The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Probabilistic Inference Over Repeated Insertion Models

Batya Kenig
Technion – Israel Institute of Technology
batyak@cs.technion.ac.il

Lovro Ilijasić
Drexel University
lovro@drexel.edu

Haoyue Ping
Drexel University
hp354@drexel.edu

Benny Kimelfeld
Technion – Israel Institute of Technology
bennyk@cs.technion.ac.il

Julia Stoyanovich
Drexel University
stoyanovich@drexel.edu

Abstract

Distributions over rankings are used to model user preferences in various settings including political elections and electronic commerce. The Repeated Insertion Model (RIM) gives rise to various known probability distributions over rankings, in particular to the popular Mallows model. However, probabilistic inference on RIM is computationally challenging, and provably intractable in the general case. In this paper we propose an algorithm for computing the marginal probability of an arbitrary partially ordered set over RIM. We analyze the complexity of the algorithm in terms of properties of the model and the partial order, captured by a novel measure termed the "cover width". We also conduct an experimental study of the algorithm over serial and parallelized implementations. Building upon the relationship between inference with rank distributions and counting linear extensions, we investigate the inference problem when restricted to partial orders that lend themselves to efficient counting of their linear extensions.

Introduction

With the availability and wealth of preference data, it has become increasingly important to develop statistical models and techniques that enable the use of this information to reason about unseen preferences and make predictions. Examples include management and analysis of *elections* (Gormley and Murphy 2008; McElroy and Marsh 2009), and recommendation systems in *electronic commerce* (Das Sarma et al. 2010). Once a statistical model has been built, these applications give rise to the problem of *reasoning*: how to efficiently use the statistical model to draw probabilistic inferences from observations.

Ranking models from the statistics literature, such as the Mallows model (Mallows 1957), the Plackett-Luce model (Plackett 1975; Luce 1959) and others (Marden 1995), play a major role in machine learning applications for analyzing preference data. Among these, the Mallows model is perhaps the most popular (Awasthi et al. 2014; Busse, Orbanz, and Buhmann 2007; Doignon, Pekeč, and Regenwetter 2004; Lu and Boutilier 2014). Generally, data about a user's preferences will take the form of arbitrary pairwise comparisons (Lu and Boutilier 2014). Such preferences can be as simple as a single paired comparison: "a is better than b", or as complex as any partially ordered set (poset) over the alternatives.

Computing the marginal probability of a poset over the Mallows distribution is not only important for probabilistic inference over arbitrary pairwise comparisons, but also plays a crucial role in learning model parameters (Lu and Boutilier 2014) (i.e., evaluating the log-likelihood under Mallows with respect to arbitrary pairwise preferences). Yet, this task is theoretically and practically intractable (Lu and Boutilier 2014). Therefore, research to date has either resorted to approximation algorithms (Lu and Boutilier 2014), or assumed very restricted forms of evidence about individual user preferences ranging from complete rankings, to top-t/bottom-t alternatives, to *partitioned preferences* (Lebanon and Mao 2008). A simple proof of hardness is via a reduction to the problem of counting the poset's linear extensions that is known to be $\#P$-complete (Brightwell and Winkler 1991). While this counting problem has been thoroughly investigated, and the properties that determine its complexity are fairly well understood (Eiben et al. 2016; Kangas et al. 2016; De Loof, De Meyer, and De Baets 2006), the question of which parameters govern the complexity of inference over Mallows remains unexplored. Furthermore, while there exist broad classes of posets that allow for tractable counting, notably, *series-parallel* (Möhring 1989) and *polytree* (Atkinson 1990), to the best of our knowledge, over Mallows, the only known tractable class of posets is that of the partitioned preferences (Lebanon and Mao 2008).

The Mallows model represents a population of users by a reference ranking $\boldsymbol{\sigma}$ and a dispersion parameter ϕ, and expresses the probability of a ranking as a function of its distance from $\boldsymbol{\sigma}$. Doignon et al. (2004) introduced the Repeated Insertion Model (RIM), an efficient generative procedure for the Mallows distribution. RIM considers every alternative of the reference ranking $\boldsymbol{\sigma} = \sigma_1, \ldots, \sigma_m$ in order, and places each alternative σ_i at a random position j inside the current list (consisting of $\sigma_1, \ldots, \sigma_{i-1}$) independently from previous insertions. RIM was shown to be the generative procedure underlying not only Mallows, but a large class of *subset choice* models that include the *Generalized Mallows* (Fligner and Verducci 1986) and *multistage ranking* (Fligner and Verducci 1988) models as a spe-

cial cases (Doignon, Pekeč, and Regenwetter 2004).

In this paper we study the problem of computing the marginal probability of a poset over a RIM distribution. A common thread in our theoretical results and empirical evaluation is that the computational complexity of this task depends not only on the structure of the poset, as is the case for counting linear extensions, but also heavily relies on the reference ranking $\boldsymbol{\sigma}$.

Contributions. (1) We devise RIMDP, an algorithm that computes the exact marginal probability of a poset over a RIM model. We prove that the complexity of this algorithm is polynomial in the size of the reference ranking $\boldsymbol{\sigma}$ and exponential in $\mathsf{cw}(\boldsymbol{\sigma})$, the *cover width* of the poset with respect to $\boldsymbol{\sigma}$. (2) We characterize two classes of posets, termed *disjoint series-parallel* and *monotonic polytree*, that refine the well known families of series-parallel and polytree posets. We provide a dynamic programming algorithm that efficiently computes the probability of these posets over Mallows. (3) We experimentally evaluate the performance of RIMDP and examine the relationship between its running time and the cover width parameter in practice. We further show significant performance gains via parallelization.

Preliminaries

In this section we present some basic notation and terminology that we use throughout the paper.

Orders and rankings. A *partially ordered set (poset)* is a binary relation $\succ$ over a set of items A that satisfies transitivity ($a \succ b$ and $b \succ c$ implies $a \succ c$) and irreflexivity ($a \succ a$ never holds). For $a, b \in A$, we say that a is a *cover* of b, denoted $a \succ\!\!\cdot\, b$, if $a \succ b$ and there is no $a' \in A$ such that $a \succ a' \succ b$. A partial order is uniquely identified by its *cover relation* $\succ\!\!\cdot$ and can be represented by the *cover graph*, where an edge directed from a to b stands for $a \succ\!\!\cdot\, b$. See for example Figure 3.

A *linear* (or *total*) order is a poset where every two items are comparable. We identify a linear order $\sigma_1 \succ \cdots \succ \sigma_m$ with the sequence $\langle \sigma_1, \ldots, \sigma_m \rangle$, called a *ranking* (*over* $\{\sigma_1, \ldots, \sigma_m\}$). If $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$ is a ranking, then $\mathrm{items}(\boldsymbol{\sigma})$ denotes the set $\{\sigma_1, \ldots, \sigma_m\}$, and $\succ_{\boldsymbol{\sigma}}$ denotes the order $\boldsymbol{\sigma}$ stands for; that is, $\sigma_i \succ_{\boldsymbol{\sigma}} \sigma_j$ whenever $i < j$. By $\boldsymbol{\sigma}(\tau)$ we denote the position of an item τ in a ranking $\boldsymbol{\sigma}$; that is, if $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$ and $\tau = \sigma_i$, then $\boldsymbol{\sigma}(\tau) = i$. A ranking $\boldsymbol{\sigma}$ is *compatible* with $\succ$ if $\boldsymbol{\sigma}(a) < \boldsymbol{\sigma}(b)$ whenever $a \succ b$. We denote by $rnk(A)$ the set of all rankings over A, and by $rnk(A \mid \succ)$ the set of rankings over A that are compatible with $\succ$.

