Efficient Rank Aggregation via Lehmer Codes

Pan Li
University of Illinois
Urbana-Champaign

Arya Mazumdar
University of Massachusetts Amherst

Olgica Milenkovic
University of Illinois
Urbana-Champaign

Abstract

We propose a novel rank aggregation method based on converting permutations into their corresponding Lehmer codes or other subdiagonal images. Lehmer codes, also known as inversion vectors, are vector representations of permutations in which each coordinate can take values not restricted by the values of other coordinates. This transformation allows for decoupling of the coordinates and for performing aggregation via simple scalar median or mode computations. We present simulation results illustrating the performance of this completely parallelizable approach and analytically prove that both the mode and median aggregation procedure recover the correct centroid aggregate with small sample complexity when the permutations are drawn according to the well-known Mallows models. The proposed Lehmer code approach may also be used on partial rankings, with similar performance guarantees.

1 Introduction

Rank aggregation is a family of problems concerned with fusing disparate ranking information, and it arises in application areas as diverse as social choice, metasearch, natural language processing, bioinformatics, and information retrieval [1, 2, 3]. The observed rankings are either linear orders (permutations) or partial (element-tied) rankings[1]. Sometimes, rankings are assumed to be of the form of a set of pairwise comparisons [4, 5]. Note that, many massive ordinal datasets arise from *ratings*, rather than actual comparisons. Rank aggregation, rather than averaging of ratings, is justified due to the fact that most raters have different rating "scales". As an example, the rating three of one user may indicate that the user liked the item, while the rating three by another user may indicate that the user disliked the item. Hence, actual preferences can only be deduced using ranked ratings.

In rank aggregation, the task at hand is to find a ranking that is at the smallest cumulative distance from a given set of rankings. Here, the cumulative distance from a set equals the sum of the distances from each element of the set, and the most frequently used distance measure for the case of permutations is the Kendall τ distance. For the case of partial rankings, the distance of choice is the Kemeny distance [6]. The Kendall τ distance between two permutations equals the smallest number of adjacent transpositions needed to convert one permutation into the other. The Kemeny distance contains an additional weighted correction term that accounts for ties in the rankings.

It is well known that for a wide range of distance functions, learning the underlying models and aggregating rankings is computationally hard [7, 8]. Nevertheless, for the case when the distance measure is the Kendall τ distance, a number of approximation algorithms have been developed that offer various trade-offs between quality of aggregation and computational complexity [9, 10]. The techniques used for aggregating permutations in a given set include randomly choosing a permutation from the set (PICK-A-PERM), pivoting via random selections of elements and divide-and-conquer approaches (FAS-PIVOT), Markov chain methods akin to PageRank, and minimum weight graph matching methods exploiting the fact that the Kendall τ distance is well-approximated by the Spearman footrule distance (SM) [11]. Methods with provable performance guarantees – PICK-A-PERM, FAS-

[1] In the mathematics literature, partial rankings are also referred to as weak orders, while the term partial order is used to describe orders of subsets of elements of a ground set. We nevertheless use the term partial ranking to denote orders with ties, as this terminology is more widely adopted by machine learning community.

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

PIVOT, and SM – give a 2-approximation for the objective function, although combinations thereof are known to improve the constant to 11/7 or 4/3 [10]. There also exists a polynomial time approximation scheme (PTAS) for the aggregation problem [12].

Unfortunately, most of these known approximate rank aggregation algorithms have high complexity for use with massive datasets and may not be implemented in a parallel fashion. Furthermore, they do not easily extend to partial rankings. In many cases, a performance analysis on probabilistic models [13] such as the Plackett-Luce model [14] or the Mallows model [15, 16], is intractable.

In this paper, we propose a new approach to the problem of rank aggregation that uses a combinatorial transform, the Lehmer code (LC). The gist of the approach is to convert permutations into their Lehmer code representations, in which each coordinate takes values independently from other coordinates. Aggregation over the Lehmer code domain reduces to computing the median or mode of a bounded set of numbers, which can be done in linear time. Furthermore, efficient conversion algorithms between permutations and Lehmer codes – also running in linear time – are known, making the overall complexity of the parallel implementation of the scheme $O(m+n)$, where m denotes the number of permutations to be aggregated, and n denotes the length (size) of the permutations. To illustrate the performance of the Lehmer code aggregators (LCAs) on permutations, we carry out simulation studies showing that the algorithms perform comparably with the best known methods for approximate aggregation, but at a significantly lower computational cost. We then proceed to establish a number of theoretical performance guarantees for the LCA algorithms: In particular, we consider the Mallows model with the Kendall τ distance for permutations and Kemeny distance for partial rankings where ties are allowed. We show that the centroid permutation of the model or a derivative thereof may be recovered from $O(\log\, n)$ samples from the corresponding distribution with high probability.

The paper is organized as follows. Section 2 contains the mathematical preliminaries and the definitions used throughout the paper. Section 3 introduces our new aggregation methods for two types of rankings, while Section 4 describes our analysis pertaining to the Mallows and generalized Mallows models. Section 5 contains illustrative simulation results comparing the performance of the LC aggregators to that of other known aggregation methods, both on simulated and real ranking data. A number of technical results, namely detailed proofs of theorems and lemmas, can be found in the supplementary material.

2 Mathematical Preliminaries

Let S denote a set of n elements, which without loss of generality we assume to be equal to $[n] \equiv \{1, 2, \ldots, n\}$. A ranking is an ordering of a subset of elements Q of $[n]$ according to a predefined rule. When $Q = [n]$, we refer to the order as a permutation (full ranking). When a ranking includes ties, we refer to it as a partial ranking (weak or bucket order). Partial rankings may be used to complete rankings of subsets of element in $[n]$ in a number of different ways [17], one being to tie all unranked elements at the last position.

