Sampling and Learning Distance-Based Probability Models for Permutation Spaces

Ekhine Irurozki

Advisors: Borja Calvo and José A. Lozano

Intelligent Systems Group

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Intelligent Systems Group

informatika fakultatea
facultad de informática

Konputazio Zientziak eta Adimen Artifizialaren Saila
Departamento de Ciencias de la Computación e Inteligencia Artificial

Sampling and learning distance-based probability models for permutation spaces

by

Ekhine Irurozki

Supervised by Borja Calvo - Jose A. Lozano

Dissertation submitted to the Department of Computer Science and Artificial Intelligenceof the University of the Basque Country as partial fulfilment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, July 30, 2014

Contents

1

Introduction

Permutations are among the richest objects in enumerative combinatorics [107]. They appear in a vast range of domains, from physics to psychology, principally, because of their versatility: permutations can be expressed in a great variety of forms such as an ordered set of items, a collection of disjoint cycles, transpositions, matrices, graphs or even Standard Young Tableaux. This wide range of expression is, in part, the reason why they can be found in many different problems, with several interpretations.

A particular interpretation of permutations is an assignment. For example, imagine a set of n tasks and a set of n workers. An assignment of one and only one task to each worker can be represented as a permutation.

A different interpretation of permutations is given by the following example. Suppose there is a set of n candidates in a presidential election and the voters are asked to label them in terms of preference, i.e. the most preferred candidate with a 1, the second preferred candidate with 2, etc. For each voter, the resulting vector is also a permutation, although under this particular interpretation it is usually denoted as ranking.

Since permutations appear in a wide variety of domains with several forms, a framework for probabilistic reasoning to deal with uncertainty on permutation spaces is needed. However, the computational cost of explicitly maintaining a probability value for each of the permutations of n items is prohibitive even for problems of medium size. This is because the cardinality of the space of permutations of n items grows factorially with n.

As a first approach for modeling distributions on permutations we could try adapting known distributions from other domains. However, classical notions for distributions on continuous or other discrete spaces cannot be easily adapted to models on permutations. Consider, for example, the notion of independence. For a distribution over binary variables, independence between two variables implies that the first variable will have value 0 or 1 regardless of the value of the second one. However, any two given positions of a permutation can not have the same value by any chance, so the classic notion of independence does not naturally translate into the permutation domain. The

Cand. A	Cand. B	Cand. C
1	3	2
2	3	1
1	2	3
2	3	1

Table 1.1: Example of rankings of a set of candidates

Task A	Task B	Task C	Task D
1	3	2	4
3	4	2	1
3	1	2	4
4	3	2	1

Table 1.2: Example of assignment of tasks to workers

conclusion is that permutation spaces need specific probability models and concepts.

Thurstone, Babington Smith, Bradley, Terry, Placket, Luce, Mallows, Marden, Meila and Diaconis have been both pioneers and the main contributors to the field of probability models on permutations. Although the literature on models on permutation spaces begins in the first decades of the 20th century, the popularity of the models dealing with rankings have raised in the last two decades. This is most likely a consequence of the explosion of the commercial applications of probabilistic reasoning with preferences. Proof of this explosion is the fact that recently, a new subfield of Machine Learning has emerged: Preference Learning [55].

However, models designed for rankings are not suited for other interpretations of permutation data. In a model on rankings the information captured is of type 'candidate i is preferred to candidate j'. Let us illustrate this idea with the example in Table 1.1, where a collection of votes over a set of candidates is given. Each voter (row) ranked the three candidates from most (1) to least (3) preferred.

Note that Candidate A is preferred to Candidate B by all the voters. Models on rankings are usually designed to maintain this kind of relative ordering information.

Consider now a different interpretation of permutations, the assignment. As an example, consider a set of four tasks and a set of four workers, all of which are labeled by integers. In Table 1.2 each row is an assignment of tasks to workers. We can see that task C is always assigned to worker 2, probably because this particular worker is very good at task C. In this case the relevant information can have the form 'task i is assigned to worker j with high probability', and thus we would expect the model to capture this kind of information.

Distributions on rankings have become very popular in areas such as machine learning. In this thesis we have tried giving a step forward filling the gap in the area of probability models on permutations when the data is not interpreted as a ranking. We are convinced that the distributions on permutations for other permutations' interpretation can be also very useful. We will later discuss several applications in fields such as machine learning.

We have focused on the distance-based probability models and extensions thereof, which are exponential families for permutations. All these families of models need a definition of a distance between permutations to control how the probability decreases as we move away from the mode. In this dissertation we discuss the use of different distances: Kendall's-τ, Cayley, Hamming and Ulam distance.

1.1 Permutations and distances

The theory of permutations is subsumed in the field of group theory. As a fundamental branch of algebra, literature on group theory is wide and diverse. It is therefore far from our aim to give a proper tutorial neither in permutation theory nor in group theory. Nevertheless, this section comprises the elemental concepts that will be recurrent on the rest of this dissertation. The interested reader can find an excellent reference on the combinatorics of permutations in [22].

Formally, a permutation of a set $\mathcal{S}$ is a one-to-one function from $\mathcal{S}$ onto $\mathcal{S}$. In this dissertation $\mathcal{S}$ is a set of n items labeled with the natural numbers $\{1, \ldots, n\}$ and permutations are represented in the classic one-line notation, i.e. as an ordered list of the set $\{1, \ldots, n\}$. In the literature they are generally denoted with Greek letters; we will mostly use σ, π and τ.

Rankings are a particular interpretation of permutations. However, mixing both terms is frequent in the literature. We will try to differentiate both concepts by restricting the use of the words ranking and ordering to the cases where that particular interpretation is considered. In general, we will use the following notation: $\sigma(j) = i$ meaning that item i is in position j; the permutation σ is represented as $\sigma = \sigma(1)\sigma(2)\ldots\sigma(n)$. As an example, for permutation $\sigma = 3412$ we say that item 3 is at position 1, i.e. $\sigma(1) = 3$.

Permutations form the algebraic group by the name of Symmetric Group, S_n, where n denotes the number of items considered in the permutations. The group operation is the composition. This means that by composing two permutations σ and π of n elements we obtain a new permutation of n items $\sigma \circ \pi$ such that $\sigma \circ \pi(j) = \sigma(\pi(j))$, which will be denoted as $\sigma\pi$. The identity element of the Symmetric Group maps each item j to position j. It is called identity permutation and denoted $e = 123\ldots n$. The Symmetric Group is non-abelian, what means that, in general, the composition is not commutative. Some exceptions to this general rule include the composition of a permutation

σ and its inverse σ^{-1}, which results in the identity, $\sigma\sigma^{-1} = \sigma^{-1}\sigma = e$, and the composition with the identity, $\sigma e = e\sigma = \sigma$.

It is crucial for the probability models studied in this thesis to measure the discrepancies between two given permutations. We will use four different distances for this task: Kendall's-τ, Cayley, Hamming and Ulam. The following four sections are devoted to each one of these distances. There we introduce them here by giving their standard definitions, some examples on their calculations and their most remarkable properties. Moreover, we the interested reader can find several surveys on distances [37], some of which are specialized on distances for permutations [36], [47], [41], or even on their landscape [106].

The right invariance is a common property to the four metrics. Right invariance means that the distance is independent of the relabeling of the items, so $d(\sigma, \pi) = d(\sigma\tau, \pi\tau)$ for every permutation τ. Particularly taking $\tau = \pi^{-1}$ and since $\pi\pi^{-1} = e$ one can w.l.o.g. write $d(\sigma, \pi) = d(\sigma\pi^{-1}, e)$, i.e., we can always take the identity permutation as the reference one. For the sake of clarity the distance is denoted as a one parameter function when the reference permutation is the identity, $d(\sigma\pi^{-1}) = d(\sigma\pi^{-1}, e)$ what simplifies the notation.

Another important property shared by three of the four distances considered is the decomposition as a sum of terms as follows:

$$d(\sigma) = \sum_{j=1}^{max} S_j(\sigma)$$

Both max and S_j depend on the distance. This decomposition is usually more informative than the distance itself and it is extensively used throughout this dissertation. For the sake of clarity, the terms in the sum are grouped in a vector $S(\sigma) = (S_1(\sigma), \ldots, S_{max}(\sigma))$ which is referred to as the distance decomposition vector.

1.1.1 Kendall's-τ distance

The Kendall's-τ distance $d_k(\sigma\pi^{-1})$ counts the number of pairwise disagreements between σ and π, i.e., the number of pairs of positions in which the items are in a particular order in σ, and the reverse order in π. We can equivalently define $d_k(\sigma\pi^{-1})$ as the minimal number of adjacent swaps to convert σ^{-1} in π^{-1}. The maximum value of the Kendall's-τ distance between two permutations is $n(n-1)/2$. It is used mainly in voting theory and it is sometimes called bubble sort distance because $d_k(\sigma\pi^{-1})$ equals the number of adjacent swaps that the bubble sort algorithm performs to order the items in $\sigma\pi^{-1}$ increasingly.

For example, if $\sigma = 21453$ and $\pi = 12345$ then $\sigma^{-1} = 21534$ and $\pi^{-1} = 12345$ and, thus, a minimal sequence of adjacent transposition (in bold) to convert σ^{-1} in π^{-1} is as follows:

Algorithm 1: Get the permutation σ consistent with the given decomposition vector $\mathbf{V}(\sigma)$

Input: $\mathbf{V}(\sigma)$, Kendall's-τ distance decomposition vector;
Output: σ consistent with $\mathbf{V}(\sigma)$
Let $\mathcal{S}$ be the ordered set $\{1, \ldots, n\}$ such that $\mathcal{S}(0) = 1, \ldots, \mathcal{S}(n-1) = n$;
for $j = 1$ **to** $n - 1$ **do**
  $\sigma(j) = \mathcal{S}(V_j(\sigma))$;
  remove j from $\mathcal{S}$
end
$\sigma(n) = \mathcal{S}(0)$;
return σ;

$$21534 \rightarrow 213\mathbf{5}4 \rightarrow 213\mathbf{45} \rightarrow \mathbf{12}345$$

and therefore, $d_k(\sigma\pi^{-1}) = 3$. Equivalently, only the pairs of positions (1,2), (3,5) and (3,4) where the items have different order in both permutations π and σ. The Kendall's-τ distance can be computed in time $O(n^2)$.

Decomposition vector

The decomposition for the Kendall's-τ distance $d_k(\sigma)$ is denoted as $\mathbf{V}(\sigma) = (V_1(\sigma), \ldots, V_{n-1}(\sigma))$ where $V_j(\sigma)$ equals the number of items smaller than $\sigma(j)$ in the tail of the permutation. It can be expressed as follows:

$$V_j(\sigma) = \sum_{i=j+1}^{n} \mathbb{1}_{\sigma(i)<\sigma(j)} \tag{1.1}$$

where $\mathbb{1}$ denotes the indicator function. It follows that $0 \leq V_j(\sigma) \leq n - j$ and $d_k(\sigma) = \sum_{j=1}^{n-1} V_j(\sigma)$ for every $\sigma \in S_n$. For $\sigma = 21534$ as above, $d(\sigma)$ can be computed by means of $\mathbf{V}(\sigma)$ as follows:

$$d(21534) = \sum_{j=1}^{n-1} V_j(21534) = 1 + 0 + 2 + 0 = 3$$

It is worth noticing that there is a bijection between each $\sigma \in S_n$ and each possible $\mathbf{V}(\sigma)$ vector. Therefore, when dealing with the Kendall's-τ distance we can use the $\mathbf{V}(\sigma)$ vector as an alternative representation of σ. Algorithm 1 shows how to obtain the permutation σ given $\mathbf{V}(\sigma)$. It follows that the conversion from $\mathbf{V}(\sigma)$ to σ and vice versa can be done in time $O(n^2)$.

Counting permutations

The number of permutations at each possible Kendall's-τ distance d, $S_k(n, d)$, is the sequence with code A008302 in the On-line Encyclopedia of Integer Sequences (OEIS)[1]. This sequence can be computed recursively as follows:

[1] `http://oeis.org/`

$$S_k(n,d) = \begin{cases} 1 & \text{if } d \leq 0 \\ S_k(n,d-1) + S_k(n-1,d) - S_k(n-1,d-n) & \text{otherwise} \end{cases}$$

The computational cost of computing $S_k(n,d)$ in the worts case is $O(n \cdot d_{max}) \equiv O(n^3)$.

1.1.2 Cayley distance

The Cayley distance $d_c(\sigma\pi^{-1})$ counts the minimum number of swaps (not necessary adjacent) that have to be made to transform σ into π. The maximum value of the Cayley distance between two permutations is $n-1$. The Cayley distance between σ and the identity, $d_c(\sigma)$, can also be defined as n minus the number of cycles of σ. Let us show this last statement in detail.

A cycle in σ is an ordered set $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ such that $\sigma(i_1) = i_2$, $\sigma(i_2) = i_3$, $\ldots$, $\sigma(i_k) = i_1$. It follows that for every $1 \leq i, j \leq n$ such that $\sigma(i) = j$, then i and j are in the same cycle. A popular representation of permutations is the cycle notation, in which its factorization in the set of disjoint cycles is given. Under the cycle notation, $\sigma = 156324$ is written as $(1)(2,5)(3,6,4)$ or equivalently $(2,5)(3,6,4)$, since any cycle of length one can be omitted.

The relations between the number of swaps and cycles was noted by Cayley, who stated that every minimal sequence of swaps to convert σ into e considers pairs of items in the same cycle. For example, a sequence of swaps to convert σ into the identity can be seen in bold in the following line.

$$156324 \rightarrow 1\mathbf{2}63\mathbf{5}4 \rightarrow 12\mathbf{36}54 \rightarrow 123\mathbf{45}\mathbf{6}$$

where the first swap orders the items in the cycle $(2,5)$ and the next two swaps order the items in the cycle $(3,6,4)$. Consequently, $d(\sigma) = 3$. The Cayley distance can be computed in $O(n)$.

Decomposition vector

Let $\mathbf{X}(\sigma) = (X_1(\sigma), \ldots, X_{n-1}(\sigma))$ be a vector where $X_j(\sigma) = 0$ if j is the largest item in its cycle in σ and $X_j(\sigma) = 1$ otherwise. Then $d_c(\sigma) = \sum_{j=1}^{n-1} X_j(\sigma)$ holds for every $\sigma \in S_n$. An example of the use of this decomposition vector $\mathbf{X}(\sigma)$ is the computation of the Cayley distance of a permutation to the identity as follows:

$$d(156324) = \sum_{j=1}^{n-1} X_j(156324) = 0 + 1 + 1 + 1 + 0$$

Note that, contrary to what happens with the Kendall's-τ distance, there is not a bijection between every possible $\mathbf{X}(\sigma)$ vector and $\sigma \in S_n$: although

each $\sigma \in S_n$ has one unique decomposition vector $\mathbf{X}(\sigma)$, the opposite is not necessarily true. Later in this dissertation we provide two algorithms that given $\mathbf{X}(\sigma)$, randomly generate a permutation σ consistent with $\mathbf{X}(\sigma)$ and assuming that every permutation consistent with $\mathbf{X}(\sigma)$ is equally probable. The conversion from a permutation σ to $\mathbf{X}(\sigma)$ and vice versa have time complexity $O(n)$.

Feller coupling

The Feller coupling is a method used to generate permutations from the uniform distribution, [8], [48] (pag. 257). We detail the Feller coupling in the following lines. Moreover, we will use it to proof that each $X_j(\sigma)$ is independent if σ is generated uniformly at random.

The Feller coupling, which is described as a sequence of n decisions, is more clearly seen when the permutation is represented in cycle notation. Recall that every permutation can be written as the product of independent cycles and that if item i is placed in position j, $\sigma(j) = i$, then items i and j are in the same cycle in σ.

The Feller coupling procedure generates a permutation by constructing a collection of cycles. This generation process considers the initial cycle (1) and selects uniformly at random the item j to place at position 1 in σ, that is, select j such that $\sigma(1) = j$. If $j = 1$, then the cycle (1) is closed. Otherwise, $\sigma(1) = j$ and thus the previous cycle is updated to $(1j)$. Then, the item j' to place at $\sigma(j)$ is selected uniformly at random. If $j' = 1$, the cycle $(1j)$ is closed. Otherwise, this first cycle is now $(1jj')$. The process continues in this way until the cycle is closed, i.e. until we select item 1 to place at any position. Then, the smallest item among those that have not been inserted in the previous cycles is selected to construct another cycle in the same way, and the process is repeated until a permutation is generated, i.e., until every item $1 \leq i \leq n$ belongs to a cycle.

Note that the generative process is carried out in n separate stages. At stage s the decision can be seen as either choosing the one items that closes the current cycle and opening a new one or either choosing one of the $n - s$ items to add to the current cycle. This means that the decision of closing the current cycle or not at stage j can be seen as independent Bernoulli random variables of parameter $1/(n - j + 1)$. Moreover, it follows that the previously defined $X_j(\sigma)$ binary variables as independent Bernoulli random variables of parameter $1/(n - j + 1)$.

Counting permutations

Recall that the Cayley distance $d_c(\sigma)$ equals n minus the number of cycles of σ. Therefore, the number of permutations at Cayley distance d, $S_c(n, d)$, equals the number of permutations with $k = n - d$ cycles. The number of permutations of n items with k cycles is given by the Stirling numbers of the

first kind, which has code A008275 in the OEIS. The sequence for $S_c(n,d)$ is given here by adapting the recurrence for the Stirling numbers as follows:

$$S_c(n,d) = \begin{cases} 1 & \text{if } d = n = 0 \\ 0 & \text{if } d = n \oplus n = 0 \\ S_c(n-1,d+1) + (n-1)S_c(n-1,d) & \text{otherwise} \end{cases}$$

where $\oplus$ denotes the XOR operator. The time complexity of computing $S_c(n,d)$ is $O(n \cdot d_{max}) \equiv O(n^2)$.

1.1.3 Hamming distance

The Hamming distance $d_h(\sigma\pi^{-1})$ counts the number of positions in which two permutations disagree. The maximum value of the Hamming distance between two permutations is, therefore, n. It is worth noticing that there is no pair of permutations σ and π such that $d(\sigma,\pi) = 1$. The Hamming distance is closely related to the concepts of fixed and unfixed points. A fixed point in σ is a position i where $\sigma(i) = i$ while the opposite is an unfixed point. Therefore, the Hamming distance from a permutation σ to the identity, $d_h(\sigma)$, counts the number of unfixed points in σ. For example, in the following permutation, the set of unfixed points is highlighted in bold; its Hamming distance to the identity is, therefore, 4.

243516

The Hamming distance can be computed in $O(n)$ time.

Decomposition vector

The notion of fixed points leads to the decomposition vector of the Hamming distance. Let $\mathbf{H}(\sigma) = (H_1(\sigma), \ldots, H_n(\sigma))$ be a vector such that $H_j(\sigma) = 0$ if j is a fixed point in σ and $H_j(\sigma) = 1$ otherwise. Then $d_h(\sigma) = \sum_{j=1}^{n} H_j(\sigma)$ holds for every $\sigma \in S_n$. Since, as we have stated, no pair of permutations is at Hamming distance 1, there can not possibly be a Hamming distance decomposition vector with just one 1 and $n-1$ zeros. Note that every $\sigma \in S_n$ has a unique $\mathbf{H}(\sigma)$ but the opposite is not necessarily true. The conversion from σ to $\mathbf{H}(\sigma)$ and the generation uniformly at random of σ consistent with a given $\mathbf{H}(\sigma)$ have complexity $O(n)$. The decomposition vector of the Hamming distance has n terms -unlike Kendall's-τ and Cayley's that have $n-1$. The distance in the previous example can be computed by means of $\mathbf{H}(\sigma)$ as follows:

$$d(243516) = \sum_{j=1}^{n} H_j(243516) = 1+1+0+1+1+0$$

Counting permutations

The question of obtaining the number of permutations at a given distance is closely related to the notion of derangement (a permutation with no fixed point) since in a permutation at Hamming distance d from the identity there are d items that form a derangement. Therefore, the number of permutations of n items at Hamming distance d is

$$S_h(n,d) = \binom{n}{d} S(d) \quad \text{where } S(d) \text{ is the number of derangements of } d \text{ items}$$

Counting the number of derangements is a recurring question and it is given in the well known On-Line Encyclopedia of Integer Sequences (OEIS) with code A000166. Even though there is no closed form for $S(d)$ the computation of $S(d)$ can be carried out efficiently in a recursive way as the following:

$$S(d) = \begin{cases} 1 & d = 0 \\ 0 & d = 1 \\ (d-1) * S(d-1) + (d-1) * S(d-2) & \text{otherwise} \end{cases} \tag{1.2}$$

This equation can be computed in time $O(d)$. Since we are interested on the derangements of at most n items, we must compute $S(d)$ for $0 \leq d \leq n$ that requires time $O(n)$.

Generating a permutation at distance d uniformly at random

In this section we deal with the generation uniformly at random of permutations at a given Hamming distance, what will be necessary for the sampling of MM and WM under the Hamming distance. We will consider two different scenarios, each for a different sampling algorithm:

i) The set of fixed and unfixed points is given. In this case, we must derange the set of unfixed items while setting each of the remaining items j at position j.
ii) A distance d is given. In this case, we must select uniformly at random $n-d$ items for deranging them while setting each of the remaining items j at position j. Selecting the items for deranging can be done by generating a permutation σ uniformly at random and deranging the first $n-d$ items in σ. Therefore, the first scenario can be seen as a particular case of this.

Note that both problems rely on generating uniformly at random a derangement of a set of items, what is approached in a recursive manner. The method for the uniformly at random generation of a derangement is based on splitting the space of derangements regarding the length of the cycle to which a particular item belongs. Let D be a set of items to derange such that $d = |D|$ and i_d the maximum item in D. Then, every derangement of the set of items D falls into one and only one of these two categories[2]:

[2] Recall that i, j such that $\sigma(i) = j$ are in the same cycle and that a cycle of length one is a fixed point.

1. Item i_d is in a cycle of length 2.
2. Item i_d is in a cycle of length greater than 2.

This trivial observation is the key for the recurrence introduced which, given a set of items D, first generates a derangement which falls into one of those groups. It works as follows:

The base case is the generation of a derangement of two items, $D = \{i_1, i_2\}$. There is just one way of deranging two items which is $\sigma(i_1) = i_2$ and $\sigma(i_2) = i_1$.

In the general case, the generation of a derangement of the items in D implies first the generation of a derangement of a set $D' \subsetneq D$. Every derangement of D can be built in one of the following ways:

- First, randomly select an item $i \in D$. Then, generate a derangement with items $D/\{i, i_d\}$. Finally set $\sigma(i) = i_d$ and $\sigma(i_d) = i$. In this way, item i_d in the resulting derangement is in a cycle of length 2. In this case, the recursion implies the generation of a derangements of the $S(d-2)$ possible derangements of $d-2$ items. Also, there are $d-1$ possible ways of selecting item i. Therefore, there are exactly $(d-2) * S(d-1)$ derangements of d items of this form.
- First, generate a derangement with items $D/\{i_d\}$. Then, randomly select an item $i \in D/\{i_d\}$. Finally set $\sigma(i) = i_d$ and $\sigma(i_d) = i$. In this way, item i_d in the resulting derangement is in a cycle of length greater than 2. In this case, the recursion implies the generation of a derangement of the $S(d-1)$ possible derangements of $d-1$ items. Also, there are $d-1$ possible ways of selecting item i. Therefore, there are exactly $(d-1)*S(d-1)$ derangements of d items of this form.

Therefore, of the total $S(d)$ derangements of d items, exactly $(d-2) * S(d-1)$ have item d in a cycle of length 2 and $(d-1) * S(d-1)$ of them have item d in a cycle of length greater than 2. Thus the probability of selecting the first branch is $(d-2) * S(d-1)/S(d)$ while the probability of the second is $1 - (d-2) * S(d-1)/S(d) = (d-1) * S(d-1)/S(d)$.

The pseudo-code of the whole generation process is given in Algorithm 2. Note that the complexity of generating a derangement of d items is $O(d)$. Note this algorithm is based on the same reasoning as the counting recurrence in Equation (1.2).

1.1.4 Ulam distance

The Ulam distance $d_u(\sigma\pi^{-1})$ counts the length of the complement of the longest common subsequence in σ and π, so the maximum value of the Ulam distance between two permutations is $n-1$. The Ulam distance between a permutation σ and the identity, $d_u(\sigma)$, equals n minus the length of the Longest Increasing Subsequence (LIS) of σ. The classical example to illustrate the Ulam distance $d_u(\sigma, \pi)$ [38] considers a shelf of books in the order specified by

Algorithm 2: *generate_derangement*(d)
This algorithm generates a uniformly at random derangement of a set of items D. Note that every possible derangement is equally probable.

Input: $D = \{i_1, i_2, \ldots, i_d\}$, set of items to derange;
Output: π, derangement of d items
if $D = \{i_1, i_2\}$ **then** $\pi(i_1) = i_2 \wedge \pi(i_2) = i_1$; `/* base case */`
else
  $D' = D/\{i_d\}$;
  i = random item in D';
  $prob = (d-1) * S(d-2)/S(d)$;
  with probability $prob$ `/*` i_d `is in a cycle of length 2 */`
    $\pi = generate\_derangement(D'/\{i\})$;
  end
  otherwise `/*` i_d `is in a cycle of length grater than 2 */`
    $\pi = generate\_derangement(D')$;
  end
  $\pi(i) = i_d$;
  $\pi(i_d) = i$;
end
return π;

σ. The objective is to order the books as specified by π with the minimum possible number of movements, where a movement consists of taking a book and inserting it in another position (delete-insert); the minimum number of movements is exactly $d_u(\sigma, \pi)$. The most relevant references of the Ulam distance include [3], [13], [104]. As a consequence of the complexity of its computation, which is $O(n \log l)$ where l is the length of the LCS of σ and π, studies for the approximate computation of the Ulam distance can be found in the literature [5], [104].

Let us illustrate the Ulam distance and the delete-insert movements with an example. For $\sigma = 561423$ the LIS is $\{123\}$ and the items in the complement of the LIS are 4, 5 and 6. It follows that the Ulam distance $d_u(\sigma) = 6 - 3 = 3$ and the sequence of possible delete-insert movements (in bold) in convert σ into the identity permutation is as follows:

$$561423 \rightarrow 56123\mathbf{4} \rightarrow 61234\mathbf{5} \rightarrow 12345\mathbf{6}$$

Decomposition vector

There is no decomposition vector for the Ulam distance. The reason is that a permutation can possibly have many LIS. For example, in $\sigma = 7615243$ there are two LIS, one consisting of items 1, 2 and 4 and the other one consisting of items 1, 2 and 3.

(a) (b)

Fig. 1.1: An example of a Ferrers diagram (a) and a SYT (b)

The Ulam distance for permutations is closely related with structures such as Ferrers diagrams and Standard Young Tableaux. Knowing how to count Standard Young Tableaux is crucial to manage Ulam distance-based probability models. In order for this manuscript to be self contained, all these concepts are introduced.

An integer partition of n is a non decreasing sequence of positive integers denoted as $\lambda = \{\lambda(1), \lambda(2), \ldots, \lambda(k)\} \vdash n$ such that $n = \sum_{i=1}^{k} \lambda(i)$. For example, $\lambda = \{4, 3, 3, 2\}$ is a partition of 12, denoted as $\{4, 3, 3, 2\} \vdash 12$, since $12 = 4{+}3{+}3{+}2$.

A partition can be graphically represented by a Ferrers diagram (FD) or Young diagram, which is a set of n boxes (or cells) arranged in table form. A FD is said to have shape $\lambda \vdash n$ if there are $\lambda(1)$ boxes in the first row, $\lambda(2)$ in the second one and so on. Figure 1.1a shows a FD of shape $\lambda = \{4, 3, 3, 2\}$.

A Standard Young Tableaux (SYT) is a FD of shape $\lambda \vdash n$ in which each number in the range $1 \ldots n$ is placed in a different box in such a way that the numbers increase along the rows and down the columns. Figure 1.1b shows a SYT of shape $\lambda = \{4, 3, 3, 2\}$.

These simple graphics are a very useful tool on representation theory or geometry for example. Since the questions of counting and generating SYT given a FD are crucial for one of our proposed sampling algorithms, the next sections are devoted to those questions entirely.

Counting SYT for a FD

The number of different SYT that can be created from a given shape is given by the Hook length formula. It does not have a closed form but has a very simple expression. First of all some notation must be introduced.

Let a FD of shape λ include box b. The hook of b, H_b, is the set of boxes to the right in the same row and below in the same column, including box b. The hook length, h_b, is the number of boxes in H_b. Figure 1.2 shows the FD of shape $\lambda = \{4, 3, 3, 2\} \vdash 12$. In Figure 1.2a, the cell at position $(2, 2)$ has been labeled as b and the hook of b, H_b, is the set of cells highlighted in gray. The hook length of b is therefore 4. In Figure 1.2b, the hook length of every box is given.

The number of different SYT in any given FD of shape λ is denoted as h_λ and it is given by the following expression.

$$h_\lambda = \frac{n!}{\prod_{b\in\lambda} h_b} \tag{1.3}$$

The Robinson-Schensted correspondence

The Robinson-Schensted correspondence (RS) is the link between SYT and the Symmetric Group. It was shown independently by Robinson [103] and Schensted [105] and it was later extended to a correspondence between matrices and semistandard Young Tableaux by Knuth [75] giving raise to the best known generalization, the Robinson-Schensted-Knuth correspondence.

The RS states that there is a one-to-one correspondence between pairs of SYT of the same shape $\lambda \vdash n$ and permutations of n items. As a consequence, the relation between the SYT of every possible shape $\lambda \vdash n$ and the number of different permutations of n items can be summarized in the following celebrated result.

$$\sum_{\lambda\vdash n} h_\lambda^2 = n!$$

The Schensted algorithm [105] generates the pair (P, Q) of SYT of the shape $\lambda \vdash n$ that maps to a given permutation $\sigma \in S_n$ by the RS. This version can be found in [3] and Section 7.1 in [22]. The inverse process of generating the permutation mapped to the pair of tableaux (P, Q) is carried out by performing the Schensted algorithm backwards. We provide the pseudo-code of this version in Algorithm 3 since it is the one that we will use later.

Let us illustrate the algorithm with the example in Figure 1.3. Let tableaux P and Q be initially as in Figure 1.3a. This algorithm iterates for n steps, deleting a box on both P and Q at each step. Also, the items in P are possibly moved during the construction of the permutation, while items in Q are not.

The algorithm starts by identifying the box in Q where item 6 (the last item) lies, that is, first row, fourth column. Set u equal to the item in the box at the first row, fourth column in P, i.e. $u = 6$. Then, the boxes at the first row, fourth column in both P and Q are removed, and since u lies in the first

(a) (b)

Fig. 1.2: The hook of cell b (a) and the hook lengths of every cell (b)

Algorithm 3: Inverse Schensted algorithm

Input: Q, P: SYT of shape $\lambda \vdash n$
Output: $\sigma \in S_n$ mapped to the pair (P, Q)
for $r = n$ **to** *1* **do**
  Choose i, j such that $Q(i, j) = r$;
  $u \leftarrow P(i, j)$;
  Remove boxes $Q(i, j)$ and $P(i, j)$;
  while $i \neq 1$ **do**
    $i \leftarrow i - 1$;
    Select j such that $P(i, j)$ is the largest entry smaller than u;
    Swap items $P(i, j)$, u;
  end
  $\sigma(r) = u$;
end

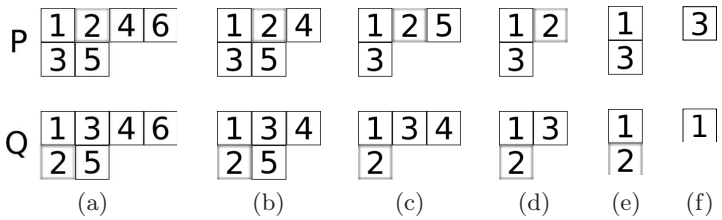


Fig. 1.3: Inverse Schensted algorithm example

row, the only remaining step is set $\sigma(6) = 6$. At the end of this iteration both P and Q are given in Figure 1.3b.

The second iteration selects the box from Q where item 5 lies, that is, the cell at the second row, second column. Therefore, $u = 5$ and the boxes at the second row, second column are deleted. Since u does not lie at the first row in P, it is swapped with the largest entry smaller than $u = 5$ in the row above, i.e., item 4 in the first row is substituted by item 5 and $u = 4$. At the end of this iteration $\sigma(5) = 4$ and both P and Q are given in Figure 1.3c.

At the end of the third iteration $\sigma(4) = 5$ and P and Q are given in Figure 1.3d. The process continues until eventually $\sigma = 312546$.

Let P and Q be two SYT of shape λ and let σ be the permutation that maps the pair of tableaux (P, Q) via the RS correspondence. The RS correspondence has several well known properties relating (P, Q) with different characteristics of σ such as the number of fixed points of σ. The one that deserves special attention is that which relates the length of the LIS of σ and the shape λ of P and Q.

Proposition 1 *The length of the LIS of σ equals the number of columns of P and Q, i.e. $\lambda(1)$ [22].*

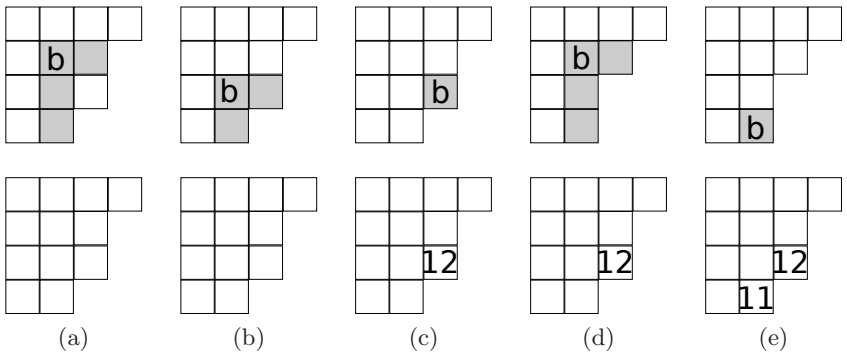


Fig. 1.4: Random generation of a SYT given a FD

Let us notice that in our example the LIS in σ is given by the items 1, 2, 4 and 6. Therefore, the length of the LIS equals 4, the same as the number of columns of the tableaux in Figure 1.3a equals $\lambda(1) = 4$, as stated in Proposition 1.

Random generation of a SYT given FD

There are several proofs of Equation (1.3) for the number of SYT of a given shape [94], [57], [54], [15]. We are particularly interested in a short probabilistic proof, [58] because they claim that it yields an algorithm that generates, uniformly at random, SYT of a given shape. We use this method for the random generation of the SYT given a FD of shape λ.

The method described in [58] to generate u.a.r. a SYT given a FD of shape $\lambda \vdash n$ is summarized in Algorithm 4. It begins by defining P as an unlabeled SYT of shape λ and labeling one cell at each iteration. In order to illustrate the process, an example is given in Figure 1.4, where the top row keeps track on the FD and the bottom one of the SYT P. The first step is to choose u.a.r. a box b in the FD, Figure 1.4a, where the hook of b is painted in gray. If b is not a corner box, as in the example, chose another box in the hook of b and name it b again, see Figure 1.4b. Repeat this process until b is a corner box, what in our running example happens in Figure 1.4c. Then, label this box in P as n and remove b from the FD. The new P can be seen in Figure 1.4d, where the process is repeated by selecting a box in the new FD. Since, again, b is not a corner box in the FD, we move through its hook until a corner box is reached, as in Figure 1.4e. Label this box in P as $n-1$ and, again, delete it from the FD. This process is repeated until every box in P has been labeled.