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 $rnk(\mathrm{items}(\boldsymbol{\sigma}))$, that is, the rankings over the items of $\boldsymbol{\sigma}$. The parameter Π, called an *insertion probability function* (or just *insertion function* for short), maps every pair of integers (i, j), with $1 \leq j \leq i \leq m$, to a probability $\Pi(i, j) \in [0, 1]$, so that for all $i = 1, \ldots, m$ we have $\sum_{j=1}^{i} \Pi(i, j) = 1$. Semantically, a ranking is generated by the following randomized process. We begin with the empty ranking, and 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 $\Pi(i, j)$ inside the current series $\langle \tau_1, \ldots, \tau_{i-1} \rangle$, pushing $\tau_j, \ldots, \tau_{i-1}$ forward and resulting in the series $\langle \tau_1, \ldots, \tau_{j-1}, \sigma_i, \tau_j, \ldots, \tau_{i-1} \rangle$. Importantly, the insertion position of each σ_i is probabilistically independent of the positions of the previous items $\sigma_1, \ldots, \sigma_{i-1}$. 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 $\mathrm{items}(\boldsymbol{\sigma})$, 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:

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

The Mallows model. 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* $\phi \in (0, 1]$, and denoted $\mathcal{M}(\boldsymbol{\sigma}, \phi)$. The model assigns to every $\boldsymbol{\tau} \in rnk(\mathrm{items}(\boldsymbol{\sigma}))$ a non-zero probability defined by

$$\Pr(\boldsymbol{\tau} \mid \boldsymbol{\sigma}, \phi) \stackrel{\mathsf{def}}{=} \frac{1}{Z} \phi^{d(\boldsymbol{\tau}, \boldsymbol{\sigma})} . \tag{2}$$

Here, $d(\boldsymbol{\tau}, \boldsymbol{\sigma})$ is *Kendall's tau* (Kendall 1938) distance between $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}$ that counts the disagreements between $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}$: $d(\boldsymbol{\tau}, \boldsymbol{\sigma}) \stackrel{\mathsf{def}}{=} \sum_{1 \leq i < j \leq m} \mathbb{1}[\boldsymbol{\tau}(\sigma_j) < \boldsymbol{\tau}(\sigma_i)]$. The indicator function $\mathbb{1}$ assigns 1 to true statements and 0 to false ones. The *normalization constant* Z is the sum of $\phi^{d(\boldsymbol{\tau}, \boldsymbol{\sigma})}$ over all $\boldsymbol{\tau} \in rnk(\mathrm{items}(\boldsymbol{\sigma}))$, and is known to be equal to $(1 + \phi)(1 + \phi + \phi^2) \ldots (1 + \cdots + \phi^{m-1})$.

Intuitively, the larger the distance of a ranking $\boldsymbol{\tau}$ is from the reference ranking $\boldsymbol{\sigma}$, the lower its probability under the Mallows model. Lower values of ϕ concentrate most of the probability mass around $\boldsymbol{\sigma}$, while $\phi = 1$ gives the uniform distribution over $rnk(\mathrm{items}(\boldsymbol{\sigma}))$. Doignon et al. (2004) showed that $\mathcal{M}(\boldsymbol{\sigma}, \phi)$ is the same as $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ where $\Pi(i, j) = \phi^{i-j} / (1 + \phi + \cdots + \phi^{i-1})$.

Inference over RIM. The inference problem we investigate in this paper is that of *computing the marginal probability of a poset*; that is, we wish to compute the probability that a ranking generated by $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ satisfies a poset $\succ$:

$$\Pr(\succ \mid \boldsymbol{\sigma}, \Pi) \stackrel{\mathsf{def}}{=} \sum_{\boldsymbol{\tau} \in rnk(\mathrm{items}(\boldsymbol{\sigma}) \mid \succ)} \Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau}) \tag{3}$$

where $\Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau})$ is given in Equation (1). We refer to this probability as *the probability of* $\succ$ *in* $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$. In terms of computational complexity, computing this probability is a hard (#P-complete) problem already for the Mallows model (Lu and Boutilier 2014).

Exact Algorithm for Bounded Cover Width

We now describe RIMDP, a dynamic-programming algorithm for computing the probability of a poset in a

RIM model. Throughout the section we fix a RIM model $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ where $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$, and a partial order $\succ$. Our goal is to compute $\Pr(\succ | \boldsymbol{\sigma}, \Pi)$. We introduce the *cover width* parameter, which is a function of $\boldsymbol{\sigma}$ and $\succ$, and show that our algorithm terminates in time that is exponential in this parameter and polynomial in m.

We denote by $A_\succ$ items that participate in the partial order $\succ$, by A^i the set of items $\{\sigma_1, \ldots, \sigma_i\}$, and by $A^i_\succ$ their intersection (i.e., the items in A^i that occur in $\succ$). Let a and b be items in $A_\succ$. Recall that a is a cover of b, denoted $a \succ\!\!\cdot\, b$, if $a \succ b$ and there is no item in $A_\succ$ between them. We say that a *covers* b *at time* i, denoted $a \succ\!\!\cdot_i b$, if $a \succ b$ and there is no item in $c \in A^i_\succ$ such that $a \succ c \succ b$. We say that a and b are *directly related at time* i if no item in $A^i_\succ$ is between them, that is, $a = b$, or $a \succ\!\!\cdot_i b$, or $b \succ\!\!\cdot_i a$. We denote by D^i the subset of items in $A^i_\succ$ that, at time i, are directly related to at least one item σ_k in $A_\succ$, where $k > i$. Formally:

$$D^i = \{\sigma \in A^i_\succ \mid \exists k \in [i+1, m] \colon \sigma \succ\!\!\cdot_i \sigma_k \text{ or } \sigma_k \succ\!\!\cdot_i \sigma\}$$

We define $\mathsf{cw}(\boldsymbol{\sigma})$, the *cover width* of $\succ$ with respect to $\boldsymbol{\sigma}$, as the maximum cardinality of D^i over $i \in \{1, \ldots, m\}$.