Rigorously, a permutation is a bijection $\sigma : [n] \to [n]$, and the set of permutations over $[n]$ forms the symmetric group of order $n!$ denoted by $\mathbb{S}_n$. For any $\sigma \in \mathbb{S}_n$ and $x \in [n]$, $\sigma(x)$ denotes the rank (position) of the element x in σ. We say that x is ranked higher than y (ranked lower than y) iff $\sigma(x) < \sigma(y)$ ($\sigma(x) > \sigma(y)$). The inverse of a permutation σ is denoted by $\sigma^{-1} : [n] \to [n]$. Clearly, $\sigma^{-1}(i)$ represents the element ranked at position i in σ. We define *the projection of a permutation* σ over a subset of elements $Q \subseteq [n]$, denoted by $\sigma_Q : Q \to [|Q|]$, as an ordering of elements in Q such that $x, y \in Q$, $\sigma_Q(x) > \sigma_Q(y)$ iff $\sigma(x) > \sigma(y)$. As an example, the projection of $\sigma = (2, 1, 4, 5, 3, 6)$ over $Q = \{1, 3, 5, 6\}$ equals $\sigma_Q = (1, 3, 2, 4)$, since $\sigma(1) < \sigma(5) < \sigma(3) < \sigma(6)$. As can be seen, $\sigma_Q(x)$ equals the rank of element $x \in Q$ in σ.

We use a similar set of definitions for partial rankings [17]. A partial ranking σ is also defined as a mapping $[n] \to [n]$. In contrast to permutations, where the mapping is a bijection, the mapping in partial ranking allows for ties, i.e., there may exist two elements $x \neq y$ such that $\sigma(x) = \sigma(y)$. A partial ranking is often represented using buckets, and is in this context referred to as a *bucket order* [17]. In a bucket order, the elements of the set $[n]$ are partitioned into a number of subsets, or buckets, $\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_t$. We let $\sigma(x)$ denote the index of the bucket containing the element x in σ, so the element x is assigned to bucket $\mathcal{B}_{\sigma(x)}$. Two elements x, y lie in the same bucket if and only if they are tied in σ. We may also define a projection of a partial ranking σ over a subset of elements $Q \subset [n]$, denoted by σ_Q, so that for $x, y \in Q$, $\sigma_Q(x) > \sigma_Q(y)$ iff $\sigma(x) > \sigma(y)$ and $\sigma_Q(x) = \sigma_Q(y)$ iff $\sigma(x) = \sigma(y)$. For a given partial ranking σ, we use $\mathcal{B}_1(\sigma), \mathcal{B}_2(\sigma), ..., \mathcal{B}_t(\sigma)$ to denote its corresponding buckets. In addition, we define $r_{k(\sigma)} \triangleq \sum_{j=1}^{k} |\mathcal{B}_j(\sigma)|$ and $l_{k(\sigma)} \triangleq \sum_{j=1}^{k-1} |\mathcal{B}_j(\sigma)| + 1$. Based on the previous discussion, $r_{\sigma(x)}(\sigma) - l_{\sigma(x)}(\sigma) + 1 = |\mathcal{B}_{\sigma(x)}(\sigma)|$ (the number of elements that are in the bucket containing x). When referring to the bucket for a certain element x, we use $\mathcal{B}_{\sigma(x)}$, $r_{\sigma(x)}$, $l_{\sigma(x)}$ whenever no confusion arises. Note that if we arbitrarily break ties in σ to create a permutation σ', then $l_{\sigma(x)} \leq \sigma'(x) \leq r_{\sigma(x)}$;

clearly, if σ is a permutation, $l_{\sigma(i)} = \sigma(i) = r_{\sigma(i)}$.

A number of distance functions between permutations are known from the social choice, learning and discrete mathematics literature [11]. One distance function of interest is based on transpositions: A transposition (a, b) is a swap of elements at positions a and b, $a \neq b$. If $|a - b| = 1$, the transposition is referred to as an adjacent transposition. It is well known that transpositions (adjacent transpositions) generate $\mathbb{S}_n$, i.e., any permutation $\pi \in \mathbb{S}_n$ can be converted into another permutation $\sigma \in \mathbb{S}_n$ through a sequence of transpositions (adjacent transpositions) [18]. The smallest number of adjacent transpositions needed to convert a permutation π into another permutation σ is known as the Kendall τ distance between π and σ, and is denoted by $d_\tau(\pi, \sigma)$. Alternatively, the Kendall τ distance between two permutations π and σ over $[n]$ equals the number of mutual inversions between the elements of the two permutations:

$$d_\tau(\sigma, \pi) = |\{(x, y) : \pi(x) > \pi(y), \sigma(x) < \sigma(y)\}|. \quad (1)$$

Another distance measure, that does not rely on transpositions, is the Spearman footrule, defined as

$$d_S(\sigma, \pi) = \sum_{x \in [n]} |\sigma(x) - \pi(x)|.$$

A well known result by Diaconis and Graham [11] asserts that $d_\tau(\pi, \sigma) \leq d_S(\pi, \sigma) \leq 2d_\tau(\pi, \sigma)$.

One may also define an extension of the Kendall τ distance for the case of two partial rankings π and σ over the set $[n]$, known as the Kemeny distance:

$$\begin{aligned} d_K(\pi, \sigma) =& |\{(x, y) : \pi(x) > \sigma(y), \pi(x) < \sigma(y)\}| \\ &+ \frac{1}{2}|\{(x, y) : \pi(x) = \pi(y), \sigma(x) > \sigma(y), \\ &\text{or } \pi(x) > \pi(y), \sigma(x) = \sigma(y), \}|. \end{aligned} \quad (2)$$

The Kemeny distance includes a component equal to the Kendal τ distance between the linear chains in the partial rankings, and another, scaled component that characterizes the distance of tied pairs of elements [17]. The Spearman footrule distance may also be defined to apply to partial rankings [17], and it equals the sum of the absolute differences between "positions" of elements in the partial rankings. Here, the position of an element x in a partial ranking σ is defined as

$$\text{pos}_\sigma(x) \triangleq \sum_{j=1}^{\sigma(x)-1} |\mathcal{B}_j(\sigma)| + \frac{|\mathcal{B}_{\sigma(x)}(\sigma)| + 1}{2}.$$

The above defined Spearman distance is a 2-approximation for the Kemeny distance between two partial rankings [17].

A permutation $\sigma = (\sigma(1), \ldots, \sigma(n)) \in \mathbb{S}_n$ may be uniquely represented via its *Lehmer code* (also called the *inversion vector*), i.e. a word of the form

$$\boldsymbol{c}_\sigma \in \mathcal{C}_n \triangleq \{0\} \times [0, 1] \times [0, 2] \times \cdots \times [0, n-1],$$

where for $i = 1, \ldots, n$,

$$\boldsymbol{c}_\sigma(x) = |\{y : y < x, \sigma(y) > \sigma(x)\}|, \quad (3)$$

and for integers $a \leq b$, $[a, b] \equiv [a, a+1, \ldots, b]$. By default, $\boldsymbol{c}_\sigma(1) = 0$, and is typically omitted. For instance, we have

e	1	2	3	4	5	6	7	8	9
σ	2	1	4	5	7	3	6	9	8
$\boldsymbol{c}_\sigma$	0	1	0	0	0	3	1	0	1

It is well known that the Lehmer code is bijective, and that the encoding and decoding algorithms have linear complexity (n) [19, 20]. Codes with similar properties to the Lehmer codes have been extensively studied under the name of *subdiagonal codes*. An overview of such codes and their relationship to Mahonian statistics on permutations may be found in [21].

