Consensus ranking under the exponential model

Marina Meilă
Department of Statistics
University of Washington
Seattle, WA 98195

Kapil Phadnis   Arthur Patterson
University of Washington
Seattle, WA 98195

Jeff Bilmes
Department of Electrical Engineering
University of Washington
Seattle, WA 98195

UW Statistics Department Technical Report 515
April 1,2007

Abstract

We analyze a popular exponential model over rankings called the generalized Mallows model. Estimating the central ranking (or consensus ranking) of this model from data is NP-hard. We obtain the following new results: (1) We show that a standard search method can estimate both the central ranking π_0 and the model parameters θ exactly. The search is $n!$ in the worst case, but is tractable when the true distribution is concentrated around its mode. (2) From a statistical point of view, we show that the generalized Mallows model is jointly exponential in (π_0, θ), and introduce the conjugate prior for this model class. (3) The sufficient statistics are the pairwise marginal probabilities that item i is prefered to item j. These probabilities are of interest in various applications. This paper provides the first exact tractable algorithm for their evaluation. Preliminary experiments confirm the theoretical predictions and compare the new algorithm and existing heuristics.

1 Introduction

Assume that we are given a set of N rankings, a.k.a *linear orderings* on n objects. For instance, the rankings represent the individual preferences of a panel of N judges, each presented with the same set of n candidate objects. The problem of *rank aggreegation* or of finding a *consensus ranking*, is formulated as finding a single ranking π_0 that best agrees with all the N rankings.[1]

Various measures of agreement have been proposed, and a good overview of these can be found in [5]. Of these, Kendall's metric [10] has been the measure of choice in many recent applications centered on the analysis of ranked data [1, 4, 11]. The Kendall distance is defined as

$$d_K(\pi, \pi_0) \; = \; \sum_{l \prec_\pi j} 1_{[j \prec_{\pi_0} l]} \tag{1}$$

In the above, π, π_0 represent permutations and $i \prec_\pi j$ ($i \prec_{\pi_0} j$) mean that l precedes j (i.e is preferred to j) in permutation π (π_0). Hence d_K is the total number of pairwise disagreements between π and π_0.

This distance was further generalized to a family of parametrized distances $d_\theta(\pi, \pi_0)$ depending on a parameter vector θ by [10]. Based on these distances defining probabilistic models of the form $P(\pi) \;\propto\; e^{-d_\theta(\pi,\pi_0)}$ is immediate. Estimating π_0 by e.g Maximum Likelihood (ML) is equivalent to finding the consensus ranking. The estimation problem in such a model is the focus of the present paper.

In section 2 we present the statistical model over rankings. Section 3 and discusses the estimation problem and prior work on it. Section 4 introduces an algorithm that simultaneously and exactly estimates (π_0, θ), but which is only tractable under a favorable data distribution.The computational properties algorithm are discussed in section 5. We analyze the estimation from a statistical point of view in section 6, showing that the model can be estimated from a set of $n(n-1)/2$ sufficient statistics, and therefore that it is jointly exponential in (π_0, θ). The conjugate prior for this model is introduced there. Section 7 looks at the inverse problem of estimating the expectation of the sufficient statistics from the model parameters. In sections 8 and respectively 9 we present experimental evaluations and a final discussion.

2 Background: Generalized Mallows models

This section follows the excellent paper of [10] to which we refer the reader interested more details. Let π denote a permutation over the set $[n] = \{1, 2, 3 \ldots n\}$, with $\pi(l)$ denoting the rank of l in π and $\pi^{-1}(j)$ denoting the

[1]In the rest of the paper, we will use the terms *permutation*, *ranking* and *linear order* interchangeably.

value in position j of π. One can uniquely determine any π by the $n-1$ integers $V_1(\pi)$, $V_2(\pi)$, $\ldots$ $V_{n-1}(\pi)$ defined as

$$V_j(\pi) \;=\; \sum_{l>j} 1_{[l \prec_\pi j]} \tag{2}$$

In other words, V_j is the number of elements in $j+1:n$ which are ranked before j by π. It follows from the above that V_j takes values in $\{0, \ldots n-j\}$. In [7] it is shown that the numbers V_j are uniformly distributed if π is sampled uniformly.

We say that a distance between permutations $d(\pi, \pi_0)$ is *right-invariant* if $d(\pi\overline{\pi}, \pi_0\overline{\pi}) = d(\pi, \pi_0)$ for any permutation $\overline{\pi}$. Requiring that a distance is right invariant means that we want it to be indifferent to the relabeling of the n objects, and it is a standard assumption in working with permutations. For any right-invariant d, we have $d(\pi, \pi_0) = d(\pi\pi_0^{-1}, \mathrm{id})$ and therefore the distance is completely determined by the function

$$D(\pi) \;\stackrel{def}{=}\; d(\pi, \mathrm{id}) \tag{3}$$

where id denotes the identical permutation id $= (1,\, 2, \ldots\, n)$.

2.1 Generalized Kendall distance

From (1) and (2) it is easy to see that the Kendall distance has a simple expression

$$D_K(\pi) \;=\; \sum_{j=1}^{n-1} V_j(\pi).$$

Therefore, [10] proposed the parametrized generalization of the Kendall distance defined by

$$D_\theta(\pi) \;=\; \sum_{j=1}^{n-1} \theta_j V_j(\pi), \qquad \theta_j \geq 0 \tag{4}$$

where $\theta = (\theta_{1:n-1})$ is a parameter vector.

The Kendall distance is a metric [12]. The generalization (4) may be assymmetric unless θ_j is constant independent of j. Therefore, in general, d_θ is not a metric.

D_θ is a versatile and convenient measure of divergence between rankings. By chosing the θ parameters to e.g. decrease with j we can emphasize the greater importance of ranking the first items in π_0 correctly relative to the

Table 1: Algorithm PiFromV. The permutation is represented by a list π indexed from $0 : n-1$. The numbers in $[n]$ are inserted into π in reverse order.

```
Algorithm PiFromV

Input Valid set of values $V_{1:n-1}$
0. Create an empty list $\pi$
1. Insert $n$ in position 0.
2. For $j = n-1 : 1$ in decreasing order
     Insert $j$ in position $V_j$ of $\pi$
Output $\pi$
```

correct ranking of items with low ranks in π_0. Variations of this model where the "emphasized ranks" j can be selected at will are also possible.

There are several equivalent algorithms for the reconstruction of π from $V_{1:n-1}$. Before we go on, we present one of them in table 1. Figure 1 shows how the algorithm works by an example. This algorithm will be used in the proof of some later results.

2.2 Generalized Mallows models

The following family of exponential models based on the divergence (4) is called the *generalized Mallows model* [10]

$$P_{\theta,\pi_0}(\pi) \;=\; \frac{e^{-d_\theta(\pi,\pi_0)}}{\psi(\theta)} \;=\; \frac{e^{-D_\theta(\pi\pi_0^{-1})}}{\psi(\theta)} \tag{5}$$

In the above, $\psi(\theta)$ is a normalization constant that does not depend on π_0. It was shown in [10] that the model (5) factors into a product of independent univariate exponential models, one for each V_j and that

$$\psi(\theta) \;=\; \prod_{j=1}^{n-1} \psi_j(\theta_j) \;=\; \prod_{j=1}^{n-1} \frac{1-e^{-(n-j+1)\theta_j}}{1-e^{-\theta_j}} \tag{6}$$

$$P[V_j(\pi\pi_0^{-1}) = r] \;=\; \frac{e^{-\theta_j r}}{\psi_j(\theta_j)} \tag{7}$$

The above models are well defined for any real values of the parameters θ. However, we will be forthwith interested only in the values $\theta_j \geq 0$, for

$$\begin{array}{lcllllll} \pi^{-1} & = & (6 & 1 & 3 & 5 & 2 & 4) \\ V & = & 1 & 3 & 1 & 2 & 1 & \end{array}$$

Reconstructing π from V						
	0					
insert 6	**6**					
	0	1				
insert 5	6	**5**				
	0	1	2			
insert 4	6	5	**4**			
	0	1	2	3		
insert 3	6	**3**	5	4		
	0	1	2	3	4	
insert 2	6	3	5	**2**	4	
	0	1	2	3	4	5
insert 1	6	**1**	3	5	2	4

Figure 1: Reconstruction of π from $V_{1:n-1}$ an example.

which the probability distribution has a maximum at $V \equiv 0$. This case corresponds to a distribution over orderings where all the high probability instances are small perturbations of the central permutation. For $\theta \equiv 0$, $P_\theta \equiv P_0$ is the uniform distribution. For $\theta_1 = \theta_2 = \ldots = \theta_{n-1}$ (5) is the *Mallows model* [12]. The size of the θ parameters controls the concentration of the distribution around its mode π_0; smaller values make the distribution closer to uniform, while larger values make it more concentrated.

