Tractable Search for Learning Exponential Models of Rankings

Bhushan Mandhani
Dept. of Computer Science
University of Washington
Seattle USA 98195

Marina Meila
Dept. of Statistics
University of Washington
Seattle USA 98195

Abstract

We consider the problem of learning the Generalized Mallows (GM) model of [Fligner and Verducci, 1986], which represents a probability distribution over all possible permutations (or rankings) of a given set of objects. The training data consists of a set of permutations. This problem generalizes the well known rank aggregation problem. Maximum Likelihood estimation of the GM model is NP-hard. An exact but inefficient search-based method was recently proposed for this problem. Here we introduce the first non-trivial heuristic function for this search. We justify it theoretically, and show why it is admissible in practice. We experimentally demonstrate its effectiveness, and show that it is superior to existing techniques for learning the GM model. We also show good performance of a family of faster approximate methods of search.

1 Introduction

Preference data, such as rankings, appear in many applications. One of them is *rank aggregation*, which looks to combine several input rankings to produce a single output ranking that best represents the input. This has applications in combining search engine rankings [Cohen et al., 1999, Dwork et al., 2001], collaborative filtering [Pennock et al., 2000], ranked voting, etc.

But often we need to go beyond rank aggregation. Instead of the single best ranking, we need a probability distribution over all possible rankings. This gives us the ability to make predictions and answer a wide range of queries such as: the probability that a particular item will occur in the top k, the most likely sequence for the top k items, whether item i will be ranked before j, the expected number of fixed points in the ranking, etc.

Several probabilistic models over rankings have been studied in statistics; see [Fligner and Verducci, 1993] for an excellent reference. Here we focus on the *Mallows model* proposed in [Mallows, 1957]. This model has become increasingly popular by itself, or as a basic block in interesting applications [Lebanon and Lafferty, 2003]. [Fligner and Verducci, 1986] extend it to obtain the *Generalized Mallows Model* (GM), in which the ranking process is interpreted as having multiple independent stages, each governed by its own dispersion parameter. In this way the GM can model processes where there is high consensus for say, the most highly ranked items, but near indifference to the ranking of other items. For example, a population can have high consensus that the economy is the most important issue at a certain time, but be rather inconsistent about the ranking of the other issues. This is true for many real world situations.

In [Meila et al., 2007] we showed that the Mallows and GM models can be estimated exactly from data by a search-based method. The search tree has $n!$ leaf nodes where n is the number of items, an order of growth reflecting the NP-hardness of the problem. In spite of this, in [Meila et al., 2007] we also introduced an effective method of search for the Mallows model. No such method is known for the GM model, leaving the exact search algorithm to the stage of a theoretical construct.

To make the exact GM model estimation practical, a good *heuristic function* is needed. We would like the heuristic to be admissible for A* search to guarantee us optimality. In this paper, we propose the first such heuristic. It differs substantially from the exist-

Appearing in Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5. Copyright 2009 by the authors.

ing solutions for the Mallows model, since it requires lower bounding a real-valued function of several discrete variables. It is no longer purely a combinatorial optimization task. Our heuristic is based on a careful analysis of this likelihood function, and imposing constraints obtained by combinatorial arguments that generalize the ideas used in the Mallows model heuristic. It is principled but efficient enough to be employed in a search that generates a very large number of nodes.

The rest of this paper is organized as follows. Sections 2 and 3 present necessary background knowledge following [Fligner and Verducci, 1986, Meila et al., 2007] about the Mallows and GM models, and their likelihood functions. Section 4 introduces our new contribution, the proposed heuristic function. We discuss related work in Section 5, present experiments in Section 6, and conclude in Section 7.

2 The Mallows & Generalized Mallows Models

Suppose we have n items labeled 1,2,...,n that are to be ranked. Any permutation π of these items represents a ranking.

Definition 1 *The Kendall Distance $d(\pi, \sigma)$ between rankings π and σ is defined as the total number of item pairs over which they disagree. They disagree over an item pair (i, j) if the relative ordering of i and j is not the same in both.*

Definition 2 *The Mallows model is the following probability distribution over all rankings π. $\mathrm{P}(\pi) = \frac{\exp(-\theta\, \mathrm{d}(\pi,\pi_0))}{\psi(\theta)}$ Ranking π_0 and $\theta \geq 0$ are the model parameters while $\psi(\theta)$ is a normalization constant.*

For $\theta > 0$, the probability of π decreases exponentially with distance from the modal ranking π_0. For $\theta = 0$, we just get the uniform distribution. For notational convenience, we will fix π_0 to (1,2,...,n) for the remainder of this section, and denote $d(\pi, \pi_0)$ by $D(\pi)$.