We propose next our generalization of Lehmer codes to partial rankings. Recall that the x-th entry in the Lehmer code of a permutation σ is the number of elements with index smaller than x that are ranked lower than x in σ (3). For a partial ranking, in addition to $\boldsymbol{c}_\sigma$, we use another code that takes into account ties according to:

$$\boldsymbol{c}'_\sigma(x) = |\{y \in [n] : y < x, \sigma(y) \geq \sigma(x)\}|. \quad (4)$$

Clearly, $\boldsymbol{c}'_\sigma(x) \geq \boldsymbol{c}_\sigma(x)$ for all $x \in [n]$. It is straightforward to see that using $\boldsymbol{c}_\sigma(x)$ and $\boldsymbol{c}'_\sigma(x)$, one may recover the original partial ranking σ. In fact, we prove next that the linear-time Lehmer encoding and decoding algorithms may be used to encode and decode $\boldsymbol{c}_\sigma$ and $\boldsymbol{c}'_\sigma$ in linear time as well.

Given a partial ranking σ, we may break the ties in each bucket to arrive at a permutation σ' as follows: For $x, y \in S$, if $\sigma(x) = \sigma(y)$,

$$\sigma'(x) < \sigma'(y) \text{ if and only if } x < y. \quad (5)$$

We observe that the entries of the Lehmer codes of σ and σ' satisfy the following relationships for all $i \in [n]$:

$$\begin{aligned} \boldsymbol{c}'_\sigma(x) &= \boldsymbol{c}_{\sigma'}(x) + \text{IN}_x - 1, \\ \boldsymbol{c}_\sigma(x) &= \boldsymbol{c}_{\sigma'}(x), \end{aligned}$$

where $\text{IN}_x = |\{y \in [n] \cap \mathcal{B}_{\sigma(x)} : y \leq x\}|$. An example illustrating these concepts is given below.

e	1	2	3	4	5	6	7	8	9
σ	1	1	2	2	3	1	2	3	3
σ'	1	2	4	5	7	3	6	8	9
$\boldsymbol{c}_{\sigma'}$	0	0	0	0	0	3	1	0	0
IN	1	2	1	2	1	3	3	2	3
$\boldsymbol{c}_{\sigma}$	0	0	0	0	0	3	1	0	0
$\boldsymbol{c}'_{\sigma}$	0	1	0	1	0	5	3	1	2

Note that IN_x, as well as $\boldsymbol{c}_\sigma$ and $\boldsymbol{c}'_\sigma$ may be computed in linear time. The encoding procedure is outlined in Algorithm 1.

Algorithm 1:
Lehmer encoder for partial rankings
Input: a partial ranking σ;
1: Set N to be the number of buckets in σ;
2: Initialize $\text{IN} = (0, 0, ..., 0) \in \mathbb{N}^n$
and $\text{BucketSize} = (0, 0, ..., 0) \in \mathbb{N}^N$;
3: **For** x **from** 1 **to** n **do**
4: $\quad \text{BucketSize}(\sigma(x)) + +$;
5: $\quad \text{IN}(x) \leftarrow \text{BucketSize}(\sigma(x))$;
6: Break ties of σ to get σ' according to (5);
7: $\boldsymbol{c}_{\sigma'} \leftarrow$ Lehmer code of σ';
Output: Output $\boldsymbol{c}_\sigma = \boldsymbol{c}_{\sigma'}$, $\boldsymbol{c}'_\sigma = \boldsymbol{c}_\sigma + \text{IN} - \mathbf{1}$;

3 Aggregation Algorithms

Assume that we have to aggregate a set of m rankings, denoted by $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$, $\sigma_k \in \mathbb{S}_n$, $1 \le k \le m$. Aggregation may be performed via the distance-based Kemeny-Young model, in which one seeks a ranking σ that minimizes the cumulative Kendall τ (Kemeny) distance d_τ (d_K) from the set Σ, formally defined as:

$$D(\Sigma, \sigma) = \sum_{i=1}^{m} d_\tau(\sigma_i, \sigma).$$

Note that when the set Σ comprises permutations only, σ is required to be a permutation; if Σ comprises partial rankings, we allow the output to be either a permutation or a partial ranking.

The LCA procedure under the Kendall τ distance is described in Algorithm 2. Note that each step of the algorithm may be executed in parallel. If no parallelization is used, the first step requires $O(mn)$ time, given that the Lehmer codes may be computed in $O(n)$ time [19, 20]. If parallelization on Σ is used instead, the time reduces to $O(m+n)$. Similarly, without parallelization the second step requires $O(mn)$ time, while coordinate parallelization reduces this time to $O(m)$. This third step requires $O(n)$ computations. Hence, the overall complexity of the algorithm is either $O(mn)$ or $O(m+n)$, depending on parallelization being used or not.

Algorithm 2: The LCA Method (Permutations)
Input: $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$, where $\sigma_i \in \mathbb{S}_n$, $i \in [n]$.
1: Compute the Lehmer codewords $\boldsymbol{c}_{\sigma_j}$ for all $\sigma_j \in \Sigma$.
2: Compute the median/mode of the coordinates:
$\hat{\boldsymbol{c}}(i) = \text{median/mode}\Big(\boldsymbol{c}_{\sigma_1}(i), \boldsymbol{c}_{\sigma_2}(i), \ldots, \boldsymbol{c}_{\sigma_m}(i)\Big)$.
3: Compute $\hat{\sigma}$, the inverse Lehmer code of $\hat{\boldsymbol{c}}$.
Output: Output $\hat{\sigma}$.