Counting permutations

The number of permutations at Ulam distance d, denoted $S_u(n, d)$, is the sequence with code A126065 in the OEIS. An approximate bound for $S_u(n, d)$

Algorithm 4: Random generation of a SYT given a FD

Input: FS of shape $\lambda \vdash n$
Output: P: SYT of shape $\lambda \vdash n$ u.a.r.
for $r = n$ **to** *1* **do**
  $b \leftarrow$ choose u.a.r. a box in the FD;
  while b *is not a corner box* **do**
    $b \leftarrow$ choose u.a.r. a box in H_b;
  end
  Label P_b as r ;
end
Return P;

which can be efficiently computed can be found in [1]. However, we will show how to compute the sequence exactly relaying on the FD, SYT, the RS correspondence and the hook length formula defined above.

As we have already stated, the number of different SYT of shape λ is given in Equation (1.3) and denoted as h_λ. It follows that there are h_λ^2 different possible pairs of STY that can be generated with the same shape. The RS correspondence defines a bijection between pairs of SYT of the same shape λ and permutations $\sigma \in S_n$ in which, as Proposition 1 asserts, $\lambda(1)$ equals the length of the LIS in σ. Since the length of the LIS in σ equals $n - d(\sigma)$ it follows that the number of permutations of n items at Ulam distance d is

$$S_u(n,d) = \sum_{\lambda \vdash n | \lambda(1) = n-d} h_\lambda^2 \tag{1.4}$$

In order to obtain $S_u(n,d)$ for every $0 \leq d < n$ with the above counting approach, every partition $\lambda \vdash n$ has to be enumerated. The asymptotic properties of the number of partitions of n, which has been object of study for years, was successfully answered by Hardy, Ramanujan, and Rademacher [61], [101]. Unfortunately, they stated that the number of partitions of n grows subexponentially with n.

1.2 Probability models on permutations

There is an extensive literature on probability models on permutation which focuses mainly on the ranking interpretation of permutations. An exceptional review on the literature on probability models can be found in [32], which besides analyzing the models, shows the intersections among them. Fundamental to the study of probability models for permutations are also the books [38], [53] and [84].

This thesis focuses on a subset of the models, the distance-based models and extensions thereof. In order to place them in the whole world of distributions for permutations, a general perspective must be considered.

It is out of our scope giving an exhaustive review of the literature. Instead, we present the most referenced models, including parametric and non-parametric models. Special attention is given to the sampling and learning operations which, after all, are the main topic of this thesis. We follow the traditional classification of the parametric models, but the reader should be aware that this classification is far from strict: some models are subsumed in others, the intersections among some models is not empty, etc. For each family of models, we provide an intuitive description followed by the formal one and references to the sampling and learning approaches proposed in the literature.

Before going into details, a note regarding the notation in the ranking domain should be given. It is very easy to confuse rankings with orderings, being this error cause of uncountable headaches. Let a set of n objects labeled as $1, \ldots, n$ be ranked. An ordering is a permutation $\sigma \in S_n$ where $\sigma(i) = j$ denotes the item j has rank i. A ranking is also a permutation $\pi \in S_n$ where $\pi(i) = j$ denotes that item i has rank j. The ranking π associated to ordering σ is given by its inverse $\pi = \sigma^{-1}$ and vice versa $\sigma = \pi^{-1}$, so a distribution on the rankings implicitly implies a distribution over the orderings.

Non-parametric Models

One way of modeling permutation distributions on permutation spaces is by means of non-parametric models. Suppose that we are given a sample of permutations of n items, $\{\sigma_1, \ldots, \sigma_m\}$. An intuitive way of summarizing the data consists on defining an $n \times n$ matrix of frequencies F such that F_{ij} is the proportion of the permutations in the sample that have item i in position j, $F_{ij} = m^{-1} \sum_s \mathbb{1}_{\sigma_s(j)=i}$. Following the notation for distributions over other domains, this frequency matrix F is usually denoted first-order marginal matrix.

Behind this intuitive summary, the problem arises when one wants to learn the distribution with the marginal probabilities in F. The difficulty is given by the fact that there are possibly many distributions with the given marginal probabilities. A possible approach consists on selecting that distribution which maximizes the entropy. It is proven in [2] that the maximum entropy distribution has the following expression.

$$P(\sigma) = exp\left(\sum_{i=1}^{n} Y_{(i,\sigma(i))} - 1\right)$$

where $Y \in \mathbb{R}^{n \times n}$ is a matrix of n^2 real coefficients. Unfortunately, the computation of matrix Y is #P-hard.

This first-order summary of the data has been considered in other domains, such as, for instance the on-line learning setting presented in [63].

If first-order marginals count the proportion of the permutations in which $\sigma(i) = j$, second-order marginals count the proportion of the permutations

in which item i_1 is in position j_1 and i_2 is in position j_2. In this way higher order marginals can be defined. Marginal information has been considered in different papers. In particular, one can find in [39] an spectral analysis that takes into account the different marginal distributions of a dataset. A more recent contribution can be found in [65] where the Fourier transform is applied on the low order marginals resulting in an approximate distribution over which inference can be performed. Also, [68] learns such an approximate distribution.

Another remarkable application is that of [83] where a non-parametric estimator based on kernel smoothing for the estimation of the population distribution of a sample of partial and complete permutations is introduced. Similarly, in [73] a visualization framework for ranking data is developed. This framework is also proved to be efficient for rankings of large n. This visualizer is based, in part, on a non-parametric estimate of the density of the rankings.

Thurstone Order Statistics Models

Thurstone models a process for determining stimulus such as the sweetness of a cookie, loudness of a sound, etc [109]. Taking the cookie example, Thurstone assumes that the sweetness of each cookie is a continuous random variable. The ranking process starts when a voter tastes the cookies and evaluates their sweetness. A ranking is obtained by assigning rank 1 to the cookie with the smallest value, rank two to the second smallest value and so on.

Formally, given $\{X_1, X_2, \ldots, X_n\}$ random variables with a continuous joint distribution $F(x_1, \ldots, x_n)$ we can define a random ranking σ in such a way that $\sigma(i)$ is the rank that X_i occupied in $X_1, X_2, \ldots, X_n$. In this way:

$$P(\sigma) = P(X_{\sigma^{-1}(1)} < X_{\sigma^{-1}(2)} < \ldots < X_{\sigma^{-1}(n)})$$

Unconstrained X_i's (and therefore F) produce all kind of distributions over S_n.

The sampling process is as complex as sampling $F(x_1, \ldots, x_n)$. The estimation of the parameters given a sample of permutations has been approached in [85], [87], [86], [112].

Multistage Models

The Multistage models [52] assume that there is a central ranking σ_0 which captures the consensus of the population. A voter generates his personal ranking in $n-1$ stages by making a decision at each stage. The accuracy of this decision ranges between 'making the correct decision' and 'making the worst possible decision' according to σ_0. Therefore, Multistage models have a distribution on the accuracy of the decision at each stage.

The Free or General Multistage model has $\binom{n}{2}$ parameters $p(r, j)$ for $1 \leq j < n$ and $0 \leq r \leq n - j$. The parameter r is associated to the accuracy at stage j, so $p(0, j)$ is the probability of making the correct choice at stage j

and $p(n-j, j)$ is the probability of making the worst possible choice at stage j.

Let $R_j(\sigma)$ be the accuracy at stage j on the ordering σ: then the probability of σ is given as follows:

$$p(\sigma) = \prod_{j=1}^{n-1} p(R_j(\sigma), j)$$

The fundamental reference on Multistage models is [52]; it includes a hierarchy on the Multistage models and sampling and learning methods.

Plackett-Luce's Model

The Plackett-Luce's model (PL) was introduced by Plackett [98], who considered an intuitive ranking process in which the choice probabilities of the items satisfy Luce's choice axiom [80]. Suppose there is a set of candidates for a presidential election. A judge generates a ranking by assigning rank one to the preferred candidate in a first stage. At a second stage assigns rank two to the preferred candidate among the remaining ones, then rank three to the preferred candidate among the remaining ones and so on until a complete ranking is generated.

Formally, let w_i be the weight associated to candidate i and proportional to its preference. The probability of each ranking σ is easily computed as follows:

$$p(\sigma) = \prod_{i=1}^{n-1} \frac{w_{\sigma^{-1}(i)}}{\sum_{j=i}^{n} w_{\sigma^{-1}(j)}}$$

Among the advantages of PL model we can highlight its natural interpretation of the ranking process. Moreover, this model can be easily extended to partial rankings. There are however some drawbacks such as the complexity of computing some marginals: even though $\sum_{\sigma|\sigma(1)=i_1} p(\sigma)$ is easily computed, the complexity of $\sum_{\sigma|\sigma(n)=i_n} p(\sigma)$ is factorial and thus prohibitive for medium-large values of n.

The most remarkable property of the PL model is that a notion of independence naturally arises. Suppose that a voter is ranking a given set of alternatives from most to least preferred. Let i_1 and i_2 be the most and second most preferred options. Note that, when choosing the third preferred alternative, PL model does not depend on whether candidate i_1 was preferred to candidate i_2 or vice versa.

This property is named L-decomposability (or Luce's-decomposability) and implies that the choice probabilities at each stage depend only on the items remaining at that stage, not on the relative ordering of the already selected items. Generalization of the L-decomposability as well as models where they hold can be found in [33], [34].

Multistage and PL models should not be confused. Both define the ranking process as a sequence of $n-1$ stages. However, the probability at each stage

in the Multistage models does not depend on the set of items at each stage. Rather, the probability at each stage depends only on the stage. Consequently, stages are independent and they could be sampled in any order.

Sampling Plackett-Luce's model is straightforward. The learning process was addressed with a Minorization-Maximization algorithm in [67] while [59] proposes a Bayesian approach.

Models Based on Paired Comparisons

Babington Smith proposes a model in which items are compared pair-wisely [10]. This means that there are $\binom{n}{2}$ parameters in the model, denoted p_{ij} which capture the probability that item i is preferred to item j.

It follows that the probability expression of a ranking σ considers every pair of items i, j such that i is preferred to j.

$$P(\sigma) \propto \prod_{(i,j)|\sigma(i)<\sigma(j)} p_{ij}$$

For the generation of a ranking, first, every distribution p_{ij} is sampled. Therefore, for every pair, a preference relation is obtained, i.e. either i is preferred to j or vice versa. If there are not circular triads (such as "i is preferred to j, j is preferred to k and k is preferred to i") a ranking can be produced. Otherwise, the ranking process is started again. Note that the chance of obtaining circular triads increases with the number of candidates.

The best known extension of this model is the Mallows-Bradley-Terry model (MBT) [81] in which the number of parameters is reduced. The estimation of a MBT has been approached with a Minorization-Maximization algorithm [67].

1.2.1 Distance-based permutation models and extensions

In this thesis we have studied three distance-based probability models: Mallows model and two extensions, Generalized Mallows and Weighted Mallows models, all of which are exponential families for permutations.

As we have seen, all the previous models are designed to deal with ranking data by capturing preference relations. We introduce here the distance-based permutation models also named distance-based ranking models and extensions thereof. The distance-based distributions owe its name to the fact that the probability of each permutation depends on its distance to a chosen permutation. The distance can be chosen in several ways; the captured information of these models will, thus, depend on the distance considered.

The original distance-based model for permutations introduced by Mallows [81] was a unimodal distribution which considered both Spearman's and Kendall's-τ distances at the same time. However, the popular version of the

Mallows models are due to Diaconis [38], who uses the original models to motivate a general class of distance-based permutation models as stated in this dissertation.

Despite being one of the first proposed families of models on permutations, it is still one of the most referenced ones. References in the literature to the distance-based models and their extensions range from application papers dealing with multi-label classification [31], label ranking [30], rank aggregation [74] or evolutionary algorithms [28] to theoretical discussions [52], [32], [51] and the analysis of their asymptotic properties [92], [49]. Moreover, distance-based models and extensions thereof have been used as basis for more complex models [89], [83], [93], [78].

Mixtures of Mallows Models also appear in the statistical literature [93], [78], [89]. Moreover, they can also be found in application papers in the combinatorial domain [64]. The authors propose a instance generator for combinatorial optimization problems based on mixtures of Mallows Models which is highly adjustable, generating landscapes of different complexity levels. By properly adjusting the few parameters of the MM, it is possible to control the complexity of the resulting instance, parametrized by the number of local optima, the sizes of the attractions basins, etc.

Mallows Model

The Mallows Model (MM) is an exponential family of probability models for permutation data. Formally, the MM is expressed as follows:

$$p(\sigma) = \frac{exp(-\theta d(\sigma, \sigma_0))}{\psi(\theta)} \quad \text{where } \psi(\theta) = \sum_{\sigma} exp(-\theta d(\sigma, \sigma_0)) \tag{1.5}$$

The central permutation, denoted σ_0, is the mode of the distribution iff the spread (or dispersion) parameter θ is greater than 0. In this case, the probability of any other permutation decays exponentially with its distance to σ_0. The dispersion parameter controls how fast this fall happens. Note that with $\theta = 0$ we obtain the uniform distribution and when $\theta < 0$ then σ_0 is the anti mode, i.e., the permutation with the lowest probability.

The function $d(\sigma, \sigma_0)$ is a distance for permutations which can be chosen in several ways. Diaconis [38] proposes the use of six different distances for this model: Kendall's-τ, Spearman's-ρ, Spearman's footrule, Cayley, Hamming and Ulam. The computational challenge of the MM is the computation of the normalization constant the value of $\psi(\theta)$. However, by using any right invariant distance, such as these six, $\psi(\theta)$ is independent of the central permutation. Moreover, based on the Moment Generating Function of the distance a closed form for $\psi(\theta)$ for some of the considered distances [51] can be obtained.

The best known variant of these models is the one that considers Kendall's-τ as the metric for permutations, also known as the Mallows ϕ-model. This is mainly for two reasons. First, among the proposed metrics, it is one of the most natural for modeling distances in the ranking domain. Second, the MM

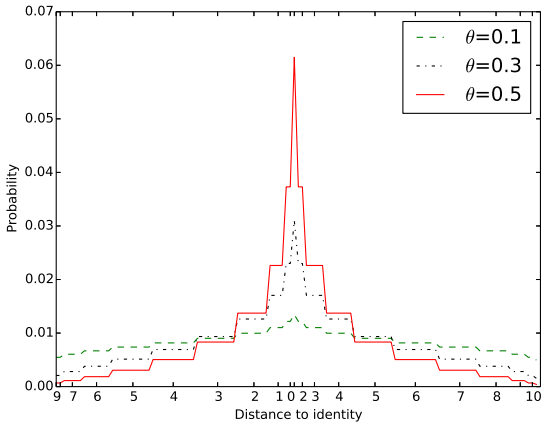


Fig. 1.5: Probability of the permutations of $n = 5$ at each Kendall's-τ distance for different θ

under Kendall's-τ metric has nice theoretical properties and as a result, some operations, such as sampling can be efficiently performed.

The Mallows ϕ-model can also be interpreted as a model based on paired comparisons. Moreover, both Mallows ϕ-model and the MM under the Cayley distance can be interpreted as Multistage models [32].

The MM is considered as the analogous of the Gaussian distribution for permutation spaces. Note that both are defined in a similar way, by specifying the mode and variance. This simile is clear by looking at Figure 1.5. This graphic shows the probability of each permutation of S_5 for three different values of θ, where the identity has been placed in the center while the rest are arranged regarding its distance to σ_0. We can also see how the larger the value of θ the more peaked the distribution. It is worth noticing that every permutation at the same distance has equal probability, assumption that can be too strict in some cases. One can easily find situations in which the disagreement between permutations depend, not only in the number of discrepancies, but also in the positions of those discrepancies. Think for example in the ranking domain, where the distance for permutations usually considered is Kendall's-τ. Two ballots that disagree in the last two candidates of an election may be closer to each other than two ballots that disagree in both the first and the last candidate. Such situation can be modeled under the Generalized Mallows Model (GMM), an extension of the MM.

Generalized Mallows Model

The Generalized Mallows Model (GMM) is the best known among the extensions of the MM. The GMM tries to break the restriction of the MM that

imposes the same probability value to every permutation at the same distance. Instead of one single spread parameter, the Generalized Mallows model (GMM) has $n-1$ spread parameters θ_j for $1 \leq j < n$, each affecting a particular position of the permutation. This allows modeling a distribution with more emphasis on the consensus of certain positions of the permutation while having more uncertainty about some others. This model is more restrictive regarding the distance metric for permutations since it requires the metric to be decomposed in $n-1$ terms as follows:

$$d(\sigma, \sigma_0) = \sum_{j=1}^{n-1} S_j(\sigma\sigma_0^{-1}) \tag{1.6}$$

For any distance that decomposes as the above equation, the GMM is defined as follows:

$$p(\sigma) = \frac{exp(\sum_{j=1}^{n-1} -\theta_j S_j(\sigma\sigma_0^{-1}))}{\psi(\boldsymbol{\theta})} \quad \text{where } \psi(\boldsymbol{\theta}) = \sum_{\sigma} exp(\sum_{j=1}^{n-1} -\theta_j S_j(\sigma\sigma_0^{-1})) \tag{1.7}$$

The most remarkable property of the GMM is related to the possibility of factorizing the distribution. If σ is an u.a.r. chosen permutation, then $S_j(\sigma)$ can be considered as random variables. If those variables are independent, the MGF of the distance can be factorized and, consequently, so do $\psi(\boldsymbol{\theta})$ and the distribution. Moreover, since the MM is a special case of the GMM in which all the dispersion parameters are equal, if the GMM under a certain distance can be factorized, the MM under the same distance can also be factorized.

The GMM under the Kendall's-τ distance is known as ϕ-component model. Among the metrics considered in this dissertation, only Kendall's-τ and Cayley can be used under the GMM since they are the only two that can be decomposed as in Equation (1.6).

Weighted Mallows Model

The decomposition of the Hamming distance has n terms and cannot be used with the GMM in Equation (1.6). In order to adapt the idea behind the GMM for the Hamming distance, we propose an extension of the MM by the name of Weighted Mallows Model (WMM). The WMM is also centered around σ_0 but the difference is that it makes use of a multidimensional dispersion parameter of length n, rather than $n-1$. For a metric decomposable in n terms the WMM is defined as follows:

$$p(\sigma) = \frac{exp(\sum_{j=1}^{n} -\theta_j S_j(\sigma\sigma_0^{-1}))}{\psi(\boldsymbol{\theta})} \quad \text{where } \psi(\boldsymbol{\theta}) = \sum_{\sigma} exp(\sum_{j=1}^{n} -\theta_j S_j(\sigma\sigma_0^{-1})) \tag{1.8}$$

Note that a WMM in which every dispersion parameter is equal is a MM. This model is only used in this thesis under the Hamming distance.

It is worth noticing that the dispersion parameters are not universal measures of spread [84]. This means that if we consider any of these models under two different distances, the dispersion parameters are not a comparable measure of uniformity since, for different metrics, the dispersion parameters are in different scales.

1.3 Motivation and overview of the dissertation

Intuitively, it seems realistic that a distance-based distribution under a particular distance will only be able to model a collection of permutations if that distance is a natural measure of disorder in the given domain. There are in the literature several theoretical, as well as experimental, references supporting this claim [25], [66], [27] and [26].

Among the distances used in this dissertation only Kendall's-τ is a natural measure of dissimilarity in the ranking domain. We could therefore deduce that an exponential model which considers Kendall's-τ will be suited for modeling voting data. This deduction is supported by [25], a theoretical study of models for permutations parametrized by distance functions. They consider several voting rules, which are, in the notation of the present dissertation, functions to recover the central permutation given a sample. This voting rules make sense from the perspective of the social choice context as the ways of choosing the winner given a sample of rankings. The authors prove that given infinite permutations generated from a model like MM under the Kendall's-τ distance, the most popular voting rules can return the consensus. They conclude that the MM under the Kendall's-τ distance makes sense from a social choice context, while MM under Cayley or Hamming do not.

Indeed, they also leave an open question regarding these distances; the authors wonder how the central permutation can be learned given a sample from models such as MM under Cayley or Hamming distances. We will explicitly address those questions in Chapters 3 and 4 avoiding the social choice interpretation of the answers.

In the domain of preference learning, one of the most popular learning approaches estimates the parameters of a sample with pair-wise comparisons (PC) [66]. The authors claim that learning by PC cannot minimize risk with respect to the Cayley, Hamming or Ulam distances.

Despite not suggesting any model based on Cayley, Hamming or Ulam distances to the preference domain, we claim that there is still room for those models in, as we said, those domains where they are a natural measure of disorder. Let us discuss the applications fields for each of the distances considered in this thesis.

The Cayley distance is discussed in Chapter 3, where several relations to other probability models are discussed. One of them is the Ewens sampling formula [46], popular in population genetics, whose implicit generative model is exactly the MM under the Cayley distance. Moreover, its interpretation as a Chinese restaurant process is stablished in the same chapter. This interpretation together with the idea of cycles as groups of items, leads to a application of the MM and GMM under the Cayley distance as clustering procedures [20]. The cycle structure of permutations is a very popular question of great interest, mainly discussed from a theoretical perspective [49], [56], [9], but with applications in, for example, theoretical physics [45]. Summarizing, whatever problem considering a probability distribution regarding the cycles in a permutation is susceptible of using the MM and GMM under the Cayley distance.

The Hamming distance is the metric for the assignments, which appear in a wide range of domains. The well-known Quadratic Assignment Problem (QAP) is perhaps the most prominent example of combinatorial problem based on assignments that arises in several domains [18]. Assignments, at the same time, are the basis for matching problems [96]. The graph isomorphism and graph matching problem, which consists in finding isomorphisms between two graphs, have several applications in areas such as pattern recognition and computer vision [97], [50]. Bipartite matching problems is another example of matching in which it is popular to specify a distribution in the space of matchings and arises in applications such as computational biology and clustering [111]. Probabilistic approaches to these problems, such as the mentioned works, could perfectly consider the Hamming based probability models introduced in this dissertation and extensions thereof.

Let permutation σ be generated from σ' by an delete-insert operation, i.e. moving item i from its position elsewhere, maintaing the rest of the items. Whether i is moved to an adjacent position or to the farthest one, the Ulam distance between them is one because the rest of the permutation is a common subsequence. This means that Ulam distance can capture non trivial alignment between permutations in a way none of the other distances can. This is interesting for streaming algorithms or error correction codes [29], [17], [47], [62] where it is usually denoted Levenstein distance. The delete-insert operator has a simile in biology, where the Ulam distance is used as an approximation of genome rearrangement distances [37]. The edit distance is the generalization of the problem which considers strings instead of permutations. The complexity of the operations under Ulam and edit distance is very similar [5]. The application area is nonetheless wider, including the popular linguistics domain. The relation between the Ulam distance and the patience sorting solitary game has been explored several times [3], [104]. Nevertheless, further relations with card games remain unexplored despite being the delete-insert operator a natural way of changing a deck of cards.

There are also empirical studies showing that the application domain strongly influences the most accurate metric for a distance-based model. In

other words, specific permutation problems are better modeled with distance-based models under a specific distance, even though the ranking interpretation is not considered. This is shown in [27] and [26] (TO SEND) where Estimation of Distribution algorithms (EDA) are used in permutation based optimization problems. EDAs are evolutionary algorithms, similar to genetic algorithms, which, instead of the classic mutation and reproduction operators, learn and sample a probability distribution of feasible solutions at each iteration [77].

The probability models considered in the EDAs have been MM under Kendall's-τ, Cayley and Ulam distances and GMM under Kendall's-τ and Cayley distances. This work includes four different permutation problems, namely Permutation Flowshop Scheduling Problem (PFSP), Quadratic Assignment Problem (QAP), Traveling Salesman Problem (TSP) and Linear Ordering Problem (LOP). As experimental results clearly show, the performance of an EDA is strongly conditioned by both the distance and the problem. As an example, the solutions reached by the MM under the Ulam distance for the LOP are the best with statistically significant differences while the solutions reached by the GMM under the Cayley distances are the best with statistically significant differences in the QAP. In this case, the second best solutions are given by the MM under the Cayley distance.

All these evidences are the motivation behind this thesis. There is not a single best distance; therefore, efficient algorithms have to be developed for different metrics in order to have computational tools to tackle problems of different nature.

The main contributions of this thesis is a set of sampling and learning algorithms for distance-based models and their generalizations under the Kendall's-τ, Cayley, Hamming and Ulam distances. Due to the factorial size of the space of permutations, efficient algorithms for these processes are required. The efficiency of these algorithms relies on several strategies such as exploiting known factorizations of the probabilities or finding efficient expressions for equations where no factorization is known. In many cases the efficiency depends on the ability to bound the search space or on counting and generating permutations uniformly at random among those at a given distance; all these aspects are covered in this dissertation. We should emphasize the fact that these auxiliary algorithms may be useful by their own in contexts other than the statistical. In fact, some of them were known to be open problems in the literature. In this way, the contents of this dissertation belong to disciplines such as computer science, statistics and combinatorics.

Every introduced sampling algorithm generates a sample assuming that the mode is the identity, $\sigma_0 = e$. In case $\sigma_0 \neq e$ one can move a sample centered around π_0 to be centered around σ_0 as follows: let $\{\pi_1, \ldots, \pi_m\}$ be a sample of permutations centered around a mode π_0. A sample $\{\sigma_1, \ldots, \sigma_m\}$ centered around σ_0 can be obtained from the previous one by composing each permutation with σ_0, i.e. $\sigma_i = \pi_i\sigma_0$ for every $1 \leq i \leq m$.

The parameters of a probability distribution are traditionally fitted via maximum likelihood estimation. For a sample $\{\sigma_1, \ldots, \sigma_m\}$ of m i.i.d. per-

mutations the MLE for the parameters of the distribution are those which maximize the likelihood function. Even though the MM is a particular case of the WMM, the MLE for the parameters of the distribution are different for each model. Therefore, we focus on each model separately. We approach the learning process in exact and/or approximate manner.

This dissertation is organized into four main chapters, each presenting the contributions associated to each distance.

Chapter 2 is devoted to the Kendall's-τ distance. It is the best known variant of the models considered in this dissertation and one of the most popular among the models for permutations. This means that it has been extensively considered in the literature. We introduce three sampling algorithms relating them with the existing approaches in the literature. Regarding the learning process, we show how to compute the MLE and experimentally evaluate the most referenced approximate algorithm.

Chapter 3 deals with the Cayley distance. The MM and GMM models under the Cayley distance are described, showing the known factorizations of the models. Then, we show how to count and generate permutations at a given distance. This is done making use of the well known Stirling numbers. Afterwards, three sampling and two learning algorithms are introduced. Finally, the algorithms are tested.

Chapter 4 is devoted to the Hamming distance based probability models. The MM and its extension, WMM under the Hamming distance are introduced. We show how to compute the normalization constant in a very efficient way. Moreover, it is adapted for the computation of the marginal and conditional distributions. Then, three sampling and two learning algorithms are introduced, whose performance is empirically analyzed.

Chapter 5 describes the MM under the Ulam distance. For this model, one sampling and one learning algorithms are introduced.

The algorithms included in this thesis are distributed in a software package. The core of the operations is written in `C++` for the sake of efficiency. Nevertheless, it has been compiled into an `R` package along with the most common operations for permutations. In this way, the package is a complete toolbox for permutations and we expect it to be helpful for a wide range of users, from the novice in the field to the experienced researcher in distributions over permutations; Chapter 6 includes a tutorial in the use of this package.

Chapter 7 concludes the dissertation summarizing the most relevant contributions.

2

Mallows and Generalized Mallows Models under the Kendall's-τ distance

The MM and GMM under the Kendall's-τ distance are the most popular variants among those considered in this dissertation. This is part due to the ability of the Kendall's-τ distance of measuring dissimilarities on the most popular application of permutations: rankings.

Motivated by the idea of helping these exponential models to be as popular for non-ranking data as they are for ranking data, we have particularly focused our efforts on the other metrics. However, no work on distance-based models for permutations can skip the Kendall's-τ distance. The Kendall's-τ distance deserves a monographic chapter, not only because its importance in the literature but also because there is no review on the learning and sampling methods.

For the sake of consistency, we do not approach this chapter as a proper review of the sampling and learning studies. Instead, the sampling algorithms proposed in this thesis have been adapted for the MM and GMM under the Kendall's-τ distance, referencing to existing similar approaches in the literature. This chapter also evaluates the performance of a classical learning algorithm. Therefore, the methods appeared here are contributions unless otherwise stated.

This chapter is organized as follows. The u.a.r. generation of permutations at a given Kendall's-τ distance is detailed in Section 2.1. Besides being interesting by its own, this result is key in a sampling algorithm. The MM and GMM models under the Kendall's-τ distance are introduced in Section 2.2. The factorization of the models are crucial for the efficient expression for the probability of each term of the decomposition vector and the expectation, all of which are included. Section 2.3 includes three new sampling algorithms, namely the Gibbs sampler, which is a well known sampling algorithm in the statistics community, the Multistage algorithm, which exploits the factorization of the models and the Distances sampler, which has a very strong combinatorial basis. Section 2.4 shows how to approximate the parameters of a sample from a MM or GMM under the Kendall's-τ distance. The experimen-

tal evaluation of every proposed algorithm is performed in Section 2.5 and Section 2.6 concludes the chapter.

2.1 Side issues

In the way towards developing efficient sampling and learning algorithms, we have encountered problems which we believe to be interesting beyond the area of probability models for permutation data. Regarding the Kendall's-τ distance, we have faced the question of u.a.r. generating a permutation at a given distance. Moreover, this approach could easily be extended to the similar problem of enumerating the permutations at a given distance.

2.1.1 Generating a permutation at distance d uniformly at random

The question of generating u.a.r. a permutation at distance d from the identity has been, until now, an open problem [6]. We propose here a novel algorithm to solve this question. It is worth noticing that this problem is usually stated in terms of inversions -an inversion is a pair (i, j) of positions of a permutation σ such that $i < j$ and $\sigma(i) > \sigma(j)$. In other words, the problem of generating u.a.r. a permutation at Kendall's-τ distance d is equivalent to the problem of generating u.a.r. a permutation with d inversions.

The question of generating u.a.r. a permutation at distance d from the identity is equivalent to generating a permutation with a given number of inversions, which, until now, was an open problem [6]. We propose here a novel algorithm to solve this question.

Recall that the Kendall's-τ distance decomposition vector of σ is denoted as $V(\sigma) = (V_1(\sigma), \ldots, V_{n-1}(\sigma))$ where each term has vales $0 \leq V_j(\sigma) \leq n - j$ and $d_k(\sigma) = \sum_{j=1}^{n-1} V_j(\sigma)$. Remarkably, there is a bijection between every permutation σ of n items and every possible distance decomposition vector.

The problem of randomly generating a permutation at distance d from the identity is thus equivalent to that of randomly generating a vector $V(\sigma)$ such that $0 \leq V_j(\sigma) \leq (n - j)$ for $1 \leq j < n$. This induces the following alternative representation for the problem: Given $n - 1$ buckets of capacity $n-1, n-2, n-3, \ldots, 1$, distribute d indistinguishable balls in such a way that any possible configuration is equally probable.

Since the number of permutations of n items at Kendall's-τ distance d is $S_k(n, d)$, then, the number of $V(\sigma)$ vectors such that $\sum_{j=1}^{n-1} V_j(\sigma) = d$ is $S_k(n, d)$; equivalently, the number of ways to distribute d indistinguishable balls into $n - 1$ buckets of capacity $n - 1, n - 2, n - 3, \ldots, 1$ is $S_k(n, d)$.

We approach this problem with a recursive procedure. Its base case is $d = 0$ where nothing is done. In the general case, we introduce $k \leq d$ balls in the first bucket and then recursively solve the problem with the rest of the $n - 2$ buckets and $d - k$ balls. The main question is how to choose k if we want every configuration to be equally probable. The insertion of k balls in

the first bucket implies that there will be $S_k(n-1, d-k)$ possible ways to introduce the $d-k$ remaining balls into the $n-2$ remaining buckets. In other words, of the $S_k(n,d)$ total possible configurations of distributing d balls in the $n-1$ buckets, exactly $S_k(n-1, d-k)$ of them will have k balls in the first bucket. Therefore, for a u.a.r. distribution of the balls into the buckets, the probability of introducing k balls in the first bucket is

$$\frac{S_k(n-1, d-k)}{S_k(n,d)}$$

The analogy with the generation of the $V(\sigma)$ vector is quite trivial. Instead of the $n-1$ buckets of capacity $n-1, n-2, n-3, \ldots, 1$ we have a vector $V(\sigma)$ of $n-1$ positions, each restricted to have values smaller than or equal to $n-1, n-2, n-3, \ldots, 1$. The base case is the situation when $d \leq 0$. In this case the output and input vectors are the same. Otherwise, we randomly choose k where the probability of setting $P(V_1(\sigma) = k)$ equals $S_k(n-1, d-k)/S_k(n,d)$. Then, we recursively solve the problem of inserting $d-k$ balls into the remaining positions of the vector, $V_2(\sigma), \ldots, V_{n-1}(\sigma)$.

The cost of generating a permutation at Kendall's-τ distance d is $O(n^2)$ if the values for $S_k(n,d)$ are known.

2.2 The MM and GMM under the Kendall's-τ distance

Formally, the MM under the Kendall's-τ distance is defined as follows:

$$p(\sigma) = \frac{exp(-\theta d_k(\sigma\sigma_0^{-1}))}{\psi(\theta)} \tag{2.1}$$

The Kendall's-τ distance decomposition vector allows writing $d_k(\sigma)$ in the form of Equation (1.1). Consequently, the following equation expresses the GMM under the Kendall's-τ distance.

$$p(\sigma) = \frac{exp(\sum_{j=1}^{n-1} -\theta_j V_j(\sigma\sigma_0^{-1}))}{\psi(\boldsymbol{\theta})} \tag{2.2}$$

If σ is chosen uniformly at random, then each $V_j(\sigma)$ can be seen as random variable with uniform distribution in the range $0, \ldots, n-j$. Moreover, under this condition, a key result states that each $V_j(\sigma)$ is independent [48]. One of the most remarkable consequences is that the GMM under the Kendall's-τ distance can be factorized and that the normalization constant can be posed as a product of $n-1$ terms [51].

$$\psi(\boldsymbol{\theta}) = \prod_{j=1}^{n-1} \psi_j(\theta_j) = \prod_{j=1}^{n-1} = \frac{1 - exp(-\theta_j(n-j+1))}{1 - exp(-\theta_j)} \tag{2.3}$$

Moreover, the probability of the values of each $V_j(\sigma)$ for the GMM under Kendall's-τ can be expressed as follows [51]:

$$p(V_j(\sigma\sigma_0^{-1}) = r) = \frac{exp(-\theta_j r)}{\psi_j(\theta_j)} \tag{2.4}$$

2.2.1 Expected distance and distance decomposition vector

The expected value of the Kendall's-τ distance under the MM is given by the following expression [51].

$$E_\theta[d_k] = \frac{n\, exp(-\theta)}{1 - exp(-\theta)} - \sum_{j=1}^{n} \frac{j\, exp(-j\theta)}{1 - exp(-j\theta)} \tag{2.5}$$

Since each $V_j(\sigma)$ is independent from the others for a uniformly at random chosen permutation σ, the expected value of each term $V_j(\sigma)$ is expressed as follows [51]:

$$E_\theta[V_j] = \frac{exp(-\theta_j)}{1 - exp(-\theta_j)} - \frac{(n-j+1)\, exp(-\theta_j(n-j+1))}{1 - exp(-\theta_j(n-j+1))} \tag{2.6}$$

2.3 Sampling

One of the strengths of the MM is its versatility: It can be motivated in several ways including a model based on paired comparisons. This flexibility has been exploded to sample a permutation as follows: for every pair of items i, j sample the probability of item i being preferred to item j, i.e., $\sigma(i) < \sigma(j)$. If, after sampling every pair of items, there are not cycles of the form $\sigma(i) < \sigma(j) < \sigma(k) < \sigma(i)$, then build the only permutation that matches the preferences. Otherwise, repeat the whole process again.

This naive method is feasible only for low values of n since, as n grows, the probability of obtaining circular triads increases. The algorithms included in this section and in the references therein outperform this naive approach and are, thus, strongly recommended.