The case $\theta_j \leq 0,\ j = 1 : n-1$ corresponds to the case where the π_0 is a minimum of the distribution, and the high probability cases are concentrated around its reverse. The cases for which the parameters θ have different signs are less interpretable and we do not consider them here.

3 The ML estimation problem

3.1 Parameter estimation.

Assume an independent sample $\pi_{1:N}$ of size N has been obtained from model (5). Then the data log-likelihood can be written as

$$\begin{aligned} l(\theta, \pi_0) &= \ln P(\pi_{1:N}; \theta, \pi_0) && (8)\\ &= -\sum_{i=1}^{N}\sum_{j=1}^{n-1} \theta_j V_j(\pi_i \pi_0^{-1}) - N \sum_{j=1}^{n-1} \ln \psi_j(\theta_j) && (9)\\ &= -N \sum_{j=1}^{n-1} \left[\theta_j \frac{\sum_{i=1}^{N} V_j(\pi_i \pi_0^{-1})}{N} - \ln \psi_j(\theta_j) \right] && (10)\\ &= -N \sum_{j=1}^{n-1} \left[\theta_j \bar{V}_j - \ln \psi_j(\theta_j) \right] && (11) \end{aligned}$$

In the above

$$\bar{V}_j = 1/N \sum_{i=1}^{N} V_j(\pi_i \pi_0^{-1}) \qquad (12)$$

is the sample expectation of $V_j(\pi\pi_0^{-1})$. It is easy to see that for any fixed π_0 the model (5) is an exponential family model [6] with parameters θ. Moreover, because the random variables V_j are independent, each V_j is distributed according to an exponential model with one parameter θ_j. This is reflected in equation (11) where the log-likelihood l decomposes into a sum of terms, each depending on a single θ_j.

Maximizing the log-likelihood to estimate θ when π_0 is known is therefore immediate. It amounts to solving the implicit equation in one variable obtained by taking the partial derivative w.r.t. θ_j in equation (11).

$$\bar{V}_j = \frac{\partial}{\partial \theta_j} \psi_j(\theta_j) \qquad (13)$$

As in [10], for each $j = 1 : n-1$, this equation is rewritten

$$\bar{V}_j = \frac{1}{e^{\theta_j} - 1} - \frac{n-j+1}{e^{(n-j+1)\theta_j} - 1}, \quad j = 1 : n-1 \qquad (14)$$

Note that $l(\theta, \pi_0)$ is log-concave in θ. Hence equation (14) has a unique solution for any j and any $\bar{V}_j \in [0, n-j]$ (see e.g [10]). This solution has in general no closed form expression, but can be obtained numerically by standard iterative algorithms for convex/concave optimization [3].

3.2 The centroid estimation problem

In the following we study the combinatorial problem of estimating the unknown mode π_0.

Let us note that finding a permutation in a discrete set can in theory be done by trying all $n!$ permutations. This is feasible for small n; $5! = 120$ is certainly no obstacle and $9! \leq 500,000$ is also within tractability limits.

for larger n various heuristic solutions have been proposed, one of the most effective being that of [8]. We briefly describe it here, but before we do so we introduce a summary of the data, which will prove pivotal to our findings. This is the matrix $Q(\pi_{1:N})$ defined as

$$Q_{jl}(\pi_{1:N}) \;=\; \frac{1}{N}\sum_{i=1}^{N} 1_{[j \prec_{\pi_i} l]} \quad \text{for}\, j,l = 1:n \tag{15}$$

In other words, $Q_{jl}(\pi_{1:N})$ is the probability that j precedes l in the sample. In the rest of the paper, when no confusion is possible, we will denote $Q(\pi_{1:N})$ simply by Q. Another useful notation will be $Q(\pi)$ which is the Q matrix corresponding to a single permutation π. The elements of $Q(\pi)$ are $\{0,1\}$ valued while the elements of $Q \equiv Q(\pi_{1:N})$ are rational numbers for any finite N.

The [8] heuristic (FV heuristic) can be described in terms of Q. Let

$$\bar{q}_l = \sum_{j=1}^{n} Q_{jl} \tag{16}$$

$\bar{q}_l$ is one less than the average rank of l in the data. Let $\bar{\pi}_0$ denote the permutation given by sorting the $\bar{q}_l$ values in increasing order. In [8] it is argued that this permutation is an unbiased estimator of π_0.

The FV heuristic starts with this permutation, plus the set of all its neighbors at $d_K = 1$; for each of these candidates, the parameters θ are estimated and the data likelihood computed. The most likely π_0 of the set is then chosen.

For $\theta_1 = \theta_2 = \ldots = \theta_{n-1} \geq 0$ (the Mallows model) the optimal π_0 does not depend on θ and the problem becomes one of finding

$$\pi_0 \;=\; \operatorname*{argmin}_{\pi'} \sum_{i=1}^{N} d_K(\pi_i, \pi') \tag{17}$$

This is precisely the consensus ranking problem. It is known that this problem is NP-hard [2], and solving it approximately has received attention in the computer science and machine learning literature. The approximation algorithm that guarantees best theoretical bounds is that of [1]; this is a randomized algorithm that achieves a factor 11/7 approximation in minimizing the r.h.s of (17).

In [4] a greedy heuristic (the CSS greedy algorithm) based on graph operations is introduced and tested. The heuristic works under slightly more general conditions, as it assumes that not all of the n items are ranked under all permutations π_i. A good discussion of the sources of difficulty for this problem is also given. This greedy heuristic achieves a factor 2 approximation. We will return to the CSS heuristic in sections 8 and 9.

Interestingly enough, none of the above works tie the concentration of the distribution to the hardness of the problem. However, intuitively, the problem should not be difficult if P_{θ,π_0} is concentrated around π_0. It is also intuitive that if the distribution is uniform, then *any* permutation will be equally qualified to be the mode. The next section exploits exactly this observation.

4 Exact ML estimation for π_0

4.1 Estimation of π_0 for θ known.

Maximizing the log-likelihood (11) w.r.t π_0 is the same as minimizing

$$\sum_{j=1}^{n-1} \theta_j \bar{V}_j \tag{18}$$

The following key observation allows us to do so. Let us denote for simplicity

$$\tilde{V}_j(\pi) \;=\; V_j(\pi\pi_0^{-1}) \tag{19}$$

If $\pi_0^{-1}(1) = r$, then $\tilde{V}_1(\pi)$ is the number of elements which come before r in π. $\bar{V}_1$, the expectation of $\tilde{V}_1$ under the sampling distribution, is the expected number of elements before r. Therefore we have:

$$\bar{V}_1 \;=\; \sum_{j\neq r} Q_{jr} \quad \text{whenever } \pi_0^{-1}(1) = r \tag{20}$$

If we only wanted to estimate the first element of π_0 we could do so using the above equation. All we have to do is to compute all column sums of Q and then choose $\pi_0^{-1}(1) = \underset{r}{\operatorname{argmin}} \sum_{l\neq r} Q_{lr}$.

This idea can be generalized by induction to all subsequent j's. Assume $\pi_0^{-1}(1) = r_1$ fixed and denote by $\pi|_{-r_1}$ the permutation over $n-1$ elements resulting from π if r_1 was removed. Then, $\tilde{V_2}(\pi)$ represents the number of items that precede $\pi_0^{-1}(2)$ in $\pi|_{-r_1}$. By averaging, it follows that

$$\bar{V_2} \ = \sum_{l \neq r_1, r_2} Q_{lr_2} \text{ whenever } \pi_0^{-1}(1:2) = (r_1, r_2) \tag{21}$$

By induction, we obtain[2]

$$\bar{V_j} \ = \sum_{l \neq r_1, r_2, \ldots r_j} Q_{lr_j} \text{ whenever } \pi_0^{-1}(1:j) = (r_1,\, r_2, \ldots,\, r_j) \tag{22}$$

It is clear from the above that in Q we have the information necessary to find the π_0 maximizing the likelihood and that an exhaustive search over all the possible permutations will obtain it.

One can represent this search a as *search tree*, whose nodes represent partial orderings $\rho = (r_1, \ldots r_j)$. Denote by $|\rho|$ the length of the sequence ρ. A node ρ with $|\rho| = j$ has $n-j$ children, represented by the sequences $\rho' = \rho | r_{j+1}$ where the symbol $|$ stands for concatenation of sequences and r_{j+1} ranges in $[n] \setminus \rho$ the set complement of ρ in $[n]$. Any path of length n through the tree starting from the root represents a permutation. It is easy to check that the tree contains exactly $n!$ nodes. A node $(r_1, \ldots r_j)$ at level $j < n$ can be thought of as the set of all permutaions that start with $r_1, \ldots, r_j$.