For permutations, the aggregation procedure may be viewed as specialized voting: The ranking σ_k casts a vote to rank x at position $x - \boldsymbol{c}_{\sigma_k}(x)$, for the case that only elements $\le x$ are considered (A vote corresponds to some score confined to $[0, 1]$). However, when σ_k is a partial ranking involving ties, the vote should account for all possible placements between $x - \boldsymbol{c}'_\sigma(x)$ and $x - \boldsymbol{c}_\sigma(x)$. More precisely, suppose that the vote cast by σ_k to place element x in position $y \in [x]$ is denoted by $v_{k\to x}(y)$. Then, one should have

$$v_{k\to x}(y) = \begin{cases} 1, & \text{for the mode,} \\ \frac{1}{\boldsymbol{c}'_\sigma(x) - \boldsymbol{c}_\sigma(x) + 1}, & \text{for the median,} \end{cases} \tag{6}$$

if and only if $y \in [x - \boldsymbol{c}'_\sigma(x), x - \boldsymbol{c}_\sigma(x)]$, and zero otherwise. Note that when the mode is used, the "positive votes" are all equal to one, while when the median is used, a vote counts only a fractional value dictated by the length of the "ranking interval".

Next, we use $V_x(y) = \sum_{k=1}^{m} v_{k\to x}(y)$ to denote the total voting score element x received to be ranked at position y. The inverse Lehmer code of the aggregator output $\hat{\sigma}$ is computed as:

$$\text{mode:} \quad \hat{\boldsymbol{c}}(x) = \arg_{y\in[x]} \max V_x(y) - 1, \tag{7}$$

$$\text{median:} \quad \hat{\boldsymbol{c}}(x) = \min\{k : \frac{\sum_{y=1}^{k} V_x(y)}{m} \ge 1/2\} - 1.$$

To compute the values $V_x(y)$ for all $y \in [x]$, the LCA algorithm requires $O(mx)$ time, which yields an overall aggregation complexity of $O(mn^2)$ when no parallelization is used. This complexity is reduced to $O(m+n^2)$ for the parallel implementation. Note that the evaluations of the V functions may be performed in a simple iterative manner provided that the votes $v_{k\to x}(y)$ are positive constants, leading to a reduction in the overall complexity of this step to $O(mn+n^2)$ when no parallelization is used. Relevant details regarding the iterative procedure may be found in Section VII of the supplementary document.

Note that the output $\hat{\sigma}$ of Algorithm 2 is a permutation. To generate a partial ranking that minimizes the Kemeny distance while being consistent[2] with $\hat{\sigma}$,

[2]We say that two partial rankings σ, π are *consistent*

one can use a $O(mn^2+n^3)$-time algorithm outlined in Section VII of the supplementary document. Alternatively, the following simple greedy method always produces practically good partial rankings with $O(mn)$ complexity: Scan the elements in the output permutation from highest ($j=1$) to lowest rank ($j=n-1$) and decide to put $\hat{\sigma}^{-1}(j+1)$ and $\hat{\sigma}^{-1}(j)$ in the same bucket or not based on which of the two choices offers smaller Kemeny distance with respect to the subset $\{\hat{\sigma}^{-1}(1),...,\hat{\sigma}^{-1}(j)\}$.

Discussion. In what follows, we briefly outline the similarities and differences between the LCA method and existing positional as well as InsertionSort based aggregation methods. Positional methods are a class of aggregation algorithms that seek to output a ranking in which the position of each element is "close" to the position of the element in Σ. One example of a positional method is Borda's algorithm, which is known to produce a 5-approximation to the Kemeny-Young problem for permutations [22]. Another method is the Spearman footrule aggregation method which seeks to find a permutation that minimizes the sum of the Spearman footrule distance between the output and each ranking in Σ. As already mentioned, the latter method produces a 2-approximation for the Kendall τ aggregate for both permutations and partial ranking. LCA also falls under the category of positional methods, but the positions on which scoring is performed are highly specialized by the Lehmer code. And although it appears hard to prove worst-case performance guarantees for the method, statistical analysis on particular ranking models shows that it can recover the correct results with small sample complexity. It also offers significant reductions in computational time compared to the Spearman footrule method, which reduces to solving a weighted bipartite matching problem and hence has complexity at least $O(mn^2+n^3)$ [23], or $O(mn)$ when implemented in MapReduce [24].

A related type of aggregation is based on InsertionSort [9, 23]. In each iteration, an element is randomly chosen to be inserted into the sequence containing the already sorted elements. The position of the insertion is selected as follows. Assume that the elements are inserted according to the identity order $e=(1,2,\ldots,n)$ so that at iteration t, element t is chosen to be inserted into some previously constructed ranking over $[t-1]$. Let $S_{t-1}=[t-1]$ and the symbol t is inserted into the ranking over S_{t-1} to arrive at σ_{S_t}, the ranking available after iteration t. If t is inserted between two adjacent elements $\sigma^{-1}_{S_{t-1}}(i-1)$ and $\sigma^{-1}_{S_{t-1}}(i)$, then one should have $\sigma_{S_t}(x)=\sigma_{S_{t-1}}(x)$ when $\sigma_{S_{t-1}}(x)\leq i-1$, $\sigma_{S_t}(x)=\sigma_{S_{t-1}}(x-1)+1$ when $\sigma_{S_{t-1}}(x)\geq i$ and $\sigma_{S_t}(t)=i$. Let $\sigma_{S_t}(t)$ denote the rank assigned to element t over S_t, the choice of which may vary from method to method. The authors of [9] proposed setting $\sigma_{S_t}(t)$ to

$$\max\left\{i\in[t-1]:\sum_{k\in[m]}1_{\sigma_k(t)<\sigma_k(\sigma^{-1}_{S_{t-1}}(i))}<\frac{m}{2}\right\},$$

or t when the above set is empty. This insertion rule does not ensure a constant approximation guarantee in the worst case (It has an expected worst-case performance guarantee of $\Omega(n)$), although it leads to a Locally Kemeny optimal solution.

We next describe how the LCA method may be viewed as an InsertionSort method with a special choice of $\sigma_{S_t}(t)$. Consider the permutation LCA method of Algorithm 2, and focus on estimating the t-th coordinate of the Lehmer code $\hat{\boldsymbol{c}}(t)$ (step 2) and the inverse Lehmer code via insertion (step 3) simultaneously. Once $\hat{\boldsymbol{c}}(t)$ is generated, it's corresponding inverse Lehmer transform may be viewed as the operation of placing the element t at position $(t-\hat{\boldsymbol{c}}(t))$ over S_t. In other words, inverting the incomplete ranking reduces to setting $\sigma_{S_t}(t)=(t-\hat{\boldsymbol{c}}(t))$, where $\sigma_{S_t}(t)$ essentially equals the mode or median of the positions of t in the rankings of Σ, projected onto S_t. The same is true of partial rankings, with the only difference being that the selection of $\sigma_{S_t}(t)$ has to be changed because of ties between elements.