Definition 1 (Cover Width)**.** *Let* $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ *be a* RIM *model over* m *items, and* $\succ$ *be a poset. The* cover width *of* $\succ$ *with respect to* $\boldsymbol{\sigma}$ *is* $\mathsf{cw}(\boldsymbol{\sigma}) = \max_{i \in \{1, \ldots m\}} \left|D^i\right|$ *where* $D^i \subseteq A^i_\succ$ *are the items that are directly related to at least one item* $\sigma_k \in A_\succ$ *where* $k > i$*.*

Example 1. *Suppose that* $\succ$ *is a linear order that forms a subsequence of* $\boldsymbol{\sigma}$ *(e.g.,* $\langle \sigma_3, \sigma_7, \sigma_{13} \rangle$*) or the reverse of* $\boldsymbol{\sigma}$ *(e.g.,* $\langle \sigma_{13}, \sigma_7, \sigma_3 \rangle$*). Then, each* D^i *consists of a single item. For illustration, on both* $\langle \sigma_3, \sigma_7, \sigma_{13} \rangle$ *and* $\langle \sigma_{13}, \sigma_7, \sigma_3 \rangle$*, we have* $D^i = \{\sigma_3\}$ *for* $i \in [3, 6]$*, and* $D^i = \{\sigma_7\}$ *for* $i \in [7, 12]$*. Hence,* $\mathsf{cw}(\boldsymbol{\sigma}) = 1$*.*

The *width* w of a poset is the cardinality of the largest set of incomparable items (*antichain*). The fastest currently known method for counting linear extensions of a general m-element poset is by dynamic programming, and runs in time $O(wm^w)$ (De Loof, De Meyer, and De Baets 2006). Given a poset $\succ$, it is easy to find cases in which its width is larger than its cover width with respect to $\boldsymbol{\sigma}$, and vice versa.

Example 2. *Consider the poset* $\succ \colon \{(\sigma_1 \succ \sigma_i) \mid i > 1\}$*. The cover width of* $\succ$ *is* 1 *because* $D^i = \{\sigma_1\}$ *for all* $i \in [1, m]$*, while its width* w *is* $m - 1$*. On the other hand, the cover width of the poset* $\boldsymbol{\tau} = \langle \sigma_2, \sigma_{10}, \sigma_4 \rangle$ *is* 2 *because* $D^i = \{\sigma_2, \sigma_{10}\}$ *for all* $i \in [4, 9]$*, while its width is* 1*.*

Let $\boldsymbol{\upsilon} \colon D^i \mapsto \{1, \ldots, i\}$ denote the placement of the items D^i inside a subranking consisting of items $A^i = \{\sigma_1, \ldots, \sigma_i\}$. The basic idea of the algorithm is as follows. We follow the RIM process and compute, at every step i, the probability of every consistent placement of the items in D^i. A placement $\boldsymbol{\upsilon}$ is *consistent* with $\succ$, denoted $\boldsymbol{\upsilon} \models \succ$, if there exists a ranking $\boldsymbol{\tau}$ over items A^i that is compatible with $\succ$ (i.e., $\boldsymbol{\tau} \in \mathit{rnk}(A^i \mid \succ)$), such that for every item $a \in D^i$ it is the case that $\boldsymbol{\tau}(a) = \boldsymbol{\upsilon}(a)$. We denote by $\mathbf{P}_i$ the set of all consistent mappings $\boldsymbol{\upsilon} : D^i \to \{1, \ldots, i\}$.

Example 3. *Let* $\succ$ *be the poset* $\langle \sigma_{21}, \sigma_5, \sigma_{10}, \sigma_6, \sigma_{20} \rangle$*. Let* $\boldsymbol{\upsilon}_1 \colon \{\sigma_5 \mapsto 12, \sigma_6 \mapsto 13\}$ *and* $\boldsymbol{\upsilon}_2 \colon \{\sigma_5 \mapsto 12, \sigma_6 \mapsto 14\}$ *be two placements of items* $\{\sigma_5, \sigma_6\} \in D^{15}$*. The mapping* $\boldsymbol{\upsilon}_1$ *is not consistent with* $\succ$*, hence* $\boldsymbol{\upsilon}_1 \notin \mathbf{P}_{15}$*, because in any ranking* $\boldsymbol{\tau} \in \mathit{rnk}(A^{15} \mid \succ)$ *the item* σ_{10} *is between* σ_5 *and* σ_6*. Therefore, items* σ_5*,* σ_6 *cannot be placed in consecutive positions. On the other hand,* $\boldsymbol{\upsilon}_2 \in \mathbf{P}_{15}$*.*

An algorithm for computing $\Pr(\succ | \boldsymbol{\sigma}, \Pi)$
Algorithm $\mathsf{RIMDP}(\mathsf{RIM}(\boldsymbol{\sigma}, \Pi))$

```
1: υ_1 ← ∅
2: if σ_1 ∈ D^1 then
3:     υ_1 ← {σ_1 ↦ 1}
4: P_1 ← {υ_1}
5: q[υ_1, 1] = 1
6: for i = 2, ..., m do
7:     for all υ ∈ P_{i−1} do
8:         for all j ∈ [1, i]: (υ_{+j} ∪ {σ_i → j}) ⊨≻ do
9:             υ' := (υ_{+j} ∪ {σ_i → j}) ∩ D^i
10:            q[υ', i] += q[υ, i − 1] × Π_σ(i, j)
11:            P_i := P_i ∪ {υ'}
12: return q[υ_∅, m]
```

Figure 1:

Let $i > 1$ and $\boldsymbol{\upsilon} \colon D^i \mapsto \{1, \ldots, i\}$ be a placement. We denote by $\boldsymbol{\upsilon}_{+j} \colon D^i \mapsto \{1, \ldots, i+1\}$ the placement obtained from $\boldsymbol{\upsilon}$ by inserting σ_{i+1} into index j; that is, for all $a \in D^i$ we have that $\boldsymbol{\upsilon}_{+j}(a) = \boldsymbol{\upsilon}(a) + 1$ whenever $\boldsymbol{\upsilon}(a) \geq j$, and otherwise $\boldsymbol{\upsilon}_{+j}(a) = \boldsymbol{\upsilon}(a)$. Likewise, we denote by $\boldsymbol{\upsilon}_{-j} \colon \left(D^i \setminus \{\sigma_i\}\right) \mapsto \{1, \ldots, i-1\}$ the mapping obtained from $\boldsymbol{\upsilon}$ by removing item σ_i from index j; that is, $\boldsymbol{\upsilon}_{-j}(a) = \boldsymbol{\upsilon}(a) - 1$ whenever $\boldsymbol{\upsilon}(a) > j$, and otherwise $\boldsymbol{\upsilon}_{-j}(a) = \boldsymbol{\upsilon}(a)$. Note that $\sigma_i \notin \boldsymbol{\upsilon}_{-j}$, hence $\boldsymbol{\upsilon}_{-j}(\sigma_i)$ is not defined even if $\sigma_i \in D^i$. In the example of Figure 2 we have that $\boldsymbol{\upsilon}^{2,4} = \{\sigma_4 \mapsto 3\}$, and $\boldsymbol{\upsilon}^{3,5} = \boldsymbol{\upsilon}^{2,4}_{+1} = \boldsymbol{\upsilon}^{2,4}_{+2} = \boldsymbol{\upsilon}^{2,4}_{+3} = \{\sigma_4 \mapsto 4\}$ is the placement that results from inserting item σ_5 into positions 1, 2, or 3. Likewise, $\boldsymbol{\upsilon}^{3,5}_{-1}$, $\boldsymbol{\upsilon}^{3,5}_{-2}$, and $\boldsymbol{\upsilon}^{3,5}_{-3}$ amount to removing item σ_5 from the first, second and third positions respectively from $\boldsymbol{\upsilon}^{3,5}$, resulting back in $\boldsymbol{\upsilon}^{2,4}$.

