top candidates according to the maximal number of the voters; the *k-approval* rule $(1, \ldots, 1, 0, \ldots, 0)$ that begins with k ones, where the winners are the in the top-k candidates according to the maximal number of voters; the *veto* rule $(1, \ldots, 1, 0)$, where the winners are the bottom candidates according to the minimal number of voters; the *k-veto* rule that ends with k zeros, where the winners are in the bottom-k candidates according to the minimal number of voters; and the *Borda* rule $(m-1, m-2, \ldots, 0)$ where the score depends linearly on the position of the candidate.

Let r be a positional scoring rule. We assume that for all $m > 0$, the scoring vector $r(\cdot, m)$ contains at least one nonzero element. We also assume that r is *normalized* in the sense that for every $m > 0$, the greatest common divisor of the numbers in $r(\cdot, m)$ is 1. These assumptions do not restrict the generality of the class of positional scoring rules (Hemaspaandra and Hemaspaandra 2007; Betzler and Dorn 2010). The rule r is *pure* if for every $m \geq 2$, the scoring vector $r(\cdot, m)$ is obtained from $r(\cdot, m-1)$ by inserting a score value at some position. All aforementioned positional scoring rules (plurality, k-approval, etc.) are pure. We say that r is *almost constant* if there is a fixed number K such that for all $m > 0$ it is the case that at least $m - K$ of the scores in $r(\cdot, m)$ are equal. For example, the class of almost constant positional scoring rules includes plurality, k-approval, veto and k-veto, as well as the rule $(2, 1, 1, \ldots, 1, 0)$ that received a considerable attention (Baumeister, Roos, and Rothe 2011). The Borda rule, on the other hand, is *not* almost constant.

A *partial* voting profile is defined similarly to an ordinary (complete) profile, except that the candidate ordering of voters is allowed to be partial. Similarly, in a *probabilistic* voting profile, we replace the total orders of voters with probability distributions over total orders. We formalize these in the next two sections. In both sections, we assume a set $\mathcal{C}$ of m candidates, and a set of n voters.

Partial Voting Profiles

A *partial voting profile* $\mathbf{P}$ is a sequence $(\sqsupset_1, \ldots, \sqsupset_n)$ of partial orders over $\mathcal{C}$. We denote by $lin(\sqsupset_i)$ the set of all linear extensions of $\sqsupset_i$, where a *linear extension* of $\sqsupset_i$ is a total order $\succ_i$ such that $c \succ_i c'$ whenever $c \sqsupset_i c'$. An *extension* of $\mathbf{P}$ is a member of $lin(\sqsupset_1) \times \cdots \times lin(\sqsupset_n)$, that is, a voting profile $\mathbf{T} = (\succ_1, \ldots, \succ_n)$ where each $\succ_i$ is a linear extension of $\sqsupset_i$. Let r be a voting rule. A candidate c is a *possible winner* if there is an extension $\mathbf{T}$ of $\mathbf{P}$ such that c is a is winner under $\mathbf{T}$. Similarly, c is a *necessary winner* if c is a winner under every extension $\mathbf{T}$ of $\mathbf{P}$.

The following is known about the complexity of determining the necessary and possible winners under positional scoring rules. Note that the rule is fixed, and the input consists of the candidate set and partial profile.

Theorem 1. (Xia and Conitzer 2011; Baumeister and Rothe 2012) *Let r be a positional scoring rule.*

- *The necessary winners can be found in polynomial time.*
- *Assume that r is pure. The possible winners can be found in polynomial time if r is either the plurality rule or the veto rule; otherwise, it is NP-complete to determine whether a given candidate is a possible winner.*

Probabilistic Voting Profiles

In a similar manner, a *probabilistic voting profile* $\mathbf{\Pi}$ is a sequence $(\pi_1, \ldots, \pi_n)$, where each π_i is a probability distribution over total orders over $\mathcal{C}$. A *sample* of $\mathbf{\Pi}$ is a voting profile $\mathbf{T} = (\succ_1, \ldots, \succ_n)$ such that each $\succ_i$ is a random total order sampled from π_i. We implicitly assume independence among voters, and therefore, the probability of a sample $\mathbf{T} = (\succ_1, \ldots, \succ_n)$ is given by

$$\Pr_{\mathbf{\Pi}}(\mathbf{T}) \stackrel{\text{def}}{=} \pi_1(\succ_1) \times \cdots \times \pi_n(\succ_n)$$

where $\pi_i(\succ_i)$ is the probability of $\succ_i$ under π_i. Hence, a probabilistic voting profile defines a probability distribution over ordinary voting profiles. In particular, for a voting rule r and a candidate c, the probability that c is a *winner* is given by $\sum_{\{\mathbf{T} | c \in r(\mathbf{T})\}} \Pr_{\mathbf{\Pi}}(\mathbf{T})$.