We define $V_j(\pi)$ to be the number of disagreements that π has with π_0 which involve the j-th ranked item and some lower ranked item in π_0. For our choice of π_0, $V_j(\pi)$ becomes the number of elements in $\{j+1, j+2, \ldots, n\}$ that are ranked above j in π. Thus, V_j can vary from 0 to $(n-j)$. It is easy to see that the following relationship holds.

$$D(\pi) = \sum_{j=1}^{n-1} V_j(\pi) \tag{1}$$

Definition 3 ([Fligner and Verducci, 1986]) *The Generalized Mallows model defines the following probability distribution over all rankings π.*

$$\mathrm{P}(\pi) = \frac{\exp(-\sum_{j=1}^{n-1} \theta_j\, V_j(\pi))}{\psi(\theta_1, \ldots, \theta_{n-1})} \tag{2}$$

Ranking π_0 and $\theta_1, \ldots, \theta_{n-1}$ are the model parameters with $\theta_j \geq 0\ \forall j$. The function ψ is the normalization constant.

Note that θ_j can be interpreted as a measure of the strength of consensus within the population that item j should rank higher than the items $(j+1)$ up to n. The larger θ_j is, the less likely it is that $V_j(\pi)$ will be large for π coming from the model. When all the θ_j parameters are constrained to be equal, the GM model reduces to the Mallows model, as is evident from Equation 1.

Theorem 1 ([Fligner and Verducci, 1986]) *When π is distributed according to the GM model, the random variables $V_1, \ldots, V_{n-1}$ are mutually independent. For all j, the distribution of V_j over $\{0, 1, \ldots, n-j\}$ is now given by:*

$$\mathrm{P}(V_j = p) = \frac{\exp(-\theta_j p)}{\sum_{q=0}^{n-j} \exp(-\theta_j q)} \tag{3}$$

The expression in the denominator of (3) is of interest, and will occur in our analysis of the likelihood function for computing the search heuristic.

$$\psi_j(\theta) = \sum_{q=0}^{n-j} \exp(-\theta q) = \frac{1 - \exp(-(n-j+1)\theta)}{1 - \exp(-\theta)} \tag{4}$$

It follows from Theorem 1 that $\psi(\theta_1, \ldots, \theta_{n-1}) = \prod_{j=1}^{n-1} \psi_j(\theta_j)$

3 Maximum Likelihood Estimation

Suppose we have a dataset of N rankings $\pi_1, \pi_2, \ldots, \pi_N$ over n items labeled 1 to n. We want to learn the parameters for the Generalized Mallows model that maximize the likelihood. Let $\bar{V_j} = \sum_{i=1}^{N} V_j(\pi_i)/N$ denote the observed mean for V_j. The log likelihood can be written as follows.

$$\log l = -N \sum_{j=1}^{n-1} \left(\theta_j \bar{V_j} + \log \psi_j(\theta_j)\right) = -N \sum_{j=1}^{n-1} f_j(\theta_j, \bar{V_j}) \tag{5}$$

where the function f_j has been introduced for notational convenience.

The log likelihood for the Mallows model is obtained simply by setting all the θ_j parameters in (5) to a single θ. Let $\bar{D} = \sum_{i=1}^{N} D(\pi_i)/N = \sum_{j=1}^{n-1} \bar{V_j}$ denote the observed mean for D. We get the following expression for the Mallows model.

$$\log l = -N\left(\theta\bar{D} + \sum_{j=1}^{n-1}\log\psi_j(\theta)\right) = -Nf(\theta,\bar{D}) \quad (6)$$

3.1 Estimating θ_j with fixed π_0

Assuming the modal ranking π_0 is known, $\bar{V_j}$ is known for all j, and we can treat f_j as a function of θ_j alone. To maximize the log likelihood each f_j is minimized separately. The following results will be later used to derive a bound for the maximum likelihood that can be attained when the ranking π_0 is some extension of a given top-k ranking (a ranking which has only its top k ranks specified). This will serve as our search heuristic, and is discussed in depth in Section 4.

Lemma 2 *The function f_j is strictly convex in θ_j for all j. The function f is strictly convex in θ.*

Lemma 3

$$h_j(\bar{V_j}) \;=\; \min_{\theta_j\geq 0} f_j(\theta_j,\bar{V_j}) \;=$$

$$\begin{cases}\log(n-j+1) & \text{if } \bar{V_j}\geq (n-j)/2\\ \theta_{j0}\bar{V_j}+\log\psi_j(\theta_{j0}) & \text{otherwise}\end{cases}$$

where $\theta_{j0}>0$ is the solution of $\psi_j'(\theta_j)/\psi_j(\theta_j) = -\bar{V_j}$. Further, the left side of this equation is a strictly increasing function of θ_j.

Since the left side of the equation above is strictly increasing, it can be solved by doing a binary search over a suitable real interval. Thus, if π_0 is known, the θ_j parameters can be determined.

