Supporting Hard Queries over Probabilistic Preferences

Haoyue Ping
New York University, USA
hp1326@nyu.edu

Julia Stoyanovich
New York Univerisity, USA
stoyanovich@nyu.edu

Benny Kimelfeld
Technion, Israel
bennyk@cs.technion.ac.il

ABSTRACT

Preference analysis is widely applied in various domains such as social choice and e-commerce. A recently proposed framework augments the relational database with a preference relation that represents uncertain preferences in the form of statistical ranking models, and provides methods to evaluate Conjunctive Queries (CQs) that express preferences among item attributes. In this paper, we explore the evaluation of queries that are more general and harder to compute.

The main focus of this paper is on a class of CQs that cannot be evaluated by previous work. These queries are provably hard since relate variables that represent items being compared. To overcome this hardness, we instantiate these variables with their domain values, rewrite hard CQs as unions of such instantiated queries, and develop several exact and approximate solvers to evaluate these unions of queries. We demonstrate that exact solvers that target specific common kinds of queries are far more efficient than general solvers. Further, we demonstrate that sophisticated approximate solvers making use of importance sampling can be orders of magnitude more efficient than exact solvers, while showing good accuracy. In addition to supporting provably hard CQs, we also present methods to evaluate an important family of count queries, and of top-k queries.

PVLDB Reference Format:
Haoyue Ping, Julia Stoyanovich, Benny Kimelfeld. Supporting Hard Queries over Probabilistic Preferences. *PVLDB*, 13(7): 1134 -1146, 2020.
DOI: https://doi.org/10.14778/3384345.3384359

1. INTRODUCTION

Preferences are statements about the relative quality or desirability of items. Preference analysis aims to derive insight from a collection of preferences. For example, in recommender systems [2, 26, 28] and in political elections [6, 10, 11, 23], we may be interested in identifying the most preferred items or sets of items, or in understanding the points of consensus or disagreement among a group of voters.

Proceedings of the VLDB Endowment, Vol. 13, No. 7
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3384345.3384359

Voter preferences are often inferred from indirect input (such as clicks on ads), or from preferences of other similar voters based on demographic similarity or on similarity over stated preferences, as in collaborative filtering, and are thus uncertain. A variety of statistical models have been developed to represent uncertain preferences [22], including the popular Mallows model [21]. There is much recent work in the machine learning and statistics communities [1, 4, 7, 10, 12, 15, 18, 19, 20], focusing specifically on learning the parameters of Mallows models or their mixtures [7, 8, 16, 20, 27]. Learning techniques for Mallows often rely on the Repeated Insertion Model (RIM) [8] — a generative model that gives rise to various distributions over rankings.

In a recent work [17], we introduced a framework for representing and querying uncertain preferences in a *Probabilistic Preference Database*, or *PPD* for short. We recall this framework here, illustrating it with an example. Consider Figure 1 that presents an instance of a polling database for the 2016 US presidential election. Each of **Candidates** and **Voters** is an *ordinary relation* (abbr. o-relation), while **Polls** is a *preference relation* (abbr. p-relation) where each tuple is associated with a preference model—Mallows in this example. Mallows models are ranking distributions parameterized by a center ranking $\boldsymbol{\sigma}$ and a dispersion parameter ϕ. We will discuss the Mallows model in Section 2.2, explaining that it is a special case of RIM [8]. The PPD formalism of [17], on which we build here, accommodates RIM preferences, and we refer to such a database as a *RIM-PPD*.

In summary, a RIM-PPD represents uncertain preferences by statistical models. Semantically, a RIM-PPD instance is a *probabilistic database* [29], where every random possible world (a deterministic database) is obtained by sampling from the stored RIM models. RIM-PPDs adopt the conventional semantics of query evaluation over probabilistic databases, associating each answer with a *confidence value*—the probability of getting this answer in a random possible world [29]. Hence, query evaluation entails *probabilistic inference*: computing the marginal probability of query answers. In the case of RIM-PPDs, query evaluation entails inference over statistical *ranking models*.

A preference relation in a possible world represents a collection of orders, each called a *session*. A tuple of a preference relation has the form $(\mathbf{s}; a; b)$, stating that in the order of session $\mathbf{s}$ item a is preferred to item b, denoted $a \succ_{\mathbf{s}} b$.

For example, the tuple $(\texttt{Ann}, \texttt{5/5}; \texttt{Sanders}; \texttt{Clinton})$ in an instance of the **Polls** relation denotes that in a poll conducted on May 5^{th}, Ann preferred Sanders to Clinton. Here, $(\texttt{Ann}, \texttt{5/5})$ identifies a session. Note that the internal rep-

Candidates (o)

candidate	party	sex	age	edu	reg
Trump	R	M	70	BS	NE
Clinton	D	F	69	JD	NE
Sanders	D	M	75	BS	NE
Rubio	R	M	45	JD	S

Voters (o)

voter	sex	age	edu
Ann	F	20	BS
Bob	M	30	BS
Dave	M	50	MS

Polls (p)

voter	date	*Preference model* $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$
Ann	5/5	$\langle$Clinton, Sanders, Rubio, Trump$\rangle$, 0.3
Bob	5/5	$\langle$Trump, Rubio, Sanders, Clinton$\rangle$, 0.3
Dave	6/5	$\langle$Clinton, Sanders, Rubio, Trump$\rangle$, 0.5

Figure 1: An instance of RIM-PPD.

resentation of a preference needs not store every pairwise comparison explicitly.

Incorporating preferences into databases facilitates preference analysis. For example, an analyst may ask whether Ann prefers Trump to both Clinton and Rubio on May 5^{th} as follows, using P to denote **Polls**:

$$Q_0() \leftarrow P(\texttt{Ann}, \texttt{5/5}; \texttt{Trump}; \texttt{Clinton}), \\ P(\texttt{Ann}, \texttt{5/5}; \texttt{Trump}; \texttt{Rubio})$$

Q_0 is a Boolean conjunctive query (CQ) that computes the marginal probability of $\{\texttt{Trump} \succ \texttt{Clinton}, \texttt{Trump} \succ \texttt{Rubio}\}$ over the Mallows model of $(\texttt{Ann}, \texttt{5/5})$.

The analyst may query preferences about the attributes of candidates, which generalizes the preferences over specific candidates. For example, using C to denote **Candidates**:

$$Q_1() \leftarrow P(\_, \_; c_1; c_2), C(c_1, \_, \texttt{F}, \_, \_, \_), C(c_2, \_, \texttt{M}, \_, \_, \_)$$

The evaluation of Q_1 computes the marginal probability that a female candidate is preferred to a male candidate over the random preferences of the users, drawn from their corresponding preference models. We refer to the values of item attributes, such as F and M, as *labels*. Q_1 is an example of an *itemwise* CQ [17], querying preferences over labels. Intuitively, itemwise CQs state a preference among constants and variables (e.g., $c_1 \succ c_2$, or $c_1 \succ \texttt{Trump}$) in addition to an independent condition on item variables (e.g., c_1 is a female candidate and c_2 is a male candidate), and this preference can be represented as a partial order of labels, named *label patterns* (e.g., $\texttt{F} \succ \texttt{M}$). Kenig et al. [17] show that, at least for the fragment of queries without self-joins, itemwise CQs are *precisely* the queries that can be evaluated in polynomial time. In a follow-up work, Cohen et al. [5] proposed a query engine that uses inference to evaluate these queries that have tractable complexity.

Problem statement. In this paper, we focus on extending RIM-PPD query evaluation to support general CQs, those that are *provably hard*. Given a non-itemwise CQ Q and an instance D of RIM-PPD, the goal is to calculate the probability that Q holds in a random possible world. This query evaluation problem is reduced to an inference problem over RIM. We investigate two types of queries beyond CQs, and also reduce their evaluation to inference over RIM. This problem statement will be refined in Section 3.3.

To get the gist of our approach, consider the query:

$$Q_2() \leftarrow P(\_, \_; c_1; c_2), C(c_1, \texttt{D}, \_, \_, e, \_), C(c_2, \texttt{R}, \_, \_, e, \_);$$

Q_2 asks for the marginal probability that a Democrat c_1 is preferred to a Republican c_2 having the same education degree e. As e is a variable, the qualified candidates for c_1 and c_2 cannot be determined ahead of time. According to the instance of **Candidates** in Figure 1, e takes on values BS and JD. Substituting e with these values in $Q_2()$, we get:

$$Q_2^{\texttt{BS}}() \leftarrow P(\_, \_; c_1; c_2), C(c_1, \texttt{D}, \_, \_, \texttt{BS}, \_), C(c_2, \texttt{R}, \_, \_, \texttt{BS}, \_);$$
$$Q_2^{\texttt{JD}}() \leftarrow P(\_, \_; c_1; c_2), C(c_1, \texttt{D}, \_, \_, \texttt{JD}, \_), C(c_2, \texttt{R}, \_, \_, \texttt{JD}, \_);$$

Note that $Q_2^{\texttt{BS}}$ and $Q_2^{\texttt{JD}}$ are both itemwise CQs, and so their evaluation is tractable. Further, according to the semantics of CQ evaluation, Q_2 holds if either $Q_2^{\texttt{BS}}$ holds or $Q_2^{\texttt{JD}}$ holds (i.e., $Q_2 = Q_2^{\texttt{BS}} \cup Q_2^{\texttt{JD}}$). Note that it is possible for a ranking to satisfy both $Q_2^{\texttt{BS}}$ and $Q_2^{\texttt{JD}}$; $\langle \texttt{Sanders}, \texttt{Trump}, \texttt{Cliton}, \texttt{Rubio} \rangle$ is an example. Therefore, $Q_2^{\texttt{BS}}$ and $Q_2^{\texttt{JD}}$ are not mutually exclusive and $\Pr(Q_2) < \Pr(Q_2^{\texttt{BS}}) + \Pr(Q_2^{\texttt{JD}})$ may hold.

More generally, a non-itemwise CQ can be decomposed into a union of itemwise CQs, but the probability of a query union is not the sum of probabilities of its individual CQs. The size of the union depends on the domain size of the instantiated variables. We propose three exact solvers for the inference problem induced by this decomposition. The first is based on the inclusion-exclusion principle, and works for a union of any label patterns. This solver, while general, does not scale well when the product of the domain sizes of the variables is large, and we use it as a performance baseline. We propose two additional exact solvers, optimized for families of label patterns that are commonly used in practice: two-label patterns and bipartite patterns that are similar to bipartite graphs.

Further, we propose approximate solvers based on Multiple Importance Sampling (MIS). We develop several flavors of approximate solvers, compare their performance, and show that they can outperform exact solvers by several orders of magnitude, while achieving good accuracy.

Finally, we expand the family of supported queries to involve *Count-Session*, returning the number of sessions satisfying a given query Q, and *Most-Probable-Session*, returning k sessions that support Q with the highest probability.

Contributions. We make the following contributions:

1. We reduce the evaluation of conjunctive queries over probabilistic preference databases to an inference problem over a union of label patterns (Section 3);

2. We develop exact solvers for CQs, Count-Session and Most-Probable-Session queries (Section 4);

3. We propose approximate solvers, based on Multiple Importance Sampling, that improve scalability, while achieving good accuracy (Section 5); and

4. We present results of an extensive experimental evaluation over real and synthetic datasets, demonstrating that (i) customized exact solvers see substantial improvement; (ii) approximate solvers are effective and scalable; (iii) evaluation is well optimized for Most-Probable-Session queries; and (iv) the implementation can handle a large number of sessions (Section 6).

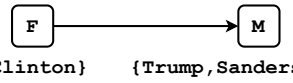


Figure 2: A label pattern over the polling database. Candidates with labels F and M are annotated below, respectively.

2. PRELIMINARIES

2.1 Preferences and Label Patterns

Let A denote a set of m items. Preference is a binary relation over A. Let $a \succ b$ denote that $a \in A$ is preferred to $b \in A$. If the preference is from a judge u, we denote it by $a \succ_u b$. The preference relation $\succ$ is irreflexive, transitive, and asymmetric.

A *preference pair* compares two items. *Pairwise preferences* are a collection of preference pairs, such as $\{a \succ b, a \succ c\}$. They can be visualized by a directed graph with items as vertices and preference pairs as edges. If the directed graph is acyclic, it represents a *partial order*. Since the relation $\succ$ is transitive, a partial order $\boldsymbol{\upsilon}$ expresses the same information as its transitive closure $tc(\boldsymbol{\upsilon})$.