An example of a probabilistic voting profile is the uniform distribution over all the completions of a partial voting profile $\mathbf{P}$. Such a probabilistic voting profile $\mathbf{\Pi} = (\pi_1, \ldots, \pi_n)$ is represented by a partial voting profile $\mathbf{P} = (\sqsupset_1, \ldots, \sqsupset_n)$ such that $\pi_i(\succ_i)$ is given by

$$\pi_i(\succ_i) \stackrel{\text{def}}{=} \begin{cases} \frac{1}{|lin(\sqsupset_i)|} & \text{if } \succ_i \text{ is a linear extension of } \sqsupset_i; \\ 0 & \text{otherwise.} \end{cases}$$

In other words, each π is the uniform distribution over total orders, a.k.a. *impartial culture* (Black 1958), subject to the precedence constraints stated in $\sqsupset_i$. The probability that candidate c is winner over a partial profile $\mathbf{P}$ is then $\frac{N}{\prod_{i=1}^{n} |lin(\sqsupset_i)|}$, where N is the number of extensions $\mathbf{T}$ of $\mathbf{P}$ in which c is winner. We denote by $\mathrm{Win}(c \mid r, \mathbf{P})$ the event that c is a winner in a random completion of $\mathbf{P}$ according to the positional scoring rule r. When $\mathbf{P}$ and r are clear from the context we denote this event by $\mathrm{Win}(c)$.

As for the complexity of probability computation, recall that *#P-hardness* means that there exists a polynomial-time (Turing) reduction from the problem of counting the solutions for a problem in NP, for instance, the number of truth assignments of a given CNF formula. A *Fully Polynomial-time Randomized Approximation Scheme* (*FPRAS* for short) for a probability function $p(x)$ is a randomized algorithm A that, given as input a problem instance x, an error bound $\epsilon > 0$ and a reliability ratio $\delta > 0$, returns an ϵ-approximation of $p(x)$ with probability $1 - \delta$, where the running time is polynomial in the size of x, in $1/\epsilon$, and in $\log(1/\delta)$. An ϵ-approximation can be *additive*, that is, in $(p(x)-\epsilon, p(x)+\epsilon)$, or *multiplicative*, that is, in $((1-\epsilon)p(x), (1+\epsilon)p(x))$.

The following is known about the complexity of computing the probability of winning for positional scoring rules.

Theorem 2. (Bachrach, Betzler, and Faliszewski 2010; Hazon et al. 2012) *The problem of computing* $\Pr(\mathrm{Win}(c \mid r, \mathbf{P}))$*, given* c *and* $\mathbf{P}$*:*

1. *is #P-hard when r is veto, Borda, and k-approval for every $k > 0$ (including plurality);*
2. *has an additive FPRAS for all positional scoring rules r.*

Later in the paper, we also consider representations of probabilisitic voting profiles by means of the *Repeated Insertion Model* (RIM) (Doignon, Pekeč, and Regenwetter 2004) and its special case of Mallows (Mallows 1957).

Our definitions are given under the semantics of *co-winners*, where an election can have multiple winners, namely, all members of $r(\mathbf{T})$. A candidate c is a *unique winner* if $r(\mathbf{T}) = \{c\}$. While we present our complexity results in the context of co-winners, all of these results apply to the semantics a single winner as well.

Main Result

The existence of an *additive* FPRAS, as stated in Theorem 2, does not imply the existence of a *multiplicative* FPRAS. The difference between the two is fundamental in probability estimation. For example, multiplicative guarantees of probabilities extend to the *division* of probabilities when estimating conditional probabilities; this is not the case for the additive approximation. Moreover, a multiplicative FPRAS allows to determine, with high probability, whether the probability is nonzero (since a multiplicative approximation is zero if and only if the exact number is zero), while an additive approximation does not.

In particular, it follows from Theorem 1 that, under conventional complexity assumptions, there is no multiplicative FPRAS for $\Pr(\mathrm{Win}(c \mid r, \mathbf{P}))$ under the pure positional scoring rules, except for plurality and veto. If there was, we would get a BPP algorithm for determining whether c is a possible winner by executing the FPRAS algorithm and testing whether the result is nonzero (implying NP $\subseteq$ BPP in contradiction to the common belief). Contrasting that, we establish an FPRAS for the *complement* probability, namely the probability of losing, for the class of almost constant scoring rules.

Theorem 3. *Let r be an almost constant positional scoring rule. There is a multiplicative FPRAS for estimating the probability of losing, that is, the problem of computing* $1 - \Pr(\mathrm{Win}(c \mid r, \mathbf{P}))$*, given c and $\mathbf{P}$.*

In particular, we conclude that there is a multiplicative FPRAS for the probability of losing for plurality, k-approval, veto, k-veto, and $(2, 1, 1, \ldots, 1, 0)$. In the next section, we discuss the proof of Theorem 3.

Approximation Algorithm

In this section, we describe a multiplicative FPRAS for the losing probability in elections over the class of almost-constant scoring rules, thereby proving Theorem 3. We adapt the well known estimation technique of Karp-Luby-Madras (Karp, Luby, and Madras 1989) for approximating the number of satisfying assignments of a DNF formula.