3.2 Search for the Modal Ranking π_0

For the Mallows model, maximum likelihood estimation of π_0 requires minimizing $\bar{D}$. This is the well-known *Kemeny ranking* problem and is NP-complete [Bartholdi et al., 1989]. For the GM model, we need to minimize $\sum_{j=1}^{n-1}\left(\theta_j\bar{V_j}+\log\psi_j(\theta_j)\right)$. Thus, the problem is no longer purely a combinatorial optimization task.

We now describe the search method we proposed in [Meila et al., 2007] for finding the ML π_0. We define the $n\times n$ matrix Q where Q_{ij} is the fraction of rankings in the dataset in which item i is ranked above j. Suppose π_0 is the ranking $(r_1, r_2, \ldots, r_n)$. $\bar{V_j}$ is the per ranking average of the number of elements in $\{r_{j+1},\ldots,r_n\}$ that are ranked above r_j. Observe that the following holds.

$$\bar{V_j} = \sum_{i=j+1}^{n} Q_{r_i r_j} \quad (7)$$

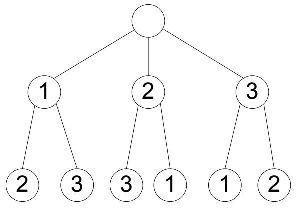


Figure 1: Search Tree for three items

Suppose we know only the top j ranks of π_0. We then know all items not in the top j and can still determine $\bar{V_j}$. It follows that we can determine $\bar{V_i}$ $\forall\, i\leq j$. From $\bar{V_i}$ we can determine θ_i. Thus, if we know the top j ranks of π_0, we can determine the portion $\sum_{i=1}^{j}\left(\theta_i\bar{V_i}+\log\psi_i(\theta_i)\right)$ of the log likelihood. The search method depends on this key observation.

Each node of the search tree is an item. A node at depth j represents a partial ranking $(r_1, r_2, \ldots, r_j)$ in which the top j ranks are known. This partial ranking is obtained by following the unique path down from the root of the tree to this node. The node will have $(n-j)$ children corresponding to the $(n-j)$ ways in which the next item can be chosen. The leaf nodes occur at depth $(n-1)$. They represent complete rankings and are the goal nodes of the search. Figure 1 shows the complete search tree for $n=3$.

In an A* search on this tree, the cost for a node x at depth j is given by $\sum_{i=1}^{j}\left(\theta_i\bar{V_i}+\log\psi_i(\theta_i)\right)$. The heuristic function is a crucial aspect of the search, and determines whether or not it will be tractable. Further, we would like it to be admissible meaning it should be a lower bound on the remaining cost for reaching a goal node. With an admissible heuristic, A* search will always return the optimal solution. We present our proposed heuristic in the next section.

4 Heuristic Functions for the Search

The search method for finding π_0 explores the same tree for both the Mallows and GM models, but they have different cost functions. The associated heuristic functions will differ too. For a node x at depth k of the tree, we will denote the heuristics for the Mallows and GM models by L and H respectively. In this paper, we show how H can be computed. The problem of computing L has been studied in [Conitzer et al., 2006] and [Meila et al., 2007]. It is instructive to first briefly discuss how L can be computed.

4.1 Mallows Model Search Heuristics

Suppose node x represents the partial ranking $(r_1,\ldots,r_k)$. The cost at x is given by $\sum_{j=1}^{k}\bar{V_j}$. L is a lower bound on the remaining cost which is

$\sum_{j=k+1}^{n-1} \bar{V}_j$. Let S be the set of items not in the partial ranking at x. Equation 7 implies that for any items $u, v \in S$, exactly one of Q_{uv} and Q_{vu} will occur in the expression $\sum_{j=k+1}^{n-1} \bar{V}_j$. If v is ranked above u, the former will occur. Otherwise, the latter will. This leads us to the following lower bound.

$$L = \sum_{u,v \in S} \min(Q_{uv}, Q_{vu}) \tag{8}$$

While computing L naively would take $O(n-k)^2$ time, using L for the parent node of x, we can compute L for x in $O(n-k)$ time.

In fact, $\sum_{j=k+1}^{n-1} \bar{V}_j$ is just the Kemeny score "$\bar{D}$" for the restriction of the input rankings to items in S. The techniques of [Conitzer et al., 2006] are thus directly applicable. They give us the following four ways for computing L, using a directed graph constructed from the input rankings: (1) L_0 is the same as (8); (2) L_1 is based on edge-disjoint cycles; (3) L_2 is based on cycles (not necessarily edge-disjoint); (4) L_3 is based on a LP relaxation of the integer linear program that gives the minimum weight feedback edge set. The relationship $L_0 \le L_1 \le L_2 \le L_3$ holds. Thus, there is a trade-off between the tightness of the bound and the cost of computing it.