We define the variables of a search algorithm

$$V_j(r_1, r_2, \ldots r_j) \ = \sum_{l \notin \{r_1, r_2, \ldots r_j\}} Q_{lr_j} \tag{23}$$

The *cost* at node $\rho = (r_1, \ldots r_j)$ is given by

$$C(r_1, \ldots r_j) \ = \ \sum_{l=1}^{j} \theta_l V_l(r_1, \ldots r_l) \tag{24}$$

This cost can be computed recursively on the tree by

$$C(r_1, \ldots r_j) \ = \ C(r_1, \ldots r_{j-1}) + \theta_j V_j(r_1, \ldots r_j) \tag{25}$$

[2]We point out that because $Q_{rr} = 0$ for all $r = 1:n$, the conditions $j \neq r$ in (20), $l \neq r_2$ in (21) and $l \neq r_j$ in what follows are superfluous and could be dropped. We prefer to keep them because this notation is in better agreement with the view of the estimation algorithm this paper is developing.

Table 2: The SearchPi algorithm outline. Node ρ stores: $\rho = r_1, \dots, r_j$, $j = |\rho|$, $C(\rho)$, $L(\rho)$; S is the set of nodes to be expanded.

```
**Initialize**
    $S = \{\rho_\emptyset\}$, $\rho_\emptyset$ = the empty sequence, $j = 0$, $C(\rho_\emptyset) = L(\rho_\emptyset) = 0$
**Do**
remove from $S$ $\rho = \underset{\rho \in S}{\operatorname{argmin}} L(\rho)$
if $|\rho| = n$ *(Return)*
    **Output** $\rho$, $L(\rho) = C(\rho)$ and **Stop**.
else *(Expand $\rho$)*
    for $r_{j+1} \in [n] \setminus \rho$
        create node $\rho' = \rho | r_{j+1}$
        $V_{j+1}(\rho') = \sum_{l \in [n] \setminus \rho'} Q_{l r_{j+1}}$
        calculate $C(\rho')$, $A(\rho')$
        $L(\rho') = C(\rho') + A(\rho')$
        store node $(\rho', j+1, C(\rho'), A(\rho'))$ in $S$
```

The tree nodes are expanded according to a search strategy like A^* . To direct the search, one also needs a lower bound $A(r_1, \dots r_j)$ on the *cost to go* from the current partial solution. We will describe possible bounds in the next section. The sum $L(\rho) = C(\rho) + A(\rho)$ represents a lower bound for any permutation in the set represented by ρ. In such a tree, search can finish with the optimal solution before the whole tree is expanded. Table 3 gives the algorithm, an A^* *Best-First (BF)* search algorithm, in pseudocode.

4.2 Simultaneous estimation of π_0 and θ.

Algorithm SearchPi is immediately extended to the more interesting case when both the centroid π_0 and the parameters θ are unknown. It is sufficient to recall that for any fixed π_0 the model (5) is an exponential family model and thus the parameter estimates depend only on the sufficient statistics $\bar{V}_{1:n-1}$. Moreover, the estimate θ_j^{ML} depends only on $\bar{V}_j$. Hence, any time a node ρ in the search tree is created, $\theta_{|\rho|}^{ML}$ can be readily computed at the node by solving (14) with $\bar{V}_j = V_{|\rho|}(\rho)$.

As mentioned before, this equation does not generally have a closed form

Table 3: The SearchPi algorithm with an admissible heuristic A. Node ρ stores: $\rho = r_1, \ldots, r_j$, $j = |\rho|$, $V_j(\rho)$, θ_j, $C(\rho)$, $L(\rho)$; S is the priority queue holding the nodes to be expanded.

```
Initialize
    $S = \{\rho_\emptyset\}$, $\rho_\emptyset$ =the empty sequence, $j = 0$, $C(\rho_\emptyset) = V(\rho_\emptyset) = L(\rho_\emptyset) = 0$
Do
remove $\rho = \operatorname*{argmin}_{\rho \in S} L(\rho)$ from $S$
if $|\rho| = n$ (Return)
    Output $\rho$, $L(\rho) = C(\rho)$ and Stop.
else (Expand $\rho$)
    for $r_{j+1} \in [n] \setminus \rho$
        create node $\rho' = \rho | r_{j+1}$
        $V_{j+1}(\rho') = V_j(r_{1:j-1}, r_{j+1}) - Q_{r_j r_{j+1}}$
    compute $V^{min} = \min_{r_{j+1} \in [n] \setminus \rho} V_{j+1}(\rho | r_{j+1})$
    calculate $A$
    for $r_{j+1} \in [n] \setminus \rho$

        $\theta_{j+1} = t_{n-j-1}(V_{j+1}(\rho'))$
        $C(\rho') = C(\rho) + \theta_{j+1} V_{j+1}(\rho')$
        $L(\rho') = C(\rho') + A$
        store node ($\rho'$, $j+1$, $V_{j+1}$, $\theta_{j+1}$, $C(\rho')$, $L(\rho')$) in $S$
```

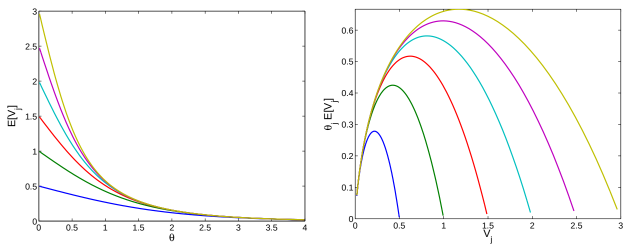

Figure 2: The dependence of $\bar{V}_j$ (left) and $\theta_j \bar{V}_j$ (right) on θ_j, for $n - j = 1 : 6$. The height of the curves increases with $n - j$.

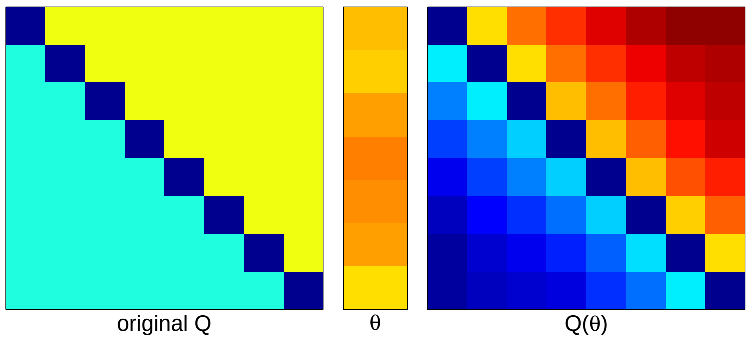

Figure 3: A matrix Q (left), the θ^{ML} vector it determines ($\pi_0 = \text{id}$) and the $E_{P_{\theta,\text{id}}}[Q(\pi)]$ matrix determined by θ^{ML}. Throughout this paper, all matrix elements are displayed on the same colorscale of $[0, 1]$.

solution. However, the values θ can be tabulated as a function of $\bar{V}$. The value of θ_j^{ML} in (14) depends only on $\bar{V}_j$ and $n - j$. Therefore, the curve $\bar{V}_{n-j}(\theta)$, and consequently its inverse which we denote $t_{n-j}(\bar{V})$ depend only on $n - j$ and not on n. This set of curves, one for each value of $n - j$ is computed off-line once and then used for any data with n up to a preset maximum value. Figure 2 shows a few of these curves.

5 Computational aspects

5.1 Admissible heuristics

We now describe possible functions $A(\rho)$ to be used in place of the cost to go. Such a function needs to satisfy two conditions: to be easily computable, and to lower bound the true cost to go. The simplest heuristic is evidently $A(\rho) = 0$.

5.1.1 Admissible heuristic for V with known θ

If the parameters θ are known, then we only need to find lower bounds on the $V_{j'}$ values for $j' > j$. They can be derived in the following way. When node ρ is expanded, after computing the V_{j+1} values for all its chidren, we find the minimum of these values

$$V^{min} \;=\; \min_{r_{j+1}\in[n]\setminus\rho} V_{j+1}(\rho|r_{j+1}). \tag{26}$$

For $j+1 < j' < n-1$ the best $V_{j'}$ on the current branch are column sums of submatrices of $Q_{[n]\setminus\rho}$, i.e

$$\begin{aligned} V_{j'}(\rho|r_{j+1},\dots r_{j'}) &= \sum_{i\in Q_{[n]\setminus\rho}} Q_{ir_{j'}} - \sum_{i\in\{r_{j+1}\dots r_{j'}\}} Q_{ir_{j'}} \\ &\geq \max[V^{min} - (j'-j)Q^{max}, 0] \;=\; a_{j'}(\rho) \end{aligned}$$

where $Q^{max} = \max_{jl} Q_{jl}$ is computed off line. Then A can be computed as $A(\rho) = \sum_{j'=j+1}^{n-1} \theta_{j'} a_{j'}(\rho)$.