Consider an election over a partial profile $\mathbf{P}$, and a positional scoring rule r. Recall that we study the uniform distribution over possible completions $\mathbf{T}$ of $\mathbf{P}$. We denote by $\mathrm{Lose}(c \mid r, \mathbf{P})$ the event that c is *not* a winner in the random completion $\mathbf{T}$ of $\mathbf{P}$ according to the positional scoring rule r. This happens if some candidate $x \in \mathcal{C}$ gains a higher number of points than c. We denote by $\mathrm{L}(x, c|r, \mathbf{P})$ the event that candidate x gains a higher number of points than c in the random extension of $\mathbf{P}$ according to the positional scoring rule r. When r and $\mathbf{P}$ are clear from the context, we denote this event simply by $\mathrm{L}(x, c)$. Therefore:

$$\Pr(\mathrm{Lose}(c)) = \Pr\Big(\bigvee_{x \in \mathcal{C} \setminus \{c\}} \mathrm{L}(x, c)\Big) \qquad (1)$$

The Karp-Luby-Madras algorithm (Karp, Luby, and Madras 1989) provides a multiplicative FPRAS for the probability $\Pr(\vee_{x \in X} E_x)$ of a disjunction X of events, under the assumption that each of the following three tasks can be performed in polynomial time.

1. Test whether E_x is true in a given sample.
2. Compute the exact probability $\Pr(E_x)$ of every E_x.
3. Sample from the posterior distribution conditioned on E_x.

The third task requires a randomized algorithm for producing a random sample s in which E_x is true, so that the probability of each sample s is $\Pr(s)/\Pr(E_x)$.

In our case, the event E_x is $\mathrm{L}(x, c)$. The first task (i.e., testing whether x gains a higher score than c in a complete profile $\mathbf{T}$) is straightforward. However, the second task is not as simple.

Theorem 4. *Computing* $\Pr(\mathrm{L}(x, c \mid r, \mathbf{P}))$*, given* $\mathbf{P}$*, x and c, is $\#P$-hard even for a single voter when r is veto, k-approval for every $k > 0$ (including plurality), or Borda.*

Nevertheless, we are still able to use the Karp-Luby-Madras algorithm by replacing the second and third tasks with approximate versions, as follows.

- We show a multiplicative FPRAS for $\Pr(\mathrm{L}(x, c))$.
- We design a polynomial-time approximate sampling algorithm from the uniform distribution of completions of $\mathbf{P}$ conditioned on $\mathrm{L}(x, c)$, that is, the probability of each completion $\mathbf{T}$ in our sampling is inside the interval

  $$(\Pr(\mathbf{T} \mid \mathrm{L}(x, c))(1-\epsilon), \Pr(\mathbf{T} \mid \mathrm{L}(x, c))(1+\epsilon))$$

  where ϵ is given as part of the input.

We can show that the above approximations allow to retain the FPRAS guarantees of the Karp-Luby-Madras algorithm. Hence, we have the following lemma.

Lemma 1. *If there is an FPRAS for* $\Pr(\mathrm{L}(x, c \mid r, \mathbf{P}))$ *and a polynomial-time approximate sampler from the uniform distributions over the completions of* $\mathbf{P}$ *conditioned on* $\mathrm{L}(x, c)$*, then there is an FPRAS for* $\Pr(\mathrm{Lose}(c \mid r, \mathbf{P}))$.

In the following sections, we devise the two approximation algorithms for the class of almost-constant positional scoring rules.

Approximating the Probability of $\mathrm{L}(x, c)$

Our approach to devising an FPRAS for $\Pr(\mathrm{L}(x, c))$ uses known approximation algorithms for the problem of counting the linear extensions of a partially ordered set. These approximation algorithms are based on the ability to sample rankings uniformly at random from the space of linear extensions of a partial order (Huber 2006; Banks et al. 2010). Such sampling gives rise to a multiplicative FPRAS for counting the linear extensions (Banks et al. 2010).

Recall our assumption that the scoring rule r over the set $\mathcal{C}$ of candidates is almost constant. We denote by K the constant such that $r(\cdot, m)$ contains at least $m-K$ positions with an identical value, which, in turn, we denote by α. Hence, at most K elements of $r(\cdot, m)$ are different from α.

For a partial order $\sqsupset$, let $\Pr(\sqsupset)$ denote the probability of sampling a linear extension of $\sqsupset$ from the uniform distribution over all permutations. That is, $\Pr(\sqsupset) = |lin(\sqsupset)|/m!$. Dividing the FPRAS for counting $lin(\sqsupset)$ by $m!$ gives an FPRAS for $\Pr(\sqsupset)$. We denote by $\overline{\Pr}(\sqsupset)$ the resulting approximation of $\Pr(\sqsupset)$. Then, in time polynomial in m, in $1/\epsilon$, and in $\ln(1/\delta)$, we get the following guarantee:

$$\Pr\Big((1-\epsilon)\Pr(\sqsupset) \leq \overline{\Pr}(\sqsupset) \leq (1+\epsilon)\Pr(\sqsupset)\Big) \geq 1-\delta$$

Let $\sqsupset_1, \ldots, \sqsupset_n$ denote the partial orders of the voters, and let $t \in \{1, \ldots, n\}$. We denote by $F_t[s_x, s_c]$ the event that the candidates x and c gain s_x and s_c points, respectively, from the first t votes (i.e., $\sqsupset_1, \ldots, \sqsupset_t$). We can then define the required probability $\Pr(\mathrm{L}(x, c))$ as:

$$\Pr(\mathrm{L}(x, c)) = \sum_{s_x > s_c} \Pr(F_n[s_x, s_c]) \tag{2}$$

The algorithm proceeds by approximating the probabilities of the events $F_t[s_x, s_c]$ for all $t \in \{1, \ldots, n\}$ and scores s_x and s_c, by dynamic programming over t.