A *linear order* or *ranking* or *permutation* is a partial order where every two items in A are comparable. Let $\boldsymbol{\tau} = \langle \tau_1, \ldots, \tau_m \rangle$ denote a ranking placing item τ_i at rank i. We denote by $\boldsymbol{\tau}(i)$ the item at rank i, by $\boldsymbol{\tau}^{-1}(\tau)$ the rank of item τ. We denote by $rnk(A)$ the set of all $m!$ permutations over the items in A.

A ranking $\boldsymbol{\tau}$ is a *linear extension* of a partial order $\boldsymbol{\upsilon}$ if $\boldsymbol{\tau}$ is consistent with $\boldsymbol{\upsilon}$ (i.e., $\forall (x \succ y) \in \boldsymbol{\upsilon}, x \succ_{\boldsymbol{\tau}} y$). We use $\Omega(\boldsymbol{\upsilon})$ to denote the set of linear extensions of $\boldsymbol{\upsilon}$.

A *sub-ranking* $\boldsymbol{\psi}$ is a ranking over a subset of the items in A, denoted by $A(\boldsymbol{\psi})$. A sub-ranking can also be consistent with a partial order $\boldsymbol{\upsilon}$. Let $\Delta(\boldsymbol{\upsilon})$ denote the set of sub-rankings that are consistent with $\boldsymbol{\upsilon}$.

Labels are values of item attributes. For example, in Figure 1, $\{\texttt{M}, \texttt{R}\}$ is the set of labels of the items `Trump` and `Rubio`. A *label pattern* (or just *pattern*) is a partial order of labels. A pattern can be represented by a directed acyclic graph g. Figure 2 presents a pattern $g_0 = \{\texttt{F} \succ \texttt{M}\}$ related to the RIM-PPD in Figure 1.

2.2 Repeated Insertion Model

The Repeated Insertion Model (RIM) is a generative ranking model that defines a probability distribution over permutations [8]. This distribution, denoted by $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$, is parameterized by a reference ranking $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$ and a function Π, where $\Pi(i, j)$ is the probability of inserting σ_i at position j. Algorithm 1 presents the RIM sampling procedure. It starts with an empty ranking, inserts items in the order of $\boldsymbol{\sigma}$, and puts item σ_i at j-th position of the current incomplete ranking with probability $\Pi(i, j)$.

Algorithm 1 RIM

Require: $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$, with $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$
1: Initialize an empty ranking $\boldsymbol{\tau} = \langle\rangle$.
2: **for** $i = 1, \ldots, m$ **do**
3: Insert σ_i into $\boldsymbol{\tau}$ at $j \in [1, i]$ with probability $\Pi(i, j)$.
4: **return** $\boldsymbol{\tau}$

EXAMPLE 2.1. $\mathsf{RIM}(\langle a, b, c \rangle, \Pi)$ *generates* $\boldsymbol{\tau}' = \langle b, c, a \rangle$ *as follows. Initialize an empty ranking* $\boldsymbol{\tau}_0 = \langle\rangle$. *At step 1,* $\boldsymbol{\tau}_1 = \langle a \rangle$ *by inserting* a *into* $\boldsymbol{\tau}_0$ *with probability* $\Pi(1, 1) = 1$. *At step 2,* $\boldsymbol{\tau}_2 = \langle b, a \rangle$ *by inserting* b *into* $\boldsymbol{\tau}_1$ *at position 1 with probability* $\Pi(2, 1)$. *Note that* b *is put before* a *since* $b \succ_{\boldsymbol{\tau}'} a$. *At step 3,* $\boldsymbol{\tau}' = \langle b, c, a \rangle$ *by inserting* c *into* $\boldsymbol{\tau}_2$ *at position 2 with probability* $\Pi(3, 2)$. *The overall probability of sampling* $\boldsymbol{\tau}'$ *is* $\Pr(\boldsymbol{\tau}' \mid \langle a, b, c \rangle, \Pi) = \Pi(1, 1) \cdot \Pi(2, 1) \cdot \Pi(3, 2)$.

The Mallows model [21], $\mathsf{MAL}(\boldsymbol{\sigma}, \phi), \phi \in [0, 1]$, is a special case of RIM. As a popular preference model, it defines a distribution of rankings that is analogous to the Gaussian distribution. Ranking $\boldsymbol{\sigma}$ is at the center. Rankings closer to $\boldsymbol{\sigma}$ have higher probabilities. For a ranking $\boldsymbol{\tau}$, its probability $\Pr(\boldsymbol{\tau} | \boldsymbol{\sigma}, \phi) \propto \phi^{dist(\boldsymbol{\sigma}, \boldsymbol{\tau})}$ where $dist(\boldsymbol{\sigma}, \boldsymbol{\tau})$ is the Kendall-tau distance between $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$: $dist(\boldsymbol{\sigma}, \boldsymbol{\tau}) = |(a, a') | a \succ_{\boldsymbol{\sigma}} a', a' \succ_{\boldsymbol{\tau}} a|$ that is the number of disagreeing preference pairs. When $\phi = 0$, only $\boldsymbol{\sigma}$ has positive probability; when $\phi = 1$, all rankings have the same probability, that is, $\mathsf{MAL}(\boldsymbol{\sigma}, 1)$ is the uniform distribution over rankings. RIM was proposed in [8] and provides an efficient and practical approach to draw rankings from the Mallows model. This is because, as was shown in [8], $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ is precisely $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$ when $\Pi(i, j) = \frac{\phi^{i-j}}{1 + \phi + \ldots + \phi^{i-1}}$.

The Approximate Mallows Posterior [20] $\mathsf{AMP}(\boldsymbol{\sigma}, \phi, \boldsymbol{\upsilon})$, is a *sampler from the posterior distribution* of $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$ conditioned on a partial order $\boldsymbol{\upsilon}$. When sampling a ranking, it follows the procedure of RIM, but the positions to insert items are constrained by $\boldsymbol{\upsilon}$. Assume that $\boldsymbol{\tau}_i$ is the current incomplete ranking when inserting σ_i. Let J denote the range of positions where inserting σ_i does not violate $\boldsymbol{\upsilon}$. Item σ_i is inserted at $j \in J$ with probability $p_j \propto \phi^{i-j}$.

EXAMPLE 2.2. $\mathsf{AMP}(\langle a, b, c \rangle, \phi, \{c \succ a\})$ *generates ranking* $\boldsymbol{\tau}' = \langle b, c, a \rangle$ *as follows. Initialize an empty ranking* $\boldsymbol{\tau}_0 = \langle\rangle$. *At step 1,* $\boldsymbol{\tau}_1 = \langle a \rangle$ *by inserting* a *into* $\boldsymbol{\tau}_0$. *At step 2,* $\boldsymbol{\tau}_2 = \langle b, a \rangle$ *by inserting* b *at position 1 with probability* $\frac{\phi}{1+\phi}$. *At step 3,* c *must be placed before* a, *so* $J = \{1, 2\}$. *Consider that* $p_1 \propto \phi^2$, $p_2 \propto \phi$, *and* $p_1 + p_2 = 1$. *So* $\boldsymbol{\tau}' = \langle b, c, a \rangle$ *by inserting* c *at position 2 with probability* $p_2 = \frac{\phi}{\phi + \phi^2}$. *The probability of sampling* $\boldsymbol{\tau}_0$ *is* $\Pr(\boldsymbol{\tau}' \mid \langle a, b, c \rangle, \phi, \{c \succ a\}) = \frac{\phi}{1+\phi} \cdot \frac{\phi}{\phi + \phi^2} = \frac{\phi}{(1+\phi)^2}$.

2.3 Labeled RIM Matching

We now recall *labeled RIM matching* [17], an inference problem that will be useful for query evaluation later. A *labeled RIM*, denoted by $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$, augments $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ with a labeling function λ, mapping each item to a finite set of its associated labels. Let $\boldsymbol{\tau}$ be a ranking of length m generated by $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$. An *embedding* of a label pattern g in $\boldsymbol{\tau}$ is a function $\delta : nodes(g) \to [1, m]$ satisfying the conditions:

1. Labels match: $\forall l \in nodes(g), l \in \lambda(\boldsymbol{\tau}(\delta(l)))$
2. Edges match: $\forall (l, l') \in edges(g), \boldsymbol{\tau}(\delta(l)) \succ_{\boldsymbol{\tau}} \boldsymbol{\tau}(\delta(l'))$

If such embedding function δ exists, we say that $\boldsymbol{\tau}$ (w.r.t. λ) *matches* (or *satisfies*) g, denoted by $(\boldsymbol{\tau}, \lambda) \models g$. When λ is clear from context, we write $\boldsymbol{\tau} \models g$. The items selected by the embedding function are the *matching items*.

EXAMPLE 2.3. *Given a ranking* $\boldsymbol{\tau}_0 = \langle$`Trump`, `Clinton`, `Sanders`, `Rubio`$\rangle$, *the labeling function* λ_0 *in Figure 1, and the pattern* g_0 *in Figure 2, there exists an embedding function* $\delta_0 = \{F \mapsto 2, M \mapsto 3\}$, *with matching items* $\boldsymbol{\tau}_0(2) =$ `Clinton` *for label* F, *and* $\boldsymbol{\tau}_0(3) =$ `Sanders` *for* M. *The edge* (F, M) *matches* `Clinton` $\succ_{\boldsymbol{\tau}_0}$ `Sanders`, *and so* $(\boldsymbol{\tau}_0, \lambda_0) \models g_0$ *with* δ_0.

The problem of *pattern matching* on labeled RIM is as follows. Given $\mathsf{RIM}_\mathsf{L}(\boldsymbol{\sigma}, \Pi, \lambda)$ and a pattern g, compute the probability that a random ranking $\boldsymbol{\tau} \sim \mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ satisfies g (w.r.t. λ). This is also the marginal probability of g over $\mathsf{RIM}_\mathsf{L}(\boldsymbol{\sigma}, \Pi, \lambda)$:

$$\Pr(g \mid \boldsymbol{\sigma}, \Pi, \lambda) = \sum_{\substack{\boldsymbol{\tau} \in rnk(A) \\ (\boldsymbol{\tau}, \lambda) \models g}} \Pr(\boldsymbol{\tau} \mid \boldsymbol{\sigma}, \Pi) \tag{1}$$

where $rnk(A)$ is the set of all $m!$ rankings over items A.

3. QUERY EVALUATION

In this section, we explain query evaluation in a RIM-PPD and refine the problem statement given in Section 1.

3.1 Conjunctive Query Evaluation

Given a Conjunctive Query (CQ) expressing preferences with a p-relation, if all atoms of p-relation refer to the same session, this query is a *sessionwise* CQ. If the sessionwise CQ is equivalent to a label pattern over each session, this is an *itemwise* CQ. Otherwise, a *non-itemwise* CQ.

In a recent paper, we showed how to reduce query evaluation of itemwise CQs to labeled RIM matching, and developed a solver for this inference problem, called Lifted Top Matching (LTM) [5]. Given an itemwise CQ Q and a RIM-PPD D, we wish to compute the marginal probability that Q is satisfied. Under the assumption that there are n independent sessions $\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ in a p-relation, we can evaluate Q over each session and aggregate the results from all sessions as follows:

$$\Pr(Q \mid D) = 1 - \prod_{i=1}^{n} (1 - \Pr(Q \mid \mathbf{s}_i))$$

Thus, query evaluation is reduced to evaluating the query over each session. For a particular session $\mathbf{s}$, we denote by $\mathsf{RIM}(\boldsymbol{\sigma}^\mathbf{s}, \Pi^\mathbf{s})$ its RIM model, by λ the labeling function of database D, and by g the label pattern corresponding to Q (as defined in Section 2.1), which leads to the labeled RIM matching problem in Section 2.3. Let $\mathsf{RIM}_\mathsf{L}(\boldsymbol{\sigma}^\mathbf{s}, \Pi^\mathbf{s}, \lambda)$ denote the labeled RIM over session $\mathbf{s}$. The probability that Q holds on session $\mathbf{s}$ is the marginal probability of g over $\mathsf{RIM}_\mathsf{L}(\boldsymbol{\sigma}^\mathbf{s}, \Pi^\mathbf{s}, \lambda)$.

$$\Pr(Q \mid \mathbf{s}) = \Pr(g \mid \boldsymbol{\sigma}^\mathbf{s}, \Pi^\mathbf{s}, \lambda) = \sum_{\substack{\boldsymbol{\tau} \in rnk(A) \\ (\boldsymbol{\tau}, \lambda) \models g}} \Pr(\boldsymbol{\tau} \mid \boldsymbol{\sigma}^\mathbf{s}, \Pi^\mathbf{s})$$

LTM calculates this probability with complexity $O(2^q m^q)$, where q is the number of nodes in g, see [5] for details.