RIMDP works as follows. For all $i = 1, \ldots, m$ and $\boldsymbol{\upsilon} \in \mathbf{P}_i$, we compute the probability that RIM positions the items in D^i exactly as in $\boldsymbol{\upsilon}$. We denote this probability by $q[\boldsymbol{\upsilon}, i]$. We compute $q[\boldsymbol{\upsilon}, i]$ for all $\boldsymbol{\upsilon} \in \mathbf{P}_i$, by dynamic programming over the index i, as follows. For $i = 1$ the set $\mathbf{P}_1$ contains either the placement $\{\sigma_1 \mapsto 1\}$ (in case $\sigma_1 \in D^1$) or, if $D^1 = \emptyset$, the empty placement. In both cases, its probability is 1. For $i > 1$, every $\boldsymbol{\upsilon} \in \mathbf{P}_i$ is produced by extending a placement $\boldsymbol{\upsilon}' \in \mathbf{P}_{i-1}$ as follows. First, item σ_i is inserted into a position $j \in [1, i]$, such that the resulting placement $\boldsymbol{\upsilon}'_{+j} \cup \{\sigma_i \mapsto j\}$ is consistent with $\succ$. Then, this placement is projected onto the items of D^i, resulting in $\boldsymbol{\upsilon}$. That is, $\left(\boldsymbol{\upsilon}'_{+j} \cup \{\sigma_i \mapsto j\}\right) \cap D^i = \boldsymbol{\upsilon}$, where $\boldsymbol{\upsilon} \cap D^i$ denotes the

projection of $\boldsymbol{\upsilon}$ onto D^i. This gives us the following:

$$q[\boldsymbol{\upsilon}, i] = \sum_{j=1}^{i} \sum_{\substack{\boldsymbol{\upsilon}' \in \mathbf{P}_{i-1}: \\ (\boldsymbol{\upsilon}'_{+j} \cup \{\sigma_i \mapsto j\}) \models \succ, \\ (\boldsymbol{\upsilon}'_{+j} \cup \{\sigma_i \mapsto j\}) \cap D^i = \boldsymbol{\upsilon}}} q[\boldsymbol{\upsilon}', i-1] \times \Pi_{\boldsymbol{\sigma}}(i, j) \qquad (4)$$

The pseudocode of the algorithm is in Figure 1. For all $i \in \{2, \ldots, m\}$, the algorithm generates the set of consistent mappings $\mathbf{P}_i$, and for each $\boldsymbol{\upsilon} \in \mathbf{P}_i$, computes $q[\boldsymbol{\upsilon}, i]$, the probability that RIM positions the items in D^i exactly as in $\boldsymbol{\upsilon}$. Recall that D^i is the subset of items in $\{\sigma_1, \ldots, \sigma_i\}$ that are directly related to at least one item in the partial order with an index larger than i. Since there are no items in $\boldsymbol{\sigma}$ with an index larger than m, then $D^m = \emptyset$. Therefore, the result of the algorithm is simply $q[\boldsymbol{\upsilon}_\emptyset, m]$ where $\boldsymbol{\upsilon}_\emptyset$ is the (single) empty mapping.

Example 4. *Consider the partial order $\succ$ in Figure 2, and in particular, the mapping $\boldsymbol{\upsilon}^{2,4} = \{\sigma_4 \mapsto 3\}$ at iteration $i = 4$. In this case, $\sigma_4 \in D^4$. Therefore, σ_4 must be inserted into position 3 at iteration $i = 4$. Also, since $\boldsymbol{\upsilon}^{2,4} \in \mathbf{P}_4$ then the position of σ_2 in the previous iteration must have been less than 3 (i.e., corresponsing to mappings $\boldsymbol{\upsilon}^{1,3}$ or $\boldsymbol{\upsilon}^{2,3}$). Therefore, $q[\boldsymbol{\upsilon}^{2,4}] = \Pi_{\boldsymbol{\sigma}}(4, 3) \times \big(q[\boldsymbol{\upsilon}^{1,3}, 3] + q[\boldsymbol{\upsilon}^{2,3}, 3]\big)$.*

The correctness of the algorithm, proved by induction on the index i, is deferred to the appendix.

Theorem 1 (Correctness)**.** *For all $i = 1, \ldots, m$ the following hold.*

1. *The set of mappings produced at the ith iteration is precisely the set $\mathbf{P}_i$ of consistent mappings.*
2. *For every $\boldsymbol{\upsilon} \in \mathbf{P}_i$ we have:*

$$q[\boldsymbol{\upsilon}, i] = \sum_{\substack{\boldsymbol{\tau} \in rnk(A^i|\succ): \\ \forall a \in D^i, \boldsymbol{\tau}(a) = \boldsymbol{\upsilon}(a)}} \Pr(\boldsymbol{\tau} \mid \boldsymbol{\sigma}, \Pi)\,.$$

The number of distinct mappings $\boldsymbol{\upsilon}$ is at most $m^{\mathsf{cw}(\boldsymbol{\sigma})}$ giving us the following complexity result.

Theorem 2. $\Pr(\succ \mid \boldsymbol{\sigma}, \Pi)$ *can be computed in time $O(m^{\mathsf{cw}(\boldsymbol{\sigma})+2})$ where m is the size of $\boldsymbol{\sigma}$ and $\mathsf{cw}(\boldsymbol{\sigma})$ is the cover width of the poset $\succ$ w.r.t. $\boldsymbol{\sigma}$.*

In particular, if the size of the poset is bounded by a constant, then the algorithm completes in polynomial time.

Classes of Posets Over Mallows

The classes *series-parallel* posets (Möhring 1989) and *polytree* posets (Atkinson 1990) exhibit certain structural properties that allow to count their linear extensions in polynomial time. In this section, we investigate how these properties can be applied for computing the marginal probability over a Mallows model $\mathcal{M}(\boldsymbol{\sigma}, \phi)$. While it remains open whether the marginal probability can be computed in polynomial time for the general classes of series-parallel and polytree posets, we answer this question affirmatively for refinements with respect to the reference ranking $\boldsymbol{\sigma}$, termed *disjoint series-parallel*, and *monotonic polytree*. The classes of posets introduced here generalize the class of posets for

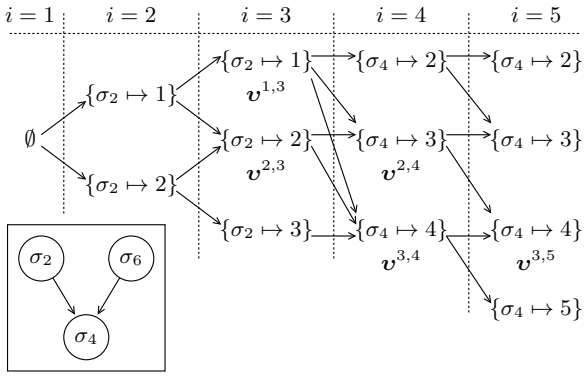


Figure 2: Examples of insertions considered in RIMDP

which Mallows is known to be computable efficiently. In particular, the class of disjoint series-parallel posets generalizes the class of *partitioned preferences* (Lebanon and Mao 2008). As in the case of the RIMDP algorithm, it is again a case where the ability to perform inference efficiently over Mallows depends on the relationship between the poset and the reference ranking.