For every partial order $\sqsupset_t$, we look at all possible combinations (s_x^t, s_c^t) of points that the candidates x and c can gain from the tth voter. The key to the efficiency of our algorithm, and the way we take advantage of the almost-constant property, is the distinction between the actual complete vote (i.e., ranking) cast by voter t, and her *voting outcome*. A voting outcome captures the information in the tth vote by specifying, for every one of the $K+1$ score values in $r(\cdot, m)$, the set of candidates that receive this score in by the tth voter.

Definition 1. *Denote by* $\mathsf{Vals}(r, m)$ *the set of distinct scores in* $r(\cdot, m)$*, and by* $2^{\mathcal{C}}$ *the set of subsets of* $\mathcal{C}$*. The voting outcome of the partial vote* $\sqsupset_t$ *is a function* $f_t : \mathsf{Vals}(r, m) \to 2^{\mathcal{C}}$ *that maps each score value to the set of candidates in* $\mathcal{C}$ *that are granted this score in* $\succ_t$.

Let $s_1 > s_2 > \cdots > s_k$ be the scores in $\mathsf{Vals}(r, m)$ where, by assumption, $k \leq K+1$. A voting outcome f_t induces a partial order $\sqsupset_t^f$ over the candidates where

$$\sqsupset_t^f \stackrel{\mathsf{def}}{=} f_t(s_1) \succ f_t(s_2) \succ \cdots \succ f_t(s_k).$$

Here, $f_t(s_i) \succ f_t(s_{i+1})$ means that all candidates mapped to s_i precede all candidates mapped to s_{i+1}, and the candidates in $f_t(s_i)$ (and $f_t(s_{i+1})$) are incomparable. We say that a voting outcome f_t is *valid* if the partial orders $\sqsupset_t$ and $\sqsupset_t^f$ are consistent. Voting outcomes resulting in invalid partial orders can be ignored, since they have a zero probability.

Lemma 2. *The number of voting outcomes over an almost constant positional scoring rule is polynomial in* m*, the number of candidates.*

Proof. Every voting outcome is determined by the assignment of candidates to one of (at most) K score values different from α (i.e., the value that appears in $m-K$ positions of $r(\cdot, m)$). Let $\mathsf{Vals}(r, m) \setminus \{\alpha\}$ be $\{s'_1, \ldots, s'_{k-1}\}$. Let d_i be the cardinality of the set $f_t(s'_i)$ (i.e., $d_i = |f_t(s'_i)|$). By definition, each d_i is at most K. Therefore, the number of voting outcomes is:

$$\begin{aligned}&\binom{m}{K}\binom{K}{d_1}\binom{K-d_1}{d_2}\cdots\binom{K-d_1-\cdots-d_{k-1}}{d_k}\\ &= \frac{m!}{(m-K)!d_1!\ldots d_k!} \leq m^K\end{aligned}$$

This completes the proof. □

Given two consistent partial orders $\sqsupset_1$ and $\sqsupset_2$, we denote by $\sqsupset_1 \wedge \sqsupset_2$ the partial order that results from combining (i.e., taking the union of) the two.

Since every voting outcome f_t induces a unique partial order $\sqsupset_t^f$, we identify f_t with $\sqsupset_t^f$. We denote by $\mathcal{O}_t[s_x^t, s_c^t]$ the combination of $\sqsupset_t$ with the set of all valid voting outcomes f_t, where x is granted s_x^t points (i.e., $x \in f_t(s_x^t)$), and c is granted s_c^t points (i.e., $c \in f_t(s_c^t)$). Formally:

$$\mathcal{O}_t[s_x^t, s_c^t] = \{\sqsupset_t \wedge \sqsupset_t^f \mid x \in f_t(s_x^t), c \in f_t(s_c^t)\} \tag{3}$$

Therefore, approximating $\Pr(\mathcal{O}_t[s_x^t, s_c^t])$ involves enumerating all partial orders $\sqsupset_t^f$ where $x \in f_t(s_x^t)$, and $c \in f_t(s_c^t)$, combining them with the partial order $\sqsupset_t$, and approximating their count. Formally:

$$\overline{\Pr}(\mathcal{O}_t[s_x^t, s_c^t]) = \sum_{\sqsupset \in \mathcal{O}_t[s_x^t, s_c^t]} \overline{\Pr}(\sqsupset) \tag{4}$$

where $\overline{\Pr}(\sqsupset)$ is calculated by applying an FPRAS for counting the linear extensions of $\sqsupset$ (Huber 2006), and dividing by $m!$. By Lemma 2, the cardinality of $\mathcal{O}_t[s_x^t, s_c^t]$ is polynomial in m for almost-constant positional scoring rules. The dynamic programming algorithm proceeds as follows.