Non-itemwise CQs are the sessionwise CQs with some variable(s) preventing label pattern reduction. In contrast to itemwise CQs, for which query evaluation has polynomial-time data complexity, the evaluation of non-itemwise CQs is #P-hard [5, Theorems 4.4 and 4.5]. To evaluate a non-itemwise CQ, we ground its variables, and rewrite it into a union of itemwise CQs. Let $V^+(Q)$ denote the set of variables to ground. For example, Q_2 in Section 1 is non-itemwise due to variable e. So $V^+(Q_2) = \{e\}$ and $Q_2 = Q_2^{\mathsf{BS}} \cup Q_2^{\mathsf{JD}}$. Note that these CQs are neither disjoint nor independent. For each session in a RIM-PPD, a union of itemwise CQs is equivalent to a union of label patterns, and the probability of Q is the sum of the probabilities of rankings that satisfy at least one pattern in the union.

3.2 Beyond Conjunctive Queries

Count-Session. A Boolean CQ Q computes the probability that Q is satisfied in a random possible world, while a Count-Session query, denoted $\texttt{count}(Q)$, computes the number of sessions satisfying Q. Since RIM-PPDs are probabilistic, $\texttt{count}(Q)$ is evaluated under the possible world semantics, and corresponds to the *expectation* of $\texttt{count}(Q)$ over the distribution of possible worlds.

Let $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ denote n sessions in a p-relation. The expectation of $\texttt{count}(Q)$ is the sum of the probabilities that the sessions satisfy Q: $\texttt{count}(Q) = \sum_{i=1}^{n} \Pr(Q|\mathbf{s_i})$.

Most-Probable-Session. For a Boolean CQ Q and an integer k, a Most-Probable-Session query, denoted $\texttt{top}(Q, k)$, finds k sessions in which Q is satisfied with the highest probability. We implement two strategies for this operator. The first calculates $\Pr(Q)$ for each session, then selects k most supportive sessions. The second strategy, named *top-k optimization*, first quickly calculates the upper bounds for all sessions, and then calculates the exact probability of sessions in descending order of their upper bounds, stopping once there are at least k sessions whose exact probability is no lower than the highest remaining upper-bound.

We will present an approach to compute the upper-bound of any pattern union using a *bipartite solver* that implements the top-k optimization in Section 4.3.2. This approach constructs a new pattern union G' with selected edges from the original G. To derive a tight upper-bound, we want to keep the edges that are hardest to satisfy. We first calculate all possible edges in G by transitive closure, then select edges using the following heuristic.

Let $\alpha(l \mid \boldsymbol{\tau})$ be the minimum position (highest rank) of items with label l in a ranking $\boldsymbol{\tau}$, and let $\beta(l \mid \boldsymbol{\tau})$ be the maximum position (lowest rank). The *ease* of an edge (l, l') to be satisfied by a random permutation from $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$ is estimated by: $\texttt{ease}(l, l' \mid \boldsymbol{\sigma}) = \beta(l' \mid \boldsymbol{\sigma}) - \alpha(l \mid \boldsymbol{\sigma})$. We construct G' with edges of small $\texttt{ease}$ values. If only one edge is selected for each pattern, G' is a union of two-label patterns, and $\texttt{top}(Q, k)$ invokes the two-label solver (see Section 4.2). Otherwise, G' is a union of bipartite patterns, and the bipartite solver is invoked (see Section 4.3).

Exact solvers have complexity exponential in the number of labels, so G' is much faster to compute. Because fewer labels and fewer edges lead to fewer constraints, more permutations satisfy G', and so $\Pr(G'|\boldsymbol{\sigma}, \Pi) \geq \Pr(G|\boldsymbol{\sigma}, \Pi)$.

3.3 Problem Statement

Queries in this paper include non-itemwise CQs, Count-Session queries, and Most-Probable-Session queries. The evaluation of these hard queries is reduced to a generalized inference problem of labeled RIM matching: given a pattern union $G = g_1 \cup \ldots \cup g_z$, compute its marginal probability over $\mathsf{RIM}_\mathsf{L}(\boldsymbol{\sigma}, \Pi, \lambda)$:

$$\Pr(G \mid \boldsymbol{\sigma}, \Pi, \lambda) = \sum_{\substack{\boldsymbol{\tau} \in rnk(A) \\ \exists g \in G, (\boldsymbol{\tau}, \lambda) \models g}} \Pr(\boldsymbol{\tau} \mid \boldsymbol{\sigma}, \Pi) \tag{2}$$

Sections 4 and 5 will present exact and approximate solvers for this problem, respectively.

4. EXACT SOLVERS

Let $\mathsf{RIM}_\mathsf{L}(\boldsymbol{\sigma}, \Pi, \lambda)$ be a labeled RIM model with reference ranking $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$. Let $G = g_1 \cup \ldots \cup g_z$ be a union

of z patterns. We are interested in the marginal probability of G over $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$ defined in Equation (2).

Equation (2) needs to enumerate $m!$ permutations. In this section, we will propose more efficient approaches.

4.1 General Solver

The general solver applies inclusion-exclusion principle:

$$\Pr(G \mid \boldsymbol{\sigma}, \Pi, \lambda) = \Pr(g_1 \cup \ldots \cup g_z \mid \boldsymbol{\sigma}, \Pi, \lambda) =$$

$$\sum_{i=1}^{z} \Pr(g_i \mid \boldsymbol{\sigma}, \Pi, \lambda) - \sum_{1 \le i_1 < i_2 \le z} \Pr(g_{i_1} \wedge g_{i_2} \mid \boldsymbol{\sigma}, \Pi, \lambda) \quad (3)$$

$$+ \ldots + (-1)^{(z-1)} \Pr(g_1 \wedge \ldots \wedge g_z \mid \boldsymbol{\sigma}, \Pi, \lambda)$$

where the conjunction $g_i \wedge \ldots \wedge g_j$ is a pattern containing all nodes and edges in $\{g_i, ..., g_j\}$.

Example 4.1. *Let $G = g_1 \cup g_2$ where $g_1 = \{l_1 \succ l_2\}$ and $g_2 = \{l_3 \succ l_4\}$. Its marginal probability over $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$ is $\Pr(g_1 \mid \boldsymbol{\sigma}, \Pi, \lambda) + \Pr(g_2 \mid \boldsymbol{\sigma}, \Pi, \lambda) - \Pr(g_3 \mid \boldsymbol{\sigma}, \Pi, \lambda)$ where $g_3 = g_1 \wedge g_2 = \{l_1 \succ l_2, l_3 \succ l_4\}$.*

The RIM inference problem for pattern unions has been reduced to a RIM inference problem for patterns, which can be solved by the LTM solver [5]. The complexity of LTM is $O(2^q m^q)$, where m is the number of items in $\boldsymbol{\sigma}$ and q is the number of nodes in one pattern [5]. The complexity of the general solver is dominated by the largest pattern conjunction $g_1 \wedge g_2 \wedge \ldots \wedge g_z$. Assuming that each g_i has q nodes, the general solver runs in $O((2m)^{q \cdot z})$. We use this solver as a baseline in our experiments.

4.2 Two-label Solver

A common class of queries concerns analysis of preferences over a pair of items. Such queries are reduced to a union of *two-label* patterns, and we call them *two-label* queries. For example, Q_2 in Section 1 is a two-label query: $Q_2() \leftarrow P(\_, \_; c_1; c_2), C(c_1, \texttt{D}, \_, \_, e, \_), C(c_2, \texttt{R}, \_, \_, e, \_)$. By instantiating e with BS and JD, Q_2 is reduced to a pattern union $G = g_1 \cup g_2$, where $g_1 = \{\{\texttt{D}, \texttt{BS}\} \succ \{\texttt{R}, \texttt{BS}\}\}$ and $g_2 = \{\{\texttt{D}, \texttt{JD}\} \succ \{\texttt{R}, \texttt{JD}\}\}$ are both two-label patterns.

Since all patterns in G only have two labels, we re-write $G = g_1 \cup \ldots \cup g_z = \bigcup_{i=1}^{z} \{l_i \succ r_i\}$. The labels $\{l_1, \ldots, l_z\}$ are the L-type labels, while $\{r_1, \ldots, r_z\}$ R-type.

Instead of calculating the probability that G is satisfied, the two-label solver calculates the probability that G is violated. Let $\boldsymbol{\tau} \not\models g$ and $\boldsymbol{\tau} \not\models G$ denote that a permutation $\boldsymbol{\tau}$ violates a pattern g and a pattern union G, respectively. Then $\boldsymbol{\tau} \not\models G$ if and only if $\forall g_i \in G, \boldsymbol{\tau} \not\models g_i$. Let $\alpha(l)$ be the minimum position (highest rank) of items with label l in a ranking, while $\beta(l)$ the maximum position (lowest rank). These are the Min/Max positions of a label in a ranking. Given a two-label pattern $g = \{l \succ r\}$ and a ranking $\boldsymbol{\tau}$, we can check whether $\boldsymbol{\tau} \models g$ by the Min/Max positions of labels. Namely, $\boldsymbol{\tau} \models g$ if $\alpha(l) < \beta(r)$ and $\boldsymbol{\tau} \not\models g$ if $\alpha(l) \ge \beta(r)$.

Algorithm 2 presents the two-label solver. It first calculates the complementary event of G by dynamic programming during RIM insertions. States are in the form of $\langle \alpha, \beta \rangle$, tracking Min positions for L-type labels and Max positions for R-type labels. States in $\mathcal{P}_i$ are generated by inserting item σ_i into the states in $\mathcal{P}_{i-1}$. Let $\langle \alpha_{i \to j}, \beta_{i \to j} \rangle$ denote a new state generated by inserting item σ_i into $\langle \alpha, \beta \rangle$ at position j; $\langle \alpha_{i \to j}, \beta_{i \to j} \rangle$ is updated from $\langle \alpha, \beta \rangle$ as follows:

- $\alpha_{i \to j}(l) = min(\alpha(l), j)$ if $l \in \lambda(\sigma_i)$ and l is L-type;
- $\beta_{i \to j}(l) = max(\beta(l), j)$ if $l \in \lambda(\sigma_i)$ and l is R-type;
- $\alpha_{i \to j}(l) = \alpha(l) + 1$ if $l \notin \lambda(\sigma_i)$ and $\alpha(l) \ge j$;
- $\beta_{i \to j}(l) = \beta(l) + 1$ if $l \notin \lambda(\sigma_i)$ and $\beta(l) \ge j$.

Algorithm 2 TwoLabelSolver

Require: $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$, $G = \bigcup_{i=1}^{z} \{l_i \succ r_i\}$
1: $\mathcal{P}_0 := \{\langle \{\}, \{\} \rangle\}, q_0 := \{\langle \{\}, \{\} \rangle \mapsto 1\}$
2: **for** $i = 1, .., m$ **do**
3: $\quad \mathcal{P}_i := \{\}$
4: $\quad$ **for** $\langle \alpha, \beta \rangle \in \mathcal{P}_{i-1}$ **do**
5: $\quad\quad$ **for** $j = 1, ..., i$ **do**
6: $\quad\quad\quad$ Generate a new state $\langle \alpha_{i \to j}, \beta_{i \to j} \rangle$ by inserting $\boldsymbol{\sigma}_i$ into $\langle \alpha, \beta \rangle$ at position j, and updating Min/Max positions according to the labeling function λ.
7: $\quad\quad\quad$ **if** $\langle \alpha_{i \to j}, \beta_{i \to j} \rangle \not\models G$ **then**
8: $\quad\quad\quad\quad \mathcal{P}_i.add(\langle \alpha_{i \to j}, \beta_{i \to j} \rangle)$
9: $\quad\quad\quad\quad q_i(\langle \alpha_{i \to j}, \beta_{i \to j} \rangle) \mathrel{+}= q_{i-1}(\langle \alpha, \beta \rangle) \cdot \Pi(i, j)$
10: **return** $1 - \sum_{\langle \alpha, \beta \rangle \in \mathcal{P}_m} q_m(\langle \alpha, \beta \rangle)$

The algorithm only tracks the states that violate G, and its complexity is $O(m^{2z+1})$.