Let $\boldsymbol{\tau}$ be a ranking over the items of $\boldsymbol{\sigma}$. We denote by $\mathrm{U}(\boldsymbol{\tau}|\boldsymbol{\sigma}, \phi)$ the *unnormalized weight* of $\boldsymbol{\tau}$ in $\mathcal{M}(\boldsymbol{\sigma}, \phi)$. By (2) we have $\mathrm{U}(\boldsymbol{\tau} \mid \boldsymbol{\sigma}, \phi) = Z \cdot \Pr(\boldsymbol{\tau} \mid \boldsymbol{\sigma}, \phi) = \phi^{d(\boldsymbol{\tau}, \boldsymbol{\sigma})}$ where Z is the Mallows normalization constant. From this point on we fix the Mallows parameters $\boldsymbol{\sigma}$ and ϕ and denote the unnormalized weight of a ranking $\boldsymbol{\tau}$ as $\mathrm{U}(\boldsymbol{\tau})$. Let $\succ$ be a poset. We denote by $\mathrm{U}(\succ)$ the unnormalized weight of $\succ$. Hence, we have $\mathrm{U}(\succ) = \sum_{\boldsymbol{\tau} \in rnk(A_{\boldsymbol{\sigma}}|\succ)} \phi^{d(\boldsymbol{\tau}, \boldsymbol{\sigma})}$.

The following sections look at classes of posets that are built using a combination operator over two posets. Let $\succ_Q$ and $\succ_P$ be two posets over disjoint sets $A_Q = \mathrm{items}(\succ_Q)$ and $A_P = \mathrm{items}(\succ_P)$, respectively. To streamline notation, we will refer to $\succ_Q$ as Q and $\succ_P$ as P.

Series-Parallel Posets

The *parallel* composition of P and Q, denoted $P||Q$, is defined over the items $A_P \cup A_Q$, with pairs of items that belong both to A_P or both to A_Q having the same relationship as they do in P and Q, respectively. In $P||Q$, a pair a, b is incomparable whenever $a \in P$ and $b \in Q$. The *series* composition of P and Q, written $P{*}Q$, is defined similarly but for every pair a, b where $a \in A_P$ and $b \in A_Q$, there is the additional relationship $a \succ b$.

We refine the notion of a parallel composition operator by relation to the reference ranking $\boldsymbol{\sigma}$. The parallel composition $P||Q$ is *disjoint with respect to* $\boldsymbol{\sigma}$ if for every pair of items $p \in A_P$ and $q \in A_Q$ it is the case that $p \succ_{\boldsymbol{\sigma}} q$. The class of *series parallel posets* is the set of posets that can be built up from single element posets using the series and parallel combination operators. The class of series-parallel posets that are *disjoint with respect to* $\boldsymbol{\sigma}$ is the set of posets that can be built using the series and *disjoint* parallel (w.r.t $\boldsymbol{\sigma}$) combination operators. A disjoint series-parallel poset is

presented in Figure 3a. The main result of this section is:

Theorem 3. *Let $\mathcal{M}(\boldsymbol{\sigma}, \phi)$ be a Mallows model and $\succ$ be a series-parallel poset that is disjoint with respect to $\boldsymbol{\sigma}$. Then $\Pr(\succ|\, \boldsymbol{\sigma}, \phi)$ can be computed in time $O(m^2)$.*

We note that disjoint series-parallel posets may have an unbounded cover width, and generally cannot be solved efficiently by the RIMDP algorithm of the previous section.

Example 5. *Consider the Mallows model $\mathcal{M}(\boldsymbol{\sigma}, \phi)$ where $\boldsymbol{\sigma}$ contains $2m$ items. Let $\succ$ be the disjoint series-parallel poset that orders the items $\{\sigma_1, \ldots, \sigma_m\}$ before $\{\sigma_{m+1}, \ldots, \sigma_{2m}\}$. By Def. 1, the cover width of $\succ$ is m.*

The class of disjoint series-parallel posets generalizes the class of partitioned preferences (Lebanon and Mao 2008). A partitioned-preference poset contains all items in $\boldsymbol{\sigma}$, and is built using only the series operator, and hence is trivially disjoint series-parallel. While ours is not as restrictive, the disjoint parallel constraint implies that every item σ that is not part of $\succ$ (i.e., $\sigma \notin \text{items}(\succ)$) is ranked either higher or lower, in $\boldsymbol{\sigma}$, than all items in $\succ$. For example, the poset of Figure 3a is defined over items $\text{items}(\succ) = \{\sigma_2, \ldots, \sigma_{10}\}$, while items $\{\sigma_1, \sigma_{11}, \ldots, \sigma_{30}\}$ are not part of the poset. One can verify that σ_1 is ranked higher, and the rest of the items (i.e., $\{\sigma_{11}, \ldots, \sigma_{30}\}$) are ranked lower, in $\boldsymbol{\sigma}$, than the items in the poset.

Lemma 1 shows how to compute the probability of a series combination operator.

Lemma 1. *Let P, Q be series-parallel posets such that $A_Q \cup A_P = A$. Then:*

$$\mathrm{U}(P{*}Q) = (\mathrm{U}(P) \times \mathrm{U}(Q)) \times \phi^{|\delta|} \tag{5}$$

where $\delta = \{(a, b) \mid a \in A_P, b \in A_Q, b \succ_{\boldsymbol{\sigma}} a\}$.

We now describe how to compute the probability of the disjoint parallel operator $P||Q$ where A_P and A_Q contain k and l items, respectively. Basically, this is the probability of generating a ranking $\boldsymbol{r}$ over items $A_P \cup A_Q$, that results from interleaving some pair of rankings $\boldsymbol{\tau}$ and $\boldsymbol{\zeta}$ over items A_P and A_Q, consistent with P and Q, respectively. Let $\boldsymbol{\tau} = \langle \tau_1, \ldots, \tau_k \rangle$, and $\boldsymbol{\zeta} = \langle \zeta_1, \ldots, \zeta_l \rangle$ denote such a pair of rankings. Since we assume disjointness with respect to $\boldsymbol{\sigma}$, it is the case that for every pair of items $\tau \in A_P$ and $\zeta \in A_Q$ we have $\tau \succ_{\boldsymbol{\sigma}} \zeta$.

We visualize the interleaving process as inserting the items of $\boldsymbol{\zeta}$, in order, into relative positions $0, \ldots, k$ of $\boldsymbol{\tau}$ while respecting the ordering constraints of $\boldsymbol{\zeta}$. Formally, inserting item ζ_i into position $j \in [0, k]$ places it between items τ_j and τ_{j+1} where τ_0 and τ_{k+1} are placeholders before and after items τ_1 and τ_k respectively.