Base Case: Computing $F_1[s_x, s_c]$. For every pair (a, b) where $a, b \in \mathsf{Vals}(r, m)$, the event $F_1[a, b]$ amounts to computing the probability of the partial orders in $\mathcal{O}_1[a, b]$ as in (4).

$$\overline{\Pr}(F_1[a, b]) = \overline{\Pr}(\mathcal{O}_1[a, b]) \tag{5}$$

Step: Computing $F_t[s_x, s_c]$. In this case, we consider all possible configurations of points (a, b) that can bring the score value to (s_x, s_c) after processing the tth vote.

$$\overline{\Pr}(F_t[s_x, s_c]) = \sum_{a,b \in r(\cdot,m)} \overline{\Pr}(F_{t-1}[s_x - a, s_c - b]) \cdot \overline{\Pr}(\mathcal{O}_t[a, b]) \tag{6}$$

where, again, $\overline{\Pr}(\mathcal{O}_t[a, b])$ is computed as in (4).

Analysis In order to show that the dynamic programming algorithm described in (2), (5), and (6) runs in polynomial time, we need to prove the following for almost constant scoring rules: (1) the set of possible score values s_x and s_c that candidates x and c can obtain in any extension of the partial profile is polynomial in m. (2) For every pair of score values $a, b \in r(\cdot, m)$ the cardinality of the set $\mathcal{O}_t[a, b]$ is polynomial in m. The former is stated in Lemma 3, and the latter in Lemma 2. Detailed proofs will be included in the complete version of the paper.

Lemma 3. *Let r be an almost constant positional scoring rule with at most K positions containing values different from α. Then, for any number of candidates $m \in \mathbb{Z}$, the number of possible scores over n votes is in $O((n + K)^K)$.*

Let ε denote the relative error of the approximation for $\Pr(\mathrm{L}(x, c))$. By (2), this is the approximation required for the probabilities of the events $F_n[s_x, s_c]$. Let ρ denote the relative error bound used for the FPRAS for counting linear extensions. In the proof of Theorem 5, we show that by setting $\rho = \frac{\ln(1+\varepsilon)}{t^2}$ for every step $t \in \{1, \ldots, n\}$ in the dynamic programming algorithm, we arrive at an FPRAS for computing $F_t[a, b]$ with a relative error factor of ε. By (2) this gives us an FPRAS for $\Pr(\mathrm{L}(x, c))$ with the required approximation.

Theorem 5. *The dynamic-programming algorithm described in* (5) *and* (6) *is an FPRAS for $F_t[a, b]$.*

Sampling Conditioned on $\mathrm{L}(x, c)$

We now consider the task of sampling from the subspace conditioned on $\mathrm{L}(x, c)$. Recall that our goal is to sample a complete profile $\mathbf{T} = \{\succ_1, \ldots, \succ_n\}$ where each $\succ_t \in lin(\sqsupset_t)$ is drawn, uniformly at random, from the subspace of rankings conditioned on the event $\mathrm{L}(x, c)$.

The algorithm samples the complete votes sequentially, starting from $\succ_1$. For each voter t, we sample a complete vote $\succ_t$, a linear extension of $\sqsupset_t$, given the total number of points that the candidates obtained in votes $\succ_1, \ldots, \succ_{t-1}$ denoted s_x^{t-1} and s_c^{t-1}. The sampling of $\succ_t$, given the candidates' current score, proceeds as follows.

1. The algorithm samples the scores (a, b) that candidates x and c are awarded in $\succ_t$. This probability corresponds to the event of generating a complete ranking that is consistent with a partial order in $\mathcal{O}_t[a, b]$ (see (3)). By the previous section, we have an FPRAS for approximating the probability $\Pr(\mathcal{O}_t[a, b])$ of this set of partial orders (see (4)).
2. The algorithm samples, uniformly at random, a partial order $\sqsupset$ from $\mathcal{O}_t[a, b]$ (see (3)). By Lemma 2, the cardinality of $\mathcal{O}_t[a, b]$ is polynomial in m.
3. The algorithm samples, uniformly at random, a linear extension $\succ_t$ of $\sqsupset$ by using the uniform sampler of (Huber 2006; Banks et al. 2010).

The idea behind the sampling is, that once the scores (a, b) of candidates x and c are determined, the actual linear extension drawn from the set rankings that extend the partial vote $\sqsupset_t$ (i.e., $lin(\sqsupset_t)$) is independent of the event $\mathrm{L}(x, c)$.

For every voter $t \in \{1, \ldots, n\}$, and every pair $(a, b) \in r(\cdot, m)$, we denote by $Q(a, b, t)$ the event that candidates x and c gain a and b points, respectively, in $\succ_t$, given their scores s_x^{t-1}, s_c^{t-1} over votes $\{\succ_1, \ldots, \succ_{t-1}\}$, and the fact that x gained more points than c (i.e., the event $L(x, c)$). Formally: $\Pr(Q(a, b, t)) = \Pr(\mathcal{O}_t[a, b] \mid \mathrm{L}(x, c), s_x^{t-1}, s_c^{t-1})$.