4.2 GM Model Search Heuristic – Concave Minimization

The cost at node x is now given by $\sum_{j=1}^{k} (\theta_j \bar{V}_j + \log \psi_j(\theta_j))$. The remaining cost is $\sum_{j=k+1}^{n-1} (\theta_j \bar{V}_j + \log \psi_j(\theta_j))$. Recall that once $\bar{V}_j$ is known for some j, θ_j is then determined from it by minimizing the function f_j as described in Lemma 3. This minimized value is $h_j(\bar{V}_j)$ where h_j is defined in the lemma. The remaining cost can be written as $\sum_{j=k+1}^{n-1} h_j(\bar{V}_j)$. The heuristic H must be a lower bound for this expression.

Our approach for computing H has three parts. First, we establish the concavity of the cost function. Second, we define a convex domain for the cost function corresponding to valid vectors of $\bar{V}_j$ values. Third, we determine the tightest such domain using graph-theoretic techniques.

Lemma 4 *The function h_j is differentiable everywhere. Its derivative is given by:*

$$h'_j(\bar{V}_j) = \begin{cases} 0 & \text{if } \bar{V}_j \ge (n-j)/2 \\ \theta_j & \text{otherwise} \end{cases}$$

where $\theta_j > 0$ is the solution of $\psi'_j(\theta_j)/\psi_j(\theta_j) = -\bar{V}_j$. Thus, h_j is strictly increasing in the interval $(0, \frac{n-j}{2})$, and then becomes constant.

Lemma 5 *The function h_j is concave.*

Lemma 6 *The multivariable function h defined below is concave.*

$$h(\bar{V}_{k+1}, \bar{V}_{k+2}, \ldots, \bar{V}_{n-1}) = \sum_{j=k+1}^{n-1} h_j(\bar{V}_j)$$

For the Mallows model search, the remaining cost has a simple combinatorial interpretation as the minimum weight feedback edge set in a directed graph [Conitzer et al., 2006]. For the GM model, we are dealing with a continuous multivariable function. Lemma 6 tells us that it is concave. We will lower bound it by minimizing it over a convex set. In fact, the constraints that define this convex set are obtained by generalizing the idea in (8) for computing L.

We compute constants $\alpha_{k+1}, \alpha_{k+2}, \ldots, \alpha_{n-1}$ such that the following inequalities hold. We describe how they are computed later.

$$\begin{array}{ccccccccc} \bar{V}_{k+1} & + & \bar{V}_{k+2} & + & \ldots & + & \bar{V}_{n-1} & \ge & \alpha_{k+1} \\ & & \bar{V}_{k+2} & + & \ldots & + & \bar{V}_{n-1} & \ge & \alpha_{k+2} \\ & & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & & & \bar{V}_{n-1} & \ge & \alpha_{n-1} \end{array} \tag{9}$$

Each constraint is linear and defines a halfspace. They together define an unbounded polyhedron in $(n-k-1)$-dimensional space. Further, this polyhedron has a unique vertex which is the intersection of the $(n-k-1)$ hyperplanes corresponding to the constraints in (9). It is determined by solving the system of linear equations obtained by converting the constraints into equalities. This system is upper triangular and is trivially solved. The unique vertex of this polyhedron is $(\alpha_{k+1} - \alpha_{k+2}, \alpha_{k+2} - \alpha_{k+3}, \ldots, \alpha_{n-1} - \alpha_n)$, where we define $\alpha_n = 0$ for notational convenience.

We will compute H by minimizing the concave function h over this polyhedron. Since each h_j is bounded, h must attain its minimum value. We know from convex analysis [Rockafellar, 1972] that if a concave function attains its minima over a convex set it must do so at an extreme point. Thus, h is minimized at the unique vertex of this polyhedron. This leads us to a key result.

Theorem 7 *Given the constraints in (9), a lower bound H for $\sum_{j=k+1}^{n-1} h_j(\bar{V}_j)$ can be computed as follows.*

$$H = \sum_{j=k+1}^{n-1} h_j(\alpha_j - \alpha_{j+1})$$

4.3 GM Model Search Heuristic – Combinatorial Minimization

We now discuss how the α_j constants are computed. Consider the set S of $(n-k)$ items not in the partial ranking at node x. We construct a graph G having a node for each item in S. For items $u, v \in S$, the edge uv is assigned $\min\{Q_{uv}, Q_{vu}\}$ as weight. Let β_j denote the weight of the minimum weight clique of $(n-j)$ nodes in the complete graph G. Note that $(\bar{V}_{j+1} + \cdots + \bar{V}_{n-1})$ represents the Kemeny score "$\bar{D}$" for some $(n-j)$ items in S. Thus, β_j or any value smaller than it works as α_{j+1}. The argument is the same as given for (8).