Example 4.2. *Let $\mathsf{RIM_L}(\boldsymbol{\sigma}_0, \Pi_0, \lambda_0)$ be a labeled RIM with $\boldsymbol{\sigma}_0 = \langle a, b, c \rangle$. Let $G = g_1 \cup g_2$ be a pattern union. We will focus on g_1 in this example. Let $g_1 = \{l_1 \succ r_1\}$. Assume that λ_0 associates items a and c with label l_1, and b with label r_1. At step 1, insert a and generate state $\langle \alpha_1, \beta_1 \rangle$ with probability $q_1(\langle \alpha_1, \beta_1 \rangle) = \Pi_0(1, 1) = 1$, where $\alpha_1 = \{l_1 \mapsto 1\}$ and $\beta_1 = \{\}$. At step 2, b must be inserted before a to violate g_1. So $\alpha_2 = \{l_1 \mapsto 2\}$, $\beta_2 = \{r_1 \mapsto 1\}$, and $q_2(\langle \alpha_2, \beta_2 \rangle) = q_1(\langle \alpha_1, \beta_1 \rangle) \cdot \Pi_0(2, 1) = \Pi_0(2, 1)$. At step 3, c must be inserted after b to violate g_1. So $\beta_3 = \{r_1 \mapsto 1\}$. Item c can be inserted either before item a generating $\alpha_3(l_1) = min(\alpha_2(l_1), 2) = 2$ with probability $\Pi_0(3, 2)$, or after a generating $\alpha_3(l_1) = min(\alpha_2(l_1), 3) = 2$ with probability $\Pi_0(3, 3)$. Both scenarios generate the same $\alpha_3(l_1) = 2$, thus their probabilities are merged by $q_3(\langle \alpha_3, \beta_3 \rangle) = q_2(\langle \alpha_2, \beta_2 \rangle) \cdot \Pi_0(3, 2) + q_2(\langle \alpha_2, \beta_2 \rangle) \cdot \Pi_0(3, 3) = \Pi_0(2, 1) \cdot (\Pi_0(3, 2) + \Pi_0(3, 3))$.*

Theorem 4.1. *Given $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$ and a union of two-label patterns G, Algorithm 2 returns $\Pr(G \mid \boldsymbol{\sigma}, \Pi, \lambda)$, the marginal probability of G over $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$.*

Proof sketch. While running RIM insertions, Algorithm 2 groups generated rankings by their Min/Max positions of labels, denoted by $\langle \alpha, \beta \rangle$. The $q(\langle \alpha, \beta \rangle)$ represents the sum of probabilities of the rankings included in $\langle \alpha, \beta \rangle$. The algorithm only tracks states violating G, and prunes states that satisfied G in past. At step i, it is enough to only consider states in $\mathcal{P}_{i-1}$ to generate new states $\mathcal{P}_i$. We prove correctness of Algorithm 2 by induction. The formal proof is provided in our technical report [25].

4.3 Bipartite Solver

A bipartite pattern is similar to a bipartite graph. The nodes are classified into two sets L and R, such that all directed edges are in the form $(l, r), l \in L, r \in R$. Labels in L and R are L-type and R-type, respectively.

With the definition of α and β in Section 4.2, an edge (l, r) in a bipartite pattern is essentially $\alpha(l) < \beta(r)$. A ranking satisfies a bipartite pattern g if it satisfies all Min/Max constraints specified by g.

For a union of bipartite patterns $G = g_1 \cup \ldots \cup g_z$, the solver tracks α for L-type labels and β for R-type labels. A permutation satisfies G if it satisfies any pattern $g \in G$.

4.3.1 Algorithm Description

The basic version of a bipartite solver works as follows. It is a Dynamic Programming algorithm that tracks the minimum positions of L-type labels and the maximum positions of R-type labels, during RIM insertion process. At step i, the first i items in $\boldsymbol{\sigma}$ are inserted, and $i!$ rankings are generated accordingly. These rankings are grouped into states in the form of $\langle \alpha, \beta \rangle$ where α maps L-type labels to their minimum positions and β maps R-type labels to their maximum positions. After all items are inserted, enumerate all states and add up the probabilities of the states satisfying at least one pattern $g_i \in G$. The complexity of this algorithm is $O(m^{qz})$, where m is the number of items in $\boldsymbol{\sigma}$, q is the number of labels per pattern, and z is the number of patterns in G.

The more sophisticated version of bipartite solver dynamically prunes labels tracked by states based on the "situations" of patterns and edges. The "situations" are {satisfied, violated, uncertain}. An edge (l, r) is satisfied if $\alpha(l) < \beta(r)$; violated if $\alpha(l) \geq \beta(r)$ after all items in l and r are inserted; uncertain if it is neither satisfied nor violated. A pattern is satisfied if all its edges are satisfied; violated if any of its edges are violated; and uncertain otherwise.

The key observation is that once an edge is satisfied by a state, this state will always satisfy this edge in the future. The same is true for an edge being violated, a pattern being satisfied, and a pattern being violated. This enables several optimization opportunities: (i) An edge is satisfied: no need to track this edge. (ii) An edge is violated: the entire pattern is violated, no need to track this pattern. (iii) A pattern is satisfied: the pattern union G is satisfied, add the probability of this state into the marginal probability, no need to track this pattern. (iv) A pattern is violated: no need to track this pattern.

In summary, the bipartite solver only needs to track labels in uncertain edges of uncertain patterns.

Algorithm 3 presents the bipartite solver that uses RIM (see Section 2.2) as basis for inference. At step i, it maintains a set of states $\mathcal{P}_i$. A state $\langle \alpha, \beta \rangle$ tracks the Min/Max positions of labels. The $\mathcal{E}_i$ maps a state $\langle \alpha, \beta \rangle$ to G_u, a union of uncertain patterns with uncertain edges in this state. Before running RIM, all patterns and edges are uncertain, so G_u=G. The probabilities of the states are tracked by q_i.

Recall that RIM sampling starts with an empty ranking. Therefore, the initial state is $\langle \{\}, \{\} \rangle$, and $\mathcal{E}_0(\langle \{\}, \{\} \rangle) = G$, $q_0(\langle \{\}, \{\} \rangle) = 1$. At step i, generate new states by inserting item σ_i into states in $\mathcal{P}_{i-1}$. If a new state already satisfies some pattern, accumulate its probability, otherwise put it into the set $\mathcal{P}_i$. When a new item σ_i is inserted into $\langle \alpha, \beta \rangle$ at position j, update $\langle \alpha_{i\to j}, \beta_{i\to j} \rangle$ as follows:

- $\alpha_{i\to j}(l) = min(\alpha(l), j)$ if $l \in \lambda(\sigma_i)$ and l is L-type.
- $\beta_{i\to j}(l) = max(\beta(l), j)$ if $l \in \lambda(\sigma_i)$ and l is R-type.
- $\alpha_{i\to j}(l) = \alpha(l) + 1$ if $l \notin \lambda(\sigma_i)$ and $\alpha(l) \geq j$.
- $\beta_{i\to j}(l) = \beta(l) + 1$ if $l \notin \lambda(\sigma_i)$ and $\beta(l) \geq j$.

EXAMPLE 4.3. *Let* $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}_0, \Pi_0, \lambda_0)$ *denote a labeled RIM where* $\boldsymbol{\sigma}_0 = \langle a, b, c, d \rangle$. *Let* $G = g_1 \cup g_2$ *be a pattern union where* $g_1 = \{l_1 \succ r_1, l_1 \succ r_2\}$, *on which we will focus right now. Assume that item a and c are associated with label* l_1, *while b with label* r_1, *d with label* r_2, *according to* λ_0. *Below are some solver execution scenarios.*

(i) At step 1, item a is inserted at position 1 with probability $\Pi_0(1,1) = 1$, *thus* $\alpha_{1\to 1}(l_1) = 1$. *(ii)If at step 2, item b is inserted before a with probability* $\Pi_0(2,1)$, $\beta_{2\to 1}(r_1) = 1$ *and* $\alpha_{1\to 1}(l_1) + 1 = 2$. *If item b is inserted after a with probability* $\Pi_0(2,2)$, $\beta_{2\to 1}(r_1) = 2$. *Edge* (l_1, r_1) *is already satisfied by this state, so there is no need to track* r_1 *any more. The* G_u *will have* $g_1 = \{l_1 \succ r_2\}$. *(iii)For the state informally represented by* $\{l_1 \mapsto 2, r_1 \mapsto 1\}$, *if at step 3, item c is inserted after b at position 2 with probability* $\Pi_0(3,2)$ *or at position 3 with probability* $\Pi_0(3,3)$, *edge* (l_1, r_1) *is violated, which leads to pattern* g_1 *getting violated. The* G_u *will remove* g_1 *and only track* g_2 *later. (iv)If at step 4, item d is inserted after a or c, pattern* g_1 *is satisfied by the new state, then G is satisfied no matter what the "situation" of* g_2 *is, and the probability of this state is accumulated into the marginal probability prob.*

Algorithm 3 BipartiteSolver

Require: $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$, $G = g_1 \cup \ldots \cup g_z$
1: $\mathcal{P}_0 := \{\langle \{\}, \{\} \rangle\}$, $\mathcal{E}_0 := \{\langle \{\}, \{\} \rangle \mapsto G\}$, $q_0 := \{\langle \{\}, \{\} \rangle \mapsto 1\}$
2: *prob* := 0
3: **for** $i = 1, .., m$ **do**
4: $\quad \mathcal{P}_i := \{\}$
5: $\quad$ **for** $\langle \alpha, \beta \rangle \in \mathcal{P}_{i-1}$ **do**
6: $\quad\quad G_u := \mathcal{E}_{i-1}(\langle \alpha, \beta \rangle)$
7: $\quad\quad$ **for** $j = 1, ..., i$ **do**
8: $\quad\quad\quad$ Generate a new state $\langle \alpha_{i\to j}, \beta_{i\to j} \rangle$ by inserting $\boldsymbol{\sigma}_i$ into $\langle \alpha, \beta \rangle$ at position j, and updating Min/Max positions according to the labeling function λ.
9: $\quad\quad\quad$ **if** $\langle \alpha_{i\to j}, \beta_{i\to j} \rangle$ violates all patterns in G_u **then**
10: $\quad\quad\quad\quad$ Ignore $\langle \alpha_{i\to j}, \beta_{i\to j} \rangle$
11: $\quad\quad\quad$ **else**
12: $\quad\quad\quad\quad p' := q_{i-1}(\langle \alpha, \beta \rangle) \cdot \Pi(i, j)$
13: $\quad\quad\quad\quad G'_u :=$ OnlyTrackUncertainPatterns(G_u)
14: $\quad\quad\quad\quad$ **if** $\exists g \in G'_u$, all edges in g are satisfied **then**
15: $\quad\quad\quad\quad\quad prob \mathrel{+}= p'$
16: $\quad\quad\quad\quad$ **else**
17: $\quad\quad\quad\quad\quad \langle \alpha_{i\to j}, \beta_{i\to j} \rangle$.onlyTrackLabelsFor($G'_u$)
18: $\quad\quad\quad\quad\quad \mathcal{P}_i.add(\langle \alpha_{i\to j}, \beta_{i\to j} \rangle)$
19: $\quad\quad\quad\quad\quad \mathcal{E}_i(\langle \alpha_{i\to j}, \beta_{i\to j} \rangle) := G'_u$
20: $\quad\quad\quad\quad\quad q_i(\langle \alpha_{i\to j}, \beta_{i\to j} \rangle) \mathrel{+}= p'$
21: **return** *prob*

THEOREM 4.2. *Given* $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$ *and a union of bipartite patterns G, Algorithm 3 returns* $\Pr(G \mid \boldsymbol{\sigma}, \Pi, \lambda)$, *the marginal probability of G over* $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$.

Proof sketch. Algorithm 3 searches for rankings that satisfy at least one pattern in G. Instead of enumerating all $m!$ rankings in the search space, the algorithm runs RIM and inspects the generated sub-rankings of length i at step $i \in [0, m]$. Generated rankings are grouped into states of the form $\langle \alpha, \beta \rangle$ by their Min/Max label positions. At step i, the set $\mathcal{P}_i$ consists of all the states that can potentially satisfy G in future items, and *prob* has accumulated the probabilities of the states that satisfied G in past. Correctness of Algorithm 2 is shown by induction. The detailed proof is given in the accompanying technical report [25].