Recall that the proposed algorithms generate a sample centered around the identity. To set a different σ_0, each permutation in the sample should be composed with σ_0.

2.3.1 Gibbs sampling algorithm

The Gibbs sampler is a Markov Chain Monte Carlo algorithm very popular in the statistical literature. In fact, two Gibbs sampling approaches were used in MM in [89]. The Gibbs sampler is based on sampling a Markov chain whose stationary distribution is the distribution of interest. Therefore, it generates samples from an approximate distribution of the MM and GMM.

The Gibbs algorithm proceeds as follows:

1. Generate uniformly at random a permutation σ.
2. Build a new permutation σ' equal to σ in all but two adjacent positions, i and $i+1$ where $1 \leq i \leq n-1$ is chosen uniformly at random. In other words, there is an inversion at position i.
3. Let $\gamma = min\{1, p(\sigma')/p(\sigma)\}$. With probability γ the algorithm accepts the candidate permutation moving the chain to it, $\sigma = \sigma'$, and goes back to 2. Otherwise, it discards σ' and goes back to step 2.

The initial samples are discarded (burn-in period) until the Markov chain approaches its stationary distribution. The algorithm iterates in steps 2 and 3 until it generates a given number of permutations.

The third step of the Gibbs algorithm involves the computation of $\gamma = min\{1, p(\sigma')/p(\sigma)\}$ for every permutation generated. It is possible to compute γ without computing $p(\sigma')$ and $p(\sigma)$ (thus reducing the computational time needed for the third step) as follows.

Let i and $i+1$ be the inverted positions and $\sigma(i) > \sigma(i+1)$. Then $\sigma'(i) < \sigma'(i+1)$ and the new solution is accepted as the number of inversions is reduced by one. On the other hand, if $\sigma(i) < \sigma(i+1)$, then $\sigma'(i) > \sigma'(i+1)$ and thus $V_i(\sigma') = V_i(\sigma) + 1$. In this case, under the MM the probability of accepting the new solution σ' is $exp(-\theta)$ while under the GMM is as follows:

$$exp(-\theta_i \cdot (v_{i+1}(\sigma) + 1) - \theta_{i+1} \cdot v_i(\sigma) + \theta_i \cdot v_i(\sigma) + \theta_{i+1} \cdot v_{i+1}(\sigma))$$

The complexity of each iteration is, thus, $O(n)$.

2.3.2 Multistage sampling algorithm

We propose a sampling algorithm able to generate samples from both the MM and GMM. We focus on the GMM as the MM is a special case of it. This algorithm is named Multistage because it exploits the multistage interpretation of the GMM. The process of generating each permutation with this sampling algorithm is divided in two steps, namely:

1. Randomly generate a distance decomposition vector $\mathbf{V}(\sigma)$ from the GMM. Recall that the distribution of each of the terms in the vector is given in Equation (2.4). Note that the probability of each term $V_j(\sigma)$ is independent of the others and, thus, sampling the vector is a process of $n-1$ independent decisions.
2. Generate the permutation σ associated to the decomposition vector $V(\sigma)$. As stated in Section 1.1.1 there is a bijection between one and the other. The generation of σ given a decomposition vector is illustrated in Algorithm 1.

Different forms of exploiting the same idea can be found in the Repeated Insertion Model [43] and in the Generalized Repeated Insertion Model [79].

2.3.3 Distances sampling algorithm

We propose a sampling algorithm based on the following two statements:

- The right invariance property, shared by all the distances considered in this thesis, implies that every permutation has the same number of permutations at each distance.
- In a MM every permutation at the same distance from the central permutation has the same probability.

Since the second statement is not true for the GMM, this sampling method is only valid to generate samples from the MM. The idea is that the probability of obtaining a permutation at distance d under the MM can be written as follows:

$$p(d) = \sum_{\sigma | d_k(\sigma,\sigma_0)=d} p(\sigma) = S_k(n,d)\frac{exp(-\theta d)}{\psi(\theta)} \tag{2.7}$$

Note that the normalization constant $\psi(\theta) = \sum_\sigma exp(-\theta d_k(\sigma\sigma_0^{-1}))$ can be expressed as the sum of $n(n-1)/2$ terms in the following way:

$$\psi(\theta) = \sum_{d=0}^{n(n-1)/2} S_k(n,d)exp(-\theta d)$$

Taking into consideration the previous expressions for the distribution, the process of simulating from the distribution can be done in three stages:

1. Randomly select the distance at which the permutation will lay. By adapting Equation (2.7), the probability of each distance is as follows:

$$p(d) = \psi(\theta)^{-1} S_u(n,d)exp(-\theta d) \quad 0 \leq d \leq n(n-1)/2 \tag{2.8}$$

2. Generate uniformly at random a permutation σ at distance d from the identity permutation e, i.e. $d_k(\sigma) = d$. This step relies on the algorithm for generating uniformly at random a permutation at a given distance shown in Section 2.1.1.

The complexity of computing the probabilities of step 1 is equivalent to the complexity of computing the number permutations per distance, $O(n^3)$, and has to be computed once per run. Moreover, it can be stored in the disk to avoid re-computation. For the second step it has been shown in Section 2.1.1 how to generate permutations at a given distance. It is based of the generation of a distance decomposition vector and has complexity $O(n^2)$.

2.4 Learning

In this section we briefly describe the best known approaches for the MLE of the parameters from a sample of m i.i.d. permutations $\{\sigma_1, \ldots, \sigma_m\}$ coming from a MM or GMM. For the sake of completeness of this dissertation we have included an experimental study of the best known approach which, at the same time, is similar to the learning strategies proposed in other chapters.

2.4.1 Mallows Model

The log-likelihood expression for the MM model is as follows:

$$\begin{aligned} Ln\ \mathcal{L}(\{\sigma_1, \sigma_2, \ldots, \sigma_m\}|\sigma_0, \boldsymbol{\theta}) &= \sum_{i=1}^{m} Ln\ p(\sigma_i|\sigma_0, \theta) \\ &= \sum_{i=1}^{m} Ln\ \frac{exp(-\theta d_k(\sigma_i, \sigma_0))}{\psi(\theta)} = \sum_{i=1}^{m}(-\theta d_k(\sigma_i\sigma_0^{-1}) - Ln\,\psi(\theta)) \\ &= -m\theta\bar{d} - mLn\,\psi(\theta) \end{aligned} \tag{2.9}$$

where $\bar{d} = \sum_{i=1}^{m} d_k(\sigma_i\sigma_0^{-1})/m$. By looking at Equation (2.9), we can see that calculating the value of σ_0 that maximizes the equation is independent of θ. Therefore the MLE estimation problem can be posed as a two step process in which first the central permutation is obtained and then the dispersion parameter for the given $\hat{\sigma}_0$ is computed.

Central permutation

A careful analysis of Equation (2.9) reveals that the permutation σ_0 that maximizes the log-likelihood can be expressed as follows:

$$\hat{\sigma}_0 = \arg\max_{\sigma_0} \sum_{s=1}^{m} -d_k(\sigma_s\sigma_0^{-1}) = \arg\min \sigma_0 \sum_{s=1}^{m} d_k(\sigma_s\sigma_0^{-1})$$

In other words, the MLE for the central permutation is given by the permutation that minimizes the sum of the Kendall's-τ distances to the sample. This problem is also known as Kemeny rank aggregation problem and it is known to be NP-hard [16].

Dispersion parameter

Once the central permutation is known, the MLE for the dispersion parameter is obtained by deriving Equation (2.9) with respect to θ and equaling to zero. The dispersion parameter is that which satisfies the following expression.

$$\frac{n-1}{exp(\theta)-1} - \sum_{k=2}^{n} \frac{k\,exp(-\theta k)}{1-exp(-\theta k)} = \bar{d} \tag{2.10}$$

where $\bar{d} = \sum_{i=1}^{m} d_k(\sigma_i \sigma_0^{-1})/m$. Note that the average distance of the sample, $\bar{d}$, equals the expected Kendall's-τ distance under a MM, whose expression can be found in Equation (2.5).

2.4.2 Generalized Mallows Model

We reproduce in this section the expression of the MLE for the parameters of a GMM under the Kendall's-τ distance which can also be found in [82]. The log-likelihood can be expressed as follows:

$$\begin{aligned} Ln\ \mathcal{L}(\{\sigma_1, \sigma_2, \ldots, \sigma_m\}|\sigma_0, \boldsymbol{\theta}) &= \sum_{i=1}^{m} Ln\ p(\sigma_i|\sigma_0, \boldsymbol{\theta}) \\ &= -\, m \sum_{j=1}^{n-1} \left(\theta_j \bar{V}_j + Ln(1 - exp(-\theta_j(n-j+1))) - Ln(1 - exp(-\theta_j))\right) \end{aligned} \tag{2.11}$$

where $\bar{V}_j = \sum_{i=1}^{m} V_j(\sigma_i \sigma_0^{-1})/m$. The MLE for the dispersion parameters are obtained by equaling to zero the derivative of Equation (2.11), and satisfy the following expression.

$$\bar{V}_j = \frac{exp(-\theta_j)}{1 - exp(-\theta_j)} - \frac{(n-j+1)\, exp(-\theta_j(n-j+1))}{1 - exp(-\theta_j(n-j+1))} \tag{2.12}$$

Note that, again, due to the independence among the $V_j(\sigma)$ variables, the average value of $\bar{V}_j$ equals its expected value, $E_\theta[V_j] = \bar{V}_j$, as can be seen in Equation (2.6).

2.4.3 Learning algorithms

As we have stated, the MLE of the parameters of a sample from a MM is carried out in two stages, the first one dealing with the central permutation and the second one with the dispersion parameter. An exhaustive analysis of more than 100 algorithms for an exact or approximate solution to the Kemeny problem can be found in [4]. Since obtaining the MLE of σ_0 is computationally expensive, a recurrent approach is to approximate σ_0 and calculate the θ for the given σ_0.

The authors of [4] conclude the well known Borda algorithm [23] offers a very good trade off between computational time and accuracy. This empirical conclusion is supported on the theoretical claim which states that the Borda algorithm returns an unbiased estimator of the solution to the Kemeny problem [52].

Let us detail how does the Borda algorithm proceed. Let $\{\sigma_1, \ldots, \sigma_m\}$ be a sample of permutations and $\mathbf{b} = (b_1, \ldots, b_n)$ be a real vector such that $b_j = \sum_{s=1}^{m} \sigma_s(j)$. Borda algorithm generates a permutation π such that $\pi(j) = 1$

for j such that b_j is the smallest entry in $\mathbf{b}$, $\pi(j) = 2$ for j such that b_j is the second smallest entry in $\mathbf{b}$, and so on.

The exact estimation of the parameters of the GMM has been addressed in [82] and [90]. We propose a method to obtain approximate MLE for the parameters of a GMM which consists in splitting the learning process. In other words, our proposed algorithm first approximates the MLE for the central permutation with the Borda algorithm and then estimates the dispersion parameters as shown in Equation (2.11).

2.5 Experiments

In this section we show the performance of the discussed algorithms in both sampling and learning processes. The experiments have been conducted on a cluster of 20 nodes each of them equipped with two Intel Xeon CPUs and 48 GB of memory.

2.5.1 Sampling experiments

The sampling algorithms are evaluated separately for the MM and GMM. However, the evaluation criteria are the same for both models. We have two evaluation criteria, each with a different evaluation procedure.

Our first concern is to show the evolution of the error of each sample as the number of permutations generated, m, increases. The procedure to analyze this evolution is as follows. For each particular setting of n and θ (resp. $\boldsymbol{\theta}$) generate several samples of size $m = 200, 400, 600, \ldots, 19800, 20000$ with each of the sampling algorithms. Measure the error of each sample and plot the results.

Since the algorithms proposed have different complexity, our second concern is the evolution of the error as the computational time increases. The evaluation procedure of this criterium is as follows. For each particular setting of n and θ (resp. $\boldsymbol{\theta}$) generate several samples for $t = 1, 2, 3, \ldots, 14, 15$ seconds with each of the sampling algorithms. Measure the error of each sample and plot the results.

The error in the MM is measured as the sum of the absolute differences between the expected distance, $E[d_k]$ (see Equation (2.5)), and the average distances of the permutations in the samples, $\bar{d} = \sum_{i=1}^{m} d(\sigma_i \sigma_0^{-1})/m$. In the GMM the error is measured as the sum of the differences between the expected V_j, $E[V_j]$ (see Equation (2.6)), and the actual $\bar{V}_j = \sum_{i=1}^{m} V_j(\sigma_i \sigma_0^{-1})/m$.

The parameter setting is as follows. The number of items in the permutations considered are $n \in \{5, 50, 100, 150\}$. The dispersion parameters in the MM case are $\theta \in \{0.1, 0.5, 1, 2, 3\}$. In the GMM case, the first of the dispersion parameters is $\theta_1 \in \{0.1, 0.5, 1, 2, 3\}$ while the rest are set such that $\theta_j = \theta_1 - (j-1)(\theta_1/2(n-2))$ for $j > 1$, i.e. θ_1 is the largest parameter while

the value of the rest decreases linearly to $\theta_{n-1} = \theta_1/2$. For each parameter configuration 10 experiments are run and the average results of them are given. The central permutation is the identity. The Gibbs algorithm discards the first n^2 permutations as part of the burning-period.

Due to the lack of space we have only introduced in this section a representative selection of the experiments. However, the complete results can be found in `http://www.sc.ehu.es/ccwbayes/members/ekhine/sup_result/`. In particular, we include in this section the results of the values of $n \in \{50, 150\}$ and the dispersion parameters $\theta \in \{0.1, 2\}$ in MM ($\theta_1 \in \{0.1, 2\}$ in GMM).

Result for MM

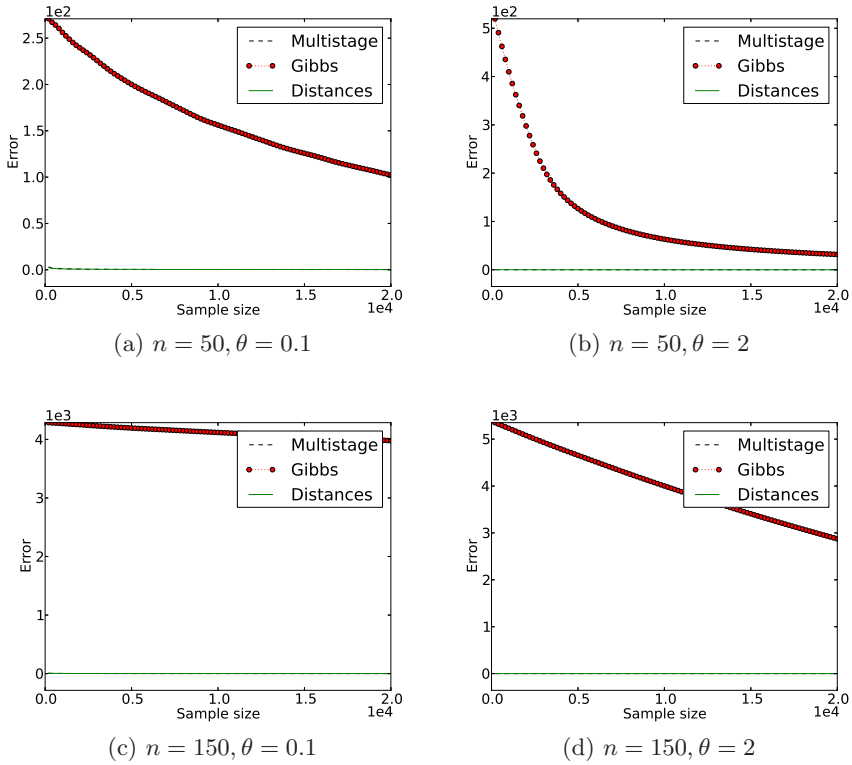
Figure 2.1 shows the evolution of the error of a sample from a MM as m increases. The results of the Multistage and Distances algorithms are almost overlapped and both close to zero. The error of the samples generated with the Gibbs algorithm decreases as m increases. However, Gibbs error is the biggest for every sample size. It quickly converges to the true distribution when n is small, specially for large values of θ when the distribution is more concentrated around the central permutation. Nevertheless, it shows a poor performance as n increases, being more notorious when the dispersion parameters decrease.

The computational time required by Gibbs is much smaller, nonetheless. As an example, Gibbs needs less than 50 milliseconds to generate the 20000 permutations of $n = 150$ items, Multistage around one second while the Distances algorithms needs 3.5 and 1.4 seconds to generate permutations from a MM of $\theta = 0.1$ and 2 respectively. The Distances is computationally more expensive because the generation of a permutation relies on the generation of a distance decomposition vector.

Therefore, the next natural test is to analyze the error when the three algorithms run for the same amount of time. The results of this test for the MM are given in Figure 2.2. Despite being the number of samples generated by the Gibbs algorithm per second much greater, we can see that the error of each sample are always the greatest. As expected, the error of the samples decrease as the computational time increases. The decrease is not prominent for the samples generated with Multistage and Distances because their error is quite stable and close to zero even for small samples sizes. It is notorious for the Gibbs sampler which, again, shows smaller error for larger values of the dispersion parameter.

Result for GMM

Figure 2.3 shows the evolution of the error of a sample as the number of permutations increases for a GMM. Note that only Multistage and Gibbs can generate from a GMM. The conclusions drawn for the GMM are similar to those drawn from the MM. While the samples generated with the Multistage have a small and almost constant error, the samples from the Gibbs are strongly



(a) $n = 50, \theta = 0.1$ (b) $n = 50, \theta = 2$

(c) $n = 150, \theta = 0.1$ (d) $n = 150, \theta = 2$

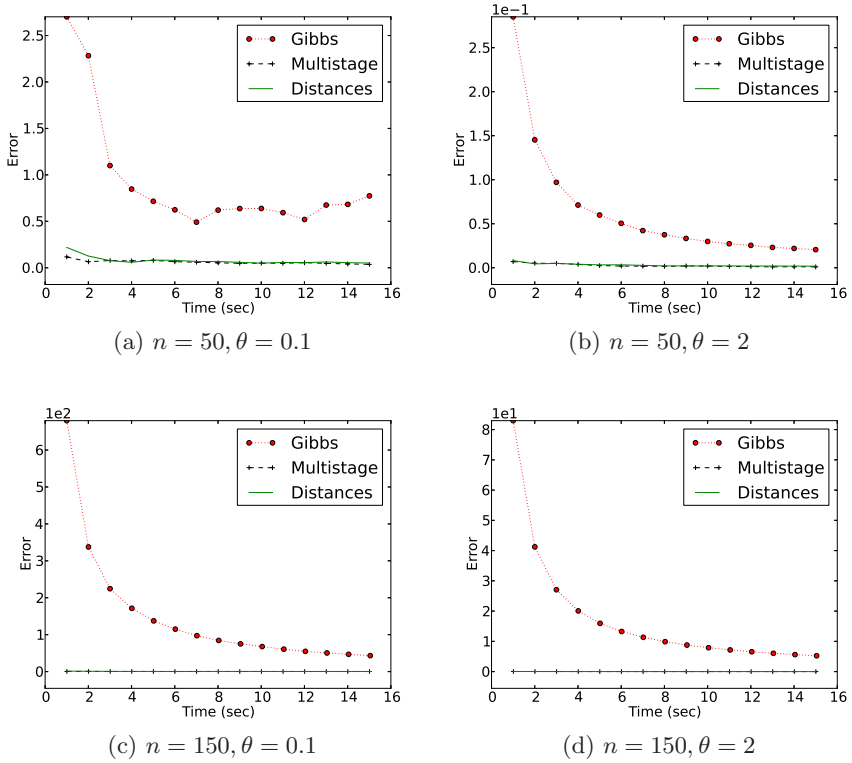
Fig. 2.1: Error of each sampling algorithm as the sample size grows for different θ and n in the MM.

influenced by n and $\boldsymbol{\theta}$, being the largest errors coming from distributions with small dispersion parameters and large number of permutations.

The time results are very similar to those in the MM, that is, Gibbs is much faster than Multistage. Therefore, the next experiment consists on running both algorithms for the same time. These results are shown in Figure 2.4. In this case again, although the Gibbs performance increases with largest $\boldsymbol{\theta}$, the Multistage proves to be the most accurate for every parameter configuration.

2.5.2 Learning experiments

In this section we deal with the approximate learning of the MM and GMM under the Kendall's-τ distance. Since no exact algorithm is considered, we work with synthetic samples generated from distributions of known parameters, σ_0 and θ for the MM (resp. $\boldsymbol{\theta}$ for the GMM). The samples are generated



(a) $n = 50, \theta = 0.1$ (b) $n = 50, \theta = 2$

(c) $n = 150, \theta = 0.1$ (d) $n = 150, \theta = 2$

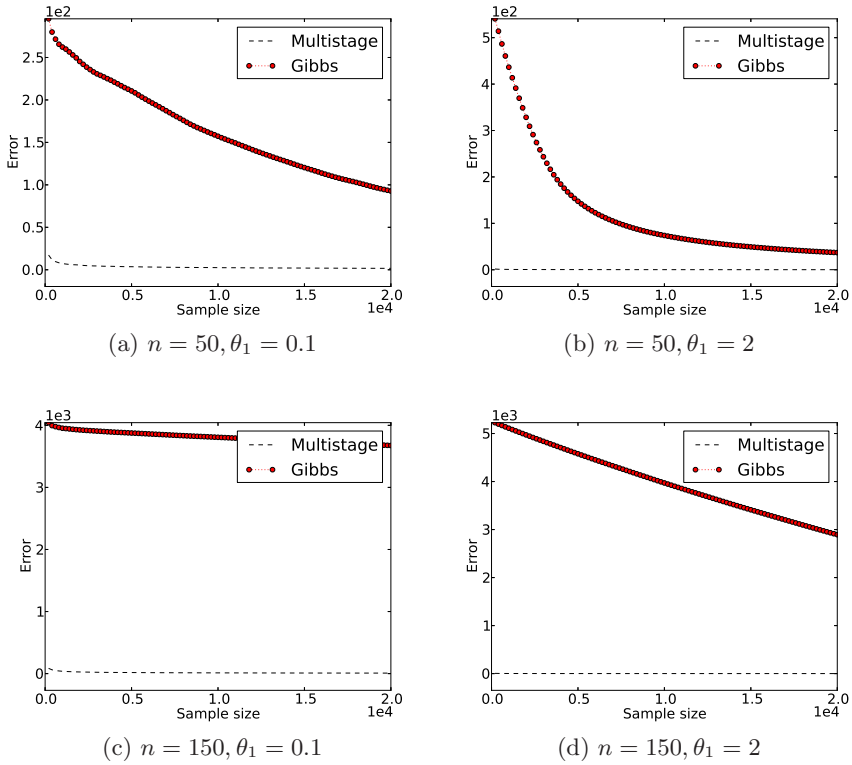
Fig. 2.2: Error of each sampling algorithm as the computational time grows for different θ and n in the MM.

with the Multistage sampling algorithm, because it is the most accurate one of those able to generate from a the MM and GMM.

The experiments in this section are designed to see the evolution of the error of the estimated parameters as the sample size increases.

We now detail the experimental procedure. For each particular setting of n and $\boldsymbol{\theta}$ generate several samples of size $m = 1000, 2000, 3000, \ldots, 9000, 10000$ with the Multistage sampling algorithm and the learn the MLE for the parameters. The central permutation is approximated with the Borda algorithm and denoted $\bar{\sigma}$. Thus $\bar{\sigma}$ is an unbiased estimator of the real σ_0. The dispersion parameters for the MM (resp. GMM) are computed as shown in Equation (2.10) (resp. Equation (2.12)) using a Newton-Raphson algorithm. Finally, denote $\bar{\mathcal{L}}$ the log-likelihoog of the sample given these parameters.

The last question is how to measure the error of the parameters estimated from a sample. Let $\bar{\mathcal{L}}$ be the log-likelihoog of the sample given the estimated parameters, and let $\mathcal{L}_0$ be the log-likelihoog of the sample given the parameters



(a) $n = 50, \theta_1 = 0.1$ (b) $n = 50, \theta_1 = 2$

(c) $n = 150, \theta_1 = 0.1$ (d) $n = 150, \theta_1 = 2$

Fig. 2.3: Error of each algorithm as the sample size grows for different $\boldsymbol{\theta}$ and n in the GMM.

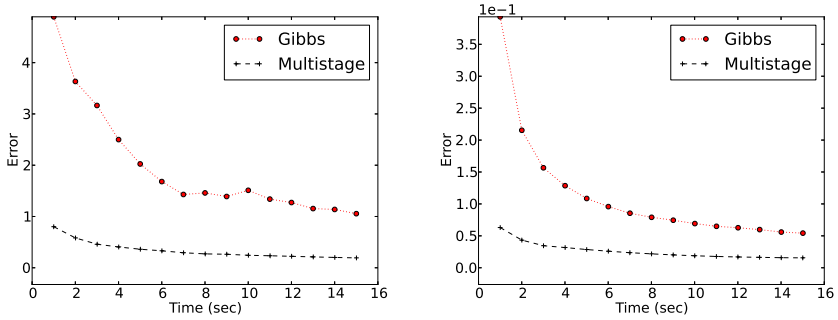
that generated the distribution, σ_0 and θ for the MM (resp. $\boldsymbol{\theta}$ for the GMM). The relative error of $\bar{\mathcal{L}}$ is given by

$$\frac{\bar{\mathcal{L}} - \mathcal{L}_0}{m\,\bar{\mathcal{L}}} \tag{2.13}$$

The sample size m in the denominator is aimed at scaling the log-likelihoog for different sample sizes. Note that the log-likelihoog results of different values of n and θ cannot be compared because they are not in the same scale. However, by measuring the error as in Equation (4.20), the results of the instances with the same n and θ and different m can be compared.

A negative value of Equation (4.20) means that the estimated parameters overfit the sample. For positive values the smaller the value, the better the estimated parameters.

The parameter configuration is the same as in the previous experimental setting.



(a) $n = 50, \theta_1 = 0.1$ (b) $n = 50, \theta_1 = 2$

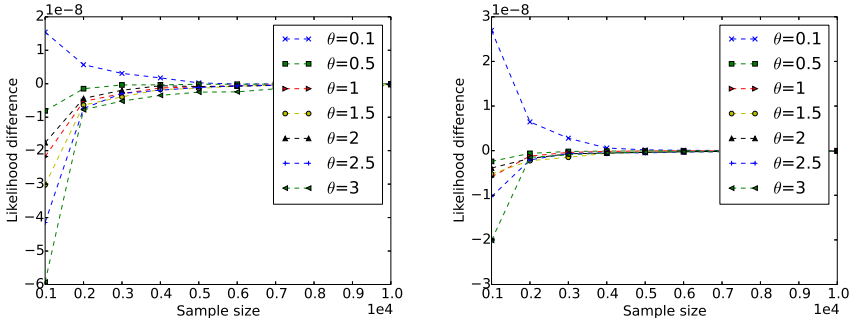
(c) $n = 150, \theta_1 = 0.1$ (d) $n = 150, \theta_1 = 2$

Fig. 2.4: Error of each sampling algorithm as the computational time grows for different $\boldsymbol{\theta}$ and n in the GMM.

The results of the estimation of a MM (resp. GMM) are given in Figure 2.5 (resp. 2.6). The conclusions are similar for both the MM and GMM. The estimated parameters are very close to those of the distribution that generated the sample, being the largest difference of $3 \cdot 10^{-8}$. The estimated σ_0 is the real one in almost all the cases. Also in most of the cases, the value of the error measured as in Equation (4.20) is negative. This means that the distribution is over-fitted.

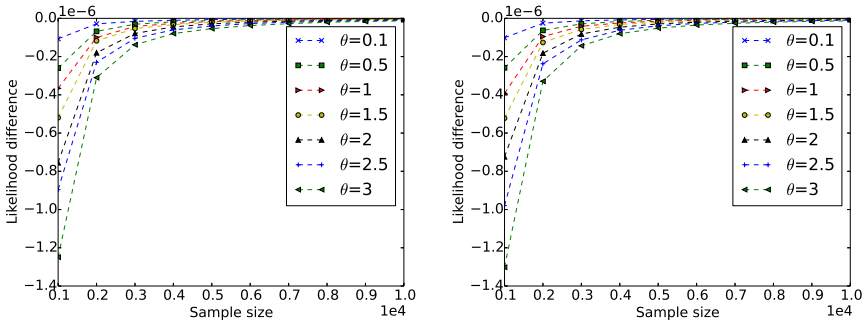
2.6 Conclusions

This chapter explores the MM and GMM under the Kendall's-τ distance. As a result of the possibility of factorizing the distribution, there are efficient expressions for the probability of a permutation, the normalization constant and the expectation of both the distance and distance decomposition vector,



(a) $n = 50$ (b) $n = 150$

Fig. 2.5: Evolution of the error of the estimated parameters for MM as m grows.



(a) $n = 50$ (b) $n = 150$

Fig. 2.6: Evolution of the error of the estimated parameters for GMM as m grows.

all of which are included in this chapter. It is also relevant the expression of the probability distribution of each term of the decomposition vector having each possible value.

This chapter is a compilation of novel and existing sampling and learning algorithms. May it also serve as a script of the following chapters by introducing the ideas of the sampling and learning algorithms.

Three different sampling algorithms are introduced. The Gibbs sampler is a popular algorithm in the statistical domain. It is a very fast algorithm which generates samples form an approximate distribution of the one of interest. The Multistage sampling algorithm exploits the possibility of interpreting the MM and GMM under the Kendall's-τ distance as a Multistage model. It

is accurate and fast. The third algorithm is called Distances sampling algorithm and has a strong combinatorial basis. It relies on counting and u.a.r. generating permutations at each possible distance. The bottle-neck that renders this algorithm as the slowest is the u.a.r. generation of permutations at each distance for Kendall's-τ distance. The latter process is, however, a novel algorithm of interest in different problems.

The experiments confirm the high performance of the Multistage sampling algorithm as it is the most accurate in the two scenarios considered, that is, at the generation of samples of a given number of permutations and at the generation of samples given a computational time limit. Therefore, we strongly recommend the use of the Multistage sampler for generating from both the MM and GMM.

Regarding the learning algorithms, the most popular algorithms are referenced. Moreover, the use of a computationally cheap algorithm is tested which, as well as fast, proved to be accurate.

In view of the results of the learning experiments, we can state that the approximate learning algorithms based on Borda lead to high quality estimator for the parameters of the distribution. Besides, it is a very quick algorithm of time complexity $O(nm)$ for samples of m permutations of n items. This means that it scales linearly with both n and m, so it can easily handle large samples of a large number of items.

3

Mallows and Generalized Mallows Models under the Cayley distance

The Cayley distance counts the number of swaps to convert a permutation into another one. It can be interpreted as the number of cycles of a permutation, which is of great interest since the asymptotic statistics of cycles in random permutations is a recurrent topic in the literature.

The MM and GMM under the Cayley distance have not been almost studied with the remarkable exception of [51]. However, their relations and intersections with other models have and most of the references in the literature come from these other models.

Among the models related to the MM and GMM under the Cayley distance, that which brings the most relevant consequences is the family of Multistage models. By posing the MM and GMM as Multistage models they can be written with simple expressions. As a consequence, operations such as calculating the probability or sampling can be done in a very efficient way.

This chapter is organized as follows. Section 3.1 includes the process for u.a.r. generating permutations at a given distance that will be needed in one of the sampling procedures. The MM and GMM under the Cayley distance are introduced in Section 3.2. It includes the factorization, the expression of the models as Multistage models and the expected values of the distance and distance decomposition vector. Moreover, the links and relation to other models are explored, providing the necessary references. Section 3.3 includes three different sampling algorithms. We propose two learning algorithms, one approximate and one exact, for both the MM and GMM which are introduced in Section 3.4. The experimental evaluation is carried out in Section 3.5 and Section 3.6 concludes the chapter.

3.1 Side issues

As in the previous chapter, we include a sampling algorithm with a strong combinatorial basis relying on the processes of counting and generating u.a.r. permutations at a given distance. Since the permutations at each distance

varies regarding Kendall's-τ or Cayley distances, the results in the previous chapter are no valid in the present one.

Since the Cayley distance $d_c(\sigma)$ can be posed as the number of cycles in σ, the sequence of the number of permutations at each Cayley distance is given by the well known Stirling numbers of the first kind. We introduce an algorithm to generate permutations at a given Cayley distance u.a.r. based on the recurrence for the calculation of the Stirling numbers. This approach could easily be extended to the similar problem of enumerating the permutations at a given distance.

3.1.1 Generating a permutation at distance d uniformly at random

The question of generating a permutation at a given distance from the identity is closely related to that of counting the permutations at a given distance. Algorithm 5 shows a recursive procedure that generates uniformly at random one of the $S_c(n,k)$ permutations of n items with k cycles.

This algorithm is a recursive procedure which, in its base case, when $k = 1$, generates a permutation of n items and 1 cycle, so it uniformly at random generates one of the $(n-1)!$ possible distinct cycles of n items. In the general case, there are two options:

1. Recursively generate a permutation of $n-1$ items and $k-1$ cycles and then place item n alone in its own cycle ($\pi(n) = n$), obtaining the k-th cycle.
2. Generate a permutation of $n-1$ items and k cycles and then randomly insert item n in any of the cycles (such that $\pi(n) \neq n$).

The probability of selecting option 1 or 2 depends on the number of permutations that can be built in each of the two ways. Let us focus on the number of permutations generated in option 1, those of n items and k cycles in which item n is alone in its own cycle. Note that for each permutation of $n-1$ items and $k-1$ cycles there is exactly one way of generating a permutation of n items and k cycles, that which results from setting $\pi(n) = n$. Therefore, from $S(n,k)$ permutations of n items and k cycles, exactly $S(n-1,k-1)$ of them have $\pi(n) = n$. As a consequence, the probability of choosing the first option 1 equals $S(n-1,k-1)/S(n,k)$, while the probability of selecting option 2 equals $1 - [S(n-1,k-1)/S(n,k)] = (n-1)S(n-1,k)/S(n,k)$.

Regarding the computational complexity of this algorithm, the process of sampling a distance is $O(n)$ given the Stirling numbers. Stirling numbers only need to be computed once and require time $O(n^2)$. The process of generating a random permutation given the distance is $O(n)$. Finally when $\sigma_0 \neq e$, the process of calculating $\sigma = \pi\sigma_0$ is also $O(n)$.

3.2 The MM and GMM under the Cayley distance

The Mallows model (MM) can be expressed as follows:

Algorithm 5: *generate_permu*(n, k)
This algorithm generates a permutation of n items with k cycles. Note that every permutation of n items with k cycles is equally probable.

Input: n, num. of items; k, num. of cycles
Output: π, permutation of n items with k cycles
if $k = 1$ **then** π=generate a cycle with the n items; `/* base case */`
else
  $prob = S(n-1, k-1)/S(n, k)$;
  with probability $prob$ `/*` n `stands in a cycle alone */`
    $\pi(1 \dots n-1) = generate\_permu(n-1, k-1)$;
    $\pi(n) = n$;
  end
  otherwise `/*` n `is in a cycle with other items */`
    $\pi(1 \dots n-1) = generate\_permu(n-1, k)$;
    ran = random number in the range $[1, n-1]$;
    $\pi(ran) = n$;
    $\pi(n) = ran$;
  end
end
return π;

$$p(\sigma) = \frac{exp(-\theta d_c(\sigma, \sigma_0))}{\psi(\theta)} \tag{3.1}$$

where $\theta \in \mathbb{R}$ is the spread parameter, σ_0 is the central permutation, $d_c(\sigma, \sigma_0)$ represents the Cayley distance between σ and σ_0 and $\psi(\theta)$ is the normalization constant $\psi(\theta) = \sum_\sigma exp(-\theta d(\sigma, \sigma_0))$.