4 Analysis of the Mallows Model

We provide next a theoretical performance analysis of the LCA algorithm under the assumption that the rankings are generated according to the Mallows and generalized Mallows Model. In the Mallows model $\mathrm{MM}(\sigma_0,\phi)$ with parameters σ_0 and ϕ, σ_0 denotes the centroid ranking and $\phi\in(0,1]$ determines the variance of the ranking with respect to σ_0. The probability of a permutation σ is proportional to $\phi^{d_\tau(\sigma_0,\sigma)}$. For partial rankings, we assume that the samples are generated from a generalized Mallows Model (GMM) whose centroid is allowed to be a partial ranking and where the distance is the Kemeny d_k, rather than the Kendall τ distance d_τ.

Our analysis is based on the premise that given a sufficiently large number of samples (permutations), one expects the ranking obtained by a good aggregation algorithm to be equal to the centroid σ_0 with high probability. Alternative methods to analytically test the quality of an aggregation algorithm are to perform a worst-case analysis, which for the LCA method appears hard, or to perform a simulation-based analysis

if for any two elements x, y, $\sigma(x)<\sigma(y)$ if and only if $\pi(x)\leq\pi(y)$ and vise versa.

which produces a comparison of the objective function values for the Kemeny-Young problem given different aggregation methods. We report on the latter study in the section to follow.

To ease the notational burden, we henceforth use $\phi_{s:t} \triangleq \sum_{k=s}^{t} \phi^k$ in all subsequent results and derivations. Detailed proofs are relegated to the supplementary material. One of our main theoretical result is the following.

Theorem 4.1. Assume that $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$, where $\sigma_k \overset{\text{i.i.d}}{\sim} \text{MM}(\sigma_0, \phi)$, $k \in [m]$, are m i.i.d. samples of the given Mallows model. If $\phi+\phi^2 < 1+\phi^n$ and $m \geq c \log \frac{n^2}{2\delta}$ with $c = \frac{2(1+q)^2}{(1-q)^4}$ and $q = \frac{\phi_{1:n-1}}{1+\phi_{3:n}}$, then the output ranking of Algorithm 2 under the mode rule equals σ_0 with probability at least $1-\delta$.

The idea behind the proof is to view the LCA procedure as an InsertionSort method, in which the probability of the event that the selected position is incorrect with respect to σ_0 is very small for sufficiently large m. Based on the lemma that follows (Lemma 4.2), one may show that if ϕ satisfies $\phi + \phi^2 < 1 + \phi^n$, the most probable position of an element in a ranking $\sigma \sim \text{MM}(\sigma_0, \phi)$ corresponds to its rank in the centroid σ_0. Given enough samples, one can estimate the rank of an element in the centroid by directly using the mode of the rank of the element in the drawn samples.

Lemma 4.2. Let $\sigma \sim \text{MM}(\sigma_0, \phi)$. Consider an element u. Then, the following two statements describe the distribution of $\sigma(u)$:

1) $\dfrac{\mathbb{P}[\sigma(u) = j+1]}{\mathbb{P}[\sigma(u) = j]} \in [\phi, \dfrac{\phi_{1:n-1}}{1+\phi_{3:n}}]$ when $\sigma_0(u) \leq j < n$.

2) $\dfrac{\mathbb{P}[\sigma(u) = j-1]}{\mathbb{P}[\sigma(u) = j]} \in [\phi, \dfrac{\phi_{1:n-1}}{1+\phi_{3:n}}]$ when $1 < j \leq \sigma_0(u)$.

In 1), the upper bound is achieved when $\sigma_0(u) = n-1$ and $j = \sigma_0(u)$, while the lower bound is achieved when $\sigma_0(u) = 1$. In 2), the upper bound is achieved when $\sigma_0(u) = 2$ and $j = \sigma_0(u)$, while the lower bound is achieved when $\sigma_0(u) = n$.

Remark 4.1. The result above may seem counterintuitive since it implies that for $\phi + \phi^2 > 1 + \phi^n$, the probability of ranking some element u at a position different from its position in σ_0 is larger than the probability of raking it at position $\sigma_0(u)$. An easy-to-check example that shows that this indeed may be the case corresponds to $\sigma_0 = (1, 2, 3, 4)$ and $\phi = 0.9$. Here, we have $\mathbb{P}[\sigma(3) = 3] = 0.2559 < \mathbb{P}[\sigma(3) = 4] = 0.2617$.

Lemma 4.2 does not guarantee that in any single iteration the position of the element will be correct, since the ranking involves only a subset of elements. Therefore, Lemma 4.3, a generalized version for the subset-projected ranking, is required for the proof.

Lemma 4.3. Let $\sigma \sim \text{MM}(\sigma_0, \phi)$ and let $A \subset [n]$. Consider an element $u \in A$. Then, the following two statements describe the distribution of $\sigma_A(u)$:

1) $\dfrac{\mathbb{P}[\sigma_A(u) = j+1]}{\mathbb{P}[\sigma_A(u) = j]} \leq \max\limits_{l \in [0, n-|A|]} \dfrac{\phi + \phi^l \phi_{2:n-l-1}}{1 + \phi^{2l} \phi_{3:n-l}}$ when $|A| > j \geq \sigma_{0,A}(u)$.

2) $\dfrac{\mathbb{P}[\sigma_A(u) = j-1]}{\mathbb{P}[\sigma_A(u) = j]} \leq \max\limits_{l \in [0, n-|A|]} \dfrac{\phi + \phi^l \phi_{2:n-l-1}}{1 + \phi^{2l} \phi_{3:n-l}}$ when $1 < j \leq \sigma_{0,A}(u)$.

Observe that the conditions that allow one to achieve the upper bound in Lemma 4.2 also ensure that the upper bounds are achieved in Lemma 4.3. Moreover, when $\phi+\phi^2 < 1+\phi^n$, the right hand sides are $\leq \frac{\phi_{1:n-1}}{1+\phi_{3:n}}$.

The next result establishes the performance guarantees for the LCA algorithm with the median operation.