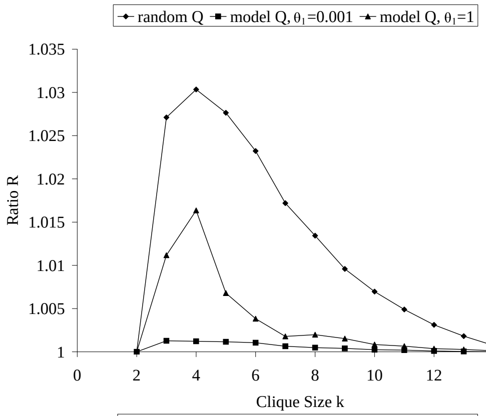
Let G^c denote the complement graph of G in which edge uv is given weight (1-$\min\{Q_{uv}, Q_{vu}\}$). Then, the maximum weight clique of size $(n-j)$ in G^c has weight $\frac{1}{2}(n-j)(n-j-1)-\beta_j$. Finding the max weight clique is NP-hard. Suppose we had some way of approximating it to a factor R, and γ_j is the approximate value we get for the clique weight. Then, we have the following.

$$\gamma_j \geq \frac{1}{R}\left(\frac{1}{2}(n-j)(n-j-1) - \beta_j\right)$$

$$\beta_j \geq \frac{1}{2}(n-j)(n-j-1) - R\gamma_j \qquad (10)$$

The right side of (10) gives us α_{j+1}. To complete the discussion, we need an algorithm for approximating the max-weight cliques in the graph G^c. While no constant factor approximations are known for arbitrary graphs, this problem is easier for graphs like G^c whose edge weights satisfy the triangle inequality. We employ a simple greedy algorithm which was shown by [Birnbaum and Goldman, 2006] to give an approximation of $R = 2$. It constructs a max-weight clique C in a stepwise manner. In each step, it adds to C the node which augments the clique weight the most. Thus, it adds the node u for which $\sum_{v \in C} w_{uv}$ is maximum. The algorithm terminates when C has reached the desired size. Note that in constructing a max-weight clique of size $(n-k)$ for G^c, the algorithm will also yield max-weight cliques of all smaller sizes. In a single sweep, we get all the γ_j values that we need to compute all the α_j values. The total time taken is O$(n-k)^2$.

The smaller R is, the higher the value we get for α_{j+1}, and the tighter the bound in (9). In practice, this greedy algorithm gives a much better approximation than $R = 2$ for the graphs G^c we are working with. These graphs are not arbitrary, and have been generated in a specified way from a matrix Q which satisfies particular constraints. Figure 2 shows the value of R obtained when estimating max-weight cliques of different sizes for the G^c corresponding to the root node of the search tree. The various input matrices Q are described in Section 6. Each plotted point is the mean value of 25 independent runs. We *always* got $R < 1.03$. Based on these results, in our experiments we used $R = 1.05$ in (10) for determining the α_j values.



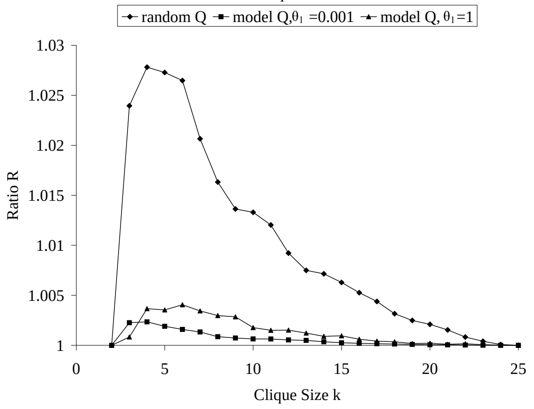


Figure 2: Approximation Ratio R given by the greedy algorithm for finding the max-weight clique for graphs G^c corresponding to different kinds of matrices Q, with $n = 15$ (top) and $n = 25$ (bottom).

5 Related Work

We saw that the maximum likelihood π_0 for the Mallows model is just the Kemeny ranking for the input rankings. Finding it has been well-studied in both AI and theoretical computer science [Conitzer et al., 2006, Ailon et al., 2005, Even et al., 1998]. However, finding the ML π_0 for the GM model is more general, and not a purely combinatorial problem. Most methods in the vast literature on finding Kemeny rankings do not easily transfer to it. Here we will focus on some that do.