5.1.2 Admissible heuristic for V with constant θ

For the special case of consensus ranking, when $\theta_j \equiv 1$, an even better heuristic can be used. Sort the off-diagonal values of Q_{lr} in increasing order, denoting the resulting sequence by

$$q_{(1)} \leq q_{(2)} \leq \;\dots\; \leq q_{(n(n-1)/2)} \tag{27}$$

The cost to go in consensus ranking is independent of θ and equal to $V_{j+1}(\rho'_{j+1}) + V_{j+2}(\rho'_{j+2}) + \dots V_{n-1}(\rho'_{n-1})$ on some (unknown) path from the current node to the bottom of the tree. Since each V_j is the sum of $n-j$ off-diagonal Q_{lr}'s, this cost to go is equal to the sum of $(n-j-1)(n-j-1+1)/2$ distinct off-diagonal elements of Q. Hence

$$A(\rho) \;=\; \sum_{l=1}^{(n-j-1)(n-j)/2} q_{(l)} \tag{28}$$

is always lower bounding the cost to go. This heuristic depends only on the level j and can be entirely computed before the search.

Variations of this idea for the case where the θ_j parameters are not constant or not known are also possible.

5.1.3 Admissible heuristics for unknown θ

If the parameters θ_j are estimated simultaneously with the central permutation π_0, then lower bounding the cost to go requires us to find lower bounds on the parameters $\theta_{j'}$, with $j' > j = |\rho|$ the current level.

Any non-zero lower bound on $\theta_{j'}$ can then be combined with the lower bounds on $V_{j'}$ described in the previous sections to produce an admissible A. The derivation of possible lower bounds for the parameters is delayed until section 7. In this case too, the bounds will be computed off-line and will depend only on the tree level j.

5.2 Number of node expansions

Let us further analyze the algorithm SEARCHPI from a computational point of view. BF algorithms with admissible heuristics are guaranteed to find the optimal solution given enough time. The stopping condition is met when the most promising node is a terminal node. This condition can be met before all nodes in the search tree are expanded. Hence, an important performance parameter for a BF algorithm is the number of nodes that it visits before it finds the optimum. This number clearly depends on the quality of the heuristic – the better a lower bound is A on the cost to go, the more nodes can be pruned from the search tree.

In our case, the worst case runnig time will be $n!$. The lower limit on the number of nodes created, given by the path of the greedy search strategy, is $n + (n-1) + \ldots + 2 = n(n+1)/2 - 1$. The number of nodes expanded by the greedy strategy is one in each level, i.e $n-1$ nodes.

A qualitative examination of the cost (24) shows that the larger the value of θ_j, the greater the advantage of the best r_j w.r.t the second best. Hence, large values of θ_j imply that the chance of a non-optimal subtree at level j to contain the optimal solution is small. In other words, when the values of the parameters are large, which corresponds to a distribution P_θ that decays fast away from the mode π_0, then the number of nodes explored will be small. For any admissible heuristic A, there are parameters θ^{ML} for which the BB algorithm will explore exactly the same nodes as the greedy algorithm and no more.

At the other end of the spectrum, if $\theta_j \approx 0$ for all j, the search is likely to be intractable. In this case are data sets sampled from an almost uniform distribution, which will have all values $Q_{lr} \approx 0.5$. Data sampled from multimodal distributions can also fall under this category[3]. For multimodal

[3]In this case, since there is no true parameter θ, we refer to the estimated θ^{ML}

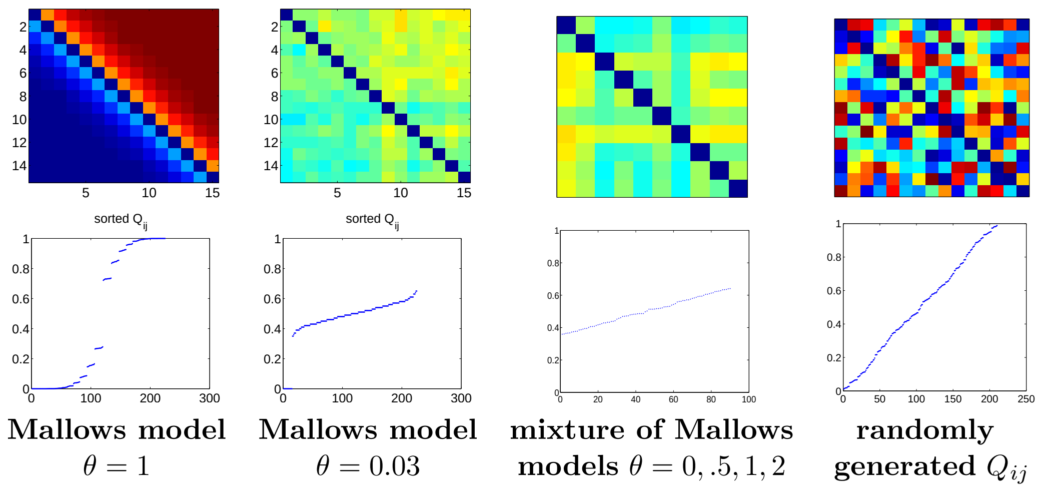

Figure 4: Various Q matrices (top row) and the plot of their sorted off-diagonal values (bottom row).

distributions, individual Q_{lr} values can take extreme values near 0 or 1, but because no consensus exists, the average Q_{lr} along a column or subcolumn will be near 0.5 as well. Examples of these cases can be seen in figure 4.

In this latter case, the algorithm can be stopped any time, and it will provide the best solution it was able to find so far. For this case, practical optimization usually involves inadmissible heuristics (e.g. beam search). We leave this avenue open for further research.

5.3 Number of operations per node.

Upon creating node $\rho' = \rho | r_{j+1} = r_1, \ldots, r_{j+1}$, the SEARCHPI algorithm needs to compute the value of $\bar{V}_{\rho'} = \sum_{l \in [n] \setminus \rho} Q_{lr_{j+1}}$. Computing this sum explicitly takes $\mathcal{O}(n-j)$ operations, which makes the time of exploring one vertical path to the terminal of a tree be $\mathcal{O}(n(n-1)+(n-1)(n-2)+\ldots+2 = \mathcal{O}(n^3)$. However, by better organizing the data we can obtain a *constant* computation time per node.

$$V_{j+1}(r_1, \ldots, r_{j+1}) = \sum_{l \neq r_1, \ldots, r_{j+1}} Q_{lr_{j+1}} \tag{29}$$

$$= \sum_{l \neq r_{1:j-1}, r_{j+1}} Q_{lr_{j+1}} - Q_{r_j r_{j+1}} \tag{30}$$

$$= V_j(r_1, \ldots, r_{j-1}, r_{j+1}) - Q_{r_j r_{j+1}} \tag{31}$$

The node $(r_1, \ldots, r_{j-1}, r_{j+1})$ is a sibling of $(r_1, \ldots, r_{j+1})$'s parent (hence an "uncle"). In our algorithm, and in any search algorithm that creates all children of a node at once, this node will have been created and its V value available by the time we need to compute $V_{j+1}(r_1, \ldots, r_{j+1})$.

To use this value, we must only make sure that no nodes are deleted from memory while their V values are still needed. This can be achieved with a counter variable associated with each $V_j(\rho)$ which signals when the value is no longer needed. Another possible solution is to pass the V values alone, as tables, down the tree. This way any node can be deleted independently of the rest of the tree. Keeping a table at a node adds a storage of $n - j$ per node.

5.3.1 Selecting the next node

This can be done efficiently if all the nodes are kept in a priority queue sorted by $L(\rho)$. Fibonacci heaps can attain constant access time, while our STL based implementation uses a binomial heap with access time logarithmic in the length of the queue.

6 Identifyability and conjugate prior

6.1 Identifyability

The matrix Q represents the sufficient statistics for the parameters π_0, θ. Because by definition

$$Q_{lj} + Q_{jl} \;=\; 1 \text{ for } l \neq j, \quad Q_{jj} = 0 \tag{32}$$

the number of free parameters in Q is at most $n(n-1)/2$.

The set $\mathcal{Q} = \{Q\}$ of matrices satisfying (32) is a convex polytope, with $n!$ extreme points given by $Q(\pi) = [1_{[l \prec_\pi j]}]_{lj}$. By the Caratheodory theorem [13], any Q in the polytope can be represented by a convex combination of at most $n(n-1)/2 + 1$ extreme points. This implies that Q can be approximated arbitrarily closely by finite data sets with N large enough. So, asymptotically, any $Q \in \mathcal{Q}$ can represent a set of sufficient statistics.