Figure 1 contains the pseudocode of the sampling algorithm. Relative approximations to the probabilities of the events $Q(a, b, t)$ for all $a, b \in r(\cdot, m)$, and $t \in \{1, \ldots, n\}$ are calculated in lines 6-9, where the algorithm applies the techniques described in the previous section in order to compute the relative approximations of these events (line 9).

The main ingredient of computing $\overline{\Pr}(Q(a, b, t))$ is an FPRAS for $\Pr(\mathrm{L}(x, c) | F_k[s_x, s_c])$. By (2), we have that

$$\overline{\Pr}(\mathrm{L}(x, c) | F_k[s_x, s_c]) = \sum_{s_x^n > s_c^n} \overline{\Pr}(F_n[s_x^n, s_c^n] | F_k[s_x, s_c]) .$$

The algorithm for computing $\overline{\Pr}(F_n[s_x^n, s_c^n] | F_k[s_x, s_c])$ is similar to the dynamic programming algorithm for computing $\overline{\Pr}(F_n[s_x^n, s_c^n])$, presented in the previous section, and is therefore omitted. In the full version of this paper we will show that $\overline{\Pr}(F_n[s_x^n, s_c^n] | F_k[s_x, s_c])$ can be approximated to within an error bound of ε by applying the FPRAS for counting linear extensions a polynomial number of times with an approximation ratio of $\rho = \frac{\ln(1+\varepsilon)}{(n-k)^2}$.

In Lemma 4 we show that the expression in line 9 of the sampling algorithm is a relative approximation for the probability of the event $Q(a, b, t)$. Formally, the probability we use for sampling a vote $\succ_t$ from the subspace conditioned on $\mathrm{L}(x, c)$ is inside the interval $(\Pr(Q(a, b, t))(1 - \varepsilon'), \Pr(Q(a, b, t))(1 + \varepsilon'))$. Since the voters cast their votes independently, then by setting $\varepsilon' = \frac{\ln(1+\varepsilon)}{n^2}$, the probability of sampling a complete profile $\mathbf{T}$ from the subspace conditioned on $\mathrm{L}(x, c)$ is inside the interval $(\Pr(\mathbf{T} \mid \mathrm{L}(x, c))(1 - \varepsilon), \Pr(\mathbf{T} \mid \mathrm{L}(x, c))(1 + \varepsilon))$, as required.

Lemma 4. *There is an FPRAS for $Q(a, b, t)$. Specifically, the following equality holds:*

$$\Pr(Q(a, b, t)) = \frac{\Pr\left(\mathrm{L}(x, c) | F_t[s_x, s_c]\right) \cdot \Pr(\mathcal{O}_t[a, b])}{\Pr\left(\mathrm{L}(x, c) | F_{t-1}[s_x^{t-1}, s_c^{t-1}]\right)}$$

Algorithm sample($\mathbf{P} = \{\sqsupset_1, \ldots, \sqsupset_n\}$,$\mathrm{L}(x, c)$)

$s_c^0 \leftarrow 0$ (initialize score for c)
$s_x^0 \leftarrow 0$ (initialize score for x)
$m \leftarrow |\mathcal{C}|$
for $t = 1, \ldots, n$ **do**
  for all $(a, b) \in r(\cdot, m)$ **do**
    $s_x := s_x^{t-1} + a$
    $s_c := s_c^{t-1} + b$
    Compute $\overline{\Pr}(Q(a, b, t))$ as follows:
    $\overline{\Pr}(Q(a, b, t)) = \frac{\overline{\Pr}(\mathrm{L}(x,c)|F_t[s_x, s_c]) \cdot \overline{\Pr}(\mathcal{O}_t[a,b])}{\overline{\Pr}(\mathrm{L}(x,c)|F_{t-1}[s_x^{t-1}, s_c^{t-1}])}$
  Randomly choose (a, b) with prob. $\overline{\Pr}(Q(a, b, t))$.
  Randomly choose $\sqsupset \in \mathcal{O}_t[a, b]$ with prob. $\frac{1}{|\mathcal{O}_t[a,b]|}$
  Randomly choose $\succ_t$ uniformly from $lin(\sqsupset)$
  $s_c^t \leftarrow s_c^{t-1} + a$
  $s_x^t \leftarrow s_x^{t-1} + b$
return $\{\succ_1, \ldots, \succ_n\}$

Figure 1: Sampling conditioned on $\mathrm{L}(x, c)$

Elections over RIM Voters

In this section, we assume that the voter preferences are represented by a parametric model called the *Repeated Insertion Model* (RIM). A voter profile is a RIM profile if every vote is represented by a RIM model. We begin by describing RIM and then show how our approach carries over to elections over RIM profiles. Specifically, we show how to efficiently compute the probability $\Pr(\mathrm{L}(x, c))$, and how to sample a complete profile conditioned on the event $\mathrm{L}(x, c)$.