We compute the unnormalized weight by dynamic programming as follows. For every $i \in [0, l]$, and $j \in [0, k]$, we denote by $s[i, j]$ the unnormalized weight of all rankings that result from interleaving the subranking $\langle \zeta_1, \ldots, \zeta_i \rangle$ with $\boldsymbol{\tau}$, where the item ζ_i is placed in the relative position $j \in [0, k]$. Denote by $\boldsymbol{\zeta}^i$ the subranking $\langle \zeta_1, \ldots, \zeta_i \rangle$. Then $s[i, j]$ is the unnormalized weight of all rankings consistent with $\boldsymbol{\zeta}^i||\boldsymbol{\tau}$ in which ζ_i is placed in relative position j. Formally,

$$s[i, j] = \sum_{\substack{\boldsymbol{r} \in rnk(\boldsymbol{\tau}||\boldsymbol{\zeta}^i):\\ \tau_j \succ_{\boldsymbol{r}} \zeta_i \succ_{\boldsymbol{r}} \tau_{j+1}}} \mathrm{U}(\boldsymbol{r}). \tag{6}$$

Recall that $\tau_j \succ_{\boldsymbol{r}} \zeta_i$ means that in $\boldsymbol{r}$, item τ_j is ranked higher than item ζ_i. Therefore, we have that:

$$\mathrm{U}(\boldsymbol{\tau}||\boldsymbol{\zeta}) = \sum_{\boldsymbol{r} \in rnk(\boldsymbol{\tau}||\boldsymbol{\zeta})} \mathrm{U}(\boldsymbol{r}) = \sum_{j=0}^{k} \sum_{\substack{\boldsymbol{r} \in rnk(\boldsymbol{\tau}||\boldsymbol{\zeta}):\\ \tau_j \succ_{\boldsymbol{r}} \zeta_l \succ_{\boldsymbol{r}} \tau_{j+1}}} \mathrm{U}(\boldsymbol{r}) = \sum_{j=0}^{k} s[l, j] \tag{7}$$

Consider the number of misorderings, with respect to $\boldsymbol{\sigma}$, that result from inserting $\zeta \in \text{items}(\boldsymbol{\zeta})$ into position j in $\boldsymbol{\tau}$. Since the parallel operator $P||Q$ is disjoint then $\tau \succ_{\boldsymbol{\sigma}} \zeta$ for every pair of items $\tau \in \boldsymbol{\tau}$ and $\zeta \in \boldsymbol{\zeta}$. Therefore, inserting any item $\zeta \in \text{items}(\boldsymbol{\zeta})$ into the relative position j in $\boldsymbol{\tau}$ (i.e., between items τ_j and τ_{j+1}), will entail precisely $k - j$ misorderings with respect to $\boldsymbol{\sigma}$. Finally, for any interleaving of the subranking $\boldsymbol{\zeta}^i$ with $\boldsymbol{\tau}$, in which ζ_i is in the relative position $j \in [0, k]$, it must be the case that ζ_{i-1} is in a relative position $t \leq j$. This gives us the following recursion:

$$s[i, j] = \begin{cases} 1 & i = 0, j = 0\,; \\ \phi^{k-j} \cdot \sum_{t=0}^{j} s[i-1, t] & i \geq 1, j \in [0, k]\,; \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

After computing $s[i, j]$ for all $i \in [0, l]$ and $j \in [0, k]$, we can compute the required probability by (7). We can compute (8) in constant time by precomuting the partial aggregations $\sum_{t=0}^{j} s[i-1, t]$. Hence, the total time for (7) is $O(m^2)$.

Polytree Posets

A poset whose cover graph is a polytree is called a polytree poset, see for example Figure 3b. A poset $\succ$ is *monotonic (resp. anti-monotonic) with respect to $\boldsymbol{\sigma}$* if for every relation $\sigma_u \succ \sigma_v$ we have that $u < v$ (resp. $u > v$).

Let $\succ$ be a monotonic polytree poset, and let $\sigma_u \succ \sigma_v$. Removing $\sigma_u \succ \sigma_v$ from $\succ$ breaks it into two monotonic posets corresponding to two polytrees: P, containing item σ_u, and Q, containing item σ_v, over disjoint sets of items A_P and A_Q respectively. By the monotonicity assumption, we have that $\tau \succ_{\boldsymbol{\sigma}} \zeta$ for every pair of items $\tau \in A_P$ and $\zeta \in A_Q$. Computing the required probability amounts to computing $\mathrm{U}(P||Q, \sigma_u \succ \sigma_v)$, the (unnormalized) probability of generating a ranking $\boldsymbol{r}$, from the Mallows distribution, such that $\boldsymbol{r} \in rnk(P||Q)$, and $\sigma_u \succ_{\boldsymbol{r}} \sigma_v$.

Every monotonic polytree poset with m items is comprised of at most $m - 1$ such constrained disjoint parallel operators. The algorithm for $\mathrm{U}(P||Q, \sigma_u \succ \sigma_v)$ is similar to the one for the disjoint parallel operator (see (8)), with the addition that we distinguish the position of item σ_u in order to maintain the constraint that $\sigma_u \succ \sigma_v$. In what follows we describe the algorithm for computing $\mathrm{U}(P||Q, \sigma_u \succ \sigma_v)$.

As before, let $\boldsymbol{\tau} = \langle \tau_1, \ldots, \tau_k \rangle$ and $\boldsymbol{\zeta} = \langle \zeta_1, \ldots, \zeta_l \rangle$ denote rankings over items A_P and A_Q that are consistent with the posets P and Q respectively. In this case, however, we distinguish the item σ_u in $\boldsymbol{\tau}$ by denoting its rank $j_u \in [1, k]$ in $\boldsymbol{\tau}$ (i.e., $\tau_{j_u} = \sigma_u$). For every triple $\langle i, j, k \rangle$ where $i \in [0, l]$, $j \in [0, k]$, and $j_u \in [1, k]$, we denote by $s[i, j, j_u]$ the

weight of all rankings that result from interleaving the subranking $\boldsymbol{\zeta}^i = \langle \zeta_1, \ldots, \zeta_i \rangle$ with $\boldsymbol{\tau} = \langle \tau_1, \ldots, \tau_k \rangle$ and where: (1) $\tau_{j_u} = \sigma_u$ (2) the item ζ_i is placed in the relative position j, and (3) item σ_v is positioned *after* item σ_u. That is, its relative position in $\boldsymbol{\tau}$ is at least j_u. Formally:

$$s[i, j, j_u] = \begin{cases} \sum_{\substack{\boldsymbol{r} \in rnk(\boldsymbol{\zeta}^i || \boldsymbol{\tau}): \\ \tau_j \succ_{\boldsymbol{r}} \zeta_i \succ_{\boldsymbol{r}} \tau_{j+1}, \\ \sigma_u \succ_{\boldsymbol{r}} \sigma_v}} \mathrm{U}(\boldsymbol{r}) & \tau_{j_u} = \sigma_u\,; \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

We then have that:

$$\mathrm{U}(\boldsymbol{\tau} || \boldsymbol{\zeta}, \sigma_u \succ \sigma_v) = \sum_{j=0}^{k} \sum_{q=1}^{j} s[l, j, q] \tag{10}$$

Since ζ_l is placed in relative position j, then item σ_v is positioned at a relative index (w.r.t $\boldsymbol{\tau}$) that is less than or equal to j. Since $\sigma_u \succ \sigma_v$ then the position q of σ_u in $\boldsymbol{\tau}$ is at most j (i.e., $q \leq j$).