Note also that for any $Q \in \mathcal{Q}$ and for any permutation π_0, there is a unique parameter vector $\theta^{ML}(\pi_0) = \underset{\theta}{\operatorname{argmax}}\, P_{\theta,\pi_0}(Q)$ (because equation (14) has a unique solution). The following result says that for any data set there is a non-negative θ estimate.

Proposition 1 *For any $Q \in \mathcal{Q}$ there exists a permutation π_0, so that $\theta_j^{ML}(\pi_0) \geq 0$.*

Proof. Since $Q_{jl} = 1 - Q_{lj}$ we have $\sum_{jl} Q_{jl} = n(n-1)/2$; therefore there is at least one column r for which $\sum_l Q_{lr} \leq (n-1)/2$. For this column, equation (14) with $j = 1$ will have a non-negative solution θ_1. We now delete column and row r from Q and proceed recursively for $j = 2 : n-1$. $\square$

This proposition justifies our focusing on the domain of non-negative θ. It shows that such a restriction is not only convenient, it is also necessary to ensure that the model is identifyable. A model P_{θ,π_0} with $\theta > 0$ is *strongly unimodal*; in such a model the probability of any inversion w.r.t π_0 is less than 0.5 [8].

While almost[4] each $Q \in \mathcal{Q}$ defines uniquely a pair $\theta^{ML}, {\pi_0}^{ML}$, the converse is not true. There are an infinity of matrices Q which produce the same $\theta^{ML}, {\pi_0}^{ML}$ (see figure 3). This equivalence class has a representative in the matrix $Q_{\theta,\pi_0} = E_{P_{\theta,\pi_0}}[Q(\pi)]$. The matrix Q_{θ,π_0} does not have a closed form expression in terms of θ. But it can be evaluated numerically by the algorithm QExpectation that we introduce in the next section.

6.2 The conjugate prior.

The existence of finite sufficient statistics implies that $P_{\theta,\pi_0}(\pi)$ is an exponential family model jointly in (θ, π_0). As such, it will have a conjugate prior, whose form is given below.

Proposition 2 *Let $\Gamma \in \mathcal{Q}$, $\nu > 0$; denote $\Gamma_\infty = Q(\mathrm{id}) \in \mathcal{Q}$, $\Theta = \mathrm{diag}(\theta, 0) \in \mathbb{R}^{n\times n}$ and Π_0 the permutation matrix associated to permutation π_0. Then*

$$P(\pi_0, \theta\,;\, \nu,\, \Gamma) \;\propto\; e^{-\nu[\mathrm{trace}\,\Gamma_\infty \Pi_0 \Gamma \Pi_0^T \Theta + \ln \psi(\theta)]} \tag{33}$$

is the conjugate prior for the parameters (π_0, θ) of model (5).

Proof. $V_j(\pi\pi_0^{-1})$ can be written as element (j,j) of $\Gamma_\infty \Pi_0 Q(\pi) \Pi_0^T$ and consequently $\ln P_{\theta,\pi_0}(\pi) = \mathrm{trace}\,\Gamma_\infty \Pi_0 Q(\pi) \Pi_0^T \Theta + \ln \psi(\theta)$. Moreover, $NQ = \sum_{i=1}^N Q(\pi_i)$. Hence,

$$\begin{aligned} P(\pi_0, \theta\,|\pi_{1:N}) &\propto P(\pi_{1:N}|\pi_0, \theta) P(\pi_0, \theta\,;\, \nu,\, \Gamma) \\ &\propto e^{-(N+\nu)[\mathrm{trace}\,\Gamma_\infty \Pi_0 \frac{NQ+\nu\Gamma}{N+\nu} \Pi_0^T \Theta + \ln \psi(\theta)]} \\ &= P(\pi_0, \theta;\, N+\nu, \frac{NQ+\nu\Gamma}{N+\nu}) \end{aligned} \tag{34}$$

[4]Except for those Q for which there are ties in π_0.

We have shown that the distribution in (33) is closed under sampling, in other words it is a conjugate prior [6]. It remains to show that the prior is integrable on $\theta_j \geq 0$, $j = 1 : n-1$. Note that taking into account (6) it is sufficient to show that the integral

$$\int_0^\infty \frac{1-e^{-j\theta}}{1-e^{-\theta}} e^{-a\theta} d\theta \; < \; \infty \tag{35}$$

for any integer $j \geq 2$ and any $a > 0$. This is easily seen by rewriting the l.h.s. of (35) as

$$\int_0^\infty [1 + e^{-\theta} + \ldots + e^{-(j-1)\theta}] e^{-a\theta} d\theta \tag{36}$$

which is a sum of j finite integrals for any $a > 0$. From (33) one can see that $a > 0$ for whenever $\nu > 0$ and Γ in the interior of $\mathcal{Q}$. $\Box$

We note that the general form of a conjugate prior family is

$$P(\pi_0, \theta\,;\, \nu,\, \Gamma) \;\propto\; h(\theta, \pi_0) e^{-\nu[\operatorname{trace}\Gamma_\infty \Pi_0 \Gamma \Pi_0^T \Theta + \ln \psi(\theta)]} \tag{37}$$

where $h(\theta, \pi_0)$ is a function that renders the prior integrable and doesn't depend on ν, Γ. Our proposition and its proof extend immediately to this general case as well.

The prior above is defined up to a normalization constant. At present we do not have a closed form formula for this constant. We also stress that the sufficient statistics Q for the model (5) *are not minimal* and the model itself, in the above parametrization, is not a minimal exponential model.

It is interesting, never the less, to interpret the prior's parameters. The ν parameter's role as "equivalent sample size" is obvious; let us now look at the matrix parameter Γ. If Γ represents an expectation matrix $E_{\theta^*,\pi^*}[Q(\pi)]$ under model (5) the conjugate prior is equivalent to having seen ν samples from a distribution centered at π^* with spread θ^*. Such a Γ can be obtained from (θ^*, π^*) by running algorithm QExpectation. However, not every $\Gamma \in \mathcal{Q}$ is an expectation under (5). If one uses another Γ in the prior, that corresponds to having seen ν samples from a distribution not in the class represented by (5).

The matrix Γ_0 obtained from $\theta^* \equiv 0$ has $(\Gamma_0)_{ij} = 0.5$ in each off-diagonal entry. This matrix corresponds to an non-informative prior w.r.t π_0, as $\theta^* \equiv 0$ represents the uniform distribution. Using a conjugate prior with Γ_0 implements a smoothing over the parameters while being non-informative w.r.t the central permutation. It can be easily verified *check this! prove this!* that there is no other $\Gamma \in \mathcal{Q}$ that is non-informative w.r.t π_0. Any other Γ

is informative w.r.t both θ and π_0. Hence, in the conjugate prior framework it is impossible to express ignorance w.r.t to the central distribution, while expressing knowledge about the parameters θ.

From an algorithmic standpoint, working with the conjugate prior is, as expected, straightforward. The full posterior, up to the normalization constant, is obtained as a summation of sufficient statistics and prior parameters. This allows one to compare the posteriors of any two models. If one is interested in the Maximum A-Posteriori (MAP) estimate, this can be readily obtained by algorithm SEARCHPI with Q replaced by $(NQ + \nu\Gamma)/(N + \nu)$.

7 Evaluating $E[Q]$ w.r.t the generalized Mallows model

The previous sections have considered the question of estimating a distribution's θ, π_0 parameters given the values Q_{jl}. Now we address the converse problem, of obtaining the theoretical Q that corresponds to a set of parameters θ, π_0.

Hence, while in (15) Q_{jl} was defined as the observed frequency of the event $j \prec_{\pi_i} l$ in the data set, here it is redefined w.r.t to the theoretical distribution P_{θ,π_0} by

$$Q_{jl}(\theta, \pi_0) \;=\; E_{P_{\theta,\pi_0}}[1_{[j \prec_\pi l]}] \tag{38}$$

W.l.o.g we can assume $\pi_0 = \text{id}$; all other cases can be obtained by applying π_0 to the rows and columns of $Q(\theta, \text{id})$. For $l > j$, $Q_{lj}(\theta, \text{id})$ represents the marginal probability of the inversion (j, l). To our knowledge, the evaluation of this probability hasn't been attempted before and it doesn't have a closed form expression (except for the trivial case $Q_{n,n-1} = V_{n-1}$. We are able to give a recursive expression for all the values Q_{jl} which can be computed in polynomial time.

The following proposition relates $Q_{jl}(\pi)$ to the $V(\pi)$ variables.

Proposition 3 *For any permutation π, any $j \in 1 : n-1$, and any $m \in 1 : n-j$ we have*[5]

$$Q_{j+m,j}(\pi) = 1 \quad \text{iff} \quad V_j > V_{j+m} + \sum_{l=j+1}^{j+m-1} 1_{[V_l \leq V_{j+m}]} \tag{39}$$

[5] We set formally $V_n = 0$ w.p. 1 throughout this section.