4.3.2 Bipartite Solver for Upper Bounds

Let $tc(g)$ be the transitive closure of pattern g. Each edge $(l, r) \in tc(g)$ represents a constraint $\alpha(l) < \beta(r)$. Let U denote the set of these constraints. By the definition of *label embedding*, any ranking $\boldsymbol{\tau}$ satisfying g must satisfy U, denoted by $\boldsymbol{\tau} \models U$, so U gives an upper bound of g.

EXAMPLE 4.4. *Let $g_0 = \{l_a \succ l_b, l_b \succ l_c\}$, a linear order $l_a \succ l_b \succ l_c$. Then $tc(g_0) = \{l_a \succ l_b, l_b \succ l_c, l_a \succ l_c\}$, and $U_0 = \{\alpha(l_a) < \beta(l_b), \alpha(l_b) < \beta(l_c), \alpha(l_a) < \beta(l_c)\}$ accordingly. If a ranking $\boldsymbol{\tau}_0 \models g_0$, $\boldsymbol{\tau}_0$ must satisfy all constraints in U_0. But if $\boldsymbol{\tau}_0 \models U_0$, it is possible that $\boldsymbol{\tau}_0 \not\models g_0$. For example $\boldsymbol{\tau}_0 = \langle b_1, a, c, b_2 \rangle$ w.r.t. $\lambda_0 = \{a \mapsto \{l_a\}, b_1 \mapsto \{l_b\}, b_2 \mapsto \{l_b\}, c \mapsto \{l_c\}\}$. In this case, $\boldsymbol{\tau}_0 \models U_0$ but $\boldsymbol{\tau}_0 \not\models g_0$.*

For a pattern union $G = g_1 \cup \ldots \cup g_z$, we can also calculate its upper bound in a similar way. Let U_i denote the upper bound constraints for $g_i \in G$. For any ranking $\boldsymbol{\tau}$, $\boldsymbol{\tau} \models G$ if and only if $\exists g_i \in G, \boldsymbol{\tau} \models g_i$. The U_i is less strict than g_i, so $\boldsymbol{\tau} \models U_i$ if $\boldsymbol{\tau} \models g_i$. Let $\mathcal{U} = U_1 \cup \ldots \cup U_z$ denote the union of upper bound constraints. Then $\boldsymbol{\tau} \models \mathcal{U}$ iff $\exists U_i \in \mathcal{U}, \boldsymbol{\tau} \models U_i$. So $\boldsymbol{\tau} \models \mathcal{U}$ if $\boldsymbol{\tau} \models G$. The $\mathcal{U}$ gives an upper bound for G.

Let U_s denote a subset of U. Note that U_s also gives an upper bound of g that is less strict than the original U, but is faster to calculate. The same conclusion applies to a union of constraint subsets. This is the principle behind the evaluation of Most-Probable-Session queries in Section 3.2.

5. APPROXIMATE SOLVERS

Exact solvers compute answers to intractable problems. To address scalability challenges that are inherent in the problem, we design approximate solvers that leverage the structure of the Mallows model, and specifically the recent results on efficient sampling from the Mallows posterior [20].

Let $\mathsf{MAL_L}(\boldsymbol{\sigma}, \phi, \lambda)$ denote a labeled Mallows model with labeling function λ. Let $G = g_1 \cup \ldots \cup g_z$ be a union of z patterns. We are interested in $\Pr(G \mid \boldsymbol{\sigma}, \phi, \lambda)$, the marginal probability of G over $\mathsf{MAL_L}(\boldsymbol{\sigma}, \phi, \lambda)$. This is also the posterior probability of G over $\mathsf{MAL_L}(\boldsymbol{\sigma}, \phi, \lambda)$, or the expectation that a sample $\boldsymbol{\tau}$ from $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$ satisfies G w.r.t. λ.

5.1 Importance Sampling for Mallows

Sampling is popular for probability estimation. For example, we can use Rejection Sampling (RS) to sample a large number of rankings from $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$ and count how many of them satisfy G. Generally, RS works well if the target probability is high, but is impractical for estimating low-probability events. Importance Sampling (IS) can effectively estimate rare events [13, 14]. IS estimates the expected value of a function $f(x)$ in a probability space P via sampling from another *proposal distribution* Q, then re-weights the samples for unbiased estimation. Assume that x is discrete, and that N samples $\{x_1, x_2, ..., x_N\}$ are generated from Q. Let $p(x) = \Pr(x \mid P)$ and $q(x) = \Pr(x \mid Q)$. The estimation is done as follows:

$$\begin{aligned} \mathbb{E}_P\big(f(x)\big) &= \sum_{x \in P} f(x) \cdot p(x) = \sum_{x \in Q} f(x) \cdot \frac{p(x)}{q(x)} \cdot q(x) \\ &= \mathbb{E}_Q\left(f(x) \cdot \frac{p(x)}{q(x)}\right) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(x_i)}{q(x_i)} f(x_i) \end{aligned} \tag{4}$$

IS re-weights each sample x_i by an *importance factor* $\frac{p(x_i)}{q(x_i)}$. When applying IS, Q is chosen to support efficient sampling and, ideally, to provide estimates $q(x)$ that are close to $p(x)$, also for efficiency reasons. To calculate $\Pr(G|\boldsymbol{\sigma}, \phi, \lambda) = \mathbb{E}(\mathbb{1}((\boldsymbol{\tau}, \lambda) \models G))$, we set $f(x) = \mathbb{1}((\boldsymbol{\tau}, \lambda) \models G)$, where ranking $\boldsymbol{\tau}$ is a sample.

5.2 From Pattern Union to Sub-ranking Union

Before diving into details of applying IS to RIM inference, let us examine the meaning of $(\boldsymbol{\tau}, \lambda) \models G$. Previously, we had $(\boldsymbol{\tau}, \lambda) \models G$ if and only if $\exists g \in G, (\boldsymbol{\tau}, \lambda) \models g$. Recall from Section 2.1 that $(\boldsymbol{\tau}, \lambda) \models g$ if there exists an embedding function δ in which labels match $(\forall l \in nodes(g), l \in \lambda(\boldsymbol{\tau}(\delta(l))))$ and edges match $(\forall (l, l') \in edges(g), \delta(l) < \delta(l'))$.

The embedding δ constructs a partial order $\boldsymbol{\upsilon} = \{\delta(l) \succ \delta(l') | (l, l') \in edges(g)\}$ so that $\boldsymbol{\tau} \in \Omega(\boldsymbol{\upsilon})$. (Recall that $\Omega(\boldsymbol{\upsilon})$ is the set of linear extensions of $\boldsymbol{\upsilon}$.) Conceptually, a pattern g can be decomposed into a union of partial orders with different embedding functions. Let $\Delta(g, \lambda)$ denote the union of partial orders decomposed from g w.r.t. λ. Then $(\boldsymbol{\tau}, \lambda) \models g$ if and only if $\exists \boldsymbol{\upsilon} \in \Delta(g, \lambda), \boldsymbol{\tau} \in \Omega(\boldsymbol{\upsilon})$. We can calculate these partial orders for all patterns in G, any permutation $\boldsymbol{\tau}$ satisfying any partial order will immediately satisfy a pattern in G, and so will satisfy G itself. In this sense, G is equivalent to a union of partial orders.

A partial order $\boldsymbol{\upsilon}$ can further be decomposed into a union of sub-rankings that are consistent with $\boldsymbol{\upsilon}$. For example, $\boldsymbol{\upsilon} = \{a \succ c, b \succ c\}$ has two sub-rankings $\boldsymbol{\psi}_1 = \langle a, b, c \rangle$ and $\boldsymbol{\psi}_2 = \langle b, a, c \rangle$. Let $\Delta(\boldsymbol{\upsilon})$ denote the union of sub-rankings from partial order $\boldsymbol{\upsilon}$. Let $\boldsymbol{\tau} \models \boldsymbol{\psi}$ denote that a permutation $\boldsymbol{\tau}$ is consistent with a sub-ranking $\boldsymbol{\psi}$. Then $\boldsymbol{\tau} \in \Omega(\boldsymbol{\upsilon})$ if and only if $\exists \boldsymbol{\psi} \in \Delta(\boldsymbol{\upsilon}), \boldsymbol{\tau} \models \boldsymbol{\psi}$. Because G is equivalent to a union of partial orders (w.r.t. λ), we have: $G = \bigcup\{\boldsymbol{\psi} | \boldsymbol{\psi} \in \Delta(\boldsymbol{\upsilon}), \boldsymbol{\upsilon} \in \Delta(g, \lambda), g \in G\}$.

Figure 3 is an example, where a union of two patterns is decomposed into three partial orders, then further into six sub-rankings. A ranking satisfies the pattern union if and only if it satisfies the sub-ranking union. Assuming z patterns are decomposed into w sub-rankings, we have $G = g_1 \cup \ldots \cup g_z = \boldsymbol{\psi}_1 \cup \ldots \cup \boldsymbol{\psi}_w$.

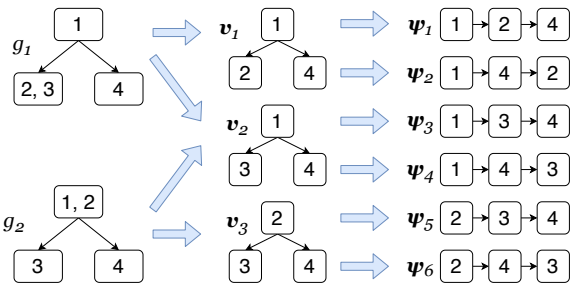


Figure 3: A union of two patterns (left), decomposed into a union of three partial orders (middle), then a union of six sub-rankings (right).

5.3 IS-AMP for a Single Sub-ranking

The pattern union G has been decomposed into w sub-rankings. Before dealing with all sub-rankings, let us see how to estimate the expectation of a single sub-ranking $\boldsymbol{\psi}$ over the Mallows model $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$.

If $\Pr(\boldsymbol{\psi} \mid \boldsymbol{\tau}, \phi)$ is low, RS is inefficient to reach accurate estimation. We can apply IS instead, using a proposal distribution that easily generates permutations satisfying $\boldsymbol{\psi}$.

Our first method, called IS-AMP, uses AMP, a state-of-the-art Mallows sampler conditioned on a partial order of items [20], to construct a proposal distribution. IS-AMP works well when the proposal distribution is around the "important region" of the probability space. Unfortunately, as we show next, AMP does not always give desirable proposal distributions, especially when there are multiple *modals* — peaks or local maxima — in the posterior distribution.

EXAMPLE 5.1. *Let $\boldsymbol{\psi}_0 = \langle \sigma_3, \sigma_1 \rangle$ be a sub-ranking for which we wish to calculate the expectation over $\mathsf{MAL}(\boldsymbol{\sigma}_0, \phi_0)$, with $\boldsymbol{\sigma}_0 = \langle \sigma_1, \sigma_2, \sigma_3 \rangle$ and $\phi_0 = 0.01$. Recall that with $\phi_0 = 0.01$, much of the probability mass of $\mathsf{MAL}(\boldsymbol{\sigma}_0, \phi_0)$ is around $\boldsymbol{\sigma}_0$. In this case IS-AMP will sample $\boldsymbol{\tau}_0 = \langle \sigma_3, \sigma_1, \sigma_2 \rangle$ very frequently, as follows: (i) Insert σ_1 into an empty ranking $\langle\rangle$. (ii) Insert σ_2 into $\langle \sigma_1 \rangle$ after σ_1 with probability $\frac{1}{1+0.01}$. (iii) Insert σ_3 into $\langle \sigma_1, \sigma_2 \rangle$ before σ_1 with probability 1. If all samples are $\boldsymbol{\tau}_0 = \langle \sigma_3, \sigma_1, \sigma_2 \rangle$, we will estimate:*

$$\begin{aligned} \textit{IS-AMP}(\boldsymbol{\psi}_0 \mid \mathsf{MAL}(\boldsymbol{\sigma}_0, \phi_0)) &\approx \frac{\Pr(\boldsymbol{\tau}_0 \mid \mathsf{MAL}(\boldsymbol{\sigma}_0, \phi_0))}{\Pr(\boldsymbol{\tau}_0 \mid \mathsf{AMP}(\boldsymbol{\sigma}_0, \phi_0, \boldsymbol{\psi}_0))} \\ &\approx \frac{9.9 \times 10^{-5}}{0.99} = 10^{-4} \end{aligned}$$

However, there are two modals *in the posterior distribution, $\boldsymbol{\tau}_0 = \langle \sigma_3, \sigma_1, \sigma_2 \rangle$ and $\boldsymbol{\tau}_1 = \langle \sigma_2, \sigma_3, \sigma_1 \rangle$. These modals are rankings that are closest to $\boldsymbol{\sigma}_0$ (in terms of Kendall-tau distance) among those that are consistent with $\boldsymbol{\tau}_0$, and so much of the probability mass of the posterior distribution is concentrated around them,* not *around $\boldsymbol{\sigma}_0$. We have:*

$$\begin{aligned} \Pr(\boldsymbol{\psi}_0 \mid \boldsymbol{\sigma}_0, \phi_0) &\geq \Pr(\boldsymbol{\tau}_0 \mid \boldsymbol{\sigma}_0, \phi_0) + \Pr(\boldsymbol{\tau}_1 \mid \boldsymbol{\sigma}_0, \phi_0) \\ &\approx 10^{-4} + 10^{-4} > \textit{IS-AMP}(\boldsymbol{\psi}_0 \mid \boldsymbol{\sigma}_0, \phi_0) \end{aligned}$$

In the example above, IS-AMP fails to effectively estimate the probability, because the posterior distribution is multi-modal. To address this issue, we design MIS-AMP, a new sampler based on AMP geared specifically at multi-modal distributions. We describe MIS-AMP next.