The Repeated Insertion Model (RIM)

An instance of RIM is defined by a generative process with two parameters, $\boldsymbol{\sigma}$ and Π, and is denoted by $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$. The parameter $\boldsymbol{\sigma}$ is a ranking $\langle \sigma_1, \ldots, \sigma_m \rangle$, referred to as a *reference ranking*. The model $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ defines a probability distribution over the sample space $lin(\{\sigma_1, \ldots, \sigma_m\})$, which is the set of all rankings over the items of $\boldsymbol{\sigma}$. The parameter Π, referred to as the *insertion probability function*, maps every pair (i, j) of integers, with $1 \leq j \leq i \leq m$, to a probability $\Pi(i, j) \in [0, 1]$, so that $\sum_{j=1}^{i} \Pi(i, j) = 1$ for all $i = 1, \ldots, m$. Semantically, a ranking is generated by the following randomized process. Beginning with the empty ranking, scan the items $\sigma_1, \ldots, \sigma_m$ in order, starting with σ_1. Each σ_i is inserted into a random position $j \in \{1, \ldots, i\}$ with probability $\Pi(i, j)$ into the current $\langle \tau_1, \ldots, \tau_{i-1} \rangle$, pushing $\tau_j, \ldots, \tau_{i-1}$ forward and resulting in $\langle \tau_1, \ldots, \tau_{j-1}, \sigma_i, \tau_j, \ldots, \tau_{i-1} \rangle$. Importantly, the insertion position of σ_i is probabilistically independent of the positions of the previous $\sigma_1, \ldots, \sigma_{i-1}$. An easy observation is that every insertion sequence gives rise to a unique ranking.

The above process defines a probability, denoted $\Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau})$, for each ranking $\boldsymbol{\tau}$ over $\{\sigma_1, \ldots, \sigma_m\}$, as follows. Let $J = \langle j_1, \ldots, j_m \rangle$ denote the *insertion vector* for $\boldsymbol{\tau}$, that is, $j_i \in [1, i]$ is the position into which σ_i is inserted into $\langle \tau_1, \ldots, \tau_{i-1} \rangle$. Then:

$$\Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau}) = \prod_{i=1}^{m} \Pi_{\boldsymbol{\sigma}}(i, j_i) \tag{7}$$

A special case of RIM is the *Mallows* model (Mallows 1957), parameterized by a reference ranking $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$ and a *dispersion parameter* $\phi \in (0, 1]$. The model assigns to every ranking $\boldsymbol{\tau}$ a probability defined by

$$\Pr(\boldsymbol{\tau} \mid \boldsymbol{\sigma}, \phi) \stackrel{\text{def}}{=} \frac{1}{Z} \phi^{d(\boldsymbol{\tau}, \boldsymbol{\sigma})} . \tag{8}$$

Here, $d(\boldsymbol{\tau}, \boldsymbol{\sigma})$ is *Kendall's tau* distance between $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}$ that counts the number of pairwise disagreements between $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}$ (Kendall 1938), and Z is a normalization factor. Doignon, Pekeč, and Regenwetter (2004) showed that the Mallows distribution with parameters $\boldsymbol{\sigma}, \phi$ is the same as $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ where $\Pi(i, j) = \phi^{i-j}/(1 + \phi + \cdots + \phi^{i-1})$.

Approximations for RIM Profiles

We consider an election with m candidates, a positional scoring rule r, and a *RIM profile* $\mathcal{R} = \{R_1, \ldots, R_n\}$ where every voter $R_i = \mathsf{RIM}(\boldsymbol{\sigma}_i, \Pi_i)$ is (represented as) a RIM model over the candidates. We begin by proving the following hardness result.

Theorem 6. *It is NP-complete to determine whether the winning probability of a candidate c is nonzero, in an election with a given RIM profile, over every pure positional scoring rule other than k-approval, and k-veto, for any $k \geq 1$, and $(2, 1, \ldots, 1, 0)$.*

Next, we discuss the proof of Theorem 6. We call a partial order $\sqsupset$ over the m candidates *all-but-one complete* if it induces a linear order over (at least) $m - 1$ of the candidates. The reductions used by Dey and Misra (2017) for proving the hardness of the possible-winner problem for all but k-approval, and k-veto, for any $k \geq 1$, and $(2, 1, \ldots, 1, 0)$, use partial orders that are all-but-one complete. Hence, we get the following.

Theorem 7. (Dey and Misra 2017) *Let r be a pure positional scoring rule that is neither k-approval, nor k-veto, for any $k \geq 1$, nor $(2, 1, \ldots, 1, 0)$. It is NP-complete to determine whether a given candidate is a possible winner, even if every vote is all-but-one complete.*

In the complete version of the paper we will include the details of how RIM can be used to emulate any all-but-one complete partial order. Therefore, for the positional scoring rules excluding k-approval, and k-veto (for any $k \geq 1$), and $(2, 1, \ldots, 1, 0)$, we reduce the possible-winner problem over an all-but-one partial profile (shown to be NP-complete (Dey and Misra 2017)) to possible-winner over a RIM profile.

Since a multiplicative FPRAS for the winning probability over RIM would solve the non-zeroness problem, we conclude from Theorem 6 that it is unlikely to exist. Here, we again seek an FPRAS for the complement probability.