Proof According to algorithm PiFromV, $j+m$ is inserted before j in position V_{j+m}. During the insertion steps following $j+m$ and preceding j, namely the insertions of $j+m-1, \ldots j+1$, $j+m$ will move right one place each time $V_l \leq V_{j+m}$, $l = j+m-1, j+m-2, \ldots j+1$. Hence, the position of $j+m$ just before j is inserted is

$$V_{j+m} + \sum_{l=j+1}^{j+m-1} 1_{[V_l \leq V_{j+m}]}, \tag{40}$$

and j is inserted after $j+m$ iff the above number is strictly smaller than V_j. □

To obtain the probability of $Q_{j+m,j} = 1$ under $P_\theta(\pi)$ we need the probability of the r.h.s. A key observation in what follows is that the V_j variables are *independent under* P_θ; therefore we will aim to express the r.h.s of (41) as a conjunction of events that involve disjoint sets of V_l variables.

We have therefore:

$$\begin{aligned} P[Q_{j+m,j} = 1] &= \\ &= \sum_{k=0}^{n-(j+m)} P[V_{j+m} = k] P[V_j - \sum_{l=j+1}^{j+m-1} 1_{[V_l \leq k]} > k] \end{aligned} \tag{41}$$

Define the probability that exactly p of $V_{j+1:j+m}$ are no larger than some k

$$A_m^p(k,j) = P[\sum_{l=j+1}^{j+m} 1_{[V_l \leq k]} = p], \quad p = 0:m \tag{42}$$

These probabilities can be computed recursively by

$$A_m^0(k,j) = (1 - P[V_{j+1} \leq k])(1 - P[V_{j+2} \leq k]) \ldots \tag{43}$$

$$(1 - P[V_{j+m} \leq k]) \tag{44}$$

$$A_1^1(k,j) = P[V_{j+1} \leq k] \tag{45}$$

$$\begin{aligned} A_m^p(k,j) &= A_{m-1}^p(k,j)(1 - P[V_{j+m} \leq k]) \\ &\quad + A_{m-1}^{p-1}(k,j) P[V_{j+m} \leq k], p \geq 1 \end{aligned} \tag{46}$$

The cumulative distribution associated to A_m^p is

$$\tilde{A}_m^p(k,j) = P[\sum_{l=j+1}^{j+m} 1_{[V_l \leq k]} \leq p] = \sum_{p'=0}^{p} A_m^{p'}(k,j) \tag{47}$$

Table 4: Computing $P_{\theta,\mathrm{id}}[j+m \prec_\pi j]$

```
Algorithm QExpectation

Input $\theta_{1:n-1}$
1. For $j = 1 : n-1$, for $k = 0 : j$
        Compute $P[V_j = k]$ by (7)
        Compute $P[V_j \leq k]$
2. For $j = 1 : n-1$, for $m = 1 : n-j$, for $k = 0 : j+m$,
    for $p = 0 : m$
        Compute $A_m^p(k,j)$ by (44-46)
        Compute $\tilde{A}_m^p(k,j)$ by (47)
3. For $j = 1 : n-1$, for $m = 1 : n-j$
        Compute $P[Q_{j+m,j}]$ by (48)
```

Once $A, \tilde{A}$ are obtained, we can express the desired probabilities in terms of $\tilde{A}_m^p$ and the distribution of the V_j's. (Below we have fixed a typo in the original report.)

$$\begin{aligned}
P[Q_{j+m,j}] &= \\
&= \sum_{k=0}^{n-(j+m)} P[V_{j+m} = k] \Big\{ P[V_j = k+1]\tilde{A}_{m-1}^0(k,j) + \\
&\quad P[V_j = k+2]\tilde{A}_{m-1}^1(k,j) + \dots P[V_j = j]\tilde{A}_{m-1}^{j+m-k-1}(k,j) \Big\} \\
&= \sum_{k=0}^{n-(j+m)} P[V_{j+m} = k] \left\{ \sum_{l=0}^{m-2} P[V_j = l+k+1]\tilde{A}_{m-1}^l(k,j) \right. \\
&\quad + P[V_j \geq k+m]1\} \qquad (48)
\end{aligned}$$

Algorithm QExpectation in table 4 summarizes the sequence of computations. Note that this algorithm relies only on the independence of the V_j variables, and not on their distributions being exponential. Therefore, it applies to a broader class of models than (5), namely to the class called *free models* in [8].

7.1 Lower bounds for θ_j from Q

Now we apply the theoretical values of the probabilities just obtained to derive lower bounds for the parameters θ that can be used to prune the search.

For this, we will analyze $Q_{j+1,j}$ the probability of an inversion between items j and $j+1$. We continue to assume w.l.o.g that $\pi_0 = \text{id}$.

The case $j = n-1$ is simple and gives an intuition of the general approach.

Proposition 4 *For Q defined as in (15), let $q = \min_{Q_{jl} \geq 0.5} Q_{jl}$. Then $\theta_{n-1} \geq \ln q/(1-q)$.*

Proof. Because $\theta_{n-1} \geq 0$, we have that $Q_{n-1,n} \geq 0.5$. Then, $q \leq Q_{n-1,n} = P[V_{n-1} = 0] = 1/(1 + e^{-\theta_{n-1}})$. From this, the desired result follows. $\Box$.

For $j < n-1$ we have from (41)

$$Q_{j+1,j} = \sum_{k=0}^{n-(j+1)} P[V_{j+1} = k] P[V_j > k] \triangleq f_j(\theta_j, \theta_{j+1}) \qquad (49)$$

After calculations we obtain

$$f_j(\theta, t) = \frac{1}{(e^\theta - 1)\psi_j(\theta)} \left[\frac{\psi_{j+1}(\theta + t)}{\psi_{j+1}(t)} - e^{-(n-j)\theta} \right] \qquad (50)$$

Proposition 5 (Conjecture) *The function f is increasing with t and decreasing with θ.*

One can prove that f is increasing with t by taking the partial derivative. This proof is omitted because it's tedious and otherwise straightforward. We conjecture the second statement, that f is decreasing with θ. The conjecture is based on extensive plots for various values of $n-j, \theta, t$. Some sample plots are shown in figure 5.

Assuming our conjecture holds, we have the following lower bound for θ_j.

Proposition 6 *For Q defined as in (38), let $q = \max_{Q_{lr} \leq 0.5} Q_{lr}$. Assume $\theta_{j+1} \geq t \geq 0$ and denote by $\tilde{\theta}_j$ the solution to*

$$f_j(\theta, t) = q \qquad (51)$$

Assuming conjecture 5 holds, $\theta_j \geq \tilde{\theta}_j$.

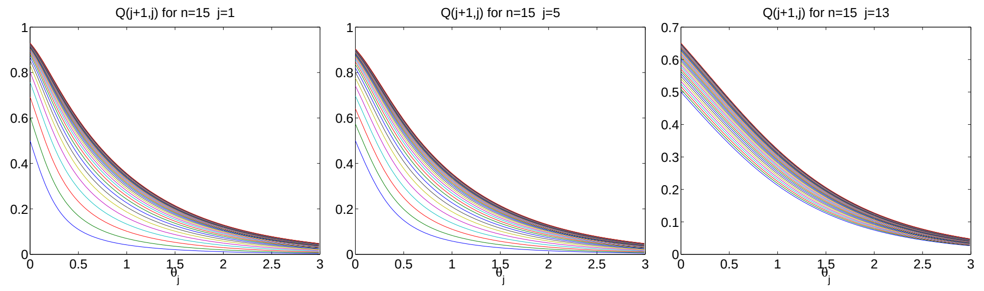

Figure 5: Plots of $Q_{j+1,j}(\theta_j, \theta_{j+1}) \equiv f_j(\theta, \theta_{j+1})$ versus θ_j for $n = 15$, $j = 1, 5, 13$ and various values of θ_{j+1}. The lowest curve corresponds to $\theta_{j+1} = 0$. Note that the curves are more spread for smaller j and for smaller θ_{j+1}.