As we have already seen, the Cayley distance induces a decomposition in an $n-1$ length vector such that $d(\sigma\sigma_0^{-1}) = \sum_{j=1}^{n-1} X_j(\sigma\sigma_0^{-1})$. Moreover, for a uniformly at random drawn σ, each $X(\sigma)$ is an independent Bernoulli random variable, see Section 1.1.2. Therefore, the GMM under the Cayley distance can be factored as a product of $n-1$ terms as follows:

$$p(\sigma) = \frac{exp(\sum_{j=1}^{n-1} -\theta_j X_j(\sigma\sigma_0^{-1}))}{\psi(\theta)} = \prod_{j=1}^{n-1} \frac{exp(-\theta_j X_j(\sigma\sigma_0^{-1}))}{\psi_j(\theta_j)}$$

The normalization constants can be very easily computed with the following expressions.

$$\psi_j(\theta_j) = (n-j)exp(-\theta_j) + 1 \tag{3.2}$$

We would like to point out that these expressions are the correction of those given in [51] which includes typos.

The probability of each position j being the largest item in its cycle under the GMM under the Cayley distance is as follows:

$$p(X_j(\sigma\sigma_0^{-1}) = r) = \begin{cases} \frac{1}{(n-j)exp(-\theta_j)+1} & \text{if } r = 0 \text{ (the cycle is closed)} \\ \frac{(n-j)exp(-\theta_j)}{(n-j)exp(-\theta_j)+1} & \text{if } r = 1 \text{ (otherwise)} \end{cases} \tag{3.3}$$

One of the main advantages of this factorization is that the normalization constant, which was defined as a sum of $n!$ terms, can be posed as a product of $n-1$ terms. Note that since the MM is the special case of the GMM in which every θ_j has the same value, the normalization constant in Equation (3.2) and the decomposition in Equation (3.3) hold for both the MM and GMM under the Cayley distance.

Although, both the MM and GMM are unimodal distributions, the GMM breaks the MM constraint that two permutations at the same distance have the same probabilities. In the case of the GMM, two permutations σ and π have the same probability if they share the same decomposition vector, i.e. $\mathbf{X}(\sigma\sigma_0^{-1}) = \mathbf{X}(\pi\sigma_0^{-1})$.

3.2.1 Expected distance and distance decomposition vector

The expected value of the Cayley distance under the MM is as follows [84] [51]:

$$E_\theta[d_c] = \sum_{j=1}^{n-1} \frac{j}{j + exp(\theta_j)} \tag{3.4}$$

The expected X_j is as follows can be obtained by deriving the logarithm of the MGF of each variable.

$$E_\theta[X_j] = \frac{n-j}{n-j+exp(\theta_j)} \tag{3.5}$$

3.2.2 Related models

In this section we explore the relations with other probabilistic models for permutations.

Non-null Feller coupling

Recall that Fellers coupling (Section 1.1.2) generates a permutation as a sequence of independent cycles. This process is divided in n stages, each of which implies choosing an item that has not already been selected. At each stage s, one out of the $n-s+1$ possible items is uniformly at random selected. Exactly one of them closes a cycle. This process generates every possible permutation with equal probability.

Consider the parametric variation of the Feller coupling with parameter λ in which, at each step of the generation of σ, the item that closes the cycle and the rest do not have the same probability. In this way, we can write this generalization as follows:

$$p(X_j(\sigma) = r) = \begin{cases} \frac{\lambda}{\lambda+n-j} & \text{if } r = 0 \text{ (the cycle is closed)} \\ \frac{n-j}{\lambda+n-j} & \text{if } r = 1 \text{ (otherwise)} \end{cases} \tag{3.6}$$

It is clear that the non-null Feller coupling and the MM under the Cayley distance describe the same generative process.

Following this generative process, let $C^{(n)} = (C_1^{(n)}, C_2^{(n)}, \ldots, C_n^{(n)})$ be a vector where $C_i^{(n)}$ equals the number of cycles of length i in a given permutation of n items. Then, the distribution on $C^{(n)}$ under the Generalized Feller coupling is given by the Ewens Sampling Formula, $ESF(\lambda)$, for any $\mathbf{a} \in \mathbb{Z}_+^n$, [108].

$$P(C^{(n)} = a) = I(\sum_{i=1}^{n} i a_i = n) \frac{n!}{\lambda^{(n)}} \prod_{j=1}^{n} \frac{\lambda^{a_j}}{j^{a_j}} \frac{1}{a_j!}$$

where $I(\cdot)$ is the indicator function and $\lambda^{(n)} = \lambda(\lambda+1)\cdots(\lambda+n+1)$. The $ESF(\lambda)$ was introduced in [46] in the context of population genetics and arises also in areas such as Bayesian statistics. Among its many references we can highlight [8] and [44]. The asymptotic distribution of this formula and its generalizations have been largely studied [49].

Random Utility models

The Latent scale model is a special case of the random utility models [102]. The latter model considers the situation in which one is given a set of options $N = \{1, \ldots, n\}$ and the goal is to select a subset of those items. Therefore, the subset choice models define a probability distribution over every possible subset of the input items.

In the latent scale model, each item j has an associated probability, l_j. This probability is independent of the probabilities of the rest of the alternatives. In this way, the probability of selecting a particular subset is as follows:

$$P(X \subseteq N) = \prod_{x \in X} l_x \prod_{x \notin X} (1 - l_x)$$

We state that any latent scale choice model can be posed as a Generalized Mallows model under the Cayley distance. This can be done taking into consideration Equation 3.3, which expresses the probability of item j being (or not) the largest item in its cycle. Given a latent scale model over n items it is possible to generate a GMM as follows. First, let $\sigma_0 = e$ where the number of items in the permutations is $n-1$ and finally let $p(X_j(\sigma\sigma)) = l_j$.

This not the first work relating choice models and ranking models. In [43] two choice models are studied. The authors conclude that every distribution generated by the latent scale model can be generated by the size independent model. The crucial tool used to prove this statement is the Repeated insertion model, which, interestingly, is a generalization of the Mallows model under the Kendall's-τ distance.

Partition models

These models define a probability distribution on the space of partitions of n items. In particular, the limit of the uniform Dirichlet-multinomial partition model tends to the Ewens sampling formula, [88]. Recall that the generative process that gives rise to the Ewens sampling formula is equivalent to the MM under the Cayley distance, as stated in the previous lines.

On the other hand, the GMM under the Cayley distance can be used to define a partition model as an extension of the previous one. In this case each θ_j is related to the weight of each item to be on a group with previous seen items.

Bayesian statistics

In [42] Bayesian versions of classical problems are studied, one of which is the matching problem. The goal is to study the expected number of fixed points under uniform and non-uniform priors. One of such non-null models is the MM under the Cayley distance.

In [60] a variation of the MM is introduced. This variation makes use of the cycle structure of the permutations. In particular, they define a prior distribution for the central permutation that uses an MM that assigns every permutation with the same cycle decomposition the same probability.

3.3 Sampling

This section introduces three sampling algorithms for the MM and two for the GMM similar to those introduced in the previous chapter. The proposed algorithms generate a sample centered around the identity. To set a different σ_0, each permutation in the sample should be composed with σ_0.

3.3.1 Gibbs sampling algorithm

The Gibbs sampler is a classical algorithm based on sampling a Markov chain whose stationary distribution is the distribution of interest. It was proposed to sample the MM under the Cayley distance in [40], where the authors also prove a quick convergence to the stationary distribution. In this section we extend that algorithm to the case of GMM under the Cayley distance.

The designed process, which is detailed in Algorithm 6, proceeds as follows:

1. Generate uniformly at random a permutation σ.
2. From the current permutation σ, the Gibbs sampler generates a candidate permutation σ'. This permutation is equal to the previous one in all but two positions which have been swapped.

Algorithm 6: Gibbs sampler algorithm

Input: m, number of samples; n, number of items
Output: $\mathcal{S}$, collection of permutations
$\sigma \leftarrow$ uniformly at random generate a permutation `/* Knuth shuffle */`
$\mathcal{S} \leftarrow \sigma$;
for $|\mathcal{S}| < m$ **do**
  for k=*1* **to** n **do** $\sigma'(k) = \sigma(k)$;
  $i, j \leftarrow$ random numbers in the range $\{1, \ldots, n\}$, where $i \neq j$;
  $swap(\sigma'(i), \sigma'(j))$;
  $\gamma = min\{1, p(\sigma')/p(\sigma)\}$;
  with probability γ
    $\sigma = \sigma'$;
  end
  $\mathcal{S} \leftarrow \sigma$;
end

3. Let $\gamma = min\{1, p(\sigma')/p(\sigma)\}$. With probability γ, the algorithm accepts the candidate permutation moving the chain to the candidate permutation, $\sigma = \sigma'$, and goes back to 2. Otherwise, it discards σ' and goes back to step 2.

The initial samples are discarded (burn-in period) until the Markov chain approaches its stationary distribution and so samples from the chain are samples from the distribution of interest.

As explained, a new permutation σ' is built by swapping 2 items i and j of σ. If both items were part of the same cycle in σ, then after the swap the cycle has been split into two new cycles and each swapped item is in a different cycle in σ'. In this case the distance decreases by one unit and the chain moves to the new permutation σ'. Alternatively, if it happens that both items i and j were in different cycles in σ, then after the swap, both cycles are merged into a single one in σ'. In this case, the distance has been increased in one unit and the chain moves to σ' with probability $p(\sigma')/p(\sigma)$. Under the MM under the Cayley distance this ratio equals $exp(-\theta)$. Under the GMM under the Cayley distance, where the probability of a permutation is $p(\sigma) \propto -\sum_{j=1}^{n-1} \theta_j X_j(\sigma\sigma_0^{-1})$ the ratio equals $exp(-\theta_k)$, where k is the item such that $X_k(\sigma\sigma_0^{-1}) = 0$ and $X_k(\sigma'\sigma_0^{-1}) = 1$. Therefore, under the GMM under the Cayley distance it is not necessary to compute the entire $\mathbf{X}(\sigma'\sigma_0^{-1})$ vector, but just the $X_k(\sigma'\sigma_0^{-1})$ of the items k in the cycles of the swapped items. The computational complexity of generating each permutation is thus, $O(n)$.

3.3.2 Multistage sampling algorithm

This novel sampling algorithm can generate permutations from both the MM and GMM under the Cayley distance by exploiting the multistage interpreta-

tion of the models. Recall that Equation (3.3) gives the probability of position j being the largest position in a cycle in π, that is $p(X_j(\pi) = 0)$. Based on Equation (3.3), the Multistage sampling algorithm generates a permutation π from the GMM as a sequence of n decisions. Moreover, this process can be carried out in two different ways, by selecting the positions in increasing order or by selecting items in decreasing order. Both approaches, forwards or backwards are detailed in this section. The former generates a permutation starting from position 1 to position n. The latter, on the other hand, generates the cycles of the permutation. Each of its n iterations assigns an item to a cycle, and the items are chosen in decreasing order.

Generating the permutation forward

The forwards generation of a permutation π can be done in n stages, in a similar way to the Feller coupling. At the first stage, the item to place in position 1 is selected, then the item at position 2, and so on. In general, by following this process, at stage j there is exactly one item that closes a cycle in π. The probability at stage j of that item that closes a cycle is $P(X_j(\pi) = 0)$ as given by Equation (3.3) while the probability of any other item is uniform, $P(X_j(\pi) = 1)/(n - j)$.

Generating the permutation backwards

The backwards generation of a permutation π is performed in n stages by generating a partition of the set of n items into the disjoint cycles that form the permutation. The process iterates by selecting the items in decreasing order and placing each of them in a cycle.

The backwards generation of a permutation from a GMM under the Cayley distance can be seen as a Chinese restaurant process, [21]. The Chinese restaurant process is motivated by the sequential clustering of items. Imagine a Chinese restaurant in which there are infinitely many tables, each of infinite capacity. The tables are labeled with the set of natural numbers and ordered according to that number. The customers come sequentially. The first customer sits at table 1. The second customer either sits at the same table or goes to the first unoccupied table, i.e. table 2. In general, each customer can either sit at any of the occupied tables or sit at the first unoccupied table.

In the simile of the Chinese restaurant process and the backwards generation of a permutation randomly from a GMM under the Cayley distance, the tables are the cycles of the permutation. The customers are the items in the permutation. The process starts with item n in the first cycle. Then, with probability $p(X_{n-1}(\pi) = 0)$ item $n - 1$ is placed in a new cycle. Otherwise, with probability $p(X_{n-1}(\pi) = 1)$, it is placed in the same cycle as item n. The same operation is repeated with the next item, $n - 2$. With probability $p(X_{n-2}(\pi) = 0)$ it is placed in a new cycle, and with probability $p(X_{n-2}(\pi) = 1)$ it is placed in any of the previous cycles. In general, with probability $p(X_j(\pi) = 0)$ item j is placed alone in a new cycle. Otherwise,

that is, with probability $p(X_j(\pi) = 1)$ it is placed in a cycle with the previous items. In the latter case, the cycle to place the item in must be chosen at random with probability proportional to the number of permutations that can be generated in any case. Following with the Chinese restaurant process simile, item j sits to the left of an uniformly at random chosen customer among those that are already sitting. In other words, the cycle in which item j will lay is chosen with probability proportional to the number of items in each cycle.

Intuition behind the dispersion parameters

Based on the backwards and forwards processes for generating permutations, these lines try to clarify the meaning of the dispersion parameters. Based on the backwards generative process it is easy to see that the larger θ_j the more likely to insert item j in a new cycle. In this way, the dispersion parameters θ_j can be considered as the measure of the similarity of item j with the items $i > j$, i.e., the larger θ_j, the more unlikely to group it in a cycle with items $i > j$.

Consider the generation of a permutation σ from a given GMM under the Cayley distance with central permutation σ_0 and dispersion parameters $\boldsymbol{\theta}$. Let θ_j have a large value with regard to the rest of the dispersion parameters. In this situation, it is likely that j is the largest item in its cycle in the generated $\sigma\sigma_0^{-1}$. This has a remarkable consequence: If we consider the minimum set of swaps to change σ_0 into σ, item j will very likely be swapped with item $j' < j$. In other words, θ_j can be seen as the weight of item j to change places with item $j' < j$. Depending on the problem, one may prefer to interpret the dispersion parameters as the strength of the item at position j to change places with the item at position $j' < j$. This can be easily carried out by inverting the permutations and working with the inverse of the permutations, σ^{-1} instead of dealing with σ.

3.3.3 Distances sampling algorithm

The process of drawing a permutation σ from a given MM is carried out by first, randomly selecting a distance between σ_0 and σ and secondly, generating σ at the given distance in an unbiased way (i.e. uniformly at random among the permutations at the given distance).

The key ideas behind the Distances sampling algorithm are the same as in the previous chapter, namely (i) under the MM, every permutation at the same distance from σ_0 is equally probable and (ii) Cayley's bi-invariance implies that every permutation has the same number of permutations at each distance. Recall that the number of permutations n items at each possible distance d is known and denoted by $S_c(n, d)$, see Section 1.1.2. Then, the probability of generating, under a MM, a permutation σ at distance d from the central permutation σ_0 is as follows:

$$p(d) = \sum_{\sigma | d_c(\sigma, \sigma_0^{-1}) = d} p(\sigma) = S_c(n,d) \frac{exp(-\theta d)}{\psi(\theta)} \tag{3.7}$$

Moreover, the normalization constant $\psi(\theta) = \sum_\sigma exp(-\theta d(\sigma \sigma_0^{-1}))$ can be expressed as the sum of n terms in the following way:

$$\psi(\theta) = \sum_{d=0}^{n-1} S_c(n,d) exp(-\theta d) \tag{3.8}$$

Therefore the sampling process of σ is as follows:

1. Randomly select the distance at which the permutation will lay using. The probabilities of each distance can be obtained from Equation (3.7) and are expressed as follows:

$$p(d) = \psi(\theta)^{-1} S_c(n,d) exp(-\theta d) \quad 0 \leq d < n \tag{3.9}$$

2. Generate uniformly at random a permutation σ at distance d from the identity permutation e, i.e. $d(\sigma) = d$. This step relies on the generating uniformly at random a permutation at a given distance shown in Section 3.1.

3.4 Learning

In this section we profusely describe the MLE for the parameters of samples from distributions under the Cayley distance. The form of the MLE for the MM and GMM under the Cayley distance are considered seprately.

3.4.1 Mallows model

In the case of the MM, the likelihood expression is given by the following equation:

$$\begin{aligned} Ln\, \mathcal{L}(\{\sigma_1, \sigma_2, \ldots, \sigma_m\} | \sigma_0, \theta) &= Ln \prod_{s=1}^{m} \frac{exp(-\theta d(\sigma_s \sigma_0^{-1}))}{\psi(\theta)} \\ &= -\theta \sum_{s=1}^{m} d(\sigma_s \sigma_0^{-1}) - m Ln\, \psi(\theta) \end{aligned} \tag{3.10}$$

By looking at Equation (3.10), we can see that calculating the value of σ_0 that maximizes the equation is independent of θ. Therefore, the MLE estimation problem can be posed as a two-step process in which, first, the central permutation and then the dispersion parameter for the given $\hat{\sigma}_0$ are obtained. The MLE for the consensus permutation under the GMM under the Cayley distance is given by the following equation.

$$\hat{\sigma}_0 = \arg\max_{\sigma_0} \sum_{s=1}^{m} -d(\sigma_s \sigma_0^{-1}) = \arg\min_{\sigma_0} \sum_{s=1}^{m} d(\sigma_s \sigma_0^{-1}) = \arg\min_{\sigma_0} \sum_{j=1}^{n-1} \bar{X}_j$$

where $\bar{X}_j = \sum_{s=1}^{m} X_j(\sigma_s \sigma_0^{-1})/m$. Since the MLE of the central permutation is the one which minimizes the sum of the distances to the permutations in the sample, finding the MLE of σ_0 can be considered as a combinatorial optimization problem. In fact, this problem is known in the literature as the swap median problem [99]. Although it is believed to be an NP-complete problem, its computational classification is still an open question [99]. In [19] the parameterized complexity of the swap median problem is studied. The authors start by introducing the definition of *non-dirty items*, which are those items that appear in the same position (which is referred to as the dominating position) in more than half of the permutations in the sample. They also prove that a central permutation places the non-dirty items in their dominating positions. Moreover, they also give a bound on the number of dirty items for which the problem is solvable in polynomial time.

Suppose that the consensus permutation $\hat{\sigma}_0$ is known, the second and last stage of the learning process of an MM concerns the estimation of the spread parameter. The MLE for the dispersion parameter for MM under the Cayley distance is the θ that satisfies the following expression:

$$\sum_{j=1}^{n-1} \frac{j}{j + exp(\theta)} = \bar{d}$$

The average distance is $\bar{d} = \sum_{s=1}^{m} d(\sigma_s \hat{\sigma}_0^{-1})/m$. This expression is obtained by deriving the likelihood in Equation (3.10) and making it equal to zero. Although there is no closed expression for θ, the solution to this equation can be easily calculated with numerical methods such as Newton-Raphson.

3.4.2 Generalized Mallows model

When a sample of permutations is to be modeled by a GMM under the Cayley distance, the likelihood expression can be given as follows:

$$\begin{aligned} Ln\ \mathcal{L}(\{\sigma_1, \sigma_2\ , \ldots, \sigma_m\}|\sigma_0, \boldsymbol{\theta}) &= \sum_{j=1}^{n-1} \sum_{i=1}^{m} -\theta_j X_j(\sigma_i \hat{\sigma}_0^{-1}) + \sum_{i=1}^{m} \sum_{j=1}^{n-1} Ln\ \psi_j(\theta_j) \\ &= \sum_{j=1}^{n-1} (\sum_{i=1}^{m} -\theta_j X_j(\sigma_i \hat{\sigma}_0^{-1}) + \sum_{i=1}^{m} Ln\ \psi_j(\theta_j)) \\ &= \sum_{j=1}^{n-1} -m(\theta_j \bar{X}_j + Ln\, \psi_j(\theta_j)) = \sum_{j=1}^{n-1} \mathcal{L}_j \end{aligned} \tag{3.11}$$

where $\bar{X}_j = \sum_{i=1}^{m} X_j(\sigma_i \hat{\sigma}_0^{-1})/m$. It is worth noticing that the MLE for the central permutation may not be the same as that which minimizes the distance

to the sample. Suppose that the MLE for σ_0 is known. Then, the MLE for the spread parameters are given by the following expression:

$$\hat{\theta}_j = Ln(n-j) - Ln(\bar{X}_j/(1-\bar{X}_j)) \tag{3.12}$$

This expression is obtained by deriving the likelihood in Equation (3.11) and making it equal to zero.The GMM learning process cannot be divided into two different stages as in the MM case, and thus the MLE calculation must be carried out simultaneously for every parameter. However, Equation (3.11) decomposes the likelihood into $n-1$ sums, being each a function on a particular $\bar{X}_j$ and θ_j. Moreover, $\hat{\theta}_j$ is also a function of $\bar{X}_j$, see Equation (3.12), so the likelihood for each position can be expressed as follows:

$$\sum_{j=1}^{n-1} \mathcal{L}_j = \sum_{j=1}^{n-1} -\bar{X}_j Ln(n-j) + (n-j)^2 + \frac{(n-j)^2}{\bar{X}_j} + \bar{X}_j Ln\frac{\bar{X}_j}{1-\bar{X}_j} - m \tag{3.13}$$

Thus, the GMM under the Cayley distance estimation problem is still a combinatorial optimization problem that can be written in the following way.

$$\begin{aligned}\hat{\sigma}_0 = \arg\max_{\sigma_0} Ln\ \mathcal{L}(\{\sigma_1,\sigma_2,\ldots,\sigma_m\}|\sigma_0,\boldsymbol{\theta}) = \arg\max_{\sigma_0} \sum_{j=1}^{n-1} \mathcal{L}_j = \\ \arg\max_{\sigma_0} \sum_{j=1}^{n-1} [-\bar{X}_j Ln(n-j) + (n-j)^2 + \frac{(n-j)^2}{\bar{X}_j} + \bar{X}_j Ln\frac{\bar{X}_j}{1-\bar{X}_j} - m]\end{aligned} \tag{3.14}$$

In order to give an efficient exhaustive algorithm that searches the space of permutations, it is of great importance to take into account the fact that each of these functions $\mathcal{L}_j$ is strictly increasing on X_j.

3.4.3 Learning algorithms

In this section, we introduce a approximate and an exact algorithm to find the MLE of the central permutation and the spread parameters for both the MM and GMM under the Cayley distance. Note that the evaluation of any candidate solution σ_0 is carried out in terms of distance in the MM and in terms of likelihood in the GMM under the Cayley distance and also that the optimal MLE for σ_0 may not be the same for both models. Both distance and likelihood can be obtained given the $\bar{\mathbf{X}}$ vector: The sum of the distances to the sample in the MM under the Cayley distance is $\sum_{s=1}^{m} d(\sigma_s\sigma_0^{-1}) = m\sum_{j=1}^{n-1} \bar{X}_j(\sigma_s\sigma_0^{-1})$ and the log likelihood expression of the GMM under the Cayley distance is given in Equation (3.13). In this way, these two algorithms optimize a function defined over the collection of $\bar{X}_j$ variables.

3.4.3.1 Approximate algorithm

The approximate algorithm that we propose proceeds in two stages. First, it generates a solution in a greedy manner and then the initial solution is improved using a Variable Neighborhood Search (VNS) [91].

The greedy process, starting from an empty permutation, iterates in n steps adding at each step an item to a position in the following way.

1. Given the partial permutation of $0 \leq j < n$ items, σ_0, evaluate every candidate solution. The candidate solutions σ'_0 are those partial permutations of $j+1$ items built by adding one more item to σ_0 in any of the $(n-j)^2$ possible ways.
2. Set as σ_0 the candidate solution that evaluates the best (ties are solved selecting one permutation at random).

This process is repeated until σ_0 is a complete permutation. The VNS performed afterwards is a approximate algorithm that makes use of two separate neighborhood systems, namely the insert and the swap. The insert operator considers as neighbors all those permutations that result from inserting an item into another position. The swap operator considers as neighbors all those permutations that result from swapping two positions. The local search for both neighborhood systems selects at each iteration the best neighbor of the current one (or selects a solution uniformly at random among those with the best evaluation in case of ties). The VNS, detailed in Algorithm 7, applies alternatively each of the neighborhood systems until both are stuck at the same local optimal solution.

Algorithm 7: Variable Neighborhood Search (VNS)

Input: σ_0 solution obtained by the greedy process
Output: σ_0 local optima for the local search with both insert and swap neighborhood systems

```
σ''_0 = σ_0;
repeat
    σ_0 = σ''_0;
    σ'_0 ← local optimum found with the local search with the insert
    neighborhood starting from σ_0;
    σ''_0 ← local optimum found with the local search with the swap
    neighborhood starting from σ'_0;
until σ_0 == σ''_0;
```

3.4.3.2 Exact algorithm

The exact algorithm implements a branch and bound search. It explores the space of partial permutations of the first j out of n items ($\sigma_0^{-1}(r) = i_r$ for $r \leq j$ and $\sigma_0^{-1}(r)$ is unknown for $r > j$). These partial permutations are generated following a deep-first search on the tree shown in Figure 3.1.

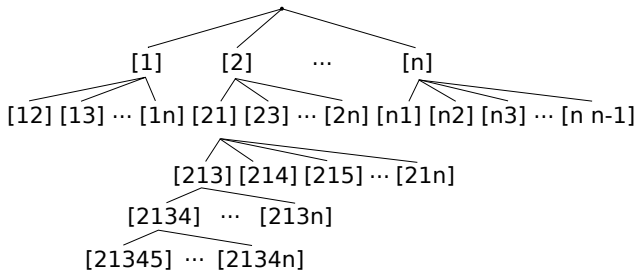


Fig. 3.1: Permutation generating tree

The fact that the visited permutations are partial permutations of the first j-th indexes makes it possible to evaluate $\bar{X}_r$ for all $r \leq j$ and thus to partially evaluate a solution. In order to illustrate this point, let us start with $\bar{X}_1$. Given a sample permutation σ_s, $X_1(\sigma_s\sigma_0^{-1})$ can be calculated as follows:

$$X_1(\sigma_s\sigma_0^{-1}) = \begin{cases} 0 & \text{if } \sigma_s\sigma_0^{-1}(1) = 1 \\ 1 & \text{if } \sigma_s\sigma_0^{-1}(1) > 1 \end{cases}$$

In the case of $\bar{X}_2$, we can proceed similarly:

$$X_2(\sigma_s\sigma_0^{-1}) = \begin{cases} 0 & \text{if } \sigma_s\sigma_0^{-1}(2) = 2 \\ 1 & \text{if } \sigma_s\sigma_0^{-1}(2) > 2 \\ ?? & \text{if } \sigma_s\sigma_0^{-1}(2) < 2 \end{cases}$$

However, in this case if $\sigma_s\sigma_0^{-1}(2) < 2$, we need to resort to $\sigma_0^{-1}(1)$ in order to calculate that value, but because we construct the partial permutation ordered, that value has been assigned in the partial permutation. Particularly if $\sigma_0^{-1}(1) = 2$, then $X_2(\sigma_s\sigma_0^{-1}) = 0$, and if $\sigma_0^{-1}(1)$ is larger than 2, then $X_2(\sigma_s\sigma_0^{-1}) = 1$.

In general, in order to calculate $\bar{X}_j$ when all the previous indexes have been assigned, we have to proceed similarly:

$$X_j(\sigma_s\sigma_0^{-1}) = \begin{cases} 0 & \sigma_s\sigma_0^{-1}(j) = j \\ 1 & \sigma_s\sigma_0^{-1}(j) > j \\ ?? & \sigma_s\sigma_0^{-1}(j) < j \end{cases}$$

In the case of $\sigma_s\sigma_0^{-1}(j) = r < j$, we can recursively calculate $\sigma_s\sigma_0^{-1}(r)$ until we reach the first k such that $(\sigma_s\sigma_0^{-1})^k(r) = r' > j$.

Unfortunately, this simple method is extremely inefficient. However, we propose a method that, by exploring the same tree as in Figure 3.1, runs in $O(m)$ (where m is the number of permutations in the sample) to evaluate $\bar{X}_j$ at each node of depth j. This algorithm is based on modifying the sample of permutations at each evaluation. Briefly speaking, the evaluation at each node of depth j is carried out using the samples modified by its parent in the tree to obtain $\bar{X}_j$ and modify the samples before continuing the search. In particular, the evaluation is carried out as follows:

Evaluation

The evaluation at each node requires the collection of samples modified by its parent in the tree as in Figure 3.1, together with the previous evaluation results $\bar{X}_r$ for $r < j$. Then, our proposed algorithm proceeds in two stages,

1. Calculate $\bar{X}_j$ as follows

$$\bar{X}_j = \sum_s X_j(\sigma_s \sigma_0^{-1})/m = \sum_s I(\sigma_s^{-1}(j) \neq \sigma_0^{-1}(j))/m \tag{3.15}$$

   where by $I(\cdot)$ we denote the indicator function.
2. For every permutation in the sample such that $\sigma_s^{-1}(j) \neq \sigma_0^{-1}(j)$, make the necessary swap in order to have $\sigma_0^{-1}(j) = \sigma_s^{-1}(j)$.

At this point, the $\bar{X}_r$ for $r \leq j$ are known and are not going to change for any node in its subtree. Although $\bar{X}_r$ for $r > j$ are not known, they can be lower bounded, as we explain in the following paragraphs. All these values of $\bar{X}_r$ for $1 \leq r < n$ are used to evaluate the current partial solution. Since $\bar{X}_r$ for $r > j$ are just lower bounds, we know that if the evaluation of the current partial solution is worse than the best solution found so far, then any complete permutation that is situated in its subtree will be even worse. In this case, the current branch can be bounded. On the other hand, if the current evaluation is better than the best evaluation found so far, continue the deep-first, branch and bound search in its subtree by passing $\bar{X}_r$ for $r \leq j$ and the modified samples to all its descendants.

Lower bound

Let us now show how to obtain a lower bound for $\bar{X}_r$ for $r > j$ at a node of level j. At each node the evaluation of $\bar{X}_j$ is given as shown in Equation (3.15). Since for all $r < j$, $\sigma_0^{-1}(r) = \sigma_s^{-1}(r)$ then item $\sigma_0^{-1}(j)$ will be in position $r > j$ of σ_s^{-1}. Therefore, Equation (3.15) can be also written as follows:

$$\bar{X}_j = \sum_s \sum_{r>j} I(\sigma_s^{-1}(r) = \sigma_0^{-1}(j))/m$$

Intuitively, $\bar{X}_j$ can be seen as the proportion x/m where m is the number of permutations in the sample and x counts how many permutations σ_s^{-1} place the item in $\sigma_0^{-1}(j)$ to the right of j. Basically, this lower bound $\bar{X}_k^*$ for $k > j$ goes through the collection of samples σ_s^{-1}, counts the number of permutations that place item q (for any value of q) to the right of the k and select the minimum among them. Therefore, a simple lower bound for $\bar{X}_k$, denoted as $\bar{X}_k^*$, can be expressed as

$$\bar{X}_k^* = min \sum_s \sum_{r>k} I(\sigma_s^{-1}(r) = q)/m \quad \text{for any possible } q \tag{3.16}$$

The computational complexity of evaluating this lower bound for every remaining $j < k$ is done in $O((n-k)m)$. Such an overload in the computational time is counterbalanced by the number of branches bounded by the algorithm, which results in an efficient algorithm, as will be shown in the experimental section.

In [19] it is shown that every consensus permutation places the non-dirty items in their dominating positions. This fact can be used when fitting an MM under the Cayley distance to reduce the search space since we will focus on just non-dirty items.

Example

Let us illustrate our proposed exact algorithm with an example. We will perform a sequence of evaluations of the branch of the tree as in Figure 3.1 corresponding to the set of partial permutations [2___], [24__], [241_]. In this process we will show how the X_j are computed, the samples modified and the lower bound obtained. Assume that the collection of samples is as follows:

$$\begin{array}{ll} \sigma_1 : 2134 & \sigma_1^{-1} : 2134 \\ \sigma_2 : 3241 & \sigma_2^{-1} : 4213 \\ \sigma_3 : 1342 & \sigma_3^{-1} : 1423 \end{array}$$

We first show the evaluation of the node at level 1 that corresponds to $\sigma_0^{-1} = [2\_\_\_]$. The first step is to compute $\bar{X}_1$ (the proportion of permutations in the sample such that $\sigma_s^{-1}(1) \neq \sigma_0^{-1}(1)$) as shown in Equation (3.15).

$$\bar{X}_1 = \sum_s I(\sigma_s^{-1}(1) \neq \sigma_0^{-1}(1))/3 = 2/3$$

Now, for the sake of efficiency of the following evaluations, we make the necessary swaps to have $\sigma_s^{-1}(1) = \sigma_0^{-1}(1)$ (remember that the current partial solution is $\sigma_0^{-1} = [2\_\_\_]$). The resulting collection, which will be used by every descendant of the current node, is as follows:

$$\begin{array}{ll} \sigma_1 : 2134 & \sigma_1^{-1} : 2134 \\ \sigma_2 : 3142 & \sigma_2^{-1} : 2413 \\ \sigma_3 : 3142 & \sigma_3^{-1} : 2413 \end{array}$$

Let us now show how the lower bound for every $r > 1$ is computed.

- The number of permutations in which item 1 is to the right of position 2 is 2.
- The number of permutations in which item 3 is to the right of position 2 is 3.
- The number of permutations in which item 4 is to the right of position 2 is 1.

Since the minimum is 1, then by using Equation (3.16) we get $\bar{X}_2^* = 1/3$. The same process is carried out for the lower bound $\bar{X}_3^*$.

- The number of permutations in which item 1 is to the right of position 3 is 0.
- The number of permutations in which item 3 is to the right of position 3 is 2.
- The number of permutations in which item 4 is to the right of position 3 is 1.

Since the minimum is 0, then $\bar{X}_2^* = 0$. Suppose that the evaluation of the current partial permutation $(2/3 + 1/3 + 0 = 1)$ is still better than that of the best solution found so far and so the algorithm continues the deep first search on the tree. Let the next node in level 2 be $\sigma_0^{-1} = [24\_\_]$. The evaluation of X_2 is

$$\bar{X}_2 = \sum_s I(\sigma_s^{-1}(2) \neq \sigma_0^{-1}(2))/3 = 1/3$$

and the set of samples

$$\begin{array}{ll} \sigma_1 : 3142 & \sigma_1^{-1} : 2413 \\ \sigma_2 : 3142 & \sigma_2^{-1} : 2413 \\ \sigma_3 : 3142 & \sigma_3^{-1} : 2413 \end{array}$$

The computation of the lower bound is now as follows:

- The number of permutations in which item 1 is to the right of position 3 is 0.
- The number of permutations in which item 3 is to the right of position 3 is 3.

The lower bound is now $\bar{X}_3^* = 0$.

If the algorithm continues by evaluating $\sigma_0^{-1} = [241\_]$, it will result in the same collection of permutations as the one above.

Partial permutations

In the context of rankings, partial permutations are generally used to consider preference information about a subset of the items. When dealing with problems relevant to the cyclic structure of arrangements the partialness can be also useful. In this sense, there can be information about certain assignments and no information about some others. Moreover, they can come as complete cycles of a subset of the items or as a partial cycles. Our proposed learning processes can be adapted to handle partial permutations by including an EM (expectation maximization) algorithm.

3.5 Experiments

We now analyze the performance of the proposed sampling and learning algorithms.

3.5.1 Sampling experiments

In this section we measure the performance of the sampling algorithms for the evaluation of a sample $\{\sigma_1, \ldots, \sigma_m\}$. The algorithms are evaluated regarding two different criteria. For each criterium a different evaluation procedure is considered.

The first one is the evolution of the error of each sample as its number of permutations, m, increases. The procedure to clarify this evolution is as follows. For each particular setting of n and θ (resp. $\boldsymbol{\theta}$) generate several samples of size $m = 200, 400, 600, \ldots, 19800, 20000$ with each of the sampling algorithms. Measure the error of each sample and plot the results.