Theorem 4.4. Assume that $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$, where $\sigma_k \overset{\text{i.i.d}}{\sim} \text{MM}(\sigma_0, \phi)$, $k \in [m]$. If $\phi < 0.5$ and $m \geq c \log \frac{2n}{\delta}$, where $c = \frac{2}{(1-2\phi)^2}$, then the output of Algorithm 2 under the median operation equals σ_0 with probability at least $1-\delta$.

The proof follows by observing that if the median of the Lehmer code $c_{\sigma_k}(t)$ over all $k \in [m]$ converges to $t - \sigma_{0,S_t}(t)$ as $m \to \infty$, then each σ_k should have $\mathbb{P}[\sigma_{k,S_t}(t) > \sigma_{0,S_t}(t)], \mathbb{P}[\sigma_{k,S_t}(t) < \sigma_{0,S_t}(t)] < 1/2$. According to the following Lemma, in this case, one needs $\phi < 0.5$.

Lemma 4.5. Let $\sigma \sim \text{MM}(\sigma_0, \phi)$ and let $A \subseteq [n]$. For any $u \in A$, the following two bounds hold:

$$1)\ \mathbb{P}[\sigma_A(u) > \sigma_{0,A}(u)] \leq \frac{\phi_{1:(|A|-\sigma_{0,A}(u))}}{\phi_{0:(|A|-\sigma_{0,A}(u))}} < \phi,$$

$$2)\ \mathbb{P}[\sigma_A(u) < \sigma_{0,A}(u)] \leq \frac{\phi_{1:\sigma_{0,A}(u)}}{\phi_{0:\sigma_{0,A}(u)}} < \phi.$$

The inequality 1) is met for $A = S$ and $\sigma_0(u) = 1$, while the inequality 2) is met for $A = S$ and $\sigma_0(u) = n$.

We now turn our attention to partial rankings and prove the following extension of the previous result for the GMM, under the LCA algorithm that uses the median of coordinate values. Note that the output of Algorithm 2 is essentially a permutation, although it may be transformed into a partial ranking via the bucketing method described in Section 2.

Theorem 4.6. Assume that $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$, where $\sigma_k \overset{\text{i.i.d}}{\sim} \text{GMM}(\sigma_0, \phi)$, $k \in [m]$. If $\phi + \phi^{1/2} < 1$ and $m \geq c \log \frac{2n}{\delta}$ with $c = \frac{2}{(1-2q')^2}$, where $q' = 1 - \frac{1}{2}\phi^{1/2} - \frac{1}{2}\phi$, then the output ranking of the LCA algorithm (see Section VII of the supplementary document) under the median operation is in Σ_0 with probability at least $1-\delta$. Here, Σ_0 denotes the set of permutations generated by breaking ties in σ_0.

The proof of this theorem relies on showing that the InsertionSort procedure places elements in their correct position with high probability. If the median is used for partial ranking aggregation, one vote is uniformly distributed amongst all possible positions in the range given by (6). To ensure that the output permutation is in Σ_0, we need to guarantee that the median of the positions of the votes for t over S_t is in $[l_{\sigma_0,S_t}(t), r_{\sigma_0,S_t}(t)]$ for large enough m (as in this case, $[l_{\sigma_0,S_t}(t), r_{\sigma_0,S_t}(t)]$ represents the bucket in σ_0 that contains t).

For a $\sigma \sim \mathrm{GMM}(\sigma_0, \phi)$, let $v(j)$ be the vote that the partial ranking σ cast for position j. Then, one requires that

$$\mathbb{E}[\sum_{k=1}^{r_{\sigma_{0,A}(u)}} v(j)] > 0.5 \text{ and } \mathbb{E}[\sum_{k=l_{\sigma_{0,A}(u)}}^{n} v(j)] > 0.5.$$

The expectations in the expressions above may be evaluated as follows (We only consider the expectation on the left because of symmetry). If the event $W = \{r_{\sigma_{S_t}(t)} \leq r_{\sigma_{0,S_t}(t)}\}$ occurs, then the vote of σ that contributes to the sum equals 1. If the event $Q = \cup_{j=1}^{n-r_{\sigma_{0,S_t}(t)}} Q_j$, where $Q_j = \{r_{\sigma_{S_t}(t)} = j + r_{\sigma_{0,S_t}(t)}, l_{\sigma_{S_t}(t)} \leq r_{\sigma_{0,S_t}(t)}\}$ occurs, then the vote that σ contributes to the sum equals $V_j = \frac{r_{\sigma_{0,S_t}(t)} - l_{\sigma_{S_t}(t)} + 1}{r_{\sigma_{S_t}(t)} - l_{\sigma_{S_t}(t)} + 1}$. Therefore, we have

$$\mathbb{E}[\sum_{k=1}^{r_{\sigma_{0,S_t}(t)}} v(k)] = \mathbb{P}[W] + \sum_{j=1}^{n-r_{\sigma_0}(u)} V_j \mathbb{P}[Q_j]. \quad (8)$$

The following lemma describes a lower bound for (8).

Lemma 4.7. Let $\sigma \sim \mathrm{GMM}(\sigma_0, \phi)$ and let $A \subseteq [n]$ be such that it contains a predefined element u. Let $A' = A - \{x \in A : x \neq u, \sigma_{0,A}(x) \leq \sigma_{0,A}(u)\}$. Define

$$\begin{aligned}
&W = \{r_{\sigma_A(u)} \leq r_{\sigma_{0,A}(u)}\},\\
&Q_j = \{r_{\sigma_A(u)} = j + r_{\sigma_{0,A}(u)},\ l_{\sigma_A(u)} \leq r_{\sigma_{0,A}(u)}\},\\
&W' = \{r_{\sigma_{A'}(u)} \leq r_{\sigma_{0,A'}(u)}\},\\
&Q'_j = \{r_{\sigma_{A'}(u)} = j + r_{\sigma_{0,A'}(u)},\ l_{\sigma'_A(u)} \leq r_{\sigma_{0,A'}(u)}\}.
\end{aligned}$$

Then, one can prove that

$$\begin{aligned}
&\mathbb{P}[W] + \sum_{j=1}^{|A|-r_{\sigma_{0,A}(u)}} V_j \mathbb{P}[Q_j]\\
\geq\ &\mathbb{P}[W'] + \sum_{j=1}^{|A'|-r_{\sigma_{0,A'}(u)}} \frac{1}{j+1} V_j \mathbb{P}[Q'_j]\\
\geq\ &1 - \tfrac{1}{2}\phi^{1/2} - \tfrac{1}{2}\phi.
\end{aligned}$$

If $\phi + \phi^{1/2} < 1$, the lower bound above exceeds 1/2. Theorem 4.6 then follows using the union bound and Hoeffding's inequality.