Proof. We first show that if $\theta_j \geq \theta_{j+1} \geq 0$ then $Q_{j+1,j} \leq 0.5$[6]

$$\begin{aligned} f(t,t) &= \frac{1}{e^t(1-e^{-t})\frac{1-e^{-(n-j+1)t}}{1-e^{-t}}}\left[\frac{\frac{1-e^{-(n-j)(2t)}}{1-e^{-2t}}}{\frac{1-e^{-(n-j)t}}{1-e^{-t}}} - e^{-(n-j)t}\right] && (52) \\ &= \frac{1}{e^t(1-e^{-(n-j+1)t})}\left[\frac{1+e^{-(n-j)t}}{1+e^{-t}} - e^{-(n-j)t}\right] && (53) \\ &= \frac{1}{e^t(1-e^{-(n-j+1)t})}\frac{1+e^{-(n-j)t}-e^{-(n-j)t}-e^{-(n-j+1)t}}{1+e^{-t}} && (54) \\ &= \frac{1}{e^t(1+e^{-t})} = \frac{1}{1+e^t} \leq \frac{1}{2} \text{ for } t \geq 0 && (55) \end{aligned}$$

Since f is increasing in the second argument, it follows that $f(t', t) \leq 0.5$ whenever $t' \geq t$. Now,

$$f_j(\theta_j, \theta_{j+1}) = Q_{j+1,j} \leq q = f_j(\tilde{\theta}, t) \quad (56)$$

Since $\theta_{j+1} \geq t$ and f_j is increasing in the second argument, it follows that necessarily $\theta_j \geq \tilde{\theta}$. □.

Using proposition 4 followed by recursive applications of proposition 6 one can recursively obtain lower bounds on $\theta_{n-1}, \theta_{n-2}, \ldots\ldots\theta_1$. Note that the bound is non-trivial (greater than 0) whenever $q < 0.5$ or $t > 0$.

[6]Alternatively, under the assumption that $\theta_{j'} > 0$ for all j', from [8] we have that the model is *strongly unimodal* and $Q_{j+1,j} \leq 0.5$.

8 Experiments

The experiments in this section evaluate various existing algorithms on the consensus ranking problem of estimating π_0. Since estimating θ adds only a small constant time per search step, we consider that this case embodies the core difficulties of the estimation problem. Exception would make the cases when θ_j has comparatively large values at large j's, signifying that the most important stages of the ranking are among the *last* ones, while getting the highly ranked elements of π_0 is less important. This case is rather unrealistic in practice.

We implemented the SearchPi in C++ with the heuristics mentioned in section 5. This algorithm is denoted in the experiments as BF. We also implemented a sub-optimal search algorithm that runs the SearchPi for a predefined amount of time (5 minutes) then continues with greedy search from the largest level j attained in the BF search. This algorithm is denoted BF-CSS (the greedy search is denoted by CSS as described below).

Although theoretically the search time should not depend on the true π_0 in all our experiments we select a random π_0 every time in order to average out in the running time of the BF algorithm any artefacts of the implementation (for example, having the first branch always be the optimal one could make the algorithm faster). We also mention that our implementation of the SearchPi is a pilot implementation not optimized w.r.t running time.

The other algorithms we compared were the FV heuristic of [8], the Greedy-Order algorithm of [4] (here denoted CSS) and the algorithm of [1] (denoted ACN here). Our implementation of the FV heuristic omits the search around $\bar{\pi}$ and therefore has a run-time complexity of $\mathcal{O}(n^2)$. The ACN algorithm is also $\mathcal{O}(n^2)$ while the greedy algorithm is $\mathcal{O}(n^3)$. In our experiments, these algorithms ran very fast (fractions of a second) in all the experiments performed.

Experiments with concentrated distributions As mentioned in section 2, the consensus ranking problem has two regimes. In the *asymptotic* regime the distribution is concentrated around its mode (θ^{ML} is large), and N is large enough that π^{ML} coincides with the true π_0. This is an easy case for the BF search, but it is also an easy case for all heuristic algorithms mentioned in section 2.

We have confirmed this experimentally, on samples with $N = 5000$ from distribution P_{θ,π_0} with random π_0 and with $\theta \equiv 1$, 1.5, 2, 3. Each experiment was replicated $n_{iter} = 10$ times. In all cases, all the heuristics returned the optimal permutation.

For this experiment, figure 6 shows the number of nodes expanded by the

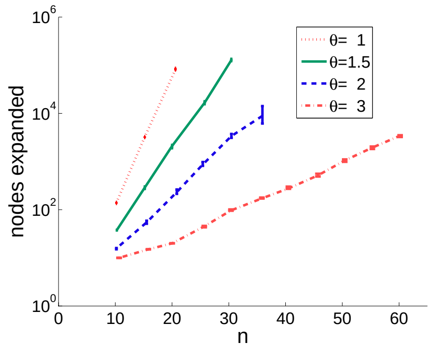

Figure 6: The average number of nodes expanded by the SearchPi with $A = 0$ for various values of n and θ. The error bars mark the minimum and maximum values over $n_{iter} = 10$ replications.

BF algorithm (the SearchPi in these experiments used the trivial $A = 0$ heuristic) as a function of θ and n.

We also ran a comparison of the heuristics FV, ACN, CSS on samples of size $N = 5000$ from a distribution with $\theta = 0.3$ (only moderately concentrated) and with $n = 10, \dots 50$. Each experiment was replicated $n_{iter} = 100$ times. For up to $n = 40$, all the heuristics returned the true permutation π_0. For these experiments, the optimal permutation was not known except for $n \leq 15$ but the large N ensures that with high probability the optimum coincides with the true π_0.

Experiments with almost uniform distribution At the other end of the spectrum is the *combinatorial* regime, where the observed permutations are distributed almost uniformly ($\theta \approx 0$) and N is relatively small so that the true π_0 is different from ${\pi_0}^{ML}$. We have simulated this case by generating $N = 100$ samples from a model with $\theta = 0.003$. The distribution being practically indistinguishable from uniform, and the Q_{ij} values being very close to 0.5, the differences in cost between various solutions are minute, and they are presented only as surrogates of a quality of the search, since the optimal π_0 is not known. For the same reason, all algorithms except SearchPi have been compared on $n_{iter} = 500$ replicated experiments.

The comparison between the heuristic algorithms is presented in figure 7.

The greedy CSS heuristic is consistently the best at all scales. Its advantage over the randomized algorithm of ACN is increasing with larger n. The true model π_0 is never optimal for this data distribution, while its es-

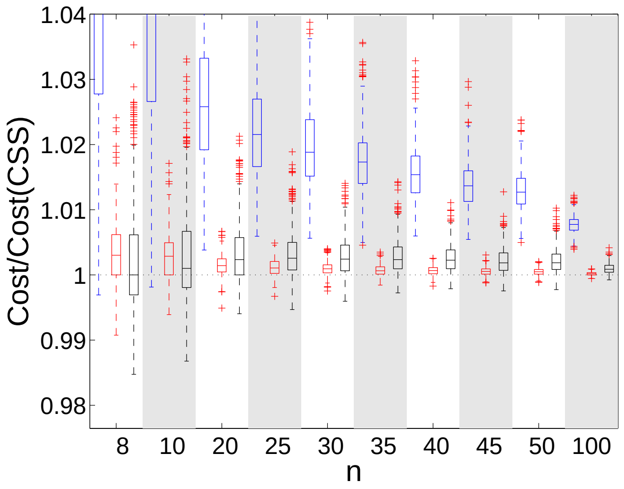

Figure 7: The cost of the true π_0 (blue), the FV (red), ACN (black) heuristics as fractions of the CSS cost. The data are $N = 100$ permutations from $P_{0.03,\pi_0}$ with random π_0. The boxplots are over $n_{iter} = 500$ replications.*redo ylabel*

timate by the FV heuristic is better but loses to the other algorithms. The "shrinking towards" 1 effect observed for larger n reflects the fact that a the larger number of values in Q are neara 0.5 when n is large. This in turn shrinks the range between the maximum and minimum cost.

Figure 8 shows comparisons between the three heuristics, the optimal BF (for $n = 8, 10$ only) and the approximate search BF-CSS. The costs are plotted as fractions of the cost BF-CSS. Therefore, the optimal BF cost always appears below or equal to 1. The experiments also show that in a large number of cases, the suboptimal BF-CSS outperforms all the other algorithms and improves on the closely related CSS greedy algorithm.

We do not claim the BF-CSS to be the ultimate approximate search heuristic. Better and faster sub-optimal searches (e.g beam-search) could be implemented. We only demonstrate by BF-CSS that the search tree approach is effective in improving the cost, or alternatively, in getting closer to a consensus, over the traditional heuristics.