For any interleaving of the subranking $\boldsymbol{\zeta}^i = \langle \zeta_1, \ldots, \zeta_i \rangle$ with $\boldsymbol{\tau}$, in which ζ_i is in the relative position $j \in [0, k]$, it must be the case that ζ_{i-1} is in a relative position $t \leq j$. Also, if $\zeta_i = \sigma_v$ then, in addition, it must also be the case that $j_u \leq j$. Finally, since we assume monotonicity, then as in the case of disjoint parallel combination operator, the number of misorderings that result from inserting item $\zeta \in A_Q$ into relative position j in $\boldsymbol{\tau}$ is precisely $k - j$. This gives us the following recursion:

$$s[i, j, j_u] = \begin{cases} 1 & i = 0, j = 0 \\ \phi^{(k-j)} \sum_{\substack{t \in [0,j], \\ q \in [1,k]}} s[i-1, t, q] & \zeta_i \neq \sigma_v, j \in [0,k], \\ & i \in [1,l] \\ \phi^{(k-j)} \sum_{\substack{t \in [0,j], \\ q \in [1,j]}} s[i-1, t, q] & \zeta_i = \sigma_v, j \in [0,k], \\ & i \in [1,l] \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

In the case where item $\zeta_i \neq \sigma_v$ then it can be inserted into any relative position j that is larger than or equal to t, the relative position of ζ_{i-1}, regardless of q, the position of σ_u in $\boldsymbol{\tau}$. Otherwise, if $\zeta_i = \sigma_v$, there is the added constraint that the position q of item σ_u in $\boldsymbol{\tau}$ must be lower than the relative position j of ζ_i in $\boldsymbol{\tau}$ (i.e., $q \leq j$).

After computing $s[i, j, j_u]$ for every triple of indexes $i \in [0, l]$, $j \in [0, k]$, and $j_u \in [1, k]$, we can compute the required probability by (10). The total time required for (10) is $O(m^3)$ because we essentially repeat the algorithm of (8) for every rank $1 \leq j_u \leq k$.

Experimental Evaluation

We now describe our implementation of the RIMDP algorithm and present experimental results. We show that the running time of RIMDP grows with the size of the partial order, and its cover width. Further, we present a parallel implementation of the algorithm and show that CPU parallelism can be leveraged to improve scalability.

We implemented RIMDP in Java 8, and executed experiments on an Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, 512GB of RAM, running 64-bit Ubuntu Linux.

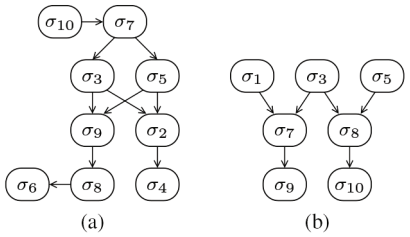


Figure 3: Poset examples: (a) disjoint series-parallel; (b) monotonic polytree

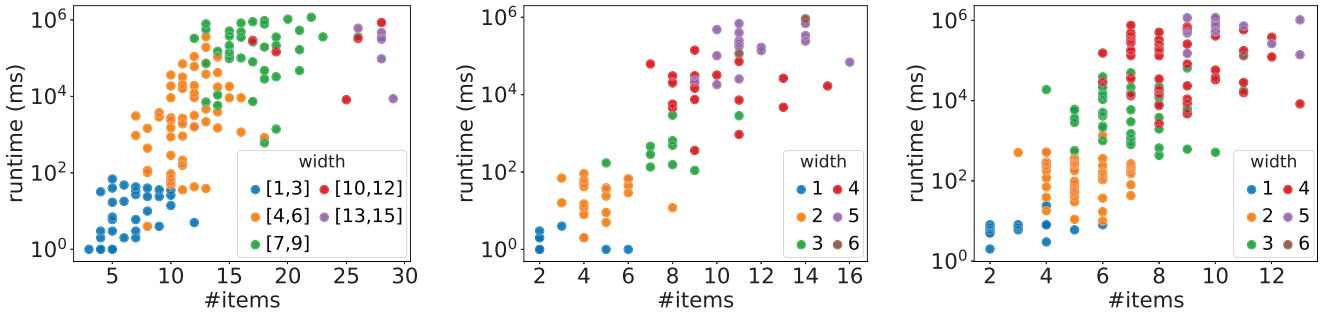
Implementation details. Recall that $q[\boldsymbol{\upsilon}, i]$ is the probability that, at time i, the RIM-generated ranking is consistent with $\boldsymbol{\upsilon}$ — the placement of items into their positions. The algorithm maintains a hash map, associating placements like $\boldsymbol{\upsilon}$ with their probabilities. If a placement is produced by multiple branches, the probabilities are summed up.

The parallel implementation of RIMDP takes the number of threads as input, and parallelizes the computation of $q[\boldsymbol{\upsilon}, i]$ for each consistent placement $\boldsymbol{\upsilon}$ at time i. To leverage parallelism, we take care to avoid synchronization between threads, which can occur at two points: when input placements are assigned to threads, and when results of parallel computation are reconciled. To avoid synchronization at placement assignment, we create separate queues for each thread. (The alternative of a common synchronized queue showed inferior results.) To avoid synchronization when reconciling the results, we use the ConcurrentHashMap data structure to efficiently perform atomic operations, and store the probability of each placement as a mutable DoubleAdder, rather than as an immutable Double.

Experimental datasets. We evaluated the running time of RIMDP on randomly generated posets that exhibit variability in both size and cover width. We experimented with three reference rankings $\boldsymbol{\sigma}$, with 30, 60 and 100 items (denoted m), and with $\phi < 1$. (In our experiments, we found that the value of ϕ has no impact on the running time of RIMDP.)

We generated a poset workload separately for each $\boldsymbol{\sigma}$ as follows. Let p_V and p_E be the probabilities that an item (resp. a relation) is added to the poset $\succ$. First, generate a set of items $P \subseteq \mathrm{items}(\boldsymbol{\sigma})$, including each item in P with probability p_V. Next, generate a random permutation of the items in P, and denote the resulting subranking by $\boldsymbol{\tau}$. Finally, add each relation $\sigma_i \succ_{\boldsymbol{\tau}} \sigma_j$ to $\succ$ with probability p_E. We generated 250 posets for each value of m. For $m = 30$ items, we set $p_V \in [0.3, 0.9]$ and $p_E \in [0.05, 0.3]$; for $m = 60$, $p_V \in [0.1, 0.5]$ and $p_E \in [0.1, 0.3]$; and for $m = 100$, $p_V \in [0.06, 0.09]$ and $p_E \in [0.1, 0.8]$. These particular settings are not prescriptive, they simply allow us to explore algorithm behavior for inputs with different characteristics.

Results and discussion. Figure 4 presents the running time of RIMDP as a function of the number of items, for three different values of m (length of $\boldsymbol{\sigma}$). These plots also present the dependency between the running time and the cover width of the poset. Results in Figure 4 are for posets



(a) m=30 (b) m=60 (c) m=100

Figure 4: RIMDP running time vs. preference set size and width; y-axis is in log-scale.