5.4 MIS-AMP for a Single Sub-ranking

We first give some general background on Multiple Importance Sampling (MIS), and will then show how it is applied to our scenario. Assume there are d proposal distributions with probability mass functions $\{q_1, ..., q_d\}$ and n_i samples generated from q_i. Let $x_{i,j}$ be the j-th sample generated from q_i. For each $x_{i,j}$, MIS not only calculates its importance factor as does IS, but it also calculates a weight w_i with which $x_{i,j}$ is sampled from q_i. Let $N = \sum_{i=1}^{d} n_i$ and $c_i = n_i/N$. Let $f(x)$ be the function of which we want to compute the expectation, and $p(x)$ be the probability mass function of the original distribution. The MIS estimator is:

$$\mathbb{E}(f(x)) = \sum_{i=1}^{d} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(x_{i,j}) \frac{p(x_{i,j})}{q_i(x_{i,j})} f(x_{i,j}) \qquad (5)$$

This estimator is unbiased if $\forall x, \sum_i w_i(x) = 1$. Vech and Guibas [30] showed that the weighting function $w_i(x) = \frac{c_i q_i(x)}{\sum_{t=1}^{d} c_t q_t(x)}$, designed to balance the contribution of each proposal distribution to the estimate, is a good choice.

Algorithm 4 GreedyModals

Require: Sub-ranking $\boldsymbol{\psi}$, Mallows model $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$
1: $S := \{\boldsymbol{\psi}\}$
2: **for** $i = 1, 2, ..., m$ **do**
3: **if** $\sigma_i \notin \boldsymbol{\psi}$ **then**
4: $S' := \emptyset$
5: **for** $\boldsymbol{\psi} \in S$ **do**
6: $J = \{j \mid dist(\boldsymbol{\psi}_{i \to j}, \boldsymbol{\sigma}) = \min_{j'=1,...|\boldsymbol{\psi}|} dist(\boldsymbol{\psi}_{i \to j'}, \boldsymbol{\sigma})\}$
7: **for** $j \in J$ **do**
8: $S'.add(\boldsymbol{\psi}_{i \to j})$.
9: $S := S'$
10: **return** S

When generating an equal number of samples from all proposal distributions (i.e., $n_1 = ... = n_d = n$ and $c_1 = ... = c_d = 1/d$), the Equation (5) can be simplified as:

$$\mathbb{E}(f(x)) = \frac{1}{d \cdot n} \sum_{i=1}^{d} \sum_{j=1}^{n} \frac{p(x_{i,j})}{\frac{1}{d} \sum_{t=1}^{d} q_t(x_{i,j})} f(x_{i,j}) \qquad (6)$$

Importance Sampling for Mallows. A good proposal distribution for IS should produce more samples in the "important region" of the target distribution—the region wherein there is a significant probability mass. So, instead of sampling with the original Mallows, we sample permutations that are consistent with the sub-ranking $\boldsymbol{\psi}$. Among all such, the ones that are nearest to Mallows center $\boldsymbol{\sigma}$ are the modals of the posterior. The samples around these modals are the important regions, and they should be effectively captured by the proposal distributions.

Our strategy is to construct Mallows models centered at these modals, and run AMP over them conditioned on the sub-ranking $\boldsymbol{\psi}$. Unfortunately, it is intractable to find a completion of a partial order that is closest, in terms of Kendall-tau distance, to a given ranking (Theorem 2 in [3]). This makes finding the modals consistent with $\boldsymbol{\psi}$ that are closest to $\boldsymbol{\sigma}$ intractable. Algorithm 4 uses a greedy heuristic to search for modals, by inserting items into $\boldsymbol{\psi}$ at positions that minimize the distance to $\boldsymbol{\sigma}$. Note that $\boldsymbol{\psi}_{i \to j}$ is a sub-ranking, with σ_i inserted into $\boldsymbol{\psi}$ at position j.

Let $S = \{\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_d\}$ denote the set of modals output by Algorithm 4. We construct $\mathsf{MAL}(\boldsymbol{\sigma}_1, \phi), \ldots, \mathsf{MAL}(\boldsymbol{\sigma}_d, \phi)$, and run AMP over each, conditioned on the sub-ranking $\boldsymbol{\psi}$, raising d proposal distributions. We are interested in the expectation of $\mathbb{1}(\boldsymbol{\tau} \models \boldsymbol{\psi})$, where $\boldsymbol{\tau} \sim \mathsf{MAL}(\boldsymbol{\sigma}, \phi)$. Note that the permutations generated by MIS-AMP will always satisfy $\boldsymbol{\psi}$, i.e., $\mathbb{1}(\boldsymbol{\tau} \models \boldsymbol{\psi}) \equiv 1$. Using Equation (6), we estimate:

$$\mathbb{E}\big(\mathbb{1}(\boldsymbol{\tau} \models \boldsymbol{\psi})\big) = \frac{1}{d \cdot n} \sum_{i=1}^{d} \sum_{j=1}^{n} \frac{p(x_{i,j})}{\frac{1}{d} \sum_{t=1}^{d} q_t(x_{i,j})} \qquad (7)$$

EXAMPLE 5.2. *We now revisit Example 5.1 and solve it by MIS-AMP. Recall that we wish to calculate the expectation of $\boldsymbol{\psi}_0 = \langle \sigma_3, \sigma_1 \rangle$ over $\mathsf{MAL}(\boldsymbol{\sigma}_0, \phi_0)$ with $\boldsymbol{\sigma}_0 = \langle \sigma_1, \sigma_2, \sigma_3 \rangle$ and $\phi_0 = 0.01$. Algorithm 4 will find two modals, $\boldsymbol{\sigma}_1 = \langle \sigma_3, \sigma_1, \sigma_2 \rangle$ and $\boldsymbol{\sigma}_2 = \langle \sigma_2, \sigma_3, \sigma_1 \rangle$ as centers of the newly constructed $\mathsf{MAL}(\boldsymbol{\sigma}_1, \phi)$ and $\mathsf{MAL}(\boldsymbol{\sigma}_2, \phi)$. MIS-AMP then draws rankings from two AMP samplers, $\mathsf{AMP}(\boldsymbol{\sigma}_1, \phi_0, \boldsymbol{\psi}_0)$ and $\mathsf{AMP}(\boldsymbol{\sigma}_2, \phi_0, \boldsymbol{\psi}_0)$. Then $\boldsymbol{\tau}_0 = \langle \sigma_3, \sigma_1, \sigma_2 \rangle$ is re-weighted as follows.*

$$
\begin{aligned}
&\textit{MIS-AMP}(\boldsymbol{\tau}_0 \mid \boldsymbol{\sigma}_0, \phi_0, \boldsymbol{\psi}_0) \\
\approx &\frac{\Pr(\boldsymbol{\tau}_0 \mid \mathsf{MAL}(\boldsymbol{\sigma}_0, \phi_0))}{\frac{1}{2}\Big(\Pr\big(\boldsymbol{\tau}_0|\mathsf{AMP}(\boldsymbol{\sigma}_1, \phi_0, \boldsymbol{\psi}_0)\big)+\Pr\big(\boldsymbol{\tau}_0|\mathsf{AMP}(\boldsymbol{\sigma}_2, \phi_0, \boldsymbol{\psi}_0)\big)\Big)} \\
\approx &\frac{9.9 \times 10^{-5}}{\frac{1}{2}(0.99 + 0.01)} \approx 2 \times 10^{-4}
\end{aligned}
$$

That is, in terms of re-weighting $\boldsymbol{\tau}_0$, MIS-AMP significantly outperforms IS-AMP in Example 5.1.

Having discussed how MIS-AMP can be used to estimate the posterior probability for a single sub-ranking, we now return to the more general problem we study in this paper, and show how MIS can be used to estimate the probability of a union of sub-rankings and a union of patterns.

5.5 MIS-AMP-Lite and MIS-AMP-Adaptive

MIS-AMP can in principle be used for a union of sub-rankings and a union of patterns. However, not unexpectedly, the challenge is that a pattern union G corresoponds to exponentially many sub-rankings, each of which in turn yields multiple modals for MIS (per Section 5.4), and so generating all sub-rankings and then using MIS-AMP for each is intractable. Instead, we develop a method for selecting a subset of subrankings of fixed size d, and ensuring that the corresponding proposal distributions cover the important regions of the posterior. We call this method MIS-AMP-lite.

Suppose that G has z patterns, and that it is equivalent to a union of w sub-rankings. MIS-AMP-lite sorts w sub-rankings in ascending order of their *estimated distance* from the Mallows center $\boldsymbol{\sigma}$, as computed by Algorithm 5. Since the sub-rankings containing modals close to $\boldsymbol{\sigma}$ are desirable, we define the distance between a sub-ranking $\boldsymbol{\psi}$ and $\boldsymbol{\sigma}$ as the minimum Kendall-tau distance between $\boldsymbol{\sigma}$ and a modal contained in $\boldsymbol{\psi}$. But identifying the closest modals is intractable, thus we estimate this distance using a greedy modal r generated in Algorithm 5. Let $dist(\boldsymbol{\psi}, \boldsymbol{\sigma})$ denote the estimated Kendall-tau distance between $\boldsymbol{\psi}$ and $\boldsymbol{\sigma}$. Each sub-ranking $\boldsymbol{\psi}$ represents a component of size proportional to $\phi^{dist(\boldsymbol{\psi},\boldsymbol{\sigma})}$ in the posterior distribution.

Since MIS-AMP-lite prunes many components in the posterior distribution, the algorithm should compensate for this pruning in the final result. Let S denote the sub-rankings in G, and $S^+ \subseteq S$ denote the set of selected sub-rankings. The compensation factor $c_{\boldsymbol{\psi}}$ for sub-ranking pruning is $c_{\boldsymbol{\psi}} = \frac{\sum_{\boldsymbol{\psi} \in S} \phi^{dist(\boldsymbol{\psi},\boldsymbol{\sigma})}}{\sum_{\boldsymbol{\psi} \in S^+} \phi^{dist(\boldsymbol{\psi},\boldsymbol{\sigma})}}$. Intuitively, the compensation factor $c_{\boldsymbol{\psi}}$ captures the portion of the probability space represented by the selected sub-rankings. MIS-AMP-lite also prunes modals, selecting d modals closest to $\boldsymbol{\sigma}$. Let M denote the set of available modals, and $M^+ \subseteq M$ denote the set of selected modals. The compensation factor c_r for modal pruning is defined similarly as for sub-rankings: $c_r = \frac{\sum_{r \in M} \phi^{dist(r,\boldsymbol{\sigma})}}{\sum_{r \in M^+} \phi^{dist(r,\boldsymbol{\sigma})}}$. Let p denote the estimate by MIS-AMP-lite over d proposal distributions without compensation. The final estimate is $\Pr(G|\boldsymbol{\sigma}, \phi) = p \cdot c_{\boldsymbol{\psi}} \cdot c_r$. We experimentally validate the compensation mechanism in Section 6.3, and show that it leads to higher accuracy.

MIS-AMP-lite requires d, the number of proposal distributions, as an input parameter. As an alternative, MIS-AMP-adaptive calls MIS-AMP-lite as a subroutine, and gradually increases the number of proposal distributions in increments of Δd until convergence. We will demonstrate the effectiveness of MIS-AMP-adaptive in Section 6.3.