To better compare the effect of the various algorithms under varying n, we present the same results now normalized by the range of the costs. The true range of costs is only known when the optimal π^{ML} is known. Then, the highest cost corresponds to the reversed π^{ML} and equals $n(n-1)/2 - C(\pi^{ML})$. In the other cases we approximate the range by the smallest cost found by any algorithm and $n(n-1)/2$ minus this value. These plots show

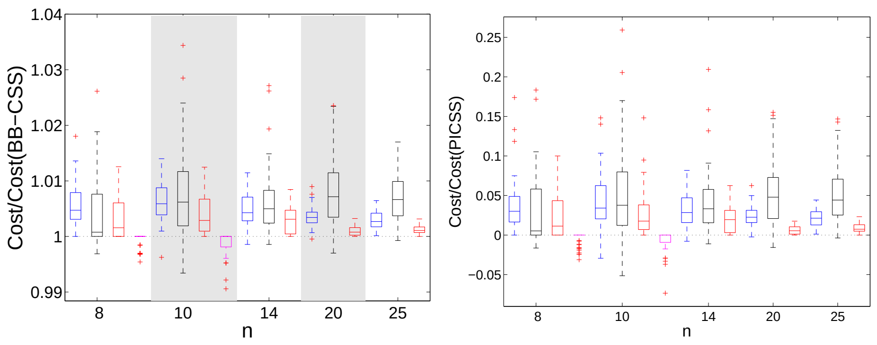

Figure 8: The cost of the FV (blue), ACN (black), CSS heuristics and of the SearchPi algorithm (magenta) as fractions of the BB-CSS cost (left). Right, the same values, now rescaled to equalize the range of possible values of the costs w.r.t n. The data are $N = 100$ permutations from $P_{0.03,\pi_0}$ with random π_0. The boxplots are over $n_{iter} = 50$ replications.

that BF-CSS improves over the other heuristics by of a few percent of the total range of costs.

Experiments with no consensus and large range of Q In this set of experiments, the data consists of a matrix Q with elements randomly sampled from $[0, 1]$ subject to the constraint $Q_{ij} + Q_{ji} = 1$ and 0 diagonal. This simulates the case of a multimodal distribution, where the permutations exhibit no consensus, but are also non-uniform. Such a setting was examined experimentally by [4]. In this problem, because the cost C can vary significantly with the choice of π_0, finding a central permutation π_0 minimizing this cost is a legitimate practical question. For instance, this task is a subtask of learning to rank in [4].

The experimental setting is identical to the previous, except that the experiments are now replicated $n_{iter} = 10$ times. Figure 9 shows the costs, as a fraction of the cost of BF-CSS. Similarly to 8, the BF algorithm improves on all heuristics for small n and the suboptimal BF-CSS improves by a few percent over the greedy algorithm (the best contender of the other heuristics) for larger values of n.

In the interest of fairness, we stress once more that the FV algorithm could be improved by local search like in [8] and that the CSS algorithm can also be improved by first finding the *strongly connected components* as described in [4].

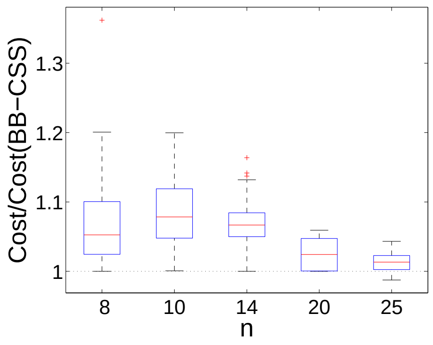

Figure 9: The cost of the greedy CSS heuristics as a fraction of the BB-CSS cost. The BB-CSS heuristic is in effect the exact BF algorithm for $n \leq 14$. The data are Q matrices with independent randomly generated entries. The boxplots are over $n_{iter} = 10$ replications.

9 Related work and discussion

This work builds directly on [10, 9] who introduced the generalized Mallows model and exploited the fact that it is an exponential family model in θ alone. As such, they use a conjugate prior on θ with a uniform prior on π_0. We have shown in section 6 such a prior is *not* the conjugate prior for θ, π_0 jointly. The normalization constant for their posterior is not computable in closed form, and it has strong similarities with the normalization constant of (33), suggesting that the latter may not be computable in closed for either. Another notable spinoff of [9] is [11] where the posterior of [9] is used as a conditional probability model over permutations, to be estimated from data by a MCMC algorithm.

To our knowledge, no authors present exact formulas for the probability of inversions Q_{jl} except for the obvious $Q_{n-1,n}$.

We have presented a new algorithm and a comparison of algorithms from various fields on the estimation of the consensus ranking. While our algorithm is certainly optimal, it is also by far the slowest. Experiments have highlighted the existing tradeoffs: in the asymptotic regime, all heuristics work well; using SearchPi is also efficient. In the combinatorial case, if we are interested in the cost only, then the differences in cost are so minute that almost any heuristic (even no optimization) will be acceptable. In other words, while the problem of consensus ranking is theoretically NP hard, minimizing the cost (approximately) is practically easy.

What is hard is finding the individual permutation that achieves best consensus in the combinatorial regime. If this is of interest, then our experiments have shown that the existing heuristics differ and that the SearchPi outperforms the other contenders when it's tractable. We are currently implementing faster and non-admissible versions of SearchPi, with the expectation that, even if exact optimization is not tractable, using a search like in SearchPi for a prespecified time can improve over greedy search.

We can show (proof omitted) that the Greedy-Order algorithm of [4] is the greedy counterpart of the SearchPi algorithm. In this sense, the good results of the CSS heuristic for larger n suggest that adding an amount of search to this already good heuristic is worthwile. In [4] missing data is also considered. Namely, it is assumed that certain items' rankings are not observed in some π_i's (these items are called *incomparable* in [4]).

The CSS heuristic has no underlying statistical assumptions, but if we make the simple assumption that items are missing at random, independently of their rank, then we can easily extend the SearchPi to this case. It suffices to notice that under such a missing data model, the estimates of Q_{jl} obtained from the subset of data where both of j, l are observed remain unbiased. Hence, the algorithm SearchPi running on

$$\hat{Q}_{jl} = 1/N_{jl} \sum_{i | j,l \text{ observed in } \pi_i} 1_{[j \prec_{\pi_i} l]} \tag{57}$$

will be a ML estimation algorithm. The same observation holds for the FV heuristic, which is based on unbiased estimation of average ranks. In [4] if k, l are incomparable in one observed ranking π_i, then the number of times k is preferred to l (l is preferred to k) is augmented by 1/2. Hence, the resulting "sufficient statistics" are now

$$\tilde{Q}_{jl} = \alpha_{jl}\hat{Q}_{jl} + (1 - \alpha_{jl})0.5 \tag{58}$$

with α_{jl} the (empirical) probability that j, l are both observed in the data. These evidently differ from the correct sufficient statistics $\hat{Q}_{jl}$ in (57). In figure 10 we present an example showing that the latter method can lead to errors.

We conclude by pointing out that with real ranking data we expect to encounter few unimodal distributions. We plan to continue this work toward the more ambitious goal of estimating parametric and non-parametric mixtures over the space of rankings.

$Q = \hat{Q}$

–	.7	.8	.9
.3	–	.7	.8
.2	.3	–	.7
.1	.2	.3	–
.6	1.2	1.6	2.4

column sums

$\tilde{Q}$

–	.52	.53	.54
.48	–	.70	.53
.47	.30	–	.52
.46	.47	.48	–
1.41	1.29	1.71	1.59

column sums

Figure 10: The correct handling of missing items. Left: a 4×4 matrix Q; if data are missing at random, the expected $\hat{Q}$ is asymptotically equal to Q. Right: the $\tilde{Q}$ matrix if items 1 and 4 are observed in $\alpha = 0.1$ fraction of cases. Under each matrix, the column sums are shown. The column sums help one see that the SearchPi , FV and the greedy CSS search return the correct permutation (1 2 3 4) on the left matrix but return the incorrect result (2 1 3 4) on $\tilde{Q}$.

References

[1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. In *The 37-th ACM Symposium on the Theory of Computing (STOC)*. Association for Computing Machinery, 2005.

[2] J. Bartholdi, C. A. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won. *Social Choice and Welfare*, 6(2):157–165, 1989.

[3] D. P. Bertsekas. *Nonlinear programming.* Athena Scientific, Cambridge, MA, 2 edition, 1999.

[4] W. C. Cohen, R. S. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[5] D. E. Critchlow. *Metric methods for analyzing partially ranked data.* Number 34 in Lecture notes in statistics. Springer-Verlag, Berlin Heidelberg New York Tokyo, 1985.

[6] M. H. DeGroot. *Probability and Statistics.* Addison–Wesley Pub. Co., Reading, MA, 1975.

[7] W. Feller. *An introduction to probability theory and its applications*, volume 1. Wiley, New York, third edition, 1968.

[8] M. A. Fligner and V. J. S. Multistage ranking models. *Journal of the American Statistical Association*, 88, 1988.

[9] M. A. Fligner and V. J. S. Posterior probability for a consensus ordering. *Psychometrika*, 55:53–63, 1990.

[10] M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society B*, 48:359–369, 1986.

[11] G. Lebanon and J. Lafferty. Cranking: combining rankings using conditional probability models on permutations. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

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

[13] R. T. Rockafellar. *Convex Analysis.* Princeton, 1970.