The second concern is the evolution of the error as the computational time increases. The evaluation procedure of this criterium is as follows. For each particular setting of n and θ (resp. $\boldsymbol{\theta}$) generate several samples for $t = 1, 2, 3, \ldots, 14, 15$ seconds with each of the sampling algorithms. Measure the error of each sample and plot the results.

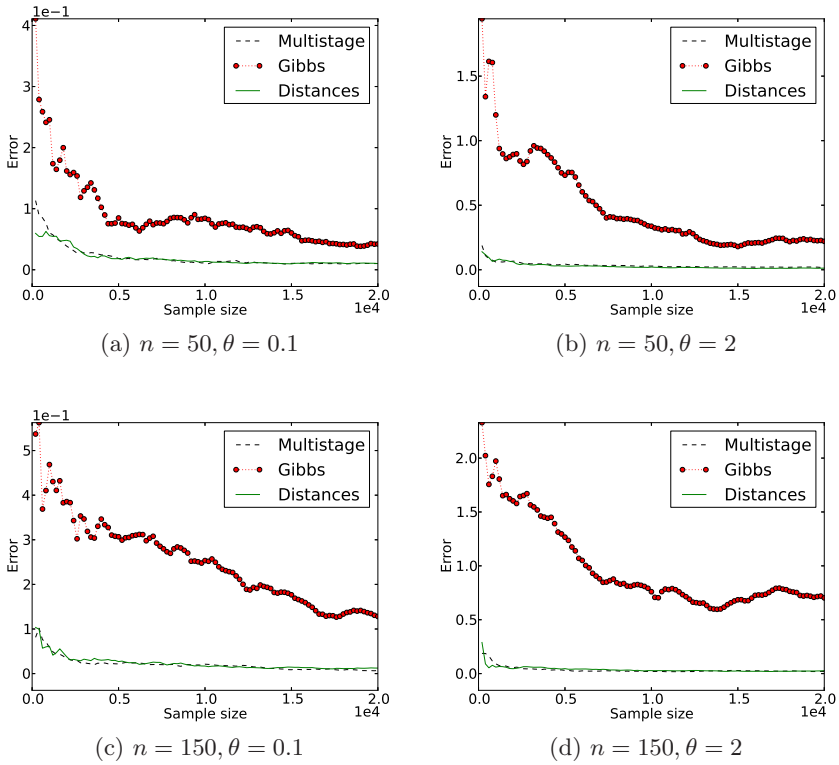
The error of sample $\{\sigma_1, \ldots, \sigma_m\}$ in the MM is measured as the sum of the differences between the average distance to the sample $\bar{d} = \sum_{i=1}^{m} d(\sigma_i, \sigma_0)/m$, and the expected distance, $E[D]$ which is given in Equation (3.4)). In the GMM the error is measured as the sum of the differences between the expected X_j, $E[\bar{X}_j]$ (see Equation (3.4)), and the actual $\bar{X}_j = \sum_{i=1}^{m} X_j(\sigma_i \sigma_0^{-1})/m$. In both cases, the central permutation is the same as that which generated the distribution, i.e., the identity.

The parameter setting is as follows. The number of items in the permutations considered are $n = \{5, 50, 100, 150\}$. The dispersion parameters in the MM case are $\theta = \{0.1, 0.5, 1, 2, 3\}$. In the GMM case, on the other hand, the first of the dispersion parameters is $\theta_1 \in \{0.1, 0.5, 1, 2, 3\}$ while the rest are set such that $\theta_j = \theta_1 - (j-1)(\theta_1/2(n-2))$ for $j > 1$, i.e. θ_1 is the largest parameter while the value of the rest decrease linearly to $\theta_{n-1} = \theta_1/2$. For each parameter configuration 10 experiments are run and the average results of them are given. W.l.o.g. the central permutation is the identity. The Gibbs algorithm, following the recommendation in [40], discards the first $n\,log(n)$ permutations as part of the burning-period.

Due to the lack of space we have only introduced in this section a representative selection of the experiments. However, the complete results can be found in `http://www.sc.ehu.es/ccwbayes/members/ekhine/sup_result/`. In particular, we include in this section the results of the values of $n = \{50, 150\}$ and the dispersion parameters $\theta = \{0.1, 2\}$ in MM ($\theta_1 = \{0.1, 2\}$ in GMM).

Result for MM

The evolution of the error as m increases is given in Figure 3.2. For equal sample sizes, the error of those samples generated with Gibbs are always worse than those generated with Multistage or Distances, which are close to zero. The error gets smaller as the sample size increases, more notoriously for



(a) $n = 50, \theta = 0.1$ (b) $n = 50, \theta = 2$

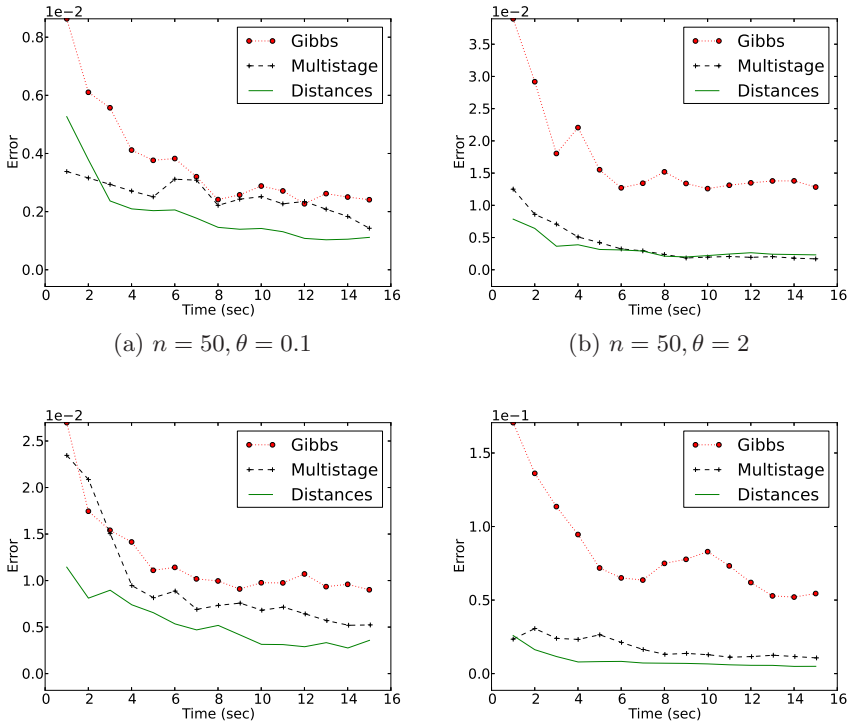
(c) $n = 150, \theta = 0.1$ (d) $n = 150, \theta = 2$

Fig. 3.2: Error of each sampling algorithm as the sample size grows for different θ and n in the MM.

the Gibbs sampler since Multistage and Distances' error is flatter and smaller in every case. The improvement of the Gibbs sampler is more evident as θ increases for a particular n.

The computational time required for the generation of the 20000 permutations of 50 items is much faster for the Gibbs, which run for less than 50 ms, than for the Distances, 200 ms, and Multistage algorithms, which required 300 ms. For permutations of 150 items, the time increases to 100 ms for the Gibbs, 1 second for the Distances and 3 seconds for the Multistage.

Given these computational times it is natural to wonder what happens when the algorithms are run for the same amount of time. The results can be found in Figure 3.3. With the remarkable exception of the instances of $n = 150$ and $\theta = 0.1$, Multistage and Distances are much more accurate than Gibbs.



(a) $n = 50, \theta = 0.1$ (b) $n = 50, \theta = 2$

(c) $n = 150, \theta = 0.1$ (d) $n = 150, \theta = 2$

Fig. 3.3: Error of each sampling algorithm as the computational time grows for different θ and n in the MM.

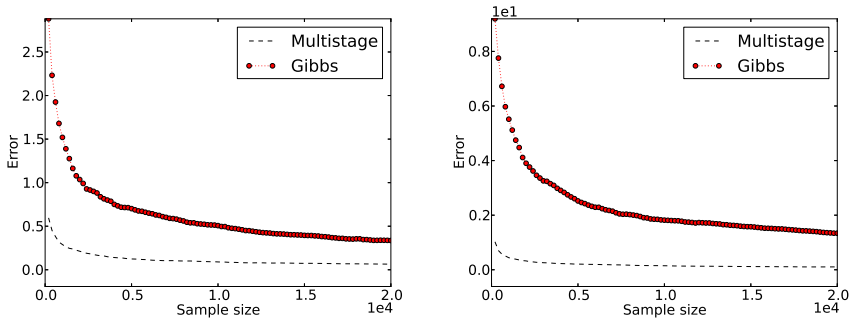
Result for GMM

The conclusions drawn for the GMM are very similar to those drawn from MM. The error of the instances of the instances generated with the Multistage have smaller error and also, the performance difference gets more evident as $\boldsymbol{\theta}$ increases. This behavior is repeated for both evaluation contexts, generation of samples of equal size, Figure 3.4 and generation of samples for the same computational time, Figure 3.5.

3.5.2 Learning experiments

In this section we test the performance of the algorithms for the estimation of the maximum likelihood parameters.

The parameter setting is as follows. The number of items of the permutations is $n \in \{5, \ldots, 13, 15, 20, 25\}$. The sample size is $m \in \{1000, 1500, 2000\}$.



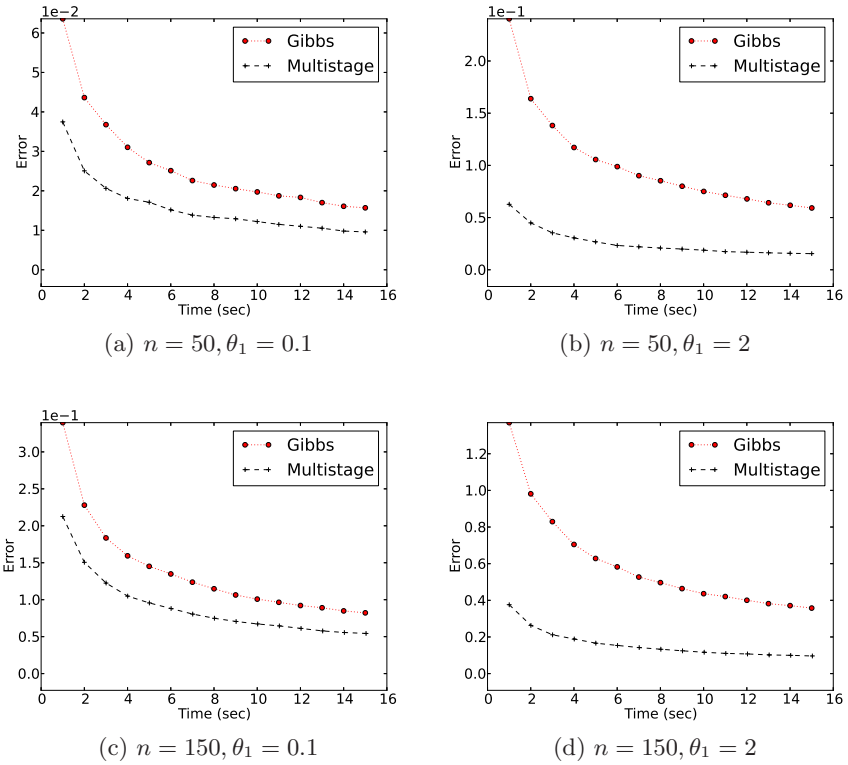
(a) $n = 50, \theta_1 = 0.1$ (b) $n = 50, \theta_1 = 2$

(c) $n = 150, \theta_1 = 0.1$ (d) $n = 150, \theta_1 = 2$

Fig. 3.4: Error of each algorithm as the sample size grows for different $\boldsymbol{\theta}$ and n in the GMM.

The dispersion parameter in MM under the Cayley distance is set as $\theta \in \{0.2, 0.4, 0.6, 0.8, 1, 1.5, 2, 3, 4, 5, 6, 7, 8$
$, 9\}$. In the GMM under the Cayley distance case each θ_j can have different values. We give $\theta_1 \in \{0.2, 0.4, 0.6, 0.8,$
$1, 1.5, 2, 3, 4, 5, 6, 7, 8, 9\}$ as reference and set $\theta_j = \theta_1 - (j-1)(\theta_1/2(n-2))$ for $j > 1$, i.e. θ_1 is the largest parameter while the value of the rest decrease linearly to $\theta_{n-1} = \theta_1/2$. The central permutation is randomly chosen. For every configuration of the parameters, the experiments are repeated 10 times and their average results are shown. Due to lack of space, we only show in this paper the results of the representative selection $m = 1000$ for both the MM and GMM under the Cayley distance[1]. We have not used the same parameter setting as in the previous section. Instead, this parameter setting is the

[1] The complete results can be found on the web `http://www.sc.ehu.es/ccwbayes/members/ekhine/permus/learning_full_results.pdf`.



(a) $n = 50, \theta_1 = 0.1$ (b) $n = 50, \theta_1 = 2$

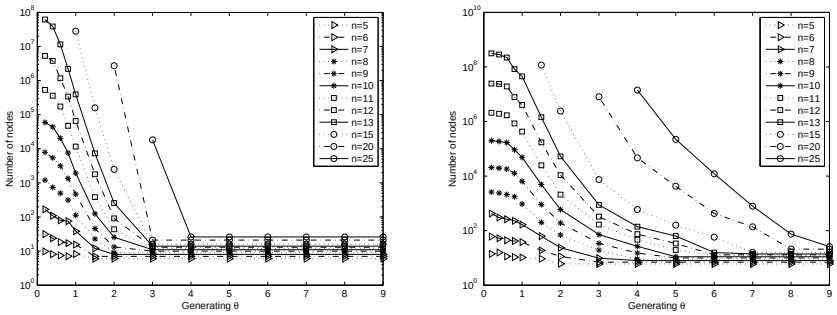
(c) $n = 150, \theta_1 = 0.1$ (d) $n = 150, \theta_1 = 2$

Fig. 3.5: Error of each sampling algorithm as the computational timegrows for different $\boldsymbol{\theta}$ and n in the GMM.

same as that used in [82], where an efficient algorithm for the GMM under the Kendall's-τ distance is introduced.

In order to analyze the quality of the exhaustive algorithm, the number of partial solutions evaluated is given. Clearly, the larger the number of bounded branches, the smaller the number of solutions evaluated by the algorithm and the less time the exhaustive search will require. Moreover, in this way the results are independent of the machine. However, in order to give an intuition of the required time, we can say that the average number of nodes visited per second for every experiment is 4000. Some of the instances are not given (see for example that of $n = 20$ and $\theta = 1$ in the MM under the Cayley distance case, Figure 3.6a). Those instances were aborted for excessing the time limit, which was set at 15 hours for the MM and 48 hours for the GMM under the Cayley distance.

Figure 3.6 shows the number of evaluations when learning from each of the samples generated under different θ (X-axis) and for different values of n.



(a) MM (b) GMM

Fig. 3.6: Number of partial solutions evaluated by the exhaustive algorithm for a sample of 1000 permutations.

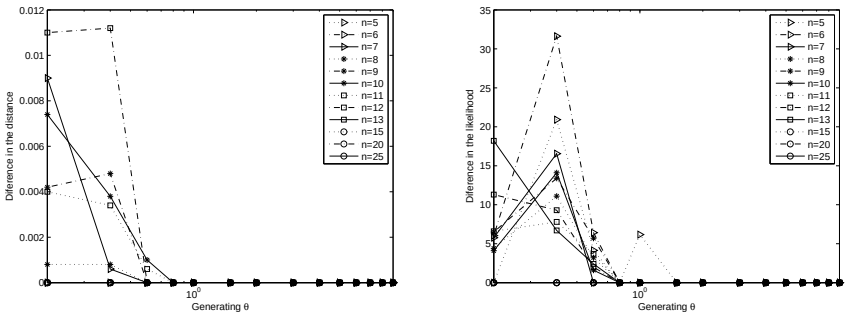
Figure 3.6a concerns the MM under the Cayley distance learning case. Due to the efficiency of the bounding strategy, instances drawn from almost uniform distributions and large values of n can be solved. Moreover, the number of partial solutions evaluated quickly decreases as the dispersion parameter that generated the sample increases. For $\theta > 4$ all these samples were correctly solved evaluating just $n + 1$ partial permutations, although there are almost no dirty items in the samples.

Similar conclusions can be drawn when fitting a GMM under the Cayley distance (see Figure 3.6b). Once again, when there is consensus in the sample the exhaustive process is very efficient. For this particular setting of the θ_j parameters, one can learn the MLE for samples of permutations generated under $\theta_1 = 4$ of $n = 25$ items in less than 5 hours. We conclude that the described algorithm happens to be an efficient learning procedure that can deal with samples of permutations up to $n = 25$ items if there is some consensus in the sample. For instances drawn from almost uniform distributions where this exhaustive algorithm is inefficient in terms of time, we propose the use of the heuristic algorithm introduced previously.

In Figure 3.7 the accuracy of the heuristic solutions are compared to that of the MLE for the central permutation. In the MM under the Cayley distance (Figure 3.7a) the quality of the heuristic solution is given as $\sum_{s=1}^{m} |d(\sigma_s, \sigma_0^+) - d(\sigma_s, \sigma_0^*)|$ where σ_0^+ and σ_0^* are the exact and heuristic solutions respectively. Conversely, in GMM under the Cayley distance (Figure 3.7b), the accuracy is the difference between the likelihood of σ_0^+ and σ_0^*. In both cases the heuristic algorithm is very accurate, being enough to have very little consensus (say $\theta = 1$ or $\theta_1 = 1$) to find the optimal solution. Moreover, the computational time was less than 300 ms for each of the 10 repetitions of the heuristic search.

These results show that the lower bounding technique for the branching algorithm is very efficient despite its naivety. Moreover, the computational

overload is worth it due to the large number of branches bounded. For samples with no consensus, where the probability of each permutation is almost the same, and large n, the exhaustive algorithm requires a large amount of time. In such a situation we may want to sacrifice the accuracy for a quick and good solution with no guarantee of being optimal. In this situation we encourage the use of the heuristic algorithm.



(a) MM, difference in the distances (b) GMM, difference in the likelihood

Fig. 3.7: Difference between the exhaustive and the heuristic solution.

3.6 Conclusions

In this chapter we have dealt with the MM and GMM under the Cayley distance. We discuss both models and their relations to other models in the literature. Moreover, we show how to exploit their properties to obtain efficient expression for calculating the probability of a permutation under these models or calculate the expected distance and distance decomposition vector.

We explore different interpretations of the sampling and learning processes, finding out that the process of generating from the distribution can be motivated as a Chinese restaurant process and the process of parameter fitting for MM can be interpreted as the combinatorial problem of finding the permutation that minimizes the sum of the distances to a given sample of permutations.

Our contributions include three sampling algorithms. The first one is the Distances sampling method, which can simulate the MM. It makes use of the Stirling numbers to generate permutations directly from the distribution. It is therefore quick and accurate. We also introduce the Multistage sampling method for simulating the MM and GMM which exploits the multistage interpretation of the GMM under the Cayley distance. Since it samples directly

from the distribution, it is as accurate as the Distances sampling algorithm. Moreover, as it is not based on the Stirling numbers it can simulate distributions over permutations of very large n. The last algorithm is an adaptation of a Gibbs sampler to the GMM. Since it does not sample directly from the distribution, it is not as accurate as the previous algorithms. However it is very fast and with a careful implementation it can simulate distributions of large n. Our results show that the MM and GMM under the Cayley distance can be efficiently sampled even for large values of n. The results of the experimental evaluation makes us encourage the use of the Distances sampler for the MM and the Multistage for the GMM.

These algorithms make use of two novel procedures to generate permutations which can be useful by themselves. The first one generates uniformly at random permutations of n items at a given distance d. The second one generates uniformly at random permutations with a given $\mathbf{X}(\sigma)$ vector, i.e. the only available information is whether or not an item is the largest item in its respective cycle, but not the cycle structure. This can done by both generative processes detailed in the Multistage sampling process, the forwards and the backwards.

Moreover, we also introduce two learning algorithms, a heuristic and an exhaustive one, for both the MM and GMM. Due to a branch and bound strategy, the exhaustive algorithm is very efficient, even for samples that have little consensus. The experimental section reports the results of the experiments for n up to 25. When there is no consensus in the sample or the number of items is very large, we propose a heuristic algorithm. This algorithm is able to learn the optimal parameters of the distribution when there is some consensus in the sample. When there is no consensus the obtained solution is very close to the optimal. Moreover, it is a very quick algorithm.

4

Mallows and Weighted Mallows Models under the Hamming distance

The Hamming distance is the most popular distance for permutations when the ranking interpretation is not considered. It appears in areas that range from coding to matching.

The MM under the Hamming distance has been considered in the literature. The normalization constant has got a closed-form expression what simplifies dealing with the distribution. Unfortunately, its most popular extension, the GMM, cannot be used under the Hamming distance. We introduce an adaptation of the GMM for the Hamming distance that we call Weighted Mallows Model (WMM). Although no closed-form for the normalization constant exists, we present an efficient way of computing it which runs in time $O(n^2)$ for permutations on n items. Moreover, we have adapted it to compute the marginal and conditional probabilities, which, in turn, are the basis to an efficient sampling method.

The organization of this chapter is as follows. Section 4.1 gives results on the fields of combinatorics and calculus that, apart of being useful by their own, are recurrent in this chapter and thus need to be as efficient as possible. The MM and WMM under the Hamming distance are detailed in Section 4.2, where, we show how to compute the normalization constant, calculate the expectation, the marginal and conditional values. Three sampling algorithms are introduced in Section 4.3. The expression of the MLE for the parameters of a given sample of permutations as well as a learning algorithm are given in Section 4.4. The experimental evaluation of the proposed algorithms is performed in Section 4.5 and Section 4.6 concludes de chapter.

4.1 Side issues

In this section we show how to efficiently compute the distance to a sample, which is used intensively during the learning process. Moreover, a combinatorial result regarding the number of permutations with at least k unfixed points

and the optimized computation of the elementary symmetric polynomial of a set of variables and its derivative are also introduced.

4.1.1 Efficient computation of the distance to a sample

In the maximum likelihood estimation of the parameters, it will be necessary to compute the sum of the distances from a permutation σ_0 to a sample of permutations a large number of times. In order for the algorithms to be efficient, this sum must be quickly computed.

If calculated naively, the sum $\sum_{s=1}^{m} d(\sigma_s, \sigma_0)$ requires time $O(nm)$. We show here how to compute the above expression in $O(n)$ for any σ_0 given an appropriated summary of the sample. In particular, we consider the bi-dimensional frequency matrix, F, of dimension $n \times n$ where $F_{i,j}$ counts the number of permutations in the sample such that have item j at position i, i.e. $F_{i,j} = \sum_{s=1}^{m} \mathbb{1}_{\sigma_s(i)=j}$.

Lemma 1 *The number of permutations with unfixed points at position j in the sample $\{\sigma_1, \ldots, \sigma_m\}$ can be computed by means of the frequency matrix F as follows.*

$$\sum_{s=1}^{m} H_j(\sigma_s \sigma_0^{-1}) = m - F_{\sigma_0^{-1}(j),j}$$

Proof. Recall that $H_j(\pi)$ equals 0 iff j is a fixed point in π. Let $\mathbb{1}$ be the indicator function and F be an $n \times n$ matrix such that $F_{i,j} = \sum_{s=1}^{m} \mathbb{1}_{\sigma_s(i)=j}$. Let $\sigma_0(i) = j$ (or equivalently $\sigma_0^{-1}(j) = i$). Note that the following relation holds.

$$H_j(\sigma\sigma_0^{-1}) = 1 - \mathbb{1}_{\sigma\sigma_0^{-1}(j)=j}$$

Therefore, the number of unfixed points in the sample $\sigma_s\sigma_0^{-1}$ at position j can be expressed by means of the frequency matrix F as follows.

$$\begin{aligned}\sum_{s=1}^{m} H_j(\sigma_s\sigma_0^{-1}) &= \sum_{s=1}^{m}(1 - \mathbb{1}_{\sigma_s\sigma_0^{-1}(j)=j}) = m - \sum_{s=1}^{m} \mathbb{1}_{\sigma\sigma_0^{-1}(j)=j} \\ &= m - \sum_{s=1}^{m} \mathbb{1}_{\sigma_s(i)=j} = m - F_{i,j} = m - F_{\sigma_0^{-1}(j),j}\end{aligned}$$

Corollary 1 *The sum of the distances from σ_0 to each of the permutations in the sample $\{\sigma_1, \ldots, \sigma_m\}$ can be computed by means of the frequency matrix F as follows.*

$$\sum_{s=1}^{m} d(\sigma_s, \sigma_0) = \sum_{s=1}^{m}\sum_{j=1}^{n} H_j(\sigma_s\sigma_0^{-1}) = \sum_{j=1}^{n}(m - F_{\sigma_0^{-1}(j),j})$$

Note that the computation of the matrix F also requires time $O(mn)$. Therefore, the computation of the F matrix is worth it if the distance to a sample is going to be computed for more than one distinct σ_0 permutation.

4.1.2 Counting permutations with at least k unfixed points

We have already shown how to count the number of permutations with exactly d unfixed points in Section 1.1.3. We deal now with the related problem of counting the number of permutations with at least k unfixed points. This count will be later useful for the computation of the marginal distribution of the WMM and for a sampling algorithm.

We start by the similar (but easier) problem of counting the number of permutations with at least k fixed points. Note that the number of permutations such that have fixed points at positions $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ is the same as the number of permutations that have fixed points at positions $1, 2, \ldots, k$. This count is denoted as $f(n, k)$ and it is easy to see that $f(n, k) = (n - k)!$.

The same situation happens when we are counting the permutations with unfixed positions, i.e., the number of permutations such that have at least k unfixed points at positions $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ is the same as the number of permutations that have unfixed points at positions $1, 2, \ldots, k$. This count is denoted as $g(n, k)$ and can easily be computed using an inclusion-exclusion approach. Before introducing the recursion, let us show how the base cases are computed.

Counting $g(n, 1)$

The set of $n!$ permutations can be divided into two groups regarding $H_1(\sigma)$, those permutations in which $H_1(\sigma) = 0$ (there is a fixed point at position 1) and those in which $H_1(\sigma) = 1$. The cardinality of the set of permutations in the former group is $f(n, 1) = (n - 1)!$. The cardinality of the latter set is therefore

$$g(n, 1) = n! - f(n, 1)$$

Counting $g(n, 2)$

The number of permutations in which both 1 and 2 are unfixed points is computed in the same way. From the complete set of $n!$ permutations we discard those in which $\sigma(1) = 1$ (this is, a set of $f(n, 1)$ permutations) and then those in which $\sigma(2) = 2$ (again, $f(n, 1)$ permutations) -note that the number of permutations such that $\sigma(2) = 2$ is equal to the number of permutations such that $\sigma(1) = 1$, that is $f(n, 1)$.

However, if we set $g(n, 2) = n! - f(n, 1) - f(n, 1)$ we are under counting, since those permutations such that $\sigma(1) \neq 1 \wedge \sigma(2) \neq 2$ are subtracted two times. Therefore, we must compensate this undercount by adding again those permutations that $\sigma(1) \neq 1 \wedge \sigma(2) \neq 2$.

$$g(n, 2) = n! - f(n, 1) - f(n, 1) + f(n, 2)$$

Counting $g(n,3)$

We can proceed in the same way for computing the number of permutations of n items at least 3 fixed points.

$$g(n,3) = n! - 3f(n,1) + 3f(n,2) - f(n,3)$$

Counting $g(n,k)$

Following the same reasoning we reach a recursive expression for $g(n,k)$ which is as follows:

$$g(n,k) = n! + \sum_{i=1}^{k}(-1)^i \binom{k}{i} f(n,i) = n! + \sum_{i=1}^{k}(-1)^i \frac{k!(n-i)!}{i!(k-i)!} \tag{4.1}$$

Equation (4.1) counts the number of permutations with at least k unfixed points in time $O(n)$.

4.1.3 Elementary Symmetric Polynomial

The efficient computation of the normalization constants for the probability distributions considered in this chapter is crucial for the efficient computation of the sampling and learning processes. We propose an expression for the normalization constant based on the calculation of the Elementary Symmetric Polynomials (ESP) on the parameters of the distribution, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$. In this section we introduce the notion of ESP and show how to compute them efficiently. Moreover, we will give an expression for their derivatives.

First of all, let us define the set $R_k = \{(i_1, \ldots, i_k) | 1 \leq i_1 < i_2 < \ldots < i_k \leq n\}$ as the set of different groups of k indices out of the n total indices $\{1, \ldots, n\}$. The ESP of degree k on a set of n variables, $\gamma_k(X_1, \ldots, X_n)$, is defined as follows:

$$\gamma_k(X_1, \ldots, X_n) = \sum_{R_k} \prod_{s=1}^{k} X_{i_s}$$

By abusing notation γ_k will be used to denote the ESP of degree k, $\gamma_k = \gamma_k(X_1, \ldots, X_n)$. The ESP γ_k can be efficiently computed with the following recursion [14]:

$$\gamma_k(X_1, \ldots, X_n) = \begin{cases} 1 & \text{if } k = 0 \\ \sum_{i=1}^{n} X_i & \text{if } k = 1 \\ \gamma_k(X_1, \ldots, X_{n-1}) + \gamma_{k-1}(X_1, \ldots, X_{n-1}) X_n & \text{otherwise} \end{cases} \tag{4.2}$$

By using the above formula, the computational complexity for computing the ESP $\gamma_k(X_1, \ldots, X_n)$ is $O(n^2)$. Note that a naive computation will require time $O(2^n)$.

Splitting the ESP

In both learning and sampling processes it is necessary to compute the ESP on all the subgroups of $n-1$ variables. The naive approach is to take each of the n possible subgroups of $n-1$ variables and compute the ESP of every subgroup as shown in Equation (4.2). This will require $O(n^3)$ time. We introduce a method for computing the ESP on all the subgroups of $n-1$ variables in $O(n^2)$.

In the next lines there is an example of the elementary symmetric polynomial on 4 variables. Each γ_k is computed by adding up every products inside the braces.

$$\gamma_1 = \begin{cases} \gamma_1^1 = \{X_1 \\ \bar{\gamma}_1^1 = \begin{cases} X_2 \\ X_3 \\ X_4 \end{cases} \end{cases} \qquad \gamma_2 = \begin{cases} \gamma_2^1 = \begin{cases} X_1X_2 \\ X_1X_3 \\ X_1X_4 \end{cases} \\ \bar{\gamma}_2^1 = \begin{cases} X_2X_3 \\ X_2X_4 \\ X_3X_4 \end{cases} \end{cases} \qquad \gamma_3 = \begin{cases} \gamma_3^1 = \begin{cases} X_1X_2X_3 \\ X_1X_2X_4 \\ X_1X_3X_4 \end{cases} \\ \bar{\gamma}_3^1 = \{X_2X_3X_4 \end{cases}$$

$$\gamma_4 = \begin{cases} \gamma_4^1 = \{X_1X_2X_3X_4 \\ \bar{\gamma}_4^1 = \{0 \end{cases}$$

As we can see, the addends are divided into two groups. By γ_k^i we denote the subset of the addends in γ_k that include the term X_i and by $\bar{\gamma}_k^i$ the addends in γ_k that do not include X_i. Recall that $R_k = \{(i_1, \ldots, i_k) | 1 \leq i_1 < i_2 < \ldots < i_k \leq n\}$. We can now state that:

$$\gamma_k^i = \sum_A \prod_{j \in A} X_j \text{ where } A = \{R \subseteq R_k | i \in R\}$$

$$\bar{\gamma}_k^i = \sum_A \prod_{j \in A} X_j \text{ where } A = \{R \subseteq R_k | i \notin R\}$$

We introduce a recursion for computing γ_k^i and $\bar{\gamma}_k^i$ for every $1 \leq i, k \leq n$ given γ_k in time $O(n^2)$. It is based in the following two relations:

$$\gamma_k^i = \bar{\gamma}_{k-1}^i X_i \quad \forall i \in \{1, \ldots, n\} \tag{4.3}$$

$$\bar{\gamma}_k^i = \gamma_k - \gamma_k^i \quad \forall i \in \{1, \ldots, n\} \tag{4.4}$$

The recursive algorithm for computing γ_k^i and $\bar{\gamma}_k^i$ for every $1 \leq i, k \leq n$ given γ_k is as follows. Let the base cases be $\gamma_0^i = \bar{\gamma}_0^i = 1$ and let γ_k for $1 \leq k \leq n$ be computed as shown in Equation (4.2). Equations (4.3) and (4.4) define a recursive procedure to compute γ_k^i and $\bar{\gamma}_k^i$ for all $1 \leq i, k \leq n$ in $O(n^2)$.

Derivatives

In the case of our probabilistic models, the variables in the ESP are exponential functions of the form $X_i = (exp(\theta_i) - 1)$. Thus, the techniques for computing the derivatives explained in [14] are not valid here. We give here an efficient expression for the first derivatives with respect to θ_i which can be obtained by the chain rule.

$$\frac{\partial \gamma_k}{\partial \theta_i} = \frac{\partial \gamma_k}{\partial X_i} \cdot \frac{\partial X_i}{\partial \theta_i} = \bar{\gamma}_{k-1}^{i} exp(\theta_i) \tag{4.5}$$

4.2 The MM and WMM under the Hamming distance

The MM under the Hamming distance can be expressed as follows:

$$p(\sigma) = \frac{exp(-\theta d(\sigma, \sigma_0))}{\psi(\theta)} \quad \text{where } \psi(\theta) = \sum_{\sigma} exp(-\theta d(\sigma, \sigma_0)) \tag{4.6}$$

Here $\theta \in \mathbb{R}$ is a dispersion parameter, σ_0 is the central permutation, $d(\sigma, \sigma_0)$ represents a distance between σ and σ_0 and $\psi(\theta)$ is the normalization constant.

As we stated in Section 1.2.1, the GMM must be used with a distance with an $n-1$ dimensional decomposition vector. This is not the case of the Hamming distance, whose decomposition vector is n dimensional. However, the idea behind the GMM can be adapted to be used with the Hamming distance with the Weighted Mallows Model (WMM). The WMM under the Hamming distances is defined as follows:

$$p(\sigma) = \frac{exp(-\sum_{j=1}^{n} \theta_j H_j(\sigma \sigma_0^{-1}))}{\psi(\boldsymbol{\theta})} \quad \text{where } \psi(\boldsymbol{\theta}) = \sum_{\sigma} exp(-\sum_{j=1}^{n} \theta_j H_j(\sigma))$$

Note that the MM is the particular case of the WMM in which every dispersion parameter has equal value. The same as the MM, the WMM is unimodal. However, the WMM allows modeling a situation in which there is a high consensus regarding the positions of a subset of the items while having a big uncertainty about the positions of some other items. Specifically, if a permutation σ comes from a distribution with central permutation such that $\sigma_0(i) = j$, the larger θ_j the more likely that $\sigma(i) = j$.

4.2.1 Normalization constants

The naive computation of the normalization constants in the MM and WMM sums over $n!$ permutations. Clearly, this sum is an important bottle-neck when dealing with distributions over permutations of more than $n = 10$ items.

Fortunately, a closed form for the normalization constants for the MM under the Hamming distance has given in [51] which is as follows:

$$\psi(\theta) = n! exp(-\theta n) \sum_{k=0}^{n} \frac{(exp(\theta)-1)^k}{k!} \tag{4.7}$$

The computational complexity of Equation (4.7) is $O(n)$ what highly improves the naive complexity of $O(n!)$.

4.2.1.1 Weighted Mallows Model under the Hamming distance

We introduce in this section an efficient expression for the normalization constant for the WMM. This expression can be found by relating the normalization constant to the moment generating function of the distance decomposition vector.