Algorithm 5 ApproximateDistance

Require: Sub-ranking $\boldsymbol{\psi}$, Mallows center $\boldsymbol{\sigma}$
1: $\boldsymbol{\tau} := \boldsymbol{\psi}$
2: **for** $i = 1, 2, ..., m$ **do**
3: **if** $\sigma_i \notin \boldsymbol{\psi}$ **then**
4: $J = \{j \mid dist(\boldsymbol{\psi}_{i \to j}, \boldsymbol{\sigma}) = \min_{j'=1,...|\boldsymbol{\psi}|} dist(\boldsymbol{\psi}_{i \to j'}, \boldsymbol{\sigma})\}$
5: $\boldsymbol{\tau} := \boldsymbol{\tau}_{i \to j}, j \in J$
6: **return** Kendall-tau$(\boldsymbol{\tau}, \boldsymbol{\sigma})$

6. EXPERIMENTAL EVALUATION

We now present results of an extensive experimental evaluation of exact and approximate solvers over six families of experimental datasets. All solvers are implemented in Python. The general solver uses LTM [5], implemented in Java, as a subroutine. We ran experiments on a 64-bit Ubuntu Linux machine with 48 cores on 4 CPUs of Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, and 512GB of RAM.

6.1 Datasets

In our experimental evaluation we use two real datasets — **MovieLens** and **CrowdRank**, and four synthetic benchmarks — **Polls**, and **Benchmarks A**, **B**, and **C**.

Benchmark-A has 33 pattern unions over the model $\mathsf{MAL}(\langle \sigma_1, \ldots, \sigma_m \rangle, 0.1)$. Each union consists of 3 bipartite patterns $\{A \succ C, A \succ D, B \succ D\}$. Items with labels C and D tend to have higher ranks than items with A and B. As a result, some pattern unions have low probabilities, allowing us to test the accuracy of approximate solvers.

Benchmark-B is a set of pattern unions with varying number of patterns, labels per pattern, and items per label. Within a pattern union, all patterns share the same edges that correspond to random partial order of labels. The number of items m is among $\{20, 50, 100, 200\}$, and Mallows $\phi = 0.1$. The number of patterns per union is 1, 2, or 3. The number of labels per pattern is 3, 4, or 5. The number of items per label is 3, 5, or 7. This benchmark tests the scalability of approximate solvers.

Benchmark-C is a set of bipartite pattern unions with varying number of patterns, labels per pattern, and items per label. The patterns within the same union share the same edges that are random bipartite directed graphs of labels. The number of items m is among $\{10, 12, 14, 16\}$, and Mallows $\phi = 0.1$. The number of patterns per union is 1, 2, or 3. The number of labels per pattern is among 2, 3, or 4. The number of items per label is 1, 3, or 5.

Polls is a synthetic database inspired by the 2016 US presidential election. The data is generated in the way of [5], with database schema as in Figure 1. The tuples in **Candidates** and **Voters**, and the values in each tuple are generated independently. Attributes party and sex have cardinality 2, geographic region cardinality 6, edu and age cardinality 6 (10-year brackets). We generate 1000 voters falling into 72 demographic groups. For each group, we generate 3 random reference rankings and 3 ϕ values $\{0.2, 0.5, 0.8\}$ to construct 9 distinct Mallows models. Each voter is randomly assigned a Mallows from her group, and a random poll date from two dates, which instantiates the relation **Polls**.

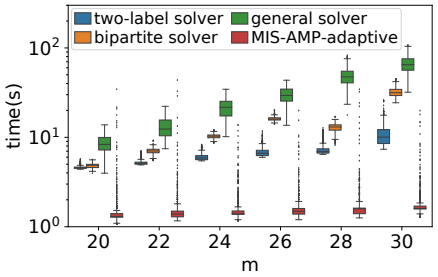
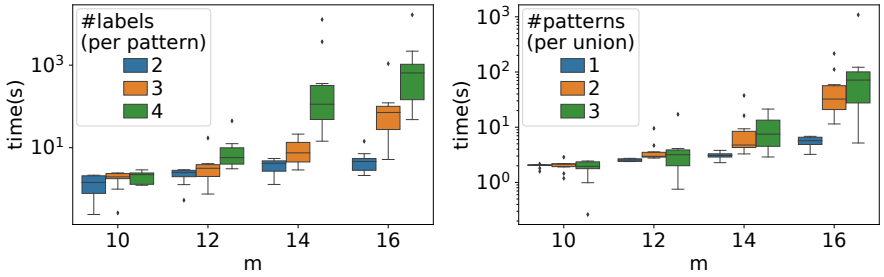


Figure 4: Evaluating a two-label query over **Polls** to compare performance of exact solvers and of MIS-AMP-adaptive.



(a) 3 patterns/union, 3 items/label **(b)** 3 labels/pattern, 3 items/label

Figure 5: Scalability of bipartite solver over **Benchmark-C**

MovieLens is a dataset of movie ratings from GroupLens (`www.grouplens.org`). In line with previous works [20, 5], we use the 200 (out of around 3900) most frequently rated movies and ratings from 5980 users who rated at least one of these movies. We learned a mixture of 16 Mallows models using a publicly available tool [27]. We store movie information in a relation M(id, title, year, genre).

CrowdRank is a real dataset of movie rankings of 50 Human Intelligence Tasks (HITs) collected on Amazon Mechanical Turk [28]. Each HIT provides 20 movies for 100 workers to rank. Then a mixture of Mallows is mined for each HIT with a publicly-available tools [27]. We selected a HIT with seven Mallows models. CrowdRank also includes worker demographics. We used a publicly available tool [24] to generate 200,000 synthetic user profiles statistically similar to the original 100 workers, with the Mallows model among the attributes.

6.2 Performance of Exact Solvers

In our first experiment, we highlight the relative performance of three exact solvers (Section 4) and the approximate solver MIS-AMP-adaptive (Section 5) over **Polls** with 20 to 30 candidates, for a Boolean CQ that all solvers can handle:

$$Q() \leftarrow P(\_,\_;l;r), C(l,p,\texttt{M},\_,\_,\_), C(r,p,\texttt{F},\_,\_,\_);$$

Q asks whether any session prefers a male candidate to a female candidate from the same party p.

Figure 4 compares the running times. Among all solvers, MIS-AMP-adaptive is the most scalable, although, as indicated by the presence of outliers, the running time of this sampling-based method varies significantly due to randomness. Among the exact solvers, the two-label solver is faster than the bipartite solver, which is in turn faster than the general solver. Importantly, MIS-AMP-adaptive is both scalable and accurate for this particular query: 77% of the instances have relative error under 1%, and 93% have relative error 10%. The highest relative error is 63%.

Next, we study the scalability of the bipartite solver over **Benchmark-C**. Recall that the complexity of bipartite solver is $O(m^{qz})$ where m is the number of items in RIM model, q is the number of labels per pattern, and z is the number of patterns per union. The qz is the total number of labels in a pattern union, which is a key parameter for bipartite solver.

Figure 5a shows the running time of bipartite solver with regards to the number of items m and number of labels per pattern, with number of patterns in the union and number of items per label both fixed at 3. The running time increases very fast with both parameters. Similarly, Figure 5b shows the running time of bipartite solver with regards to the number of items in RIM model and number of labels per pattern, with number of patterns in union and number of items per label both fixed to be 3. The running time increases very fast with both parameters. Nonetheless, bipartite solver is practical for lower values of m.

In the next experiment, we evaluate the performance of the general solver over **Benchmark-A**, where each pattern union G has 3 patterns: $G = g_1 \cup g_2 \cup g_3$. The solver applies inclusion-exclusion principle to generate pattern conjunctions as follows:

$$G = g_1 + g_2 + g_3 - (g_1 \wedge g_2) - (g_1 \wedge g_3) - (g_2 \wedge g_3) + (g_1 \wedge g_2 \wedge g_3)$$

That is, G is decomposed into 7 patterns, and LTM is called to compute the probability for each of them. Figure 6 presents the running time of LTM as a function of the number of patterns in a conjunction, showing an exponential increase in running time.

Next, we test performance of the **top-k optimization** on **Polls** with 16 candidates. The query is the following. (Note that it contains a self-join.)

$$\begin{aligned} Q() \leftarrow & P(\_, date; c_1; c_2), P(\_, date; c_1; c_3), P(\_, date; c_1; c_4), \\ & C(c_1, p, \_, \_, \_, \texttt{NE}), C(c_2, p, \_, \_, \_, \texttt{MW}), date = \texttt{"5/5"}, \\ & C(c_3, \_, \_, age, \_, \texttt{NE}), C(c_4, \_, \texttt{M}, \_, \texttt{BA}, \_), age = 50; \end{aligned}$$

Figure 7 displays the running times of evaluating this query under $k = [1, 10, 100]$. Three tallest bars represent the simple strategy of calculating all sessions. The lower bars with 2 colors represent top-k optimization. The "1-edge" label means calculating upper bounds of all sessions by selecting only one edge from each pattern. The "full" label below "1-edge" is the amount of time spent on evaluating exact probabilities of sessions in descending order of their upper bounds until there are k sessions having probabilities higher the probabilities or upper bounds of rest sessions. The "2-edge" label means selecting 2 edges for more accurate upper bounds. As a result, the "full" label below "2-edge" means fewer sessions to calculate. In Figure 7, applying "1-edge" and "2-edge" speeds up the evaluation of $k = 1$ by 5.2 times and 8.2 times, respectively. Even for $k = 100$, the speedup of applying "1-edge" and "2-edge" reaches 1.6 and 2.1, respectively.

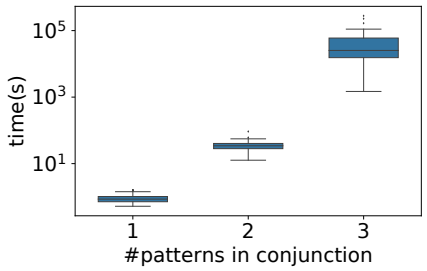


Figure 6: General solver running time increases exponentially with # patterns in conjunction for **Benchmark-A**.

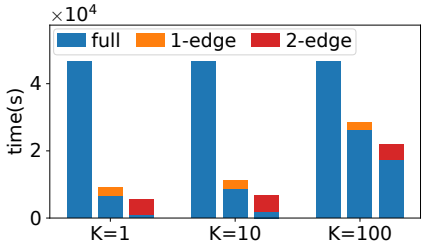


Figure 7: The top-k optimization works well over **Polls**. The tallest "full" bars are baseline. The "1-edge" and "2-edge" bars first quickly compute upper bounds of all sessions.

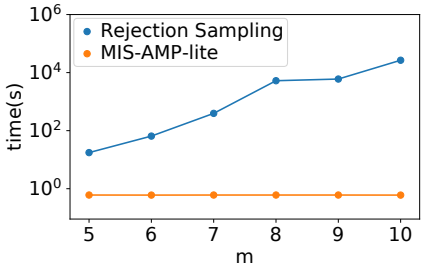
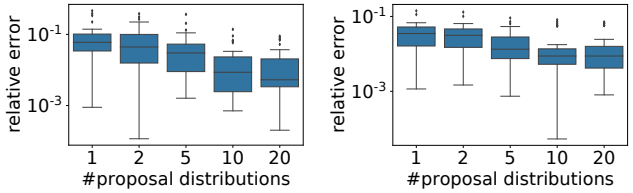
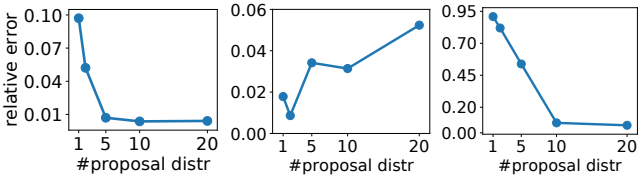


Figure 8: RS does not scale as well as MIS-AMP-lite for query $\sigma_m \succ \sigma_1$ over $\mathsf{MAL}(\langle \sigma_1, \ldots, \sigma_m \rangle, 0.1)$.



(a) Benchmark-A **(b) Benchmark-C**, 3 patterns/union, 3 labels/pattern, 3 items/label



(a) typical **(b)** atypical **(c)** no comp. for **(b)**

Figure 10: MIS-AMP-lite over **Benchmark-A**.

Figure 9: Multi-proposal distributions improve accuracy.