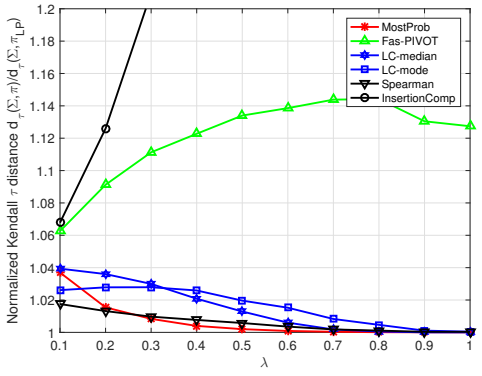

Figure 1: The normalized Kendall τ distance between the set σ and the aggregate vs the parameter λ of the MM.

5 Performance Evaluation

We next evaluate the performance of the LCA algorithms via experimental methods and compare it to that of other rank aggregation methods using both synthetic and real datasets. For comparative analysis, we choose the Fas-Pivot and FasLP-Pivot (LP) methods [10], InsertionSort with Comparison (InsertionComp) from [9], and the optimal Spearman Footrule distance aggregator (Spearman) [11]. For the randomized algorithms Fas-Pivot and FasLP-Pivot, the pivot in each iteration is chosen randomly. For InsertionSort with Comparison, the insertion order of the elements is also chosen randomly. Furthermore, for all three methods, the procedure is executed five times, and the best solution is selected. For Fas-Pivot and FasLP-Pivot, we chose the better result of Pick-A-Perm and the given method, as suggested in [10].

In the context of synthetic data, we only present results for the Mallows model in which the number of ranked items equals $n = 10$, and the number of rankings equals $m = 50$. The variance parameter was chosen according to $\phi = e^{-\lambda}$, where λ is allowed to vary in $[0, 1]$. For each parameter setting, we ran 50 independent simulations and computed the average cumulative Kendall τ distance (normalized by m) between the output ranking and Σ, given as $D_{av} = \frac{D(\sigma,\Sigma)}{m}$. We then normalized the D_{av} value of each algorithm by that of FasLP-Pivot, since FasLP-Pivot always offered the best performance. The results are depicted in Fig. 1. Note that we used MostProb to describe the most probable ranking, which is the centroid for the Mallows Model.

Note that for parameter values $\lambda \geq 0.6$ LCA algorithms perform almost identically to the best aggregation method, the LP-based pivoting scheme. For smaller values of λ, the performance differences are negligible; but the LCS method has significantly

Table 1: Rank aggregator comparison for the Sushi dataset (permutations)

m	10	50	200	1000	5000
Fas-Pivot	14.51	15.98	16.18	16.38	16.06
FasLP-Piovt	13.59	15.00	15.33	15.39	15.39
InsertionComp	15.87	16.60	16.70	16.80	16.65
Spearman	14.41	15.24	15.54	15.56	15.61
LC-median	14.03	15.25	15.57	15.58	15.74
LC-mode	14.19	15.33	15.46	15.47	15.49

Table 2: Rank aggregator comparison for the Jester dataset (permutations)

m	50	200	1000	5000	10000
Fas-Pivot	2102	2137	2144	2127	2127
FasLP-Piovt	1874	1915	1920	1922	1921
InsertionComp	2327	2331	2337	2323	2390
Spearman	1900	1936	1935	1937	1937
LC-median	1932	1962	1965	1966	1965
LC-mode	1973	1965	1962	1964	1965

smaller complexity, which in the parallel implementation mode equals $O(n+m)$. Note that the InsertionSort Comp method performs poorly, although it ensures local Kemeny optimality.

We also conducted experiments on a number of real-world datasets. To test the permutation LCA aggregation algorithms, we used the Sushi ranking dataset [25] and the Jester dataset [26]. The Sushi dataset consists of 5000 permutations involving $n = 10$ types of sushi. The Jester dataset contains scores in the continuous interval $[-10, 10]$ for $n = 100$ jokes submitted by 48483 individuals. We chose the scores of 14116 individuals who rated all 100 jokes and transformed the rating into permutations by sorting the scores. For each dataset, we tested our algorithms by randomly choosing m many samples out of the complete list and by computing the average cumulative Kendall τ distance normalized by m via 50 independent tests. The results are listed in the Table 1 and Table 2.

To test our partial ranking aggregation algorithms, we used the complete Jester dataset [26] and the Movielens dataset [27]. For the Jester dataset, we first rounded the scores to the nearest integer and then placed the jokes with the same integer score in the same bucket of the resulting partial ranking. We also assumed that the unrated jokes were placed in a bucket ranked lower than any other bucket of the rated jokes. The movielens dataset contains incomplete lists of scores for more than 1682 movies rated by 943 users. The scores are integers in [5], so that many ties are present. We chose the 50 most rated movies and 500 users who rated these movies with largest coverage. Similarly as for the Jester dataset, we assumed

Table 3: Rank aggregator comparison for the Jester dataset (partial rankings)

m	50	200	1000	5000	10000
Fas-Pivot	1265	1280	1279	1279	1281
FasLP-Piovt	1264	1280	1279	1279	1281
InsertionComp	1980	1967	1956	1949	1979
Spearman	1272	1284	1281	1281	1282
LC-median	1275	1287	1284	1283	1287
LC-mode	1311	1304	1289	1283	1283

Table 4: Rank aggregator comparison for the Movielens dataset (partial rankings)

m	20	50	100	200	500
Fas-Pivot	328.8	344.4	350.3	351.4	353.3
FasLP-Piovt	328.6	344.4	350.3	351.4	353.5
InsertionComp	386.3	390.2	392.6	393.1	393.0
Spearman	332.9	347.3	352.5	353.5	355.4
LC-median	334.2	350.4	355.4	355.9	359.1
LC-mode	340.1	353.5	357.5	359.0	360.0

that the unrated movies were tied for the last position. In each test, we used the iterative method described in Section 3 to transform permutations into partial rankings. Note that for the Footrule-optimal method for aggregating partial rankings, we used the algorithm in Section 3.1.2 of [28]. Moreover, when computing the Kemeny distance between two partial rankings of (2), we omitted the penalty incurred by ties between unrated elements, because otherwise the iterative method would yield too many ties in the output partial ranking. We used the following formula to assess the distance between two incomplete partial rankings (9):

$$d_\tau(\pi,\sigma) = |\{(x,y) : \pi(x) > \sigma(y), \pi(x) < \sigma(y)\}| + \frac{1}{2}|\{(x,y) : [\pi(x) = \pi(y), \sigma(x) > \sigma(y),\ x,y \text{ rated by } \pi] \text{ or } [\pi(x) > \pi(y), \sigma(x) = \sigma(y),\ x,y \text{ rated by } \sigma]\}|. \quad (9)$$

The results are listed in Table 3 and Table 4. The parallelizable, low-complexity LCA methods offer very similar performance to that of the significantly more computationally demanding LP pivoting algorithm.