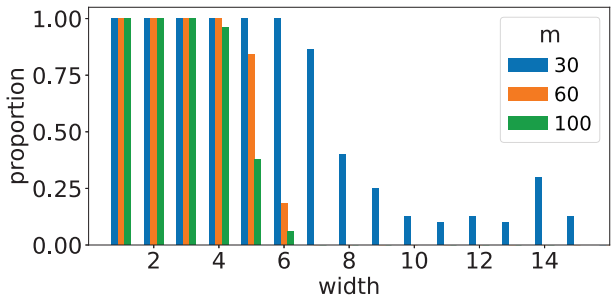


Figure 5: Proportion of posets computed in 20 min.

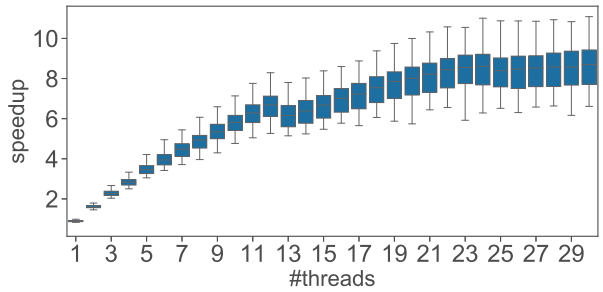


Figure 6: Speedup of parallel RIMDP for $m = 100$.

for which RIMDP terminated within 20 minutes. Figure 5 presents the proportion of the posets for each value of m that completed within 20 minutes, as a function of cover width.

In line with the complexity of $O(m^{\mathsf{cw}(\boldsymbol{\sigma})})$, we see that the execution time is generally dominated by m, the size of the reference ranking, and the cover width $\mathsf{cw}(\boldsymbol{\sigma})$ of the poset. Fixing m and the poset size (i.e., #items), we see that the instances with the largest execution time also have the largest width. From Figure 4, we see that RIMDP scales much better on lower values of m. Specifically, for $m = 30$ (Figure 4a) it is able to handle posets with up to 28 items, and width up to 15, in 20 minutes. These numbers reduce to 16 and 6 when $m = 60$ (Figure **??**), and to 13 and 6 when $m = 100$ (Figure **??**).

Figure 6 presents the speed-up achieved by the parallel implementation of RIMDP as a function of the number of threads, computed by dividing the running time of the sequential algorithm by the running time of the parallel version. We observe nearly-linear speed-up for up to 12 threads. Speed-up drops slightly at 13 and 25 threads because of communication between chips: There are 12 cores per chip on our machine, all computation is performed on a single chip for up to 12 threads, on two chips for between 13 and 24 threads, and on three chips for 25 to 30 threads.

Conclusions and Future Work

In this paper we explored the problem of computing the marginal probability of a poset over a RIM distribution. We presented the RIMDP algorithm and showed that its complexity is governed by the cover width parameter that depends on both the poset and the reference ranking. We empirically evaluated RIMDP and showed that it lends itself to effective parallelization. We also embarked on an exploration of the relationship between inference over rank distributions and counting of linear extensions, by studying two simple classes of posets whose linear extensions can be counted efficiently.

While RIM has many desirable properties stemming from its simplicity, it suffers from limited flexibility. Improving upon this deficit, Meek and Meila (2014) introduce the *recursive inversion models*—a generative model that enhances the insertion procedure of RIM with a *merge* operator over rankings. A similar line of work is the hierarchical *riffled independence models* of (Huang, Kapoor, and Guestrin 2012). A ranking distribution is said to be *riffle independent* if it can be generatively modeled as producing partial rankings on disjoint item sets, and then interleaving them. Including Mallows as a special case, probabilistic inference over these models is intractable as well. As part of future work we intend to investigate how the techniques introduced in this paper can be applied to more sophisticated models as above.

We also plan to investigate the *parameterized complexity* (Flum and Grohe 2006) of inference over RIM, with the cover width playing the role of the parameter. In particular, we will explore whether inference is *Fixed-Parameter Tractable* (FPT), meaning that the running time is polynomial (with a fixed degree), except that the *coefficient* of the polynomial can have a super-polynomial dependence on the

cover width. In reference, counting linear extensions is *not* FPT (under conventional assumptions) when parameterized by the treewidth of the cover graph (Eiben et al. 2016).

Acknowledgments

The authors thank Rina Dechter for insightful discussions and comments. This work was supported in part by the US National Science Foundation (NSF) Grants No. 1464327 and 1539856, by the US-Israel Binational Science Foundation (BSF) Grant No. 2014391, and by the Israel Science Foundation (ISF) Grant No. 1295/15. Benny Kimelfeld is a Taub Fellow, supported by the Taub Foundation.

References

Atkinson, M. D. 1990. On computing the number of linear extensions of a tree. *Order* 7(1):23–25.

Awasthi, P.; Blum, A.; Sheffet, O.; and Vijayaraghavan, A. 2014. Learning mixtures of ranking models. *CoRR* abs/1410.8750.

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

Busse, L. M.; Orbanz, P.; and Buhmann, J. M. 2007. Cluster analysis of heterogeneous rank data. In *ICML*, 113–120. ACM.

Das Sarma, A.; Das Sarma, A.; Gollapudi, S.; and Panigrahy, R. 2010. Ranking mechanisms in twitter-like forums. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, 21–30. New York, NY, USA: ACM.

De Loof, K.; De Meyer, H.; and De Baets, B. 2006. Exploiting the lattice of ideals representation of a poset. *Fundam. Inf.* 71(2,3):309–321.

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.

Eiben, E.; Ganian, R.; Kanga, K.; and Ordyniak, S. 2016. Counting Linear Extensions: Parameterizations by Treewidth. In Sankowski, P., and Zaroliagis, C., eds., *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 39:1–39:18. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

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.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Gormley, I. C., and Murphy, T. B. 2008. A mixture of experts model for rank data with applications in election studies. *Ann. Appl. Stat.* 2(4):1452–1477.

Huang, J.; Kapoor, A.; and Guestrin, C. 2012. Riffled independence for efficient inference with partial rankings. *J. Artif. Intell. Res. (JAIR)* 44:491–532.

Kangas, K.; Hankala, T.; Niinimäki, T.; and Koivisto, M. 2016. Counting linear extensions of sparse posets. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 603–609. AAAI Press.

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

Lebanon, G., and Mao, Y. 2008. Non-Parametric Modeling of Partially Ranked Data. *Journal of Machine Learning Research* 9:2401–2429.

Lu, T., and Boutilier, C. 2014. Effective sampling and learning for mallows models with pairwise-preference data. *Journal of Machine Learning Research* 15(1):3783–3829.

Luce, R. D. 1959. *Individual Choice Behavior: A theoretical analysis*. Wiley.

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

Marden, J. I. 1995. *Analyzing and Modeling Rank Data*. Chapman & Hall.

McElroy, G., and Marsh, M. 2009. Candidate gender and voter choice: Analysis from a multimember preferential voting system. *Political Research Quarterly*.

Meek, C., and Meila, M. 2014. Recursive inversion models for permutations. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 631–639.

Möhring, R. H. 1989. *Computationally Tractable Classes of Ordered Sets*. Dordrecht: Springer Netherlands. 105–193.

Plackett, R. L. 1975. The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 24(2):193–202.