In summary, all exact solvers have exponential complexity with query size. The two-label solver is the fastest, while the bipartite solver is also efficient and can be used also for two-label queries as a special case. These two solvers are also effective in scope of the top-k optimization.

6.3 Performance of Approximate Solvers

Rejection Sampling is inefficient for rare events. We constructed a simple low-probability query $\sigma_m \succ \sigma_1$ for $\mathsf{MAL}(\boldsymbol{\sigma}, 0.1)$, where $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$. When increasing m, the $\Pr(\sigma_m \succ \sigma_1 | \boldsymbol{\sigma}, 0.1)$ decreases exponentially, and RS needs $\mathrm{EXP}(m)$ samples for convergence. In this experiment, we generate 6 Mallows models with $m \in \{5, 6, 7, 8, 9, 10\}$. For each Mallows, we run RS and MIS-AMP-lite 10 times. The exact values of $\Pr(\sigma_m \succ \sigma_1 | \boldsymbol{\sigma}, 0.1)$ are pre-calculated. RS stops running when the estimated probability is within 1% relative error. (Note that this is an optimistic stopping condition for RS, since the algorithm would not yet be able to determine that it converged.) MIS-AMP-lite is set to have only 1 proposal distribution. Figure 8 shows that RS running time increases exponentially with m, while MIS-AMP-lite is much more scalable.

MIS-AMP-lite over Benchmark-A, Benchmark-C. The number of proposal distributions is a critical parameter for MIS-AMP-lite. In this experiment, MIS-AMP-lite is executed with 1, 2, 5, 10, 20 proposal distributions.

Figure 9 gives the distributions of relative errors of MIS-AMP-lite as a function of the number of proposal distributions on **Benchmark-A** and **Benchmark-C** with the number of patterns in union, number of labels per pattern, and number of items per label fixed to be 3. Accuracy improves as the number of proposal distributions increases, and plateaus at around 20 distributions. Overall, MIS-AMP-lite shows low relative error.

Figure 10a complements these cumulative results, showing accuracy of MIS-AMP-lite on a specific instance, where 10 distributions is a good choice. Further, we investigated an atypical instance in Figure 10b. Its relative error was reduced mainly by the compensation, and adding proposal distributions kept increasing accuracy after turning off compensation, as shown in Figure 10c.

MIS-AMP-lite over Benchmark-C. To test the effectiveness of compensation systematically, we ran MIS-AMP-lite with one proposal distribution over **Benchmark-C**. Figure 11 shows that the accuracy of most instances improved by compensation (blue dots under the red line), especially those near the lower right corner, corresponding to instances where relative error was very high (close to 100%) before compensaiton, and was reduced dramatically by applying compensation.

MIS-AMP-adaptive over Benchmark-B. MIS-AMP-adaptive has two stages, proposal distribution construction and sampling. Figure 12a shows the overhead due to proposal distribution construction, fixing 100 items in Mallows model and 3 patterns in union. As expected, the overhead increases sharply with the number of labels, especially when there are many items per label. But once proposal distributions are constructed, sampling converges quickly.

Figure 12b shows the sampling time, fixing 2 patterns in union and 5 items per label. The sampling time increases only moderately with the number of items in Mallows model, and the query size (number of labels) doesn't have significant impact on sampling time. Note that due to the random-

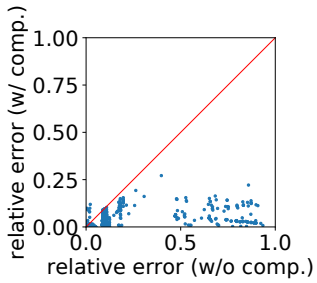
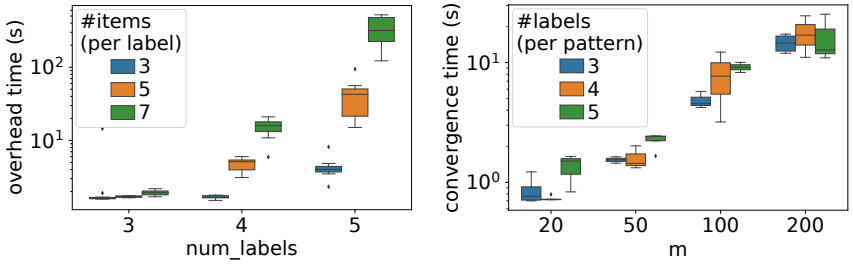


Figure 11: Compensation of MIS-AMP-lite improves the accuracy of estimation on **Benchmark-C**.



(a) Overhead time **(b)** Convergence time

Figure 12: Scalability of MIS-AMP-adaptive over **Benchmark-B**.

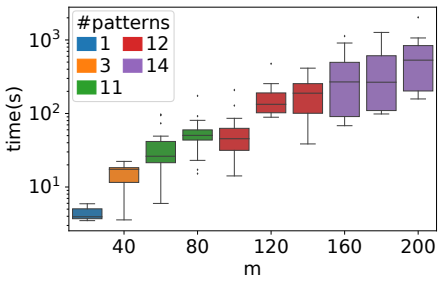


Figure 13: MIS-AMP-adaptive runtime over **MovieLens**.

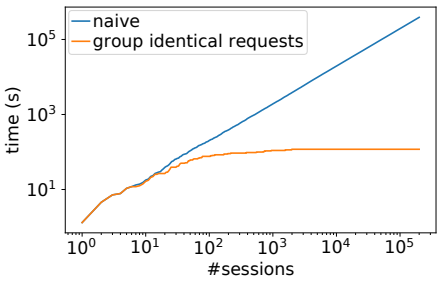


Figure 14: Scalability over 200K sessions in **CrowdRank**.

ness of sampling procedure, here we repeated the sampling 3 times and select the median value to plot in figure.

MIS-AMP-adaptive over MovieLens. We vary the number of movies m from 40 to 200 to test scalability with:

$$Q() \leftarrow P(\_; 223; 111), P(\_; x; 111), P(\_; x; y),$$
$$M(x, \_, year_1, genre), year_1 >= 1990,$$
$$M(y, \_, year_2, genre), year_2 < 1990;$$

The query asks whether the movie *Clerks* (id 223) is preferred to *Taxi Driver* (id 111), and whether some movie released after 1990 is preferred to a movie before 1990 and also to *Taxi Driver*. Figure 13 shows the running time of MIS-AMP-adaptive over the sessions. Note that when number of movies m increases, there are more genres in the dataset, yielding more patterns in the pattern union.

In summary, the approximate solvers are scalable and accurate. Multiple proposal distributions help them reach the important regions of the target distribution.

6.4 Scalability over Sessions

When evaluating a query, multiple sessions may share the same RIM model and pattern union. RIM-PPD groups identical requests before invoking inference solvers, realizing performance gains. We illustrate scalability in the number of sessions using a query that asks whether a user prefers a movie with the leading actor of their gender to a movie with the leading actor around their age. Focus on short (< 90 min) movies that are preferred to some Thriller.

$$Q() \leftarrow P(v; m_1; m_2), P(v; m_2; m_3), V(v, sex, age),$$
$$M(m_1, \_, sex, \_, \texttt{short}), M(m_2, \_, \_, age, \texttt{short}),$$
$$M(m_3, \texttt{Thriller}, \_, \_);$$

Figure 14 shows the results of running the general solver over **CrowdRank** with 200,000 sessions. The naive implementation runs in linear time in the number of sessions, while grouping requests quickly converged after 118 seconds.

7. CONCLUSIONS

In this work, we developed methods for answering computationally hard queries over probabilistic preferences, where we enable users to express preferences over item attributes in the form of values or variables. To evaluate this class of hard queries, we developed a general solver that applies inclusion-exclusion principle. Then, we took the optimization opportunities in two-label patterns and bipartite patterns, significantly reducing query evaluation time. Scalability was further improved by approximate solvers, where we studied the posterior distributions of pattern unions over Mallows models, and applied Multiple Importance Sampling to effectively estimate the Mallows posterior probability.

Future directions include supporting additional aggregation queries and incorporating probabilistic preference models beyond RIM [9, 19].

8. ACKNOWLEDGMENTS

The work of Julia Stoyanovich was supported in part by NSF Grants No. 1916647, 1926250, and 1934464. The work of Benny Kimelfeld was supported in part by BSF Grant No. 2017753 and ISF Grant No. 1295/15.

9. REFERENCES

[1] P. Awasthi, A. Blum, O. Sheffet, and A. Vijayaraghavan. Learning mixtures of ranking models. In *NIPS*, pages 2609–2617, 2014.

[2] S. Balakrishnan and S. Chopra. Two of a kind or the ratings game? adaptive pairwise preferences and latent factor models. *Frontiers of Computer Science*, 6(2):197–208, 2012.

[3] F. Brandenburg, A. Gleißner, and A. Hofmeier. Comparing and aggregating partial orders with kendall tau distances. In *WALCOM*, pages 88–99, 2012.

[4] L. M. Busse, P. Orbanz, and J. M. Buhmann. Cluster analysis of heterogeneous rank data. In *ICML*, pages 113–120, 2007.

[5] U. Cohen, B. Kenig, H. Ping, B. Kimelfeld, and J. Stoyanovich. A query engine for probabilistic preferences. In *SIGMOD*, pages 1509–1524, 2018.

[6] P. Diaconis. A generalization of spectral analysis with applications to ranked data. *Annals of Statistics*, 17(3):949–979, 1989.

[7] W. Ding, P. Ishwar, and V. Saligrama. Learning mixed membership mallows models from pairwise comparisons. *CoRR*, abs/1504.00757, 2015.

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

[9] M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society B*, 43:359–369, 1986.

[10] I. C. Gormley and T. B. Murphy. A latent space model for rank data. In *ICML*, 2006.

[11] I. C. Gormley and T. B. Murphy. A mixture of experts model for rank data with applications in election studies. *The Annals of Applied Statistics*, 2(4):1452–1477, 12 2008.

[12] J. Huang, A. Kapoor, and C. Guestrin. Riffled independence for efficient inference with partial rankings. *J. Artif. Intell. Res.*, 44:491–532, 2012.

[13] H. Kahn. Random sampling (monte carlo) techniques in neutron attenuation problems–i. *Nucleonics*, 6(5):27–passim, 1950.

[14] H. Kahn. Random sampling (monte carlo) techniques in neutron attenuation problems–ii. *Nucleonics*, 6(6):60–65, 1950.

[15] T. Kamishima and S. Akaho. Supervised ordering by regression combined with thurstone's model. *Artif. Intell. Rev.*, 25(3):231–246, 2006.

[16] B. Kenig, L. Ilijasic, H. Ping, B. Kimelfeld, and J. Stoyanovich. Probabilistic inference over repeated insertion models. In *AAAI*, pages 1897–1904, 2018.

[17] B. Kenig, B. Kimelfeld, H. Ping, and J. Stoyanovich. Querying probabilistic preferences in databases. In *PODS*, pages 21–36, 2017.

[18] G. Lebanon and J. D. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *ICML*, pages 363–370, 2002.

[19] G. Lebanon and Y. Mao. Non-parametric modeling of partially ranked data. In *NIPS*, pages 857–864, 2007.

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

[21] C. L. Mallows. Non-null ranking models. *Biometrika*, 44:114–130, 1957.

[22] J. I. Marden. *Analyzing and modeling rank data*. CRC Press, 1995.

[23] G. McElroy and M. Marsh. Candidate gender and voter choice: Analysis from a multimember preferential voting system. *Political Research Quarterly*, 63(4):822–833, 2010.

[24] H. Ping, J. Stoyanovich, and B. Howe. Datasynthesizer: Privacy-preserving synthetic datasets. In *SSDBM*, pages 42:1–42:5, 2017.

[25] H. Ping, J. Stoyanovich, and B. Kimelfeld. Supporting hard queries over probabilistic preferences, 2020. arXiv:2003.06984 [cs.DB].

[26] A. D. Sarma, A. D. Sarma, S. Gollapudi, and R. Panigrahy. Ranking mechanisms in twitter-like forums. In *WSDM*, pages 21–30, 2010.

[27] J. Stoyanovich, L. Ilijasic, and H. Ping. Workload-driven learning of mallows mixtures with pairwise preference data. In *WebDB*, pages 1–6, 2016.

[28] J. Stoyanovich, M. Jacob, and X. Gong. Analyzing crowd rankings. In *WebDB*, pages 41–47, 2015.

[29] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[30] E. Veach and L. J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *SIGGRAPH*, pages 419–428, 1995.