[Cohen et al., 1999] propose a greedy algorithm for finding the Kemeny ranking. It successively chooses the items ranked 1 to n in that order. In step j, among the $(n-j+1)$ remaining candidates for rank j, it chooses the one that minimizes $\bar{V}_j$. The function to be minimized is $\bar{D} = \sum_{j=1}^{n-1} \bar{V}_j$. Their greedy strategy is a natural and intuitive choice. Moreover, it can be easily generalized to estimate π_0 for the GM model where we need to minimize $\sum_{j=1}^{n-1} h_j(\bar{V}_j)$. In step j, we now choose for rank j the item that minimizes $h_j(\bar{V}_j)$. However, we know from Lemma 4 that h_j is increasing. Thus, $h_j(\bar{V_j})$ is minimized by just minimizing $\bar{V_j}$. Their greedy algorithm works unchanged for the GM model.

[Fligner and Verducci, 1988] propose a class of models which have a ranking π_0 as parameter and which specify the distribution for each V_j. The random variables V_j are defined exactly as in Section 2. They call such a model strongly unimodal if $\mathrm{P}(V_j = x)$ is a decreasing function of $x \; \forall j$. π_0 will clearly be the mode for such a model. Suppose π_0 is $(r_1, r_2, \ldots, r_n)$, and R_i denotes the rank of item r_i in some ranking π drawn from the distribution. They show that for a strongly unimodal model, the expectations of the random variables R_i satisfy $E(R_1) < E(R_2) < \ldots < E(R_n)$. Thus, if we sort the items in increasing order of average ranks in the dataset, the resulting ranking σ_0 is likely to be close to π_0. For estimating π_0, they propose a local search starting at σ_0. At each step, it moves to the neighbor that maximizes the likelihood. The neighbors of a ranking σ are the $(n-1)$ rankings obtained by swapping two adjacent items in σ. The search terminates when a local maxima is reached. The Mallows and GM models are both strongly unimodal, and this method can be used to estimate π_0 for them.

[Fligner and Verducci, 1993] is a rich collection of papers covering work on probabilistic models for rankings in the statistics community. It is interesting to note that many of these models, while being simple to define, are quite difficult to treat analytically. In the GM model, the independence of the random variables V_j is the key to it being analytically tractable and having interpretable parameters.

6 Experiments

Our first set of experiments evaluate the performance of our proposed heuristic for the GM model search. We are interested in both the tractability of search, and the quality of the results obtained. The second set look to validate the GM model by fitting it to a small but real dataset of rankings.

6.1 Evaluation of Heuristic Performance

Recall that the matrix Q represents all that we need to know about the input dataset. We worked with two kinds of input matrices Q. A "model Q" matrix is obtained from a set of rankings sampled from an instance of the GM model itself. The θ_j parameters of the model decreased linearly so that $\theta_{n-1} = 0.5\theta_1$. Thus, θ_1 determined all the θ_j values and was a parameter that we varied. We used a set of 1000 rankings from this model. The second kind of input Q matrix we used was obtained by setting each Q_{ij} with $i < j$ to a random value in $[0, 1]$. Since $Q_{ij} + Q_{ji} = 1$, the remaining values are automatically determined. We will refer to such a Q matrix as "random Q". Note that the choice of synthetic input data allows us to carefully evaluate how the heuristic performance depends on properties of the input data. This would be hard to do with a real dataset.

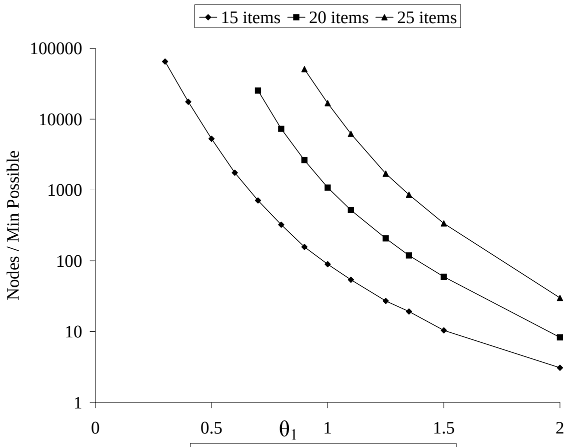
For each configuration of input parameters, we repeated the experiment 25 times i.e., we generated 25 different Q matrices and used each as input. The values reported are the mean value over 25 runs. Our implementation was in Java. It had access to 1500 MB memory and ran on a 2.8 GHz Linux machine.

Figure 3 evaluates the tractability of the search when employing our proposed heuristic. The top plot measures the number of nodes generated as a multiple of the minimum possible number which is $n(n+1)/2$. The search generates this minimum number when it follows a straight path from root node to goal, thereby expanding exactly $(n-1)$ nodes. As θ_1 increases, the number of nodes generated decreases exponentially. Thus, the more consensus there is in the input rankings, the better the performance of the heuristic. For large enough θ_1, the search will be optimal and generate the minimum possible number of nodes. When θ_1 becomes small enough, the search runs out of memory. Note that the complete search tree does not fit in memory even for $n = 15$. Thus, the search would fail without the heuristic. The bottom plot of Figure 3 shows the total running time and follows a similar trend, as we would expect.