In particular, the process of finding the efficient expression for the normalization starts by expressing it as a function of the moment generating function (MGF) of $\mathbf{H}(\sigma) = (H_1(\sigma), \ldots, H_n(\sigma))$ under the uniform distribution. Then, we will find an alternative expression for the probability generating function (PGF) by making use of a Taylor expansion. Finally, we will relate both PGF and MGF.

The notation used is as follows. Let $\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1(\sigma), \ldots, \varepsilon_n(\sigma))$ be a binary vector such that $\varepsilon_j(\sigma) = 1 - H_j(\sigma)$. Assuming that σ comes from the uniform distribution, then $\boldsymbol{\varepsilon}(\sigma)$ is a binary random vector. We denote by $P_0(\boldsymbol{\varepsilon}(\sigma) = \boldsymbol{\varepsilon})$ the probability of a permutation σ under the uniform distribution of having the distance decomposition $\mathbf{1} - \boldsymbol{\varepsilon}$. Being the multivariate (joint) MGF of the random vector $\mathbf{X}$ defined as $M_{\mathbf{X}}(\mathbf{t}) = E[\prod_{j=1}^{n} exp(t_j X_j)]$, the normalization constant, $\psi(\boldsymbol{\theta})$, can be posed as a function of the MGF of $\boldsymbol{\varepsilon}(\sigma)$ under the uniform distribution.

$$\begin{aligned}
\psi(\boldsymbol{\theta}) &= n! \sum_{\mathbf{H} \in \{0,1\}^n} P_0(\mathbf{H}(\sigma) = \mathbf{H}) exp(-\sum_j \theta_j H_j(\sigma)) \\
&= n! \sum_{\varepsilon \in \{0,1\}^n} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) exp(-\sum_j \theta_j (1 - \varepsilon_j)) \\
&= n! \sum_{\varepsilon \in \{0,1\}^n} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) exp(-\sum_j \theta_j) exp(\sum_j \theta_j \varepsilon_j) \\
&= n! exp(-\sum_j \theta_j) \sum_{\boldsymbol{\varepsilon} \in \{0,1\}^n} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) exp(\sum_j \theta_j \varepsilon_j) \\
&= n! exp(-\sum_j \theta_j) M_{\boldsymbol{\varepsilon}}(\boldsymbol{\theta})
\end{aligned}$$

Note that there can not be any $\mathbf{H}(\sigma)$ such that $\sum_{j=1}^{n} H_j(\sigma) = 1$. Therefore, the probability of such a vector is zero.

We consider the multivariate case of the PGF of $\boldsymbol{\varepsilon}(\sigma)$ under the uniform distribution, which is defined as follows:

$$f_{\boldsymbol{\varepsilon}}(\mathbf{t}) = f_{\boldsymbol{\varepsilon}}(t_1, \ldots, t_n) = E[t_1^{\varepsilon_1} \cdots t_n^{\varepsilon_n}] = \sum_{(\varepsilon_1, \ldots, \varepsilon_n)} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) t_1^{\varepsilon_1} \cdots t_n^{\varepsilon_n}$$

The Taylor expansion of a multivariate function $f(\mathbf{t})$ around $\mathbf{t} = \mathbf{1}$ is:

$$f(\mathbf{t}) = \sum_{k=0}^{\infty} \frac{1}{k!} \sum_{x_1 + \cdots + x_n = k} \binom{k}{x_1 \cdots x_n} \left. \frac{\partial^k f}{\partial t_1^{x_1} \ldots \partial t_n^{x_n}} \right|_{\mathbf{t}=\mathbf{1}} (t_1 - 1)^{x_1} \cdots (t_n - 1)^{x_n}$$

In order to give the Taylor expansion for $f_{\boldsymbol{\varepsilon}}$ we need to know its derivative for variable t_i:

$$\begin{aligned}
\frac{\partial f_{\boldsymbol{\varepsilon}}}{\partial t_i} &= \sum_{(\varepsilon_1, \ldots, \varepsilon_n)} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) t_1^{\varepsilon_1} \cdots \varepsilon_i t_i^{\varepsilon_i - 1} \cdots t_n^{\varepsilon_n} \\
&= \sum_{(\varepsilon_1, \ldots, \varepsilon_n) | \varepsilon_i = 0} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) t_1^{\varepsilon_1} \cdots 0 t_i^{0-1} \cdots t_n^{\varepsilon_n} \\
&\quad + \sum_{(\varepsilon_1, \ldots, \varepsilon_n) | \varepsilon_i = 1} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) t_1^{\varepsilon_1} \cdots 1 t_i^{1-1} \cdots t_n^{\varepsilon_n} \\
&= 0 + \sum_{(\varepsilon_1, \ldots, \varepsilon_n) | \varepsilon_i = 1} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) \prod_{j \neq i} t_j^{\varepsilon_j}
\end{aligned}$$

Its evaluation around $\mathbf{t} = (1, \ldots, 1)$ is :

$$\begin{aligned}
\left. \frac{\partial f_{\boldsymbol{\varepsilon}}}{\partial t_i} \right|_{\mathbf{t}=\mathbf{1}} &= \sum_{(\varepsilon_1, \ldots, \varepsilon_n) | \varepsilon_i = 1} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) 1^{\varepsilon_1} \cdots 1 \cdot 1^{1-1} \cdots 1^{\varepsilon_n} \\
&= \sum_{(\varepsilon_1, \ldots, \varepsilon_n) | \varepsilon_i = 1} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) 1
\end{aligned}$$

Note that this is equivalent to the probability under the uniform distribution of a permutation with a fixed point at i, i.e. the number of permutations of $n-1$ items divided by $n!$, $(n-1)!/n!$. The second order derivative with respect to t_i equals 0. The second order cross partial derivatives equal the probability under the uniform distribution of a permutation σ in which i_1 and i_2 are unfixed points, i.e. the number of permutations of n item with fixed points in i_1 and i_2 divided by $n!$

$$\left. \frac{\partial^2 f_{\boldsymbol{\varepsilon}}}{\partial t_{i_1} \partial t_{i_2}} \right|_{\mathbf{t}=\mathbf{1}} = \sum_{(\varepsilon_1, \ldots, \varepsilon_n) | \varepsilon_{i_1} = 1, \varepsilon_{i_2} = 1} P_0(\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1, \ldots, \varepsilon_n)) 1^{\varepsilon_1} \cdots 1 \cdot 1^{1-1} \cdots 1^{\varepsilon_n}$$

Then, in general, the k-th order cross partial derivatives equal:

$$\left. \frac{\partial^k f_{\boldsymbol{\varepsilon}}}{\partial t_{i_1} \ldots \partial t_{i_k}} \right|_{\mathbf{t}=\mathbf{1}} = \frac{(n-k)!}{n!}$$

Since $(\varepsilon_1, \ldots, \varepsilon_n) \in \{0,1\}^n$ then $\binom{k}{\varepsilon_1 \cdots \varepsilon_n} = k!$ and since $\partial^k f_{\boldsymbol{\varepsilon}} / \partial t_i^k = 0$ for $k > 1$ the Taylor series cannot be expanded more than $n+1$ terms. Therefore, the Taylor expansion around $\mathbf{1}$ can be equivalently written as (recall that $R_k = \{(i_1, \ldots, i_k) | 1 \leq i_1 < i_2 < \ldots < i_k \leq n\}$):

$$\begin{aligned} f_{\boldsymbol{\varepsilon}}(\mathbf{t}) &= \sum_{k=0}^{n} \frac{1}{k!} \sum_{R_k} \binom{k}{\varepsilon_{i_1} \cdots \varepsilon_{i_n}} \frac{\partial^k f_{\boldsymbol{\varepsilon}}}{\partial t_{i_1} \ldots \partial t_{i_k}} \bigg|_{\mathbf{t}=\mathbf{1}} \prod_{s=1}^{k} (t_{i_s} - 1) \\ &= \sum_{k=0}^{n} \frac{1}{k!} \sum_{R_k} \frac{k!(n-k)!}{n!} \prod_{s=1}^{k} (t_{i_s} - 1) \\ &= \sum_{k=0}^{n} \frac{(n-k)!}{n!} \sum_{R_k} \prod_{s=1}^{k} (t_{i_s} - 1) \end{aligned}$$

If γ_k denotes the Elementary Symmetric Polynomial (ESP) of degree k, then $\gamma_k((t_1 - 1), \ldots, (t_n - 1)) = \sum_{r \in R_k} \prod_{i \in r} (t_i - 1)$ and therefore:

$$f_{\boldsymbol{\varepsilon}}(\mathbf{t}) = \sum_{k=0}^{n} \frac{(n-k)!}{n!} \gamma_k((t_1 - 1), \ldots, (t_n - 1))$$

An efficient formulation for the computation of ESP, $\gamma_k((t_1 - 1), \ldots, (t_n - 1))$ is given in Section 4.1.3. Note that $M_{\boldsymbol{\varepsilon}}(\boldsymbol{\theta}) = E[\prod_{j=1}^{n} exp(\theta_j \varepsilon_j)] = E[\prod_{j=1}^{n} exp(\theta_j)^{\varepsilon_j}]$ and $f_{\boldsymbol{\varepsilon}}(\mathbf{t}) = E[\prod_{j=1}^{n} {t_j}^{\varepsilon_j}]$, so $f_{\boldsymbol{\varepsilon}}(exp(\boldsymbol{\theta})) = M_{\boldsymbol{\varepsilon}}(\boldsymbol{\theta})$. Thus, the probability generating function can be given as follows.

$$M_{\boldsymbol{\varepsilon}}(\boldsymbol{\theta}) = \sum_{k=0}^{n} \frac{(n-k)!}{n!} \gamma_k((exp(\theta_1) - 1), \ldots, (exp(\theta_n) - 1))$$

Finally, the normalization constant can thus be given as follows:

$$\begin{aligned} \psi(\boldsymbol{\theta}) &= n! exp(-\sum_j \theta_j) M_{\boldsymbol{\varepsilon}}(\boldsymbol{\theta}) \\ &= exp(-\sum_j \theta_j) \sum_{k=0}^{n} (n-k)! \gamma_k((exp(\theta_1) - 1), \ldots, (exp(\theta_n) - 1)) \end{aligned} \quad (4.8)$$

The computational complexity of Equation (4.8) is $O(n^2)$, the same as the complexity of the computation of the ESP.

4.2.2 Expected value, marginal and conditional probabilities

In this section we deal with the expressions for the expected value of the distance in MM and the expected value of the distance decomposition vector in WMM. We also consider the marginal and conditional probabilities for the

WMM under the Hamming distance based on Equation (4.8). Since the MM is a particular case of the WMM in which every θ_j has equal value, the results of the WMM model can be applied for both.

The following theorem expresses the expected value of the distance under the MM as well as the expected value of the distance decomposition vector under the WMM.

Theorem 1 *The expected value of the distance under the MM with the Hamming distance and dispersion parameter θ is as follows:*

$$E_\theta[d_h] = \frac{n\sum_{k=0}^{n}\frac{(exp(\theta)-1)^k}{k!} - exp(\theta)\sum_{k=0}^{n-1}\frac{(exp(\theta)-1)^k}{k!}}{\sum_{k=0}^{n}\frac{(exp(\theta)-1)^k}{k!}}$$

Proof. First of all, note that the expected distance under an exponential model such as those considered in this dissertation can be derived from the MGF in the following way [51].

$$E_\theta[d_h] = \left.\frac{\partial Ln\, M(t)}{\partial t}\right|_{t=-\theta}$$

where $M(t)$ stands for the moment generating function of the distance under the uniform distribution. In Section 4.2.1 we have shown how to efficiently compute the normalization constants of the MM and WMM. The expressions are based on the MGF. Although we only detail the MGF for the WMM (Equation (4.8)), MGF for the MM can be obtained by forcing every θ_j to have the same value. It is expressed as follows.

$$M_D(t) = exp(tn)\sum_{k=0}^{n}\frac{(exp(-t)-1)^k}{k!}$$

The expression of the expected value relies on the derivative of the MGF of D with respect to t, which is as follows.

$$\frac{\partial M_D(t)}{\partial t} = exp(tn)\left(n\sum_{k=0}^{n}\frac{(exp(-t)-1)^k}{k!} - \sum_{k=1}^{n}\frac{k(exp(-t)-1)^{k-1}exp(-t)}{k!}\right)$$

Finally, the expectation of variable D is expresses as follows.

$$E_\theta[d_h] = \frac{n\sum_{k=0}^{n}\frac{(exp(\theta)-1)^k}{k!} - exp(\theta)\sum_{k=0}^{n-1}\frac{(exp(\theta)-1)^k}{k!}}{\sum_{k=0}^{n}\frac{(exp(\theta)-1)^k}{k!}}$$

Theorem 2 *The expected value of the distance decomposition vector $\mathbf{H}(\sigma)$ under the WMM with the Hamming distance and dispersion parameter $\boldsymbol{\theta}$ is as follows:*

$$E_{\boldsymbol{\theta}}[\mathbf{H}] = \left(1 - \frac{\sum_{k=1}^{n}(n-k)!exp(\theta_1)\bar{\gamma}_{k-1}^{1}}{\sum_{k=0}^{n}(n-k)!\gamma_k}, \ldots, 1 - \frac{\sum_{k=1}^{n}(n-k)!exp(\theta_n)\bar{\gamma}_{k-1}^{n}}{\sum_{k=0}^{n}(n-k)!\gamma_k}\right)$$

where $\gamma_k = \gamma_k((exp(\theta_1)-1),\ldots,(exp(\theta_n)-1))$ *and* $\bar{\gamma}_k^i = \bar{\gamma}_k^i((exp(\theta_1)-1),\ldots,(exp(\theta_n)-1))$ *denotes the ESP of degree* k *of the previous set of variables except for* $\{(exp(\theta_i)-1)$.

Proof. We now show how to obtain a computationally cheap expression for the expected value of the distance decomposition vector $\mathbf{H}$ in the WMM under the Hamming distance. Recall that $\boldsymbol{\varepsilon}$ is a binary vector related with $\mathbf{H}$ by $\varepsilon_j = 1 - H_j$ for every j. The expected value of vector $\mathbf{H}$, $E_\theta[\mathbf{H}]$ is given by a vector $\mathbf{E} = (E_1,\ldots,E_n)$ in which E_i is the expected value of H_i. The next line shows how to obtain $\mathbf{E}$ from $M_{\boldsymbol{\varepsilon}}(\mathbf{t})$. Let $\mathbf{b}$ and $\mathbf{1}$ be two binary vectors of length n where $\mathbf{1} = (1,\ldots,1)$ and note that the additions and subtractions are vector sums and subtractions.

$$\begin{aligned} E_\theta[\mathbf{H}] &= \sum_{\boldsymbol{b}} P(\mathbf{H}=\boldsymbol{b})\boldsymbol{b} = E_{\boldsymbol{\theta}}[\mathbf{1}-\boldsymbol{\varepsilon}] = \mathbf{1} - E_{\boldsymbol{\theta}}[\boldsymbol{\varepsilon}] \\ &= \mathbf{1} - \left(\frac{\partial Ln\, M_{\boldsymbol{\varepsilon}}(\mathbf{t})}{\partial t_1}\bigg|_{\mathbf{t}=\boldsymbol{\theta}}, \ldots, \frac{\partial Ln\, M_{\boldsymbol{\varepsilon}}(\mathbf{t})}{\partial t_n}\bigg|_{\mathbf{t}=\boldsymbol{\theta}} \right) \end{aligned}$$

The MGF of vector $\boldsymbol{\varepsilon}$ has been obtained in the computation of the normalization constant in Equation (4.8). It is expressed as follows.

$$M_{\boldsymbol{\varepsilon}}(\mathbf{t}) = \sum_{\boldsymbol{\varepsilon}} P(\mathbf{H}=\boldsymbol{\varepsilon}) \prod_{j=1}^{n} exp(t_i\,\varepsilon_i) = \sum_{k=0}^{n} \frac{(n-k)!}{n!}\gamma_k$$

where $\gamma_k = \gamma_k(exp(t_1)-1,\ldots,exp(t_n)-1)$.

The partial derivative of $M_{\boldsymbol{\varepsilon}}$ is as follows.

$$\frac{\partial M_{\boldsymbol{\varepsilon}}(\mathbf{t})}{\partial t_i} = \sum_{k=1}^{n} \frac{(n-k)!}{n!} exp(t_i)\bar{\gamma}_{k-1}^i$$

where $\bar{\gamma}_k = \gamma_k(exp(t_1)-1,\ldots,exp(t_n)-1)$ denotes the ESP of the variables $(exp(t_1)-1,\ldots,exp(t_n)-1)$ with the exception of $exp(t_i)-1$.

Finally, the expected value is given by the following expression.

$$E_{\boldsymbol{\theta}}[\mathbf{H}] = \mathbf{1} - \left(\frac{\sum_{k=1}^{n}(n-k)!exp(\theta_1)\bar{\gamma}_{k-1}^1}{\sum_{k=0}^{n}(n-k)!\gamma_k}, \ldots, \frac{\sum_{k=1}^{n}(n-k)!exp(\theta_n)\bar{\gamma}_{k-1}^n}{\sum_{k=0}^{n}(n-k)!\gamma_k} \right)$$

Due to the factorial size of permutation spaces, the naive computation of the marginal distribution of a WMM under the Hamming distance is infeasible. We introduce a method for computing such marginal distributions by adapting the reasoning used in Equation (4.8) to sum over the subset of permutations of interest.

We consider two disjoint sets of items, A and B, and two sets of permutations, $fix(A)$ and $unfix(B)$. The set $fix(A)$ includes every permutation in

which j is a fixed point for every $j \in A$. We define the set $unfix(B)$ in the same way, as the set of permutations that have an unfixed point at position $j \in B$. Recall that by $g(n,k)$ we denote the number of permutations of n items in which there are at least k unfixed points and a recursion to compute it is given in Equation (4.1).

Theorem 3 *Let $a = |A|$ and $b = |B|$. The marginal distribution of the set of permutations in which every $i \in A$ is a fixed point and every $j \in B$ is an unfixed point -the permutations in the intersection $fix(A) \cap unfix(B)$- is as follows:*

$$\sum_{\sigma \in fix(A) \cap unfix(B)} p(\sigma) = \frac{\sum_{\sigma \in fix(A) \cap unfix(B)} exp(\sum_{j=1}^{n} -\theta_j H_j(\sigma))}{\psi(\boldsymbol{\theta})}$$
$$= \frac{exp(-\sum_{j \notin A} \theta_j) \sum_{k=0}^{n-a-b} g(n-a-k,b) \bar{\gamma}_k^{AB}(T_1, \ldots, T_n)}{\sum_{k=0}^{n} g(n,k) \gamma_k(T_1, \ldots, T_n)} \tag{4.9}$$

where $T_i = (exp(\theta_i) - 1)$. Note that by $\bar{\gamma}_k^{AB}((exp(\theta_1) - 1), \ldots, (exp(\theta_n) - 1))$ we denote the ESP of degree k of the set of variables $\{(exp(\theta_j) - 1) | j \notin A \cup B\}$.

Proof. Recall that $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$ denotes a binary vector and $\boldsymbol{\varepsilon}(\sigma) = (\varepsilon_1(\sigma), \ldots, \varepsilon_n(\sigma))$ a binary random vector such that $\varepsilon_j(\sigma) = 1 - H_j(\sigma)$. Let $a = |A|$, $b = |B|$. The cardinalities of the sets of permutations are $|fix(A)| = (n-a)!$ and $|fix(B)| = g(n,b))$. Therefore, $|fix(A) \cap fix(B)| = g(n-a,b))$. Let α and β range over the sets A and B respectively.

$$\sum_{\sigma\in fix(A)\cap unfix(B)} exp(\sum_{j=1}^{n} -\theta_j H_j(\sigma))$$
$$= g(n-a,b) \sum_{\mathbf{H}\in\{0,1\}^n} P_0(\boldsymbol{H}(\sigma)=\mathbf{H}) exp(-\sum_j \theta_j H_j)$$
$$= g(n-a,b) \sum_{\boldsymbol{\varepsilon}|\varepsilon_\alpha=1\wedge\varepsilon_\beta=0} P_0(\boldsymbol{\varepsilon}(\sigma)=\boldsymbol{\varepsilon}) exp(-\sum_j \theta_j(1-\varepsilon_j))$$
$$= g(n-a,b) \sum_{\substack{(\varepsilon_1,\ldots,\varepsilon_n)|\\ \varepsilon_\alpha=1\wedge\varepsilon_\beta=0}} P_0(\boldsymbol{\varepsilon}(\sigma)=\boldsymbol{\varepsilon}) exp(-\sum_j \theta_j) exp(\sum_j \theta_j\varepsilon_j)$$
$$= g(n-a,b) exp(\sum_j -\theta_j) \sum_{\substack{(\varepsilon_1,\ldots,\varepsilon_n)|\\ \varepsilon_\alpha=1\wedge\varepsilon_\beta=0}} P_0(\boldsymbol{\varepsilon}(\sigma)=\boldsymbol{\varepsilon}) exp(\sum_j \theta_j\varepsilon_j)$$
$$= g(n\text{-}a,b) exp(\sum_j \text{-}\theta_j) \sum_{\substack{(\varepsilon_1,\ldots,\varepsilon_n)|\\ \varepsilon_\alpha=1\wedge\varepsilon_\beta=0}} P_0(\boldsymbol{\varepsilon}(\sigma)=\boldsymbol{\varepsilon}) exp(\sum_{j\in A}\theta_j 1 + \sum_{j\in B}\theta_j 0 + \sum_{j\notin AB}\theta_j\varepsilon_j)$$
$$= g(n-a,b) exp(\sum_j -\theta_j) exp(\sum_{j\in A}\theta_j) \sum_{\substack{(\varepsilon_1,\ldots,\varepsilon_n)|\\ \varepsilon_\alpha=1\wedge\varepsilon_\beta=0}} P_0(\boldsymbol{\varepsilon}(\sigma)=\boldsymbol{\varepsilon}) exp(\sum_{j\notin AB}\theta_j\varepsilon_j)$$
$$= g(n-a,b) exp(\sum_{j\notin A} -\theta_j) \sum_{\substack{(\varepsilon_1,\ldots,\varepsilon_n)|\\ \varepsilon_\alpha=1\wedge\varepsilon_\beta=0}} P_0(\boldsymbol{\varepsilon}(\sigma)=\boldsymbol{\varepsilon}) exp(\sum_{j\notin A\cup B}\theta_j\varepsilon_j)$$
$$= g(n-a,b) exp(\sum_{j\notin A} -\theta_j) M(\boldsymbol{\theta})$$

The multivariate case of the conditional probability generating function is defined as:

$$f(\mathbf{t}) = \sum_{\substack{(\varepsilon_1,\ldots,\varepsilon_n)|\\ \varepsilon_\alpha=1\wedge\varepsilon_\beta=0}} P(\boldsymbol{\varepsilon}(\sigma)=(\varepsilon_1,\ldots,\varepsilon_n)) \prod_{j\notin A\cup B} t_j^{\varepsilon_j}$$

In the same way as in Section 4.2.1.1, the Taylor expansion of a multivariate function is used. Therefore, we proceed by expressing the derivative of $f(\mathbf{t})$ for variable t_i for $i \notin \{A\cup B\}$.

$$\frac{\partial f}{\partial t_i} = \sum_{\substack{(\varepsilon_1,\ldots,\varepsilon_n)|\\ \varepsilon_\alpha=1\wedge\varepsilon_\beta=0\wedge\varepsilon_i=1}} P(\boldsymbol{\varepsilon}(\sigma)=(\varepsilon_1,\ldots,\varepsilon_n)) \prod_{j\notin\{A\cup B\}} t_j^{\varepsilon_j}$$

$$\left.\frac{\partial f}{\partial t_i}\right|_{\mathbf{t}=\mathbf{1}} = \sum_{\substack{(\varepsilon_1,\ldots,\varepsilon_n)|\\ \varepsilon_\alpha=1\wedge\varepsilon_\beta=0\wedge\varepsilon_i=1}} P(\boldsymbol{\varepsilon}=(\varepsilon_1,\ldots,\varepsilon_n)) 1^{\varepsilon_1}\cdots 1\cdot 1^{1-1}\cdots 1^{\varepsilon_n}$$

In general, the k-th order cross partial derivatives are equivalent to the probability under the uniform distribution of a permutation with a fixed point

at i among those having fixed points at positions $\alpha \in A$ and unfixed points at positions $\beta \in B$. Then, in general, the k-th order cross partial derivatives equal:

$$\left.\frac{\partial^k f}{\partial t_{i_1} \dots t_{i_k}}\right|_{\mathbf{t}=\mathbf{1}} = \frac{g(n-a-k,b)}{g(n-a,b)}$$

Since $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_n) \in \{0,1\}^n$ then $\binom{k}{\varepsilon_1 \cdots \varepsilon_n} = k!$. Since $\partial^k f / \partial t_i^k = 0$ for $k > 1$ the Taylor series cannot be expanded more than $n-a-b+1$ terms. Recall that $R_k = \{(i_1, \dots, i_k) | 1 \leq i_1 < i_2 < \dots < i_k \leq n\}$. Therefore, the Taylor expansion around $\mathbf{a} = \mathbf{1} = (1, \dots, 1)$ can be equivalently written as:

$$\begin{aligned} f(\mathbf{t}) &= \sum_{k=0}^{n-a-b} \frac{1}{k!} \sum_{R_k} \binom{k}{\varepsilon_1 \cdots \varepsilon_n} \left.\frac{\partial^k f}{\partial t_{i_1} \dots t_{i_k}}\right|_{\mathbf{t}=\mathbf{1}} \prod_{s=1}^{k} (t_{i_s} - 1) \\ &= \sum_{k=0}^{n-a-b} \frac{1}{k!} \sum_{R_k} \frac{k! g(n-a-k,b)}{g(n-a,b)} \prod_{s=1}^{k} (t_{i_s} - 1) \\ &= \sum_{k=0}^{n-a-b} \frac{k! g(n-a-k,b)}{g(n-a,b)} \sum_{R_k} \prod_{s=1}^{k} (t_{i_s} - 1) \end{aligned}$$

Let $\bar{\gamma}_k^{AB}$ denotes the Elementary Symmetric Polynomial of degree k of the set of items not included in $A \cup B$.

$$f(\mathbf{t}) = \sum_{k=0}^{n-a-b} \frac{g(n-a-k,b)}{g(n-a,b)} \bar{\gamma}_k^{AB}((t_1 - 1), \dots, (t_1 - 1))$$

Since $f(exp(\boldsymbol{\theta})) = M(\boldsymbol{\theta})$, the probability generating function can be given as follows.

$$M(\boldsymbol{\theta}) = \sum_{k=0}^{n-a-b} \frac{g(n-a-k,b)}{g(n-a,b)} \bar{\gamma}_k^{AB}((exp(\theta_1) - 1), \dots, (exp(\theta_n) - 1))$$

The sum of $exp(\sum_{j=1}^{n} -\theta_j H_j(\sigma))$ for the subset of permutations of interest, i.e., $fix(A) \cap unfix(B)$, is as follows:

$$\begin{aligned} &\sum_{\sigma \in fix(A) \cap unfix(B)} exp(\sum_{j=1}^{n} -\theta_j H_j(\sigma)) = g(n-a,b) exp(\sum_{j \notin A} -\theta_j) M(\boldsymbol{\theta}) \\ &= exp(\sum_{j \notin A} -\theta_j) \sum_{k=0}^{n-a-b} g(n-a-k,b) \bar{\gamma}_k^{AB}((exp(\theta_1) - 1), \dots, (exp(\theta_n) - 1)) \end{aligned}$$

Regarding the computational complexity of this calculation two different aspects must be considered. On the one hand, the complexity of computing $g(n-a, k+b)$ is $O(n-a)$. On the other hand, the complexity of computing

$\bar{\gamma}_k^{AB}$ is $O((n-a-b)^2)$. Therefore, the complexity of the calculation of the marginal $\sum_{\sigma \in fix(A) \cap unfix(B)} p(\sigma)$ is $O(max\{(n-a),(n-a-b)^2\})$.

It is well known that the conditional distribution can posed in terms of the joint distribution as follows:

$$p(X|Y) = \frac{p(X \cap Y)}{p(Y)}$$

This definition can be easily translated to the WMM under the Hamming distance as stated in the following Lemma.

Corollary 2 *Let A, A', B and B' be four disjoint sets of items. The probability of the permutations having fixed points at positions $j \in A'$ and unfixed points at positions $j \in B'$ given that the items $j \in A$ are fixed points and the items $j \in B$ are unfixed points is as follows.*

$$p(fix(A') \cap unfix(B') | fix(A) \cap unfix(B)) = \frac{\sum_{\sigma \in fix(A \cup A') \cap unfix(B \cup B')} p(\sigma)}{\sum_{\sigma \in fix(A) \cap unfix(B)} p(\sigma)} \tag{4.10}$$

The computational complexity in this case is thus the same as the complexity of the marginal computation.

4.3 Sampling

In this section we show how to generate permutations from both the MM and WMM with similar approaches to those in the previous chapters.

The three algorithms generate samples assuming that the mode is the identity. In case $\sigma_0 \neq e$ one can move a sample centered around e to be centered around σ_0 by composing each permutation with σ_0.

4.3.1 Gibbs sampling algorithm

We have adapted the Gibbs sampler to generate samples from an approximate distribution of the MM and WMM. The Gibbs algorithm proceeds as follows:

1. Generate a permutation σ u.a.r.
2. Build a new permutation σ' equal to σ in all but two positions chosen u.a.r. These two positions are swapped.
3. Let $\gamma = min\{1, p(\sigma')/p(\sigma)\}$. With probability γ the algorithm accepts the candidate permutation moving the chain to it, $\sigma = \sigma'$, and goes back to 2. Otherwise, it discards σ' and goes back to step 2.

The initial samples are discarded (burn-in period) until the Markov chain approaches its stationary distribution and so samples from the chain are samples from the distribution of interest. Then, the above process in repeated until the algorithm generates a given number of permutations.

The third step of the Gibbs algorithm involves the computation of $\gamma = min\{1, p(\sigma')/p(\sigma)\}$ for every permutation generated. It is possible to compute γ without computing $p(\sigma')$ and $p(\sigma)$ thus reducing the computational time needed for the third step. The details differ for the MM and WMM.

For the MM case let d be $d = d(\sigma) - d(\sigma')$. Taking into account that σ' was generated by swapping two items in σ it follows that $-2 \leq d \leq 2$. If $d \geq 0$ the new permutation is accepted and the chain moves to σ'. Otherwise, the probability of accepting the new permutation σ' is as follows:

$$exp(-\theta d)$$

For the WMM case, let σ' be generated from σ by swapping positions i and j. If $H_j(\sigma') + H_i(\sigma') \leq H_j(\sigma) + H_i(\sigma)$, then the algorithm accepts the new permutation and the chain moves to σ'. Otherwise, the ratio of accepting σ' is follows:

$$exp(-H_j(\sigma') \cdot \theta_j - H_i(\sigma') \cdot \theta_i + H_j(\sigma) \cdot \theta_j + H_i(\sigma) \cdot \theta_i)$$

The computation of each permutation has complexity $O(n)$. It is thus a quick algorithm. However, we should remark that this is an approximate algorithm.

4.3.2 Chain sampling algorithm

We propose a method for generating permutations from a MM or a WMM based on the chain rule. The process of generating each permutation can be divided in two stages. In the first stage, it randomly generates a distance decomposition vector, $\boldsymbol{H}(\sigma)$. The sampling finishes by uniformly at random generating a permutation σ consistent with the given $\boldsymbol{H}(\sigma)$.

The generation of the random distance decomposition vector $\boldsymbol{H}(\sigma)$ is carried out with the well known chain rule for probability distributions. The chain rule expresses the joint distribution as a product of conditional distributions.

In this way, the Chain sampling algorithm uses the conditional probabilities of the WMM given in Equation (4.10) to sample each of the terms in the decomposition vector $\boldsymbol{H}(\sigma)$.

The complete process for the generation of $\boldsymbol{H}(\sigma)$ can be found in Algorithm 8. The process of randomly generating a permutation finishes by generating uniformly at random a permutation σ consistent with $\boldsymbol{H}(\sigma)$. We refer the reader to Section 1.1.3 where the random generation of a permutation given the subsets of items that are fixed or unfixed is given. A careful implementation of the Chain algorithm leads to a computational complexity of $O(n^2)$, see Appendix 4.3.2.

Algorithm 8: Random generation of $\boldsymbol{H}(\sigma)$ from the MM or WMM

Input: θ (resp. $\boldsymbol{\theta}$) dispersion parameters in MM (resp. WMM)
Output: $\boldsymbol{H}(\sigma)$ random distance decomposition vector
$A = B = \emptyset$;
for $j \leftarrow 1$ **to** n **do**
  $prob = p(fix(j)|fix(A) \cap unfix(B))$ as shown in Corollary 2;
  with probability $prob$ `/* j is fixed */`
    $h_j(\sigma) = 0$;
    $A = A \cup \{j\}$;
  end
  otherwise `/* j is unfixed */`
    $h_j(\sigma) = 1$;
    $A = B \cup \{j\}$;
  end
end

Optimizing the Chain sampling algorithm

Each iteration j of the Chain sampling algorithm is divided in two steps, namely:

- Compute the conditional probability of j being a fixed point conditioned to the previous sets of fixed and unfixed points, A and B. In other words, sum the probability of the permutations:

$$\sum_{\sigma \in fix(\{i\}))} p(\sigma|fix(A) \cap unfix(B))$$

- Insert item j either in set A or in set B.

By Corollary 2 and Theorem 3, the sum of the probability on the first step can be posed as

$$\frac{exp(-\sum_{j \notin A \cup \{i\}} \theta_j) \sum_{k=0}^{n} g(n-a-1-k, b) \bar{\gamma}_k^{AB \cup \{i\}}(T_1, \ldots, T_n)}{exp(-\sum_{j \notin A} \theta_j) \sum_{k=0}^{n} g(n-a-k, b) \bar{\gamma}_k^{AB}(T_1, \ldots, T_n)}$$

where $T_i = (exp(\theta_1) - 1)$. The code optimization explodes the following two considerations. First, the number of permutations of n items with at least k unfixed points, denoted $g(i, j)$, can be precomputed in $O(n^2)$ for all $1 \leq i, j \leq n$. Second, the set $A \cup B$ at iteration j equals $\{j+1, j+2, \ldots, n\}$. It follows that $\bar{\gamma}_k^{AB \cup \{i\}}$ can be obtained by splitting $\bar{\gamma}_k^{AB}$ in time $O(n-a-b-1)$ as shown in Section 4.1.3, Equations (4.3) and (4.4).

4.3.3 Distances sampling algorithm

The Distances sampler can be used for generating from the MM under the Hamming distance but not for generating from the WMM.

The probability under the MM of a permutation at distance d is as follows:

$$p(d) = \sum_{\sigma | d_h(\sigma, \sigma_0^{-1}) = d} p(\sigma) = S_h(n,d) \frac{exp(-\theta d)}{\psi(\theta)} \tag{4.11}$$

where $S_h(n,d)$ denotes the number of permutations at distance d. Note that the normalization constant $\psi(\theta) = \sum_\sigma exp(-\theta d(\sigma))$ can be expressed as the sum of n terms in the following way:

$$\psi(\theta) = \sum_{d=0}^{n} S_h(n,d) exp(-\theta d) \tag{4.12}$$

Therefore, the Distances sampling process divides the process of generating a permutation from a given MM in two different stages as follows:

1. Randomly select the distance d considering the probabilities of Equation (4.11).
$$p(d) = \psi(\theta)^{-1} S_h(n,d) exp(-\theta d) \tag{4.13}$$
2. Uniformly at random generate one permutation among those at distance d from the identity. Recall that the random generation of permutations at a prescribed Hamming distance is explained in Section 1.1.3.