Recall that to apply the Karp-Luby-Madras algorithm for estimating the losing probability of a distinguished candidate c, we again need to be able to perform the following in

polynomial time for candidates $x \neq c$: *(a)* compute the probability $\Pr(\mathrm{L}(x, c))$, *and (b)* sample a complete profile from the subspace conditioned on $\mathrm{L}(x, c)$. We prove that these are possible (the details will be given in the full version of the paper). In fact, in contrast to Theorem 3, this result applies not only to every almost-constant rule, but rather to *every* positional scoring rule. The main result is then as follows.

Theorem 8. *Let r be a positional scoring rule. There is a multiplicative FPRAS for estimating the probability of losing, given a RIM profile $\mathcal{R}$ and a candidate c.*

Conclusions

We developed a multiplicative FPRAS for the probability of losing in elections with positional scoring rules. We focused on two models of probabilistic voters: a uniform distribution over the completions of incomplete profiles, for which we covered the almost-constant scoring rules, and RIM, for which we covered all positional scoring rules. Immediate open problems for future investigation include rules beyond almost-constant ones in the first model, approximating the winning probability in the case of plurality and veto, and approximation of expressive queries over election outcomes in the context of a database (Kimelfeld, Kolaitis, and Stoyanovich 2018).

References

Bachrach, Y.; Betzler, N.; and Faliszewski, P. 2010. Probabilistic possible winner determination. In *AAAI*.

Banks, J.; Garrabrant, S.; Huber, M. L.; and Perizzolo, A. 2010. Using TPA to count linear extensions. *CoRR* abs/1010.4981.

Baumeister, D., and Rothe, J. 2012. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. *Inf. Process. Lett.* 112(5):186–190.

Baumeister, D.; Roos, M.; and Rothe, J. 2011. Computational complexity of two variants of the possible winner problem. In *AAMAS*, 853–860.

Betzler, N., and Dorn, B. 2010. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *J. Comput. Syst. Sci.* 76(8):812–836.

Black, D. 1958. *The Theory of Committees and Elections*. Cambridge: Cambridge University Press.

Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds. 2016. *Handbook of Computational Social Choice*. Cambridge University Press.

Brightwell, G., and Winkler, P. 1991. Counting linear extensions. *Order* 8(3):225–242.

Bubley, R., and Dyer, M. 1999. Faster random generation of linear extensions. *Discrete Mathematics* 201(1):81 – 88.

Dey, P., and Misra, N. 2017. On the exact amount of missing information that makes finding possible winners hard. In *MFCS*, 57:1–57:14.

Doignon, J.-P.; Pekeč, A.; and Regenwetter, M. 2004. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika* 69(1):33–54.

Fligner, M. A., and Verducci, J. S. 1986. Distance based ranking models. *Journal of the Royal Statistical Society. Series B (Methodological)* 48(3):359–369.

Fligner, M. A., and Verducci, J. S. 1988. Multistage ranking models. *Journal of the American Statistical Association* 83(403):892–901.

Hazon, N.; Aumann, Y.; Kraus, S.; and Wooldridge, M. 2012. On the evaluation of election outcomes under uncertainty. *Artificial Intelligence* 189:1 – 18.

Hemaspaandra, E., and Hemaspaandra, L. A. 2007. Dichotomy for voting systems. *Journal of Computer and System Sciences* 73(1):73 – 83.

Huber, M. 2006. Fast perfect sampling from linear extensions. *Discrete Mathematics* 306(4):420–428.

Karp, R. M.; Luby, M.; and Madras, N. 1989. Monte-carlo approximation algorithms for enumeration problems. *J. Algorithms* 10(3):429–448.

Karzanov, A., and Khachiyan, L. 1991. On the conductance of order markov chains. *Order* 8(1):7–15.

Kendall, M. G. 1938. A new measure of rank correlation. *Biometrika* 30(1/2):81–93.

Kimelfeld, B.; Kolaitis, P. G.; and Stoyanovich, J. 2018. Computational social choice meets databases. In *IJCAI*, 317–323. ijcai.org.

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

Mallows, C. L. 1957. Non-null ranking models. i. *Biometrika* 44(1-2):114–130.

Sinclair, A., and Jerrum, M. 1989. Approximate counting, uniform generation and rapidly mixing markov chains. *Inf. Comput.* 82(1):93–133.

Talvitie, T.; Kangas, K.; Niinimäki, T. M.; and Koivisto, M. 2018a. Counting linear extensions in practice: MCMC versus exponential monte carlo. In *AAAI*.

Talvitie, T.; Kangas, K.; Niinimäki, T. M.; and Koivisto, M. 2018b. A scalable scheme for counting linear extensions. In *IJCAI*, 5119–5125.

Talvitie, T.; Niinimäki, T.; and Koivisto, M. 2017. The mixing of markov chains on linear extensions in practice. In *IJCAI*, 524–530. AAAI Press.

Xia, L., and Conitzer, V. 2011. Determining possible and necessary winners given partial orders. *J. Artif. Intell. Res.* 41:25–67.