We now compare the quality of the estimates given by our method with those given by other existing methods. In particular, we compare with the greedy algorithm of [Cohen et al., 1999] and the local search method of [Fligner and Verducci, 1988], both of which we described previously. We will denote them by "Greedy" and "FV" respectively. We denote our method by SRT (Search over Rankings Tree).

We compare the log likelihood of the models estimated by the different methods. For this experiment, we use the random Q matrix as input instead of the model Q. It represents a more challenging dataset and is extreme in several ways. First, it can be shown that in general such a matrix cannot be obtained from a set of full rankings. Second, as every element of Q is a sample from the uniform distribution over $[0, 1]$, the $\bar{V_j}$ values that are computed during the search represent sums of $n' = (n-j)$ independent random variables.[1] Hence, for all but the smallest n' values, in fact

[1]The $\bar{V_j}$ values computed on various branches of the search tree are *not independent* but each $\bar{V_j}$ separately is composed of independent random variables.



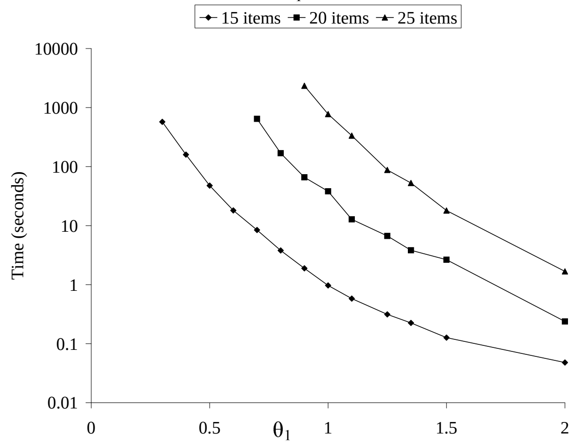
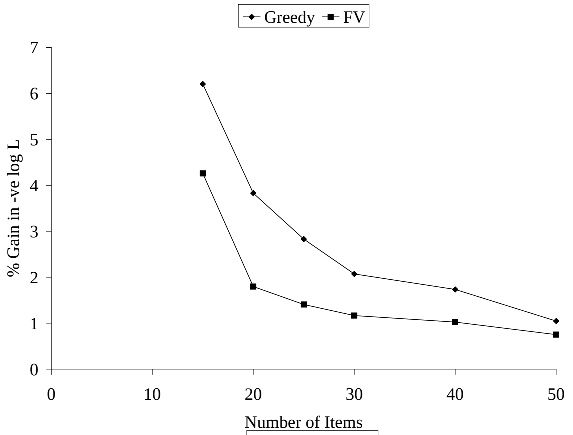


Figure 3: Top: Number of search tree nodes generated, as a multiple of the minimum possible number. Bottom: Running time for the A* search.



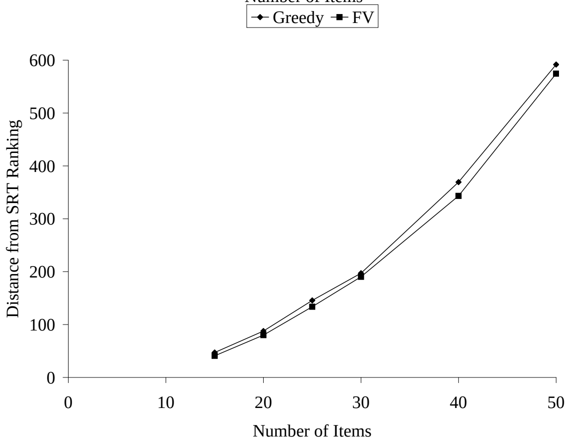


Figure 4: Top: Percentage increase in negative log likelihood obtained from other methods compared to that given by SRT. Bottom: Kendall distance of the ranking π_0 obtained from other methods from the π_0 given by SRT.

for n' around 10 or larger, $\bar{V_j}$ will obey the Central Limit Theorem and will be distributed around $n'/2$ with standard deviation $\sqrt{n'}/2$. In our experiments, the range of $\bar{V_j}$ at a branch of the search tree was of the order of one standard deviation. For example, for $n = 50, n-1 = 49$ the range of the $\bar{V_1}$ at the first branching was $[49/2 \pm 3.5] = [21, 28]$ out of a possible range $[0, 49]$. Values of $\bar{V_j}$ near $(n-j)/2$ induce θ_j estimates near 0. Adding to this the fact that the function $f_j(\theta_j, \bar{V_j})$ varies slowly around $(0, (n-j)/2)$ leads to the conclusion that the range of values for the log likelihood $\sum_{j=1}^{n-1} f_j$ for random Q is very small. Further, it shrinks with increasing n at a rate of $1/\sqrt{n}$. Thus, the difference in log likelihood between the optimal π_0 and the worst is small and decreases as n increases. We give this explanation to show that if the advantage of SRT decreases with increasing n, this is at least partly the reason.