Therefore, the time complexity of the generation of each permutation using this method is $O(n)$ given the sequence $S_h(n,d)$. Summarizing, this is a fast algorithm for the generation of exact samples from the MM model. Unfortunately, this algorithm can not generate samples from the WMM model.

4.4 Learning

The parameters of a probability distribution are fitted in this dissertation via maximum likelihood estimation. In [34] it is stated that the maximum likelihood estimate of the parameters of a L-decomposable distribution, such as the MM, can be done by iterative scaling. Unfortunately, this only includes the dispersion parameters.

For a sample of m i.i.d. permutations $\{\sigma_1, \ldots, \sigma_m\}$ the MLE for the parameters of the distribution are those which maximize the likelihood function. Even though the MM is a particular case of the WMM, the MLE for the parameters of the distribution are different for each model. Therefore, we focus on each model separately.

4.4.1 Mallows Model

The log-likelihood expression for the MM model is as follows:

$$Ln\ \mathcal{L}(\{\sigma_1,\sigma_2,\ldots,\sigma_m\}|\sigma_0,\theta) = \sum_{i=1}^{m} Ln\ p(\sigma_i|\sigma_0,\theta) = \sum_{i=1}^{m} Ln\ \frac{exp(-\theta d(\sigma_i,\sigma_0))}{\psi(\theta)}$$
$$= \sum_{i=1}^{m}(-\theta d(\sigma_i\sigma_0^{-1}) - Ln\,\psi(\theta)) = -m\theta\bar{d} - mLn\,\psi(\theta) \tag{4.14}$$

where $\bar{d} = \sum_{i=1}^{m} d(\sigma_i\sigma_0^{-1})/m$. By looking at Equation (4.14), we can see that calculating the value of σ_0 that maximizes the equation is independent of θ. Therefore the MLE estimation problem can be posed as a two step process in which first the central permutation is obtained and then the dispersion parameter for the given $\hat{\sigma}_0$ is computed.

Central permutation

The MLE for the central permutation is given by the permutation that minimizes the sum of the distances to the sample. Let us pose the problem in a different way. Let F be the frequency matrix of the sample. The problem of finding the permutation that minimizes the distance to the sample is equivalent to the problem of selecting one entry of F per row and column in such a way that their sum is maximum. Actually, this is equivalent to the linear assignment problem (LAP) when the objective is to maximize the sum of the assignment. We denote this MLE for the central permutation as σ_{LAP}. The Hungarian algorithm solves this problem in $O(n^3)$.

Dispersion parameter

Once the central permutation is known, the MLE for the dispersion parameter is obtained by deriving Equation (4.14) respect θ and equaling to zero.

$$\frac{d\left(-m\theta\bar{d} - mLn\,\psi(\theta)\right)}{d\theta} = 0 \quad \Rightarrow \quad -m\bar{d} - m\frac{d\psi(\theta)/d\theta}{\psi(\theta)} = 0$$

The MLE for θ is then given by the θ that satisfies the next equation:

$$\frac{exp(\theta)\sum_{k=0}^{n-1}\frac{(exp(\theta)-1)^k}{k!} - n\sum_{k=0}^{n}\frac{(exp(\theta)-1)^k}{k!}}{\sum_{k=0}^{n}\frac{(exp(\theta)-1)^k}{k!}} + \bar{d} = 0 \tag{4.15}$$

Although no closed form for θ in Equation (4.15) exists, root finding algorithms such as Newton-Raphson can efficiently recover θ.

As a summary, the estimation of the MLE for the parameters of a MM model can be done in polynomial time.

4.4.2 Weighted Mallows Model

In this section we describe the maximum likelihood estimation process of a given sample coming from a WMM model. The log-likelihood can be expressed as follows:

$$Ln\ \mathcal{L}(\{\sigma_1,\sigma_2,\ldots,\sigma_m\}|\sigma_0,\boldsymbol{\theta}) = \sum_{i=1}^{m} Ln\ p(\sigma_i|\sigma_0,\boldsymbol{\theta}) \tag{4.16}$$

$$\begin{aligned}
&= \sum_{i=1}^{m} Ln\ \frac{exp\left(\sum_{j=1}^{n} -\theta_j H_j(\sigma_i\sigma_0^{-1})\right)}{\psi(\boldsymbol{\theta})} \\
&= \sum_{i=1}^{m}\left(\sum_{j=1}^{n} -\theta_j H_j(\sigma_i\sigma_0^{-1}) - Ln\,\psi(\boldsymbol{\theta})\right) = m\sum_{j=1}^{n} -\theta_j\bar{H}_j - mLn\,\psi(\boldsymbol{\theta}) \\
&= -\,m\left(\sum_{j=1}^{n}\theta_j\bar{H}_j + Ln\sum_{k=0}^{n}(n-k)!\gamma_k((exp(\theta_1)-1),\ldots,(exp(\theta_n)-1))\right)
\end{aligned} \tag{4.17}$$

where $\bar{H}_j = \sum_{i=1}^{m} H_j(\sigma_i\sigma_0^{-1})/m$. Note that the permutation that maximizes the likelihood does need to be the same as the permutation that minimizes the sum of the distances to the permutations in the sample, i.e., the central permutation for the MM may not be the same for the MM and WMM.

In contrast to the MM estimation process, the learning can not be divided into different stages for the estimation of the different parameters. However, the same as in the MM case, the expression for the dispersion parameters can be posed as a function of the central permutation. In particular, the dispersion parameters $\boldsymbol{\theta}$ for a given σ_0 are given by equaling to zero the derivative of the likelihood respect to θ_i:

$$\frac{\partial\left(-m\sum_{j=1}^{n}\theta_j\bar{H}_j - mLn\,\psi(\boldsymbol{\theta})\right)}{\partial\theta_i} = 0 \quad \rightarrow \quad -m\bar{H}_j - m\frac{\partial\psi(\boldsymbol{\theta})/\partial\theta_i}{\psi(\boldsymbol{\theta})} = 0$$

In other words, given a permutation σ_0, the MLE for $\boldsymbol{\theta}$ is that which satisfies the following system of equations:

$$\frac{\sum_{k=1}^{n}(n-k)!exp(\theta_i)\bar{\gamma}_{k-1}^{i}}{\sum_{k=0}^{n}(n-k)!\gamma_k} + \bar{H}_i = 0 \qquad 1 \le i \le n \tag{4.18}$$

By abusing notation, γ_k denotes the ESP $\gamma_k((exp(\theta_1)-1),\ldots,(exp(\theta_n)-1))$ and $\bar{\gamma}_{k-1}^{i}$ denotes the ESP of all variables except for $(exp(\theta_i)-1)$, which is the ESP of degree $k-1$ of the set of variables $\{(exp(\theta_1)-1),\ldots,(exp(\theta_n)-1)\} \setminus \{(exp(\theta_i)-1)\}$. The efficient computation of γ_k and a fast algorithm for obtaining $\bar{\gamma}_k^i$ given γ_k^i can be found in Section 4.1.3. Note that there is no closed-form expression for the dispersion parameters in Equation (4.18),

the system of equations must be solved with numerical methods such as the multidimensional Newton-Raphson.

Summarizing, by expressing $\boldsymbol{\theta}$ as a function of σ_0 with Equation (4.18), we have posed the learning of a WMM as a combinatorial problem of finding the permutation σ_0 that maximizes Equation (4.17).

Although the computational complexity of estimating the parameters of a WMM under the Hamming distance is not known, the authors conjecture that it is NP-hard. We propose an approximate algorithm for fitting a WMM which is based on its asymptotic properties.

4.4.3 Approximate MLE

We propose an approximate algorithm for obtaining the MLE for the parameters of a sample of permutations generated from a WMM under the Hamming distance, $\{\sigma_1, \ldots, \sigma_m\}$. This algorithm is based on the asymptotic properties of the model. The problem statement is as follows. Let F be the frequency matrix on a sample $\{\sigma_1, \ldots, \sigma_m\}$ generated from a WMM and let σ_{LAP} be the permutation that maximizes the objective function in the LAP in F, i.e., σ_{LAP} maximizes the sum of the assignments given F. We claim that σ_{LAP} is an asymptotically unbiased estimator for the central permutation of $\{\sigma_1, \ldots, \sigma_m\}$.

For the sake of the readability of the proof, we give first two auxiliary lemmas. Let us first state the notation. Consider two natural numbers $1 \leq i_1, i_2 \leq n$ and two sets of permutations, X and Y. Let X be the set of permutations that fix position i_1, i.e., $X = \{\sigma | \sigma(i_1) = i_1\}$ and Y the set of permutations in which item i_2 is at position i_1, $Y = \{\sigma | \sigma(i_1) = i_2\}$ for a particular $i_2 \neq i_1$. We consider also a WMM under the Hamming distance centered w.l.o.g. around the identity, $\sigma_0 = e$, and where every dispersion parameter is higher than zero, $\theta_j > 0$ for every j.

Lemma 2 *The probability of the set X is greater than the probability of the set Y.*

$$\sum_{\sigma \in X} p(\sigma) = \frac{exp(\sum_{j=1}^{n} -\theta_j H_j(\sigma))}{\psi(\theta)} \quad > \quad \sum_{\sigma' \in Y} p(\sigma') = \frac{exp(\sum_{j=1}^{n} -\theta_j H_j(\sigma'))}{\psi(\theta)} \tag{4.19}$$

Proof. In order to see that $\sum_{\sigma \in X} p(\sigma) > \sum_{\sigma \in Y} p(\sigma)$, we will first construct a bijection between the permutations in X and those in Y. Note first that $|X| = |Y| = (n-1)!$. The bijection ϕ associates to every $\sigma \in X$ a permutation $\phi(\sigma) \in Y$ in the following way:

$$\phi(\sigma)(i_1) = i_2 \quad \wedge \quad \phi(\sigma)(\sigma^{-1}(i_2)) = i_1 \quad \wedge \quad \phi(\sigma)(i) = \sigma(i) \;\; \forall i \neq i_1, i \neq \sigma^{-1}(i_2)$$

In other words, $\phi(\sigma)$ is generated by swapping items i_1 and i_2 in σ. Therefore, for $\sigma, \pi \in X$ such that $\sigma \neq \pi$ then $\phi(\sigma) \neq \phi(\pi)$. In order to see Equation 4.19 we show below the relation between $H(\sigma)$ and $H(\phi(\sigma))$.

$$H_i(\sigma) = \begin{cases} 0 \text{ if } i = i_1 \\ 0 \text{ if } i = \sigma^{-1}(i_2) \wedge \sigma(i_2) = i_2 \\ 1 \text{ if } i = \sigma^{-1}(i_2) \wedge \sigma(i_2) \neq i_2 \end{cases}$$

$$H_i(\phi(\sigma)) = \begin{cases} 0 \text{ if } i = i_1 \\ 1 \text{ if } i = \sigma^{-1}(i_2) \wedge \sigma(i_2) = i_2 \\ 1 \text{ if } i = \sigma^{-1}(i_2) \wedge \sigma(i_2) \neq i_2 \\ H_i(\sigma) \text{ if } i \neq i_1 \wedge i \neq \sigma^{-1}(i_2) \end{cases}$$

It follows that $H_j(\sigma) \leq H_j(\phi(\sigma))$ for every j. Consequently,

$$exp(\sum_{j=1}^{n} -\theta_j H_j(\sigma)) > exp(\sum_{j=1}^{n} -\theta_j H_j(\phi(\sigma)))$$

and finally $\sum_{\sigma \in X} p(\sigma) > \sum_{\sigma \in Y} p(\sigma)$.

Lemma 3 *Let F be a square matrix. If the entries in the main diagonal are the maximum in their respective row, $F_{ii} > F_{ij}$ for all $i, j \neq i$, then the solution to the LAP problem when the objective is to maximize the sum is the identity permutation.*

Proof. Let $\sigma \in S_n$. By assumption, we know that $F_{ii} > F_{i\sigma(i)}$ for all $\sigma(i) \neq i$ and all $i = \{1, \ldots, n\}$. It follows that $\sum_{i=1}^{n} F_{ii} > \sum_{i=1}^{n} F_{i\sigma(i)}$ for all $\sigma \neq e$

Theorem 4 *Let F be the frequency matrix of a sample from the WMM under the Hamming distance centered around $\sigma_0 = e$. The solution to the LAP in F is an asymptotically unbiased estimator for the consensus permutation.*

Proof. In the context of permutations, we understand an asymptotically unbiased estimator, σ^m, as that for which, as the sample size m grows, the following holds:

$$\lim_{m \to \infty} E[d(\sigma^m, \sigma_0)] = 0$$

As m increases, $\lim_{m\to\infty} F_{ij} = \sum_{\sigma|\sigma(i)=j} p(\sigma)$. Lemma 2 shows that the probability $\sum_{\sigma|\sigma(i)=i} p(\sigma) > \sum_{\sigma|\sigma(i)=j} p(\sigma)$ for all $j \neq i$, thus $F_{ii} > F_{ij}$ for all $i, j \neq i$. Lemma 3 shows that for such a matrix F, the solution to the LAP consists on the identity permutation. In other words, as the number of samples tends to infinity, the probability of obtaining σ^m_{LAP} as a solution to the LAP problem tends to 1, i.e.,

$$\lim_{m \to \infty} p(\sigma^m_{LAP} \neq \sigma_0) = 0$$
$$\lim_{m \to \infty} p(\sigma^m_{LAP} = \sigma_0) = 1$$

Back to the definition of an asymptotically unbiased estimator we have that

$$\lim_{m\to\infty} E[d(\sigma^m_{LAP},\sigma_0)] = \lim_{m\to\infty} \sum_{\sigma'} d(\sigma',\sigma_0)p(\sigma')$$
$$= \lim_{m\to\infty} \left(\sum_{\sigma'\neq\sigma^m_{LAP}} d(\sigma',\sigma_0)0 + d(\sigma^m_{LAP},\sigma_0)1 \right) = 0$$

what concludes the proof.

Since σ_{LAP} is an asymptotically unbiased estimator for the consensus permutation of the WMM under the Hamming distance, we propose using σ_{LAP} as an approximate MLE for the consensus permutation. Fortunately, the LAP problem can be solved in polynomial time. One can find several algorithms for the LAP such as [72].

The MLE for the dispersion parameters are obtained by solving the system of equations in (4.18). A multidimensional Newton-Raphson implementation is provided in [100].

4.5 Experiments

This section is devoted to show the efficiency of the proposed algorithms in terms of computational time and accuracy. We will first deal with the sampling algorithms and then with the learning algorithms.

4.5.1 Sampling

In this chapter we propose three different sampling algorithms. The Distances algorithm generates samples from the MM, the Chain algorithm from the MM and WMM while the Gibbs algorithm, on the other hand, generates samples from approximations of both MM and GMM. The sampling experiments are designed to compare the performance of the algorithms with same two different criteria as the previous sections.

First, we compare the evolution of the error of the samples as the m grows. For each particular setting of n and θ (resp. $\boldsymbol{\theta}$) generate several samples of size $m = 200, 400, 600, \ldots, 19800, 20000$ with each of the sampling algorithms. Measure the error of each sample and plot the results.

We also compare the evolution of the error as the computational time increases. For each particular setting of n and θ (resp. $\boldsymbol{\theta}$) generate several samples for $t = 1, 2, 3, \ldots, 14, 15$ seconds with each of the sampling algorithms. Measure the error of each sample and plot the results.

The error in the MM is measured as the sum of the differences between the expected distance, $E[D]$, and the actual distances of the permutations in

the samples. In the WMM the error is measured as the sum of the differences between the expected H_j, $E[H_j]$, and the actual $\bar{H}_j$. The expression of the expectations are given in Theorem 2.

The parameter setting is as follows. The number of items in the permutations considered are $n \in \{5, 50, 100, 150\}$. The dispersion parameters in the MM case are $\theta \in \{0.1, 0.5, 1, 2, 3\}$. In the WMM case, the first of the dispersion parameters is $\theta_1 \in \{0.1, 0.5, 1, 2, 3\}$ while the rest are set such that $\theta_j = \theta_1 - (j-1)(\theta_1/2(n-1))$ for $j > 1$, i.e. θ_1 is the largest parameter while the value of the rest decrease linearly to $\theta_n = \theta_1/2$. For each parameter configuration 10 experiments are run and the average results of them are given. The central permutation is the identity. The Gibbs algorithm discards the first n^2 permutations as part of the burning-period.

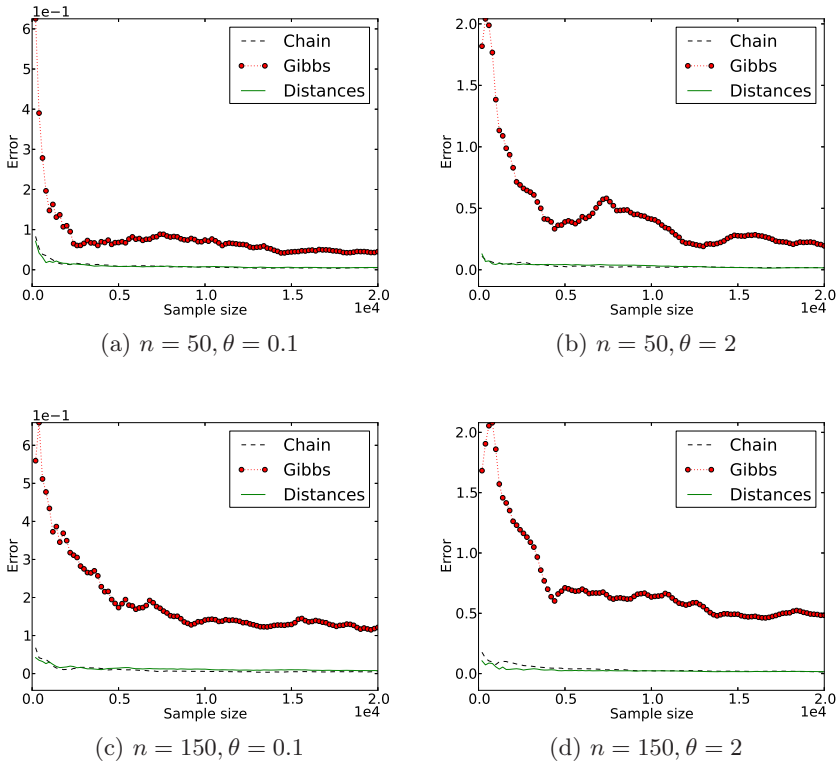
We have only introduced in this dissertation a representative selection of the experiments. In particular, we include in this chapter the results of the values of $n \in \{50, 150\}$ and the dispersion parameters $\theta \in \{0.1, 2\}$ in MM ($\theta_1 \in \{0.1, 2\}$ in WMM). The complete results can be found in `http://www.sc.ehu.es/ccwbayes/members/ekhine/sup_result/`.

Results for MM

By looking at Figure 4.1 we can see the evolution of the error as the size of the generated sample grows. Note that the error of the Distances and Chain algorithms are similar and are almost overlapped. The error of the Gibbs is always larger than the error of the other two algorithms, and these differences between the error increases with θ. However, the Gibbs is a very fast algorithm. The computational time required for the generation of the samples increases linearly with m, but the time required by the Gibbs is insignificant with respect to the time required by the other two. Gibbs required less than 50 milliseconds for each of the instances considered here, the Distances required around 100 milliseconds and Multistage around 600 ms for the instances of $n = 50$ and around 5 seconds for the instances of $n = 150$. Therefore, the question that naturally arises is what happens if all the algorithms are run for the same time.

By looking at Figure 4.2 we can see the evolution of the error as the computational time given grows. The distances sampling algorithm has the best trade-off between error and computational time. The Chain and Gibbs sampling algorithms have a similar performance when the distribution to sample is almost uniform. However, as θ grows and the sample gets more peaked, the Chain algorithm is clearly more accurate than the Gibbs.

Consequently, we can state that the method with best trade-off between computational time and accuracy is the Distances sampling algorithm. On the other hand, the Chain is as accurate as the Distances method but slower. The reason to keep it in consideration is that the Distances sampling algorithm can not generate samples from the WMM while the Chain can.



(a) $n = 50, \theta = 0.1$ (b) $n = 50, \theta = 2$

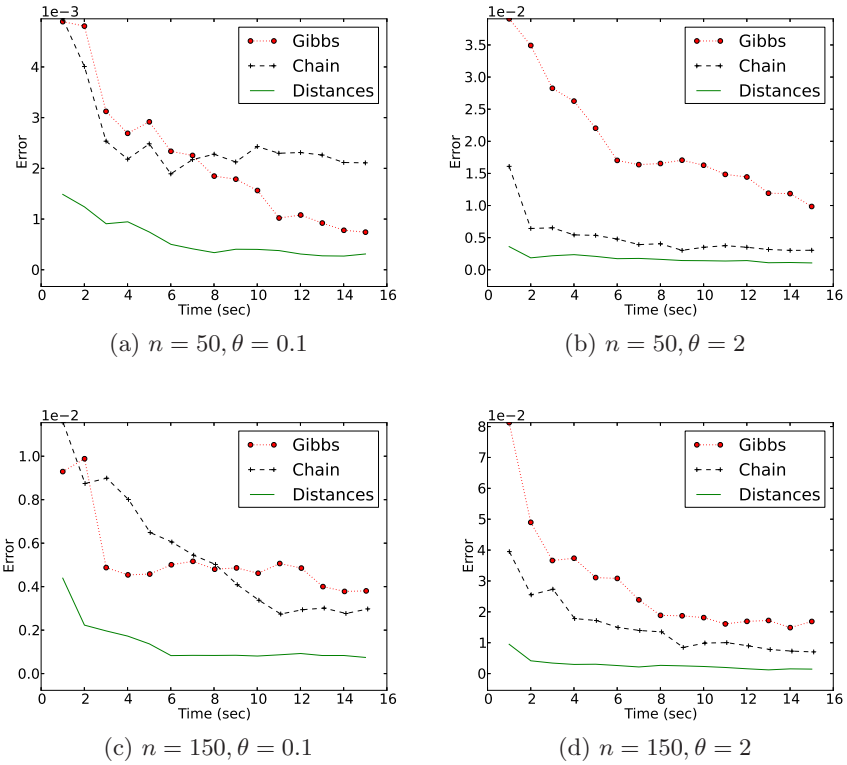
(c) $n = 150, \theta = 0.1$ (d) $n = 150, \theta = 2$

Fig. 4.1: Error of each algorithm as the sample size grows for different θ and n in the MM.

Result for WMM

The results relative to the evolution of the error of the generated sample (from a WMM) as m grows are given in Figure 4.3. Recall that for the WMM only Chain and Gibbs algorithms can be applied. Similar conclusions can be drawn from the results of the MM and WMM. As the sample size grows, the error slowly decreases for the Gibbs while being stable and close to zero for the Chain sampler. Again, the Chain sampling algorithm is much slower than the Gibbs but, at the same, time much more accurate. The running times are similar to the MM distribution.

Figure 4.4 shows the result of running both algorithms for the same computational time. Again, when the WMM is close to the uniform distribution the error results are similar, but the Chain algorithm outperforms the Gibbs for non-uniform distributions. Moreover, this difference increases as $\boldsymbol{\theta}$ increases.



(a) $n = 50, \theta = 0.1$ (b) $n = 50, \theta = 2$

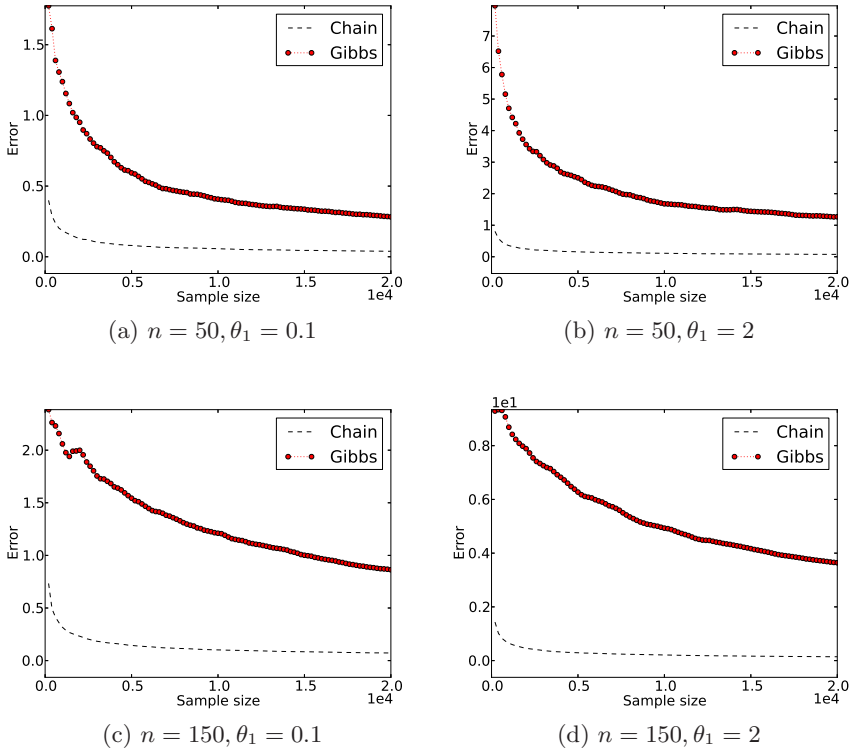
(c) $n = 150, \theta = 0.1$ (d) $n = 150, \theta = 2$

Fig. 4.2: Error of each sampling algorithm as the sample size computational time grows for different θ and n in the MM.

4.5.2 Learning

Recall that the MLE for the central permutation of the MM can be obtained in polynomial time with the well known Hungarian algorithm and the dispersion parameter, θ, for a given σ_0 can be computed with a Newton-Raphson algorithm. Therefore, we omit the MM from the learning experiments and focus thus on the WMM.

We have previously shown that the σ_{LAP} is an asymptotically unbiased estimator for the consensus permutation of a sample from a WMM. Therefore, this experimental section is designed to show how the quality of the estimated parameters evolve as the sample size, m, increases. Given that no efficient exact method for the MLE for the central permutation is known, the evaluation process will consist on generating a sample from the model centered around σ_0, learning the parameters and evaluating them w.r.t. σ_0. W.l.o.g. the consensus permutation is the identity, $\sigma_0 = e$. Among the proposed sam-



(a) $n = 50, \theta_1 = 0.1$ (b) $n = 50, \theta_1 = 2$

(c) $n = 150, \theta_1 = 0.1$ (d) $n = 150, \theta_1 = 2$

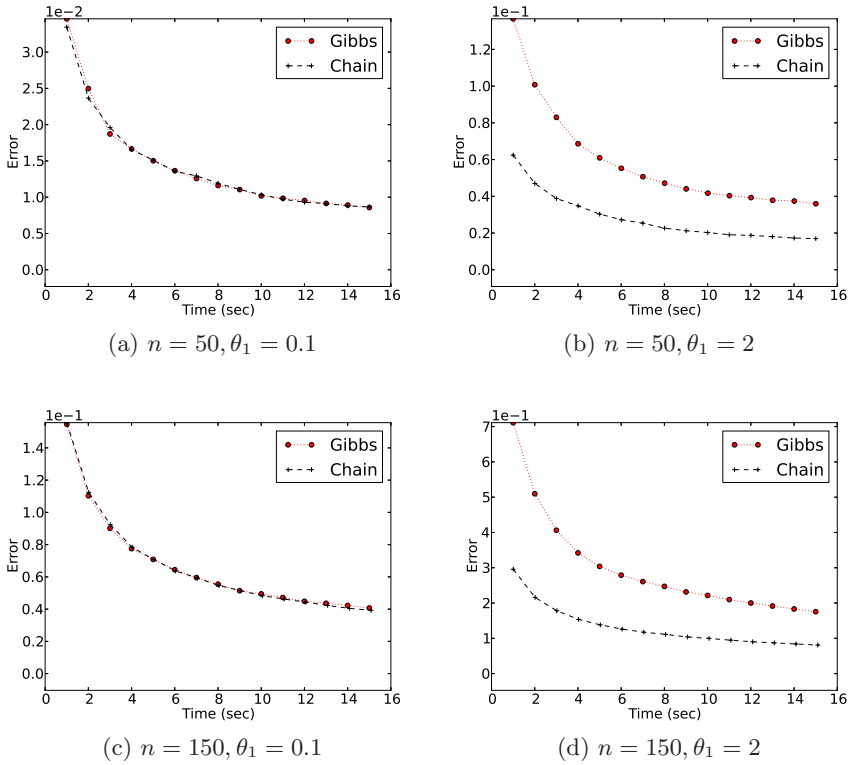
Fig. 4.3: Error of each sampling algorithm as the sample size grows for different $\boldsymbol{\theta}$ and n in the WMM.

pling algorithms the Chain is the most accurate and thus the one we will use. The evaluation criteria can be defined in different ways, we have chosen the following:

1. Compare the estimated central permutation and that which generated the sample
2. Compare the likelihood of the sample given the parameters that generated the sample with the likelihood of the sample with those estimated.

The evaluation procedure for each particular setting of n and $\boldsymbol{\theta}$ is as follows: Generate several samples of size $m = 1000, 2000, 3000, \ldots, 9000, 10000$ with the Chain sampling algorithm. Estimate σ_{LAP} and then:

1. Measure the Hamming distance between the actual and the estimated central permutations, $d(\sigma_{LAP}, \sigma_0)$.
2. Obtain the associated dispersion parameters for σ_{LAP} and compute the likelihood of the sample given σ_{LAP}, denoted as $\mathcal{L}_{LAP}$. Compute the like-



(a) $n = 50, \theta_1 = 0.1$ (b) $n = 50, \theta_1 = 2$

(c) $n = 150, \theta_1 = 0.1$ (d) $n = 150, \theta_1 = 2$

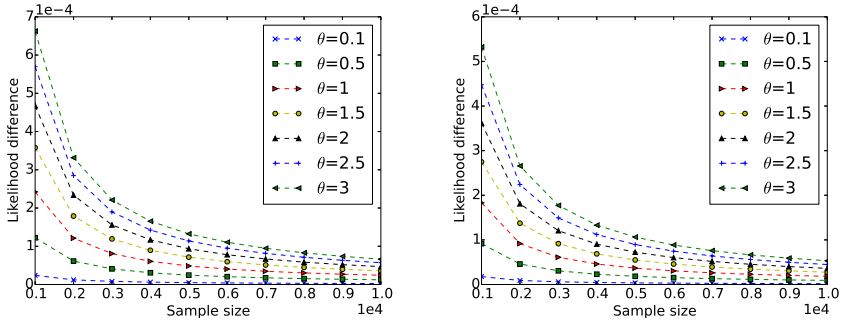
Fig. 4.4: Error of each sampling algorithm as the sample size computational time grows for different $\boldsymbol{\theta}$ and n in the WMM.

lihood of the sample given the parameters that generated the distribution, $\mathcal{L}_0$. The relative error of $\mathcal{L}_{LAP}$ is given by

$$\frac{\mathcal{L}_{LAP} - \mathcal{L}_0}{m\, \mathcal{L}_{LAP}} \tag{4.20}$$

The sample size m in the denominator is aimed to put in the same scale the likelihood for different samples sizes.

The parameter setting is as follows. The number of items in the permutations considered are $n \in \{5, 50, 100, 150\}$. The first of the dispersion parameters is $\theta_1 \in \{0.1, 0.5, 1, 2, 3\}$ while the rest are set such that $\theta_j = \theta_1 - (j-1)(\theta_1/2(n-1))$ for $j > 1$, i.e. θ_1 is the largest parameter while the value of the rest decrease linearly to $\theta_n = \theta_1/2$. For each parameter configuration 10 experiments are run and the average results of them are given. W.l.o.g. the central permutation is the identity.



(a) $n = 50$ (b) $n = 150$

Fig. 4.5: Evolution of the error of the estimated parameters for GMM as m grows.

Results for WMM

Figure 4.5 shows the evolution of the likelihood as the sample size grows, where the error is measured as shown in Equation (4.20). Each plot includes the results of every dispersion parameter of a given n. The plots clearly show that as m increases the error in the likelihood decreases. Even in the cases where the estimated central permutation is correct, the estimated dispersion parameters are better estimated with larger sample sizes, resulting on a better likelihood.

It is important to note that the results are only comparable for the instances of the same n and $\boldsymbol{\theta}$. This means that even though the results of an instance of parameters n and m have smaller error for small values of θ than for large values of $\boldsymbol{\theta}$ we can not state that the former results are better than the latter.

4.6 Conclusions

The Hamming distance is the most popular distance for permutations which arises in coding and matching problems. The MM under the Hamming distance can be thus thought as a probability model on matchings, for example, in which there is a"correct" matching and any other matching is less probable as its similarities with the correct one decrease. The WMM is an adaptation of the GMM for the Hamming distance and models situations in which the correct matching of certain items is more important than the matching of others.

The computation of the normalization constant is usually the bottleneck of the MM and GMM. The first result presented in this chapter is an efficient

expression of the normalization constant for the WMM under the Hamming distance. This expression is the key to efficiently compute the expected distance, the expected number of fixed points and the marginal and conditional probabilities.

We propose three different sampling algorithms. The Gibbs sampler is an adaptation of the well known Monte Carlo algorithm. It is very fast but it generates samples from approximations to the MM and GMM of interest. The Chain sampling algorithm, which can generate samples from MM and GMM, makes use of the well known chain rule for probability. The chain rule computes the joint distribution using conditional probabilities. Therefore, our proposed method to compute conditional probabilities is crucial for this sampling algorithm. The last proposed sampling algorithm is the Distances, which is based on counting and u.a.r. generating permutations at a given distance. Thus, it has a very strong combinatorial basis. It is a fast and accurate algorithm which can only generate samples from a MM. Summarizing, the experimental evaluation

Regarding the learning process, we show how to estimate the parameters of a sample from a MM in polynomial time. Despite the complexity of the estimation of WMM is not known, we conjecture that it is NP-complete. We propose a very efficient learning algorithm and show that the estimated parameters are asymptotically unbiased estimators of the real ones.

We have tried to give a step forward the popularization of the Hamming distance-based probability models for permutations by providing diverse, efficient and accurate algorithms for the most critical operations for distributions: calculating the probability, marginalizing, conditioning, computing the expectation and, probably the most important, sampling and learning.

5

Mallows Model under the Ulam distance

Among the models considered in this dissertation, the Ulam based probability models are certainly the most challenging. The most significant impediment is the unfruitfulness of the combinatorial approaches used to tackle the other distance related models. Instead, almost every operation related to permutations under the Ulam distance needs to be posed entirely as problems over alternative combinatorial objects: the standard young tableaux (SYT). Fortunately, there is a mechanism to transform from the permutation domain to the SYT domain, which is the Robinson-Schensted correspondence (RS). As a consequence, the algebraic background of the present chapter is the most demanding among the chapters in this dissertation.

Moreover, no decomposition is known for the Ulam distance and no factorization is known for the MM under the Ulam distance. Our proposed approach for dealing with the MM under the Ulam distance on permutations of n items is based on the enumeration of all the integer partitions of n. It follows that the complexity of our proposed approach is conditioned by the complexity of enumerating the partitions of n items, which grows subexponentially with the number of items.

This chapter is organized as follows. First of all, in Section 5.1, we detail the uniformly at random generation of a permutation at Ulam distance d, which will be useful for the sampling process. Moreover, we will state a computational solution to overcome the computational challenges found when dealing with the MM on permutations of large n. Section 5.2 explicitly defines the MM under the Ulam distance. We include an efficient computation of the normalization constant, crucial for both the computation of the probability of a permutation and its expected distance under the MM. Sampling the MM under the Ulam distance is addressed in Section 5.3, where the Gibbs and Distances algorithms are adapted for the MM under the Ulam distance. Section 5.4 shows how to estimate the parameters of a sample from a MM. Finally, Section 5.5 evaluates the proposed methods and Section 5.6 concludes the chapter.

5.1 Side issues

The Distances sampling algorithm, which has been discussed in previous chapters, is adapted for the MM under the Ulam distance. Therefore, we must be able to count and generate permutations at a given distance u.a.r. The problem of counting the permutations at a given distance is explained in Section 1.1.4. We show how to use the RS correspondence and the SYT for the task of generating u.a.r. permutations at a given distance here. Again, this method can be easily adapted to enumerate permutations at a given distance.