Concluding remarks. An open question is to determine if a constant-factor-approximation result holds for LCA. The problem may be approached by observing that for any $\pi, \sigma \in \mathbb{S}_n$, $d_\tau(\pi,\sigma) \geq ||c_\pi - c_\sigma||_{\ell_1}$ [29, 30], and showing that the optimal $\min_\pi \sum_i d_\tau(\pi,\sigma_i)$ can be constantly approximated by $\min_\pi \sum_i ||c_\pi - c_\sigma||_{\ell_1}$ whose optimal solution is the median LCA. Another interesting problem is to extend the LCA method to other forms of rank aggregation, such as the k-center problem [31, 32], or the min-max problem [33].

Acknowledgment. The work was funded by the NSF grants CCF 1642550 and 1527636.

Reference

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 89–96.

[2] Tie-Yan Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[3] Minji Kim, Farzad Farnoud, and Olgica Milenkovic, "Hydra: gene prioritization via hybrid distance-score rank aggregation," *Bioinformatics*, p. btu766, 2014.

[4] Sahand Negahban, Sewoong Oh, and Devavrat Shah, "Iterative ranking from pair-wise comparisons," in *Advances in Neural Information Processing Systems*, 2012, pp. 2474–2482.

[5] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz, "Pairwise ranking aggregation in a crowdsourced setting," in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 193–202.

[6] John G Kemeny, "Mathematics without numbers," *Daedalus*, vol. 88, no. 4, pp. 577–591, 1959.

[7] Andrew Davenport and Jayant Kalagnanam, "A computational study of the kemeny rule for preference aggregation," in *AAAI*, 2004, vol. 4, pp. 697–702.

[8] John Bartholdi, Craig A Tovey, and Michael A Trick, "Voting schemes for which it can be difficult to tell who won the election," *Social Choice and welfare*, vol. 6, no. 2, pp. 157–165, 1989.

[9] Cynthia Dwork, Ravi Kumar, Moni Naor, and D Sivakumar, "Rank aggregation revisited," 2001.

[10] Nir Ailon, Moses Charikar, and Alantha Newman, "Aggregating inconsistent information: ranking and clustering," *Journal of the ACM (JACM)*, vol. 55, no. 5, pp. 23, 2008.

[11] Persi Diaconis and Ronald L Graham, "Spearman's footrule as a measure of disarray," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 262–268, 1977.

[12] Claire Kenyon-Mathieu and Warren Schudy, "How to rank with few errors," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM, 2007, pp. 95–103.

[13] Michael A Fligner and Joseph S Verducci, *Probability models and statistical analyses for ranking data*, vol. 80, Springer, 1993.

[14] Francois Caron and Arnaud Doucet, "Efficient bayesian inference for generalized bradley–terry models," *Journal of Computational and Graphical Statistics*, vol. 21, no. 1, pp. 174–196, 2012.

[15] Tyler Lu and Craig Boutilier, "Learning mallows models with pairwise preferences," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 145–152.

[16] Guy Lebanon and John Lafferty, "Cranking: Combining rankings using conditional probability models on permutations," in *ICML*. Citeseer, 2002, vol. 2, pp. 363–370.

[17] Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D Sivakumar, and Erik Vee, "Comparing and aggregating rankings with ties," in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2004, pp. 47–58.

[18] Richard P Stanley, *Enumerative combinatorics*, Number 49. Cambridge university press, 2011.

[19] Martin Mareš and Milan Straka, "Linear-time ranking of permutations," in *Algorithms–ESA 2007*, pp. 187–193. Springer, 2007.

[20] Wendy Myrvold and Frank Ruskey, "Ranking and unranking permutations in linear time," *Information Processing Letters*, vol. 79, no. 6, pp. 281–284, 2001.

[21] Vincent Vajnovszki, "Lehmer code transforms and mahonian statistics on permutations," *Discrete Mathematics*, vol. 313, no. 5, pp. 581–589, 2013.

[22] Don Coppersmith, Lisa Fleischer, and Atri Rudra, "Ordering by weighted number of wins gives a good ranking for weighted tournaments," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, 2006, pp. 776–782.

[23] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar, "Rank aggregation methods for the web," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 613–622.

[24] Karthik Kambatla, Georgios Kollias, and Ananth Grama, "Efficient large-scale graph analysis in mapreduce," 2012.

[25] Toshihiro Kamishima, "Nantonac collaborative filtering: recommendation based on order responses," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 583–588.

[26] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[27] F Maxwell Harper and Joseph A Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 19, 2016.

[28] Frans Schalekamp and Anke van Zuylen, "Rank aggregation: Together we're strong," in *Proceedings of the Meeting on Algorithm Engineering & Expermiments*. Society for Industrial and Applied Mathematics, 2009, pp. 38–51.

[29] Alexander Barg and Arya Mazumdar, "Codes in permutations and error correction for rank modulation," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3158–3165, 2010.

[30] Da Wang, Arya Mazumdar, and Gregory W. Wornell, "Compression in the Space of Permutations," *IEEE Transactions on Information Theory*, vol. 61, no. 12, pp. 6417–6431, 2015.

[31] Pranjal Awasthi, Avrim Blum, Or Sheffet, and Aravindan Vijayaraghavan, "Learning mixtures of ranking models," in *Advances in Neural Information Processing Systems*, 2014, pp. 2609–2617.

[32] Flavio Chierichetti, Anirban Dasgupta, Ravi Kumar, and Silvio Lattanzi, "On learning mixture models for permutations," in *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*. ACM, 2015, pp. 85–92.

[33] Pan Li and Olgica Milenkovic, "Multiclass minmax rank aggregation," *arXiv preprint arXiv:1701.08305*, 2017.