In order to compare over a larger range of n values, we modify SRT to revert to beam search when it runs out of memory rather than not return any solution. The top plot of Figure 4 shows how much the negative log likelihood for π_0 given by other methods exceeds that given by SRT. As n increases, the degree to which it outperforms decreases as we explained above. The bottom plot of Figure 4 shows that though the log likelihood values for the other methods are only a few percent higher, they actually represent π_0 estimates that are quite far from the π_0 given by SRT.

While this section has focused on comparing SRT to its potential competitors, it is natural to ask how much the computationally expensive lower bound H helps the A* search. Hence, we tried to run the search with the trivial zero lower bound. The results were off the chart, and the algorithm did not terminate even for $n = 15$.

6.2 Validation on Real Data

The search for the GM model is computationally more expensive compared to the Mallows model due to the more complicated cost function. Here we look to determine if this extra cost is justified.

Table 1: Negative log likelihood of the Mallows and GM models on Neftochim data from 5-fold CV

	-Log Likelihood	
	GM	Mallows
Group 1	7.370	8.037
Group 2	7.367	7.676
Group 3	7.965	8.001
All Groups	7.413	8.043

Our data consists of rankings of seven different concerns regarding the construction of the Neftochim chemical plant in Bulgaria in 1999. They were provided by three groups of participants. The first group consisted of 193 people living in towns close to the proposed plant. The second group consisted of 47 people in a town further away than the first group. The final group consisted of 20 experts which included engineers and environmentalists. We would expect to see consensus in the rankings provided by different groups, and the Mallows and GM models to fit this data well.

For each group separately and for all groups together, we did ML estimation of both the Mallows and GM models using SRT, and compared their goodness of fit. We did 5-fold cross-validation, and computed the negative log likelihood per ranking. Table 1 shows that the GM model clearly provides a better fit for this data than the Mallows model. Further, the other methods (Greedy and FV) failed to find the ML π_0 even in this simple setting.

7 Conclusions

While ML estimation of the Mallows model has been well-studied as the rank aggregation or Kemeny ranking problem, the estimation of the GM model has been virtually neglected since the original work of [Fligner and Verducci, 1986]. However, the GM model often fits data much better than Mallows due to its generality. For example, it can model the scenario of there being high consensus about the top ranked items, and little about the others.

The A* search method we proposed in [Meila et al., 2007] for learning these two models was ineffective for the GM model in the absence of a suitable heuristic function. This paper provides the first non-trivial admissible heuristic for the GM model search. As the reader has seen, this cost function is considerably more complex than that for the Mallows model. Consequently, developing our lower bound H required the combination of techniques from convex analysis and approximation algorithms for graphs. Moreover, our method strikes a reasonable tradeoff between computational cost and accuracy.

Our method turns the A* search from a theoretical tool into a tractable algorithm that is exact with high probability. While it remains significantly more computationally expensive than the other approximate methods, we experimentally showed that it outperforms them in accuracy. Finally, it is more general and has the potential to extend to missing data, Bayesian inference and other related problems.

References

Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. In *STOC*, pages 684–693, New York, NY, USA, 2005. ACM.

J.J. Bartholdi, C.A. Tovey, and M.A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

Benjamin E. Birnbaum and Kenneth J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing lps. In *APPROX-RANDOM*, pages 49–60, 2006.

William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. Artif. Intell. Res. (JAIR)*, 10:243–270, 1999.

Vincent Conitzer, Andrew J. Davenport, and Jayant Kalagnanam. Improved bounds for computing Kemeny rankings. In *AAAI*, 2006.

Cynthia Dwork, S. Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

Michael A. Fligner and Joseph S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society*, 1986.

Michael A. Fligner and Joseph S. Verducci. Multistage ranking models. *Journal of the American Statistical Association*, 83, 1988.

Michael A. Fligner and Joseph S. Verducci, editors. *Probability Models and Statistical Analyses for Ranking Data.* Springer-Verlag, 1993.

Guy Lebanon and John Lafferty. Conditional models on the ranking poset. In *Advances in NIPS*, 2003.

C. L. Mallows. Non null ranking models. *Biometrika*, 1957.

Marina Meila, Kapil Phadnis, Arthur Patterson, and Jeff Bilmes. Consensus ranking under the exponential model. In *UAI*, 2007.

David M. Pennock, Eric Horvitz, and C. Lee Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *AAAI/IAAI*, pages 729–734, 2000.

R.T. Rockafellar. *Convex Analysis.* Princeton University press, 1972.