It is also shown how to tackle the computational challenges found when dealing with the MM on permutations of large n, say $n = 100$. The difficulties are due to the exponential growth of the number of partitions of n and a proportional increase of the required memory.

5.1.1 Generating a permutation at distance d uniformly at random

The uniformly at random generation of a permutation at distance d is a recurrent question in this dissertation. The Ulam distance case, however, is the most peculiar case since it relies on alternative combinatorial structures such as the SYT, FD and RS correspondence, all of which are detailed in Section 1.1.4.

Recall that the RS correspondence claims that there is a bijection between pairs of SYT of the same shape and permutations. Therefore, the RS correspondence allows us breaking the problem of randomly generating a permutation at a given Ulam distance on three simpler stages as follows.

1. Randomly select a FD of shape $\lambda \vdash n$. The probability of selecting each shape λ is proportional to the number of permutations at distance d that can be generated with it. Let us detail this last point.
   The Hook length formula in Equation (1.3) shows how to compute the number of different SYT of shape λ that can be generated, h_λ. Therefore, there are h_λ^2 different possible pairs of STY of the given shape λ. The RS correspondence defines a bijection between pairs of SYT of the same shape λ and permutations, so the number of possible permutations that are generated from a FD of shape λ is h_λ^2.
   Moreover, as stated in Proposition 1, the length of the LIS of σ generated from a FD of shape λ equals $\lambda(1)$. Since the length of the LIS of σ equals $n - d(\sigma)$, the probability of shape λ is as follows.
   $$p(\lambda) \propto h_\lambda^2 \quad \forall\, \lambda \text{ such that } \lambda(1) = n - d$$
2. Uniformly at random generate two SYT of shape λ, namely P and Q. The random generation of SYT of shape λ has been addressed in [58].
3. Generate σ given the SYT P and Q with the inverse Schensted algorithm (see Algorithm 3).

5.1.2 Computational remarks

All the processes introduced in this chapter with the exception of the Gibbs sampler, consider at least one of these two data structures: the sequence of the number of permutations at each distance, $\{S_u(n,0), S_u(n,1), \ldots, S_u(n,n-1)\}$, and the sequence of different FD of every shape $\lambda \vdash n$. The former is a sequence of n numbers whose computation relies on the enumeration of the latter.

As we have already stated, the number of FD is exactly the number of integer partitions of n, which, at the same time, grows subexponentially with n. Clearly, as n increases, maintaining every FD in the memory becomes absolutely prohibitive. To overcome this problem, we have designed a framework that does not store every FD in the RAM. This version needs a preprocess phase in which the necessary information is generated and stored into files in the local disk. Then, the sampling and learning algorithms work in the same way as explained, but with the information in the files instead of the memory.

This technique consists on generating the files which store the relevant information, that is, the shape of every FD of n boxes, the number of permutations that can be generated with the given shape and the sequence of the number of permutations at each distance. Supposing we are dealing with distributions on permutations of n items, this approach generates $n+1$ files following this structure.

- The information about the FD of every shape $\lambda \vdash n$ are divided in n different files. Any two FD of shapes $\lambda, \lambda' \vdash n$ are in the same file iff $\lambda(1) = \lambda'(1)$. In this way, every FD that generates permutations at the same distance is stored in the same file. For each FD we store the partition $\lambda \vdash n$ and the number of permutations that can be generated with it, h_λ.
- One file stores the sequence $S_u(n,d)$ for every value of d.

It is worth noting that only combinatorial information is included in these files. This means that these files only need to be computed once for each distribution of permutations of n items. For example, there is no need of recomputing the files for sampling distributions of different θ (as long as they consider permutations of the same number of items, n). On the other hand, the learning process for distributions on permutations of n items only needs the file with the sequence $\{S_u(n,0), S_u(n,1), \ldots, S_u(n,n-1)\}$.

5.2 MM under the Ulam distance

The MM under the Ulam distance can be written as follows.

$$p(\sigma) = \frac{exp(-\theta d_u(\sigma\sigma_0^{-1}))}{\psi(\theta)} \tag{5.1}$$

This distribution is not decomposable as far as the authors know. However, by exploiting the same ideas that lead to the Distances sampling method,

i.e., right invariance and equal probability for the permutations at the same distance, it yields an efficient expression for the normalization constant.

$$\psi(\theta) = \sum_{d=0}^{n-1} S_u(n,d) exp(-\theta d) \tag{5.2}$$

Since there is not decomposition vector for the Ulam distance, there is no GMM nor WMM under Ulam.

5.2.1 Expected distance

The expected value of the distance under the MM with the Ulam distance is given by the following:

$$E_\theta[d_u] = \frac{\sum_{d=0}^{n-1} S_u(n,d) exp(-\theta d) d}{\psi(\theta)} \tag{5.3}$$

5.3 Sampling

The random generation from a MM under the Ulam distance is addressed by means of the Gibbs and the Distances sampling algorithms. Recall that the proposed algorithms generate a sample centered around the identity. To set a different σ_0, each permutation in the sample should be composed with σ_0.

5.3.1 Gibbs sampling algorithm

The Gibbs sampler generates samples from an approximate distribution of the MM under the Ulam distance as follows:

1. Generate uniformly at random a permutation σ.
2. Generate uniformly at random two integers, i, j in the range $[1, n]$.
3. Build a new permutation σ' as follows:

$$\sigma' = \begin{cases} \sigma(1), \ldots, \sigma(i-1), \sigma(i+1), \ldots, \sigma(j), \sigma(i), \sigma(j+1), \ldots, \sigma(n) & \text{if } i < j \\ \sigma(1), \ldots, \sigma(j-1)\sigma(i), \sigma(j), \ldots, \sigma(i-1), \sigma(j+1), \ldots, \sigma(n) & \text{if } i > j \end{cases}$$

4. Let $\gamma = min\{1, p(\sigma')/p(\sigma)\}$. With probability γ the algorithm accepts the candidate permutation moving the chain to it, $\sigma = \sigma'$, and goes back to 2. Otherwise, it discards σ' and goes back to step 2.

The initial samples are discarded. The above process is repeated until the algorithm generates a given number of permutations. In previous chapters we showed how to compute $min\{1, p(\sigma')/p(\sigma)\}$ without computing $p(\sigma')$ and $p(\sigma)$ what results on a computational time reduction. Unfortunately, there is no such efficient method for the Ulam distance and thus, the complexity of generating each permutation is $O(n \log n)$.

5.3.2 Distances sampling algorithm

As shown in previous chapters, the probability of obtaining a permutation at Ulam distance d from the identity permutation is as follows.

$$p(d) = \sum_{\sigma | d_u(\sigma, \sigma_0^{-1}) = d} p(\sigma) = S_u(n, d) \frac{exp(-\theta d)}{\psi(\theta)} \tag{5.4}$$

Recall that the normalization constant is given in Equation (5.2). In this way, the process of sampling a permutation from a MM under the Ulam distance can be done in two stages.

1. Randomly select a distance d. Taking into account Equation (5.4), the probability of each distance is as follows:

$$p(d) = \psi(\theta)^{-1} S_u(n, d) exp(-\theta d) \quad 0 \leq d < n \tag{5.5}$$

2. Uniformly at random generate a permutation at distance d from e as shown in Section 5.1.1.

5.4 Learning

This section approaches the maximum likelihood estimation for the parameters of the distribution given a sample of m i.i.d. permutations $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$. The log likelihood of the MM under the Ulam distance is given by the following expression.

$$\begin{aligned} Ln\, \mathcal{L}(\{\sigma_1, \sigma_2, \ldots, \sigma_m\} | \sigma_0, \theta) &= \sum_{s=1}^{m} Ln\, p(\sigma_s | \sigma_0, \theta) = Ln \prod_{s=1}^{m} \frac{exp(-\theta d_u(\sigma_s \sigma_0^{-1}))}{\psi(\theta)} \\ &= -\theta \sum_{s=1}^{m} (d_u(\sigma_s \sigma_0^{-1})) - m Ln\, \psi(\theta) \end{aligned} \tag{5.6}$$

By looking at Equation (5.6), we can see that calculating the value of σ_0 that maximizes the equation is independent of θ. Therefore the maximum likelihood estimation problem for the MM under the Ulam distance can be posed as a two step process in which first the MLE for the central permutation $\hat{\sigma}_0$ is obtained and then the MLE for the dispersion parameter $\hat{\theta}$ for the given $\hat{\sigma}_0$ is calculated.

The MLE for the dispersion parameter θ is obtained by equaling to zero the derivative, which is given by the following equation:

$$\bar{d} = \frac{\sum_{d=0}^{n-1} S_u(n, d) exp(-\theta d) d}{\psi(\theta)} \tag{5.7}$$

Algorithm 9: Select the set-median permutation of a given sample

Input: $\{\sigma_1, \ldots, \sigma_m\}$, a sample of m permutations;
Output: σ_0, set-median permutation of the sample
$\mathbf{d} = (d_1, \ldots, d_m) = (0, \ldots, 0)$;
for *i = 1* **to** *m-1* **do**
  for *j = i+1* **to** *m* **do**
    $\mathbf{d}_i = \mathbf{d}_i + d_u(\sigma_i, \sigma_j)$;
    $\mathbf{d}_j = \mathbf{d}_j + d_u(\sigma_i, \sigma_j)$;
  end
end
$\sigma_0 = \sigma_i$ where $i = \arg\min_j d_j$;
return σ_0;

The average distance is $\bar{d} = \sum_{i=1}^{m} d_k(\sigma_i \sigma_0^{-1})/m$. Note that again $\bar{d}$ as expressed in Equation (5.7) equals the distance expectation of the MM in Equation (5.3).

There is no closed expression for the previous formula. However, we can efficiently solve it with a root finding method.

5.4.1 Approximate algorithm

We propose an approximate algorithm for the MLE for the σ_0 based on adapting classical approaches for similar problems.

Problems consisting on finding the permutation that minimizes the sum of the distances to the permutations in the sample are often called median problems. Median problems are usually NP-complete. Among the distances consider in this dissertation, the only median problem for which a polynomial time solution is known is the Hamming distance. Although the question of the complexity of the median problem for the Ulam distance is open, it is supposed to be NP-complete. Similar problems such as the Transposition median problem [12] and the Median String problems [35], both related to the Ulam distance, are NP-complete.

An approximate solution for the String median problem consists on looking for the permutation in the sample that minimizes the sum of the distances, [95]. This approach is called set-median problem. Our proposed approach for the MLE of the parameters of a sample takes a similar approach and selects as the estimated σ_0 the set median permutation of the sample. An optimized version of the set-median algorithm in which the pairwise distances is only computed once is given in Algorithm 9.

We propose an algorithm for approximating the MLE for the parameters of a given sample which works in two stages, as follows:

1. Select the set median permutation under the Ulam distance as the MLE for σ_0.

2. Compute the dispersion parameter with a root finding algorithm such as Newton-Raphson.

The accuracy of the set median permutation as an estimator for the central permutation increases with m. Nonetheless, increasing m is a double-edged sword since the complexity of computing the set median permutation is $O(m^2\, n\, log\, n)$, i.e., increases quadratically with the sample size.

5.5 Experiments

This section is devoted to test the performance of the sampling and learning algorithms introduced in this chapter.

5.5.1 Sampling experiments

In this section we measure the performance of the sampling algorithms for the generation of a sample $\{\sigma_1, \ldots, \sigma_m\}$.

The algorithms are evaluated regarding two different criteria. For each criterium a different evaluation procedure is considered, which are as follows.

As in previous chapters, we first consider the evolution of the error of each sample as its number of permutations, m, increases. Again, we use the same procedure. For each particular setting of n and θ generate several samples of size $m = 200, 400, 600, \ldots, 19800, 20000$ with each of the sampling algorithms. Measure the error of each sample and plot the results.

We are also concerned with the evolution of the error as the computational time increases since, again, the Gibbs sampler is much faster than the Distances algorithm. Again, for each particular setting of n and θ generate several samples for $t = 1, 2, 3, \ldots, 14, 15$ seconds with each of the sampling algorithms. Measure the error of each sample and plot the results.

The error of sample $\{\sigma_1, \ldots, \sigma_m\}$ in the MM is measured as the difference between the average distance to the sample $\bar{d} = \sum_{i=1}^{m} d(\sigma_i, \sigma_0)/m$, and the expected distance, $E[d_u]$, which is given in Equation (5.3). The central permutation is the identity.

The parameter setting is as follows. The number of items in the permutations considered are $n \in \{5, 50, 100\}$. The dispersion parameters in the MM case are $\theta \in \{0.1, 0.5, 1, 2, 3\}$. For each parameter configuration 10 experiments are run and the average results of them are given. W.l.o.g. the central permutation is the identity. The Gibbs algorithm discards the first n^2 permutations as part of the burning-period.

In particular, we include in this section the results of the values of $n = \{50, 100\}$ and the dispersion parameters $\theta = \{0.1, 2\}$ and refer the interested reader to `http://www.sc.ehu.es/ccwbayes/members/ekhine/sup_result/` for the complete results. Recall the we have defined two computational approaches: the first one consists on storing both the sequence $S_u(n, d)$ for every

d and the collection of FD in the RAM memory. The second one, as explained in Section 5.1.2, is to save those structures to disk. Experiments of $n = \{5, 50\}$ follow the former approach. However, the amount of memory required for the experiments of $n = 100$ was unafordable as the memory required is proportional to the number of partitions of 100, which is about 2×10^8. Therefore, for the experiment of $n = 100$ we follow the approach detailed in Section 5.1.2, that is, the FD were first stored into files. In this case, the process of storing the data in the disks took about half an hour.

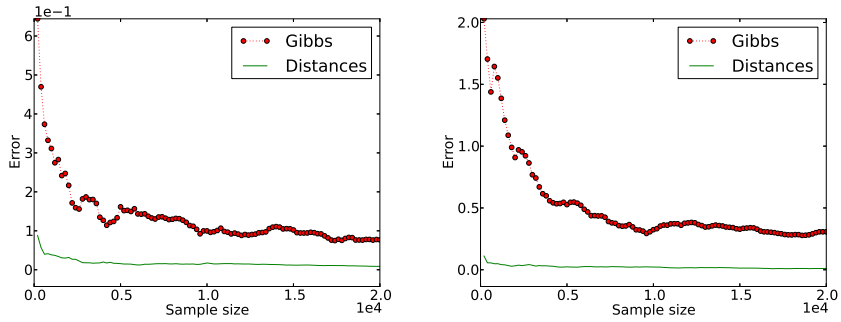
Figure 5.1 shows the evolution of the error of the sample as m grows. Clearly, the error of the sample decreases as m increases for both sampling algorithms. However, the error of the Distances sampler is always smaller than the error of Gibbs, being notorious the fact that, for every n and θ, the error of a sample of $m = 20000$ generated with Gibbs is larger than the error of a sample of $m = 200$ permutations of Distances. Moreover, the differences in the error of both algorithms increase with θ.

The computational time for the generation of the 20000 permutations of $n = 50$ with the Gibbs algorithm is about 250 ms. For the same n, the computational time needed by the Distances algorithm varies regarding the dispersion parameter: for almost uniform distributions, $\theta = 0.1$, the largest sample was generated in less than 2.5 seconds and for $\theta = 2$ it required about 1.4 seconds. Let us detail where does the time difference come from.

Among the operations involved on the random generation of a permutation from a MM (see Section 5.3.2), the one that contributes in the time difference the most is the random selection of a FD at a target distance d, which is detailed in Section 5.1.1. Let $\bar{d}_1$ be the average target distance under a MM of $\theta = 0.1$, $\bar{d}_1 = E_{0.1}[d_u]$ and equivalently, $\bar{d}_2 = E_2[d_u]$. Clearly, this target distance will be on average different when the dispersion parameter is 0.1 or 2. The aforementioned step requires randomly choosing a FD of shape λ among those such that $\lambda(1) = n - d$. It turns out that the set shapes is, on average, larger for $\bar{d}_1$ than for $\bar{d}_2$, that is $|\{\lambda|\lambda(1) = n - \bar{d}_1\}| >> |\{\lambda|\lambda(1) = n - \bar{d}_2\}|$. Consequently, the larger the set, the longer will take to randomly select the FD.

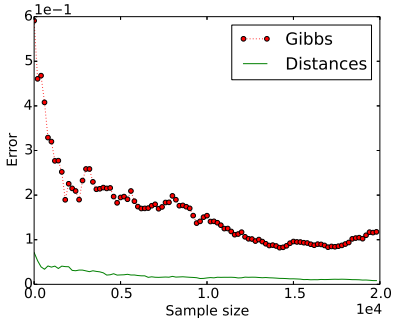
For $n = 100$, the Gibbs algorithm needed about 200 ms to generate the largest sample size while the Distances sampler needed about 1.4 seconds to generate from the almost uniform distributions and about 1.2 seconds to generate from the most peaked distributions. Notice that these referred times are smaller than the computational time of the experiments of $n = 50$. This is because for the sampling experiments of $n = 100$ do not include the time of saving the FD to disk, which, as stated, took about half an hour.

Our next concern is to evaluate the algorithm in the case when both are allowed to run for the same time, Figure 5.2. Both algorithms get smaller errors for larger computational times, and the decrease is more notorious for the Gibbs. The error of the Distances is always smaller than error of the Gibbs sampler and the difference between both algorithms gets more evident as θ and n increase.

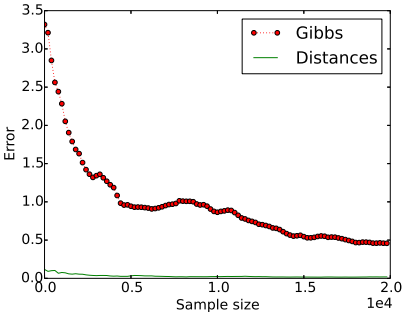


(a) $n = 50, \theta = 0.1$

(b) $n = 50, \theta = 2$



(c) $n = 100, \theta = 0.1$



(d) $n = 100, \theta = 2$

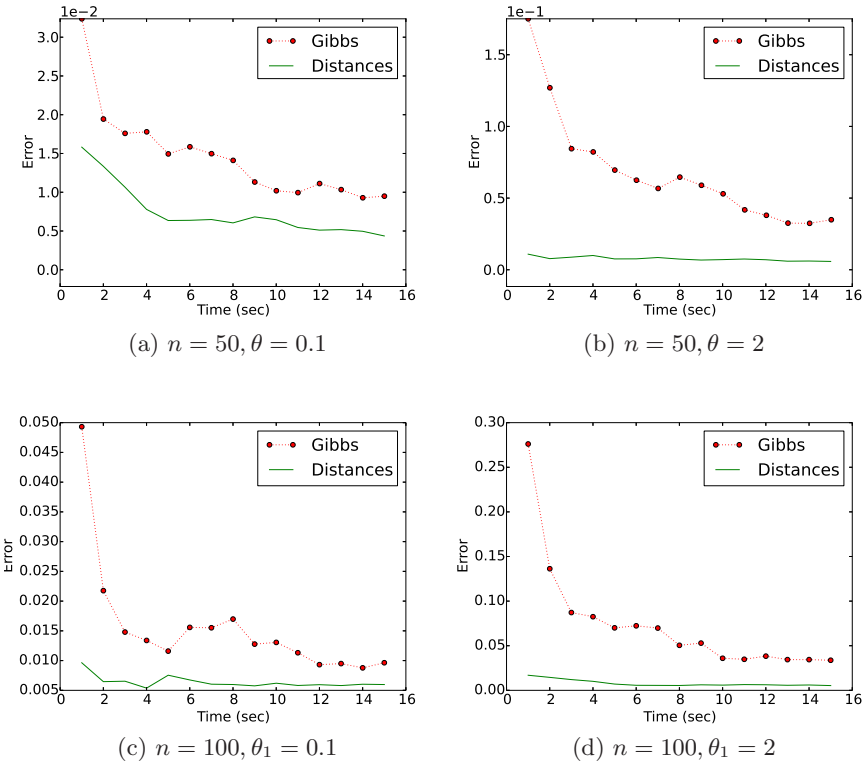
Fig. 5.1: Error of each sampling algorithm as the sample size grows for different θ and n in the MM.

5.5.2 Learning experiments

In this section we deal with the approximate learning of the MM under the Ulam distance. The learning algorithm is evaluated with synthetic samples generated from distributions of known parameters, σ_0 and θ, using the Distances sampling algorithm, which is the most accurate one.

The experiments in this section are designed to see the evolution of the error of the estimated parameters as the sample size increases.

We now detail the experimental procedure. For each particular setting of n and θ generate several samples of size $m = 1000, 2000, 3000, \ldots, 9000, 10000$. Learn the MLE for the parameters. The central permutation is approximated with the set median permutation, denoted $\bar{\sigma}$. The dispersion parameters for the MM are computed as shown in Equation (5.7) using a with a Newton-Raphson algorithm. Finally, denote by $\bar{\mathcal{L}}$ the likelihood of the sample under the generative model $(\bar{\sigma}, \theta)$. The relative error of $\bar{\mathcal{L}}$ is given by



(a) $n = 50, \theta = 0.1$ (b) $n = 50, \theta = 2$

(c) $n = 100, \theta_1 = 0.1$ (d) $n = 100, \theta_1 = 2$

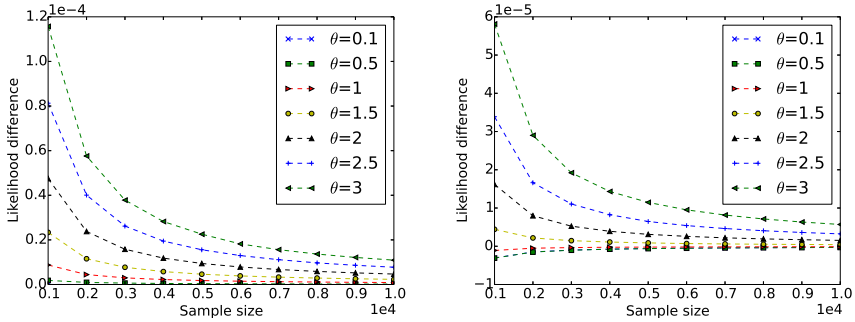
Fig. 5.2: Error of each sampling algorithm as the computational time grows for different θ and n in the MM.

$$\frac{\bar{\mathcal{L}} - \mathcal{L}_0}{m\,\bar{\mathcal{L}}} \tag{5.8}$$

where $\mathcal{L}_0$ is the likelihood of the sample given the parameters that generated the distribution, σ_0 and θ. The sample size m in the denominator is aimed to put in the same scale the likelihood for different sample sizes.

The parameter setting is as follows. The number of items in the permutations considered are $n \in \{50, 100, 100\}$. The dispersion parameters are $\theta \in \{0.1, 0.5, 1, 2, 3\}$. For each parameter configuration 10 experiments are run and the average results of them are given. The central permutation is the identity.

Figure 5.3 shows the evolution of the estimated parameters as the sample size grows, measured as shown in Equation (5.8). Each graphic includes the results of every dispersion parameter of a given n. The graphics clearly show that as m increases the likelihood of the estimated parameters tend to



(a) $n = 50$ (b) $n = 100$

Fig. 5.3: Evolution of the error of the estimated parameters for the MM as m grows.

the likelihood of the true parameters. Even in the cases where the estimated central permutation is correct, the estimated dispersion parameters are better estimated with larger sample sizes, resulting on a better likelihood.

5.6 Conclusions

This chapter is devoted to the MM under the Ulam distance. Almost every section relies on the ability of counting and generating permutations at a given Ulam distance. At the same time, these problems are completely reformulated in the domain of Standard Young Tableaux. This is possible due to the Robinson-Schensted correspondence, which makes possible changing from the domain of SYT to permutations and vice versa.

In order to be able to operate with permutations of a large number of items, we have to move from the combinatorial arena to the computational arena. Given that the amount of memory quickly becomes unaffordable, our approach consists on working with the data in the disks instead of the RAM.

By exploiting Ulam's right invariance and the fact that every permutation at the same distance has equal probability, we are able to give an efficient expression for the normalization constant, leading to efficient computation of the probability of a permutation and the expression of the expected value of the distance under the MM.

In this chapter we have adapted the Distances sampling algorithm for sampling permutations from a MM under the Ulam distance. It is entirely based on the ability of counting and generating permutations at a given Ulam distance. The Gibbs algorithm has also been adapted to sample the MM. It is faster than Distances sampler but samples from an approximate distribution of the MM. The experiments show that Distances outperforms the Gibbs

sampler in terms of log-likelihood of the sample in the two tested scenarios: (i) generating a sample of fixed size and (ii) generating a sample by running each algorithm the same amount of time.

A learning algorithm is also introduced. It obtains an approximate MLE for the central permutation and then the dispersion parameter for it. It is based on the fact that equivalent approaches are taken in related problems.

As a summary, we can state that both learning and sampling algorithms have a strong combinatorial basis.

6

PerMallows: an R package for MM, GMM and WMM

In order to contribute to the efficient reasoning on permutations, we have packed the sampling and learning algorithms described in this dissertation in an `R` package, `PerMallows`. The package can be freely download from the CRAN repository at `http://cran.r-project.org/web/packages/PerMallows/index.html`. In this chapter we show how to use the `PerMallows` package by providing useful examples.

For the sake of clearness, we have tried to give a unified interface for the functions included here by parametrizing the functions by distance. In order to provide a complete framework for reasoning on permutations, `PerMallows` includes functions to work with permutations.

In order to follow the conventions in the `R` community, the number of items in the permutations, n, is denoted `perm.length` in `PerMallows` and both notations will be used interchangeably through out this chapter.

In this section we show how to use the `PerMallows` package. `PerMallows` includes functions to generate permutations, as well as the most common operators and distance related functions. The metrics considered are Kendall's-τ, Cayley, Hamming and Ulam. Moreover, `PerMallows` implements the algorithms for probability distributions introduced in [71, 70, 69]. The probability models considered are Mallows (MM) and Generalized Mallows (GMM).

The number of items in the permutations, n, is denoted `perm.length` in `PerMallows`.

6.1 Permutations

Generation

The most basic function consists of generating permutations. The permutations are coded as vectors of the first n natural numbers where each item appears once and once only. They can be defined by hand as follows:

```
> sigma <- c(1, 5, 6, 4, 2, 3)
> sigma

[1] 1 5 6 4 2 3
```

The validity of a vector as a permutation can be checked with the function `is.permutation`.

```
> is.permutation(perm = sigma)

[1] TRUE

> is.permutation(c(0, 1, 5, 4, 2, 3))

[1] FALSE

> is.permutation(c(1, 8, 9, 2, 5, 3))

[1] FALSE
```

The identity permutation is that which maps every item i to position i. It can be created with the `identity.permutation` function.

```
> identity.permutation(perm.length = 6)

[1] 1 2 3 4 5 6
```

The generation of a permutation uniformly at random is supported via the `runif.permutation` function.

```
> runif.permutation(n = 2, perm.length = 6)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    3    6    4    1    5    2
[2,]    2    3    5    1    4    6
```

The generation of the set of every possible permutation of n items is carried out with the `permutations.of` function. Recall that the number of permutations of n items increases factorially with n and it is thus computationally expensive to generate every permutation of $n \geq 10$. By default, the `alert` argument is set to true. When `alert` is `TRUE` and n is greater than 9, an alert message is shown.

```
> permutations.of(perm.length = 3, alert = FALSE)

     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    3    2
[3,]    2    1    3
[4,]    2    3    1
[5,]    3    1    2
[6,]    3    2    1
```

```
> permutations.of(perm.length = 10)

Do you really want to generate all  3628800  permutations? (Y/N)
[1] "Process cancelled"
```

The collection of permutations generated is stored as a 2-dimensional matrix. It is also possible to read such a matrix from disk using the function `read.permutation.file`, which also checks if every row is a valid permutation.

```
> path = system.file("test.txt", package = "PerMallows")
> sample = read.permutation.file(path)
```

Together with the `PerMallows` package we provide some small datasets that will be used as running examples throughout this reference manual.

```
> data("data.1")
> data.1

     V1 V2 V3 V4
[1,]  1  3  2  4
[2,]  2  1  3  4
[3,]  1  4  3  2
[4,]  1  2  4  3
[5,]  2  3  4  1
```

Another way of generating permutations is by ranking the ratings of a set of items. Suppose we have the results of five students in three different tests. An example of such a file is given in 'data.order'. After reading the file, we can get the ranks of each student with the system function `order.ratings`.

```
> data("data.order")
> data.order

   V1   V2  V3  V4   V5
1 0.1 4.20 2.0 9.4 9.00
2 6.3 2.11 0.1 5.7 4.00
3 9.0 4.50 7.1 6.3 0.21

> order.ratings(ratings = data.order)

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    2    5    4
[2,]    3    2    5    4    1
[3,]    5    2    4    3    1
```

Operations

Two permutations can be composed and the result is a permutation.

```
> sigma <- c(1, 5, 6, 4, 2, 3)
> pi <- c(3, 5, 1, 2, 6, 4)
> compose(perm1 = sigma, perm2 = pi)

[1] 6 2 1 5 3 4
```

Be aware that the composition is not commutative.

```
> compose(perm1 = sigma, perm2 = pi)

[1] 6 2 1 5 3 4

> compose(perm1 = pi, perm2 = sigma)

[1] 3 6 4 2 5 1
```

Our implementation of the composition allows one of the arguments to be a collection of permutations. In this case, every permutation in the sample is composed with the permutation in the other argument, resulting in a new collection of permutations.

```
> tau <- c(2, 1, 3, 4)
> compose(perm1 = data.1, perm2 = tau)

     V2 V1 V3 V4
[1,]  3  1  2  4
[2,]  1  2  3  4
[3,]  4  1  3  2
[4,]  2  1  4  3
[5,]  3  2  4  1
```

A useful summary of a sample is given by the number of permutations in the sample in which item i appears at position j. This is usually denoted as the frequency matrix or first order marginal matrix. The current package supports it via the `freq.matrix` function. The parameter `perm` can be either a sample of permutations or a single permutation. The frequency matrix of a single permutation is usually denoted permutation matrix. We will illustrate the `freq.matrix` function using the well known APA dataset.

The American Psychological Association (APA) dataset includes 15449 ballots of the election for the president in 1980. Each voter ranked at least one of the five candidates. Along with this package we distribute the 5738 ballots that ranked the five candidates by the name of `data.apa`. This dataset has been largely used in the literature. In particular, one can find in [39] a spectral analysis that takes into account the first order marginal of the dataset which can be computed as follows:

```
> data("data.apa")
> freq.matrix(perm = data.apa)
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.1835134 0.2647264 0.2288254 0.1746253 0.1483095
[2,] 0.1350645 0.1876961 0.2466016 0.2467759 0.1838620
[3,] 0.2804113 0.1673057 0.1382015 0.1829906 0.2310910
[4,] 0.2042524 0.1693970 0.1897874 0.2028581 0.2337051
[5,] 0.1967585 0.2108749 0.1965842 0.1927501 0.2030324
```

Another basic operation for permutations is inversion. The inverse of a permutation can be obtained using the function `inverse.permutation`.

```
> inverse.permutation(perm = sigma)

[1] 1 5 6 4 2 3
```

The argument `perm` can be a single permutation or a collection of permutations. It is worth noting that the inverse of the inverse of any permutation is itself.

```
> inverse.permutation(inverse.permutation(c(1, 4, 5, 3, 2)))

[1] 1 4 5 3 2
```

The present package also includes operators for the manipulation of permutations. The `swap` function, for example, swaps two prescribed items.

```
> swap(perm = identity.permutation(6), i = 1, j = 3)

[1] 3 2 1 4 5 6
```

The `insert.at` function takes an item from position i and inserts it after position j.

```
> insert.at(perm = identity.permutation(6), i = 5, j = 2)

[1] 1 2 5 3 4 6
```

An inversion at position $1 \leq i < n$ occurs when the items at positions i and $i + 1$ are swapped.

```
> inversion.at(perm = identity.permutation(6), 1)

[1] 2 1 3 4 5 6
```

Distances

We will now show how to deal with functions related with the distances between permutations included in this package. These functions include the argument `dist.name` which, unless otherwise stated, is one of the following: `kendall`, `cayley`, `hamming` or `ulam`. Moreover, its default value is `kendall`.

The following computes the distance between `perm1` and `perm2` for a given metric.

```
> pi

[1] 3 5 1 2 6 4

> sigma

[1] 1 5 6 4 2 3

> distance(perm1 = sigma, perm2 = pi, dist.name = "cayley")

[1] 4
```

The arguments `perm2` and `dist.name` can be omitted. In that case, `perm2` is assumed to be the identity, that means that the following code computes the Kendall's-τ distance between `pi` and e.

```
> distance(perm1 = pi)

[1] 6
```

As we have stated, the distance from a permutation to the identity can be decomposed in a vector. Clearly, this decomposition is different regarding the metric in question. In particular, the decomposition of the Kendall's-τ distance is a vector of $n-1$ integer terms, while for the Cayley distance it is a binary vector of $n-1$ terms and for the Hamming distance it is a binary vector of n terms. These decompositions are computed by the following function:

```
> sigma

[1] 1 5 6 4 2 3

> v.vector <- permutation2decomposition(perm = sigma,
+     dist.name = "kendall")
> v.vector

[1] 0 3 3 2 0

> x.vector <- permutation2decomposition(perm = sigma,
+     dist.name = "cayley")
> x.vector

[1] 0 1 1 0 0

> h.vector <- permutation2decomposition(perm = sigma,
+     dist.name = "hamming")
> h.vector

[1] 0 1 1 0 1 1
```

Given that there are possibly many longest increasing subsequences in a permutation, there is no decomposition of the Ulam distance. The possible values for the `dist.name` are therefore, `kendall`, `cayley` and `hamming` and being the default value `kendall`.

The `PerMallows` package can also perform the inverse operation, that is, given a decomposition vector and a metric, obtain a permutation consistent with the vector. Recall that for Cayley and Hamming there are possibly many permutations with a particular decomposition. In these situations, the following function recovers uniformly at random one of the permutations consistent with the decomposition vector.

```
> decomposition2permutation(vec = v.vector, dist.name = "kendall")

[1] 1 5 6 4 2 3

> decomposition2permutation(vec = x.vector, dist.name = "cayley")

[1] 1 5 6 4 2 3

> decomposition2permutation(vec = h.vector, dist.name = "hamming")

[1] 1 3 5 4 6 2
```

The `PerMallows` package implements a function to obtain the list of the cycles in which the permutation decomposes.

```
> cycles = permutation2cycles(perm = sigma)
```

The `cycle2str` function can be used in order to friendly display the cycles.

```
> cycle2str(cycles)

(1)(5 2)(6 3)(4)
```

Also, the inverse operation consisting of building a permutation given the list of cycles is supported.

```
> cycles2permutation(cycles = cycles)

[1] 1 5 6 4 2 3
```

`PerMallows` includes a function to count the number of permutations of n items at distance `dist.value` for a given distance.

```
> count.perms.distance(perm.length = 6, dist.value = 2,
+     dist.name = "ulam")

[1] 181
```

The most computationally expensive version of any function is usually that concerning the Ulam distance. In the case where one expects to work repeatedly with a model under the Ulam distance of a particular `perm.length` number of items (being `perm.length` large, say `perm.length`> 50), it is a good idea to generate auxiliary files including the count of permutations at each distance and later operating with those files. The operations that can be accelerated are counting and generating random permutations, learning and sampling. The files are generated with the following function:

```
> generate.aux.files(perm.length = 6)

[[1]]
[1] 6
```

The fastest version of the function to count the number of permutations at a given Ulam distance is the same as the one above but the optional parameter `disk` is set to `TRUE`. In this version, the data is read from the files instead of computing it.

```
> count.perms.distance(perm.length = 6, dist.value = 4,
+     dist.name = "ulam", disk = TRUE)

[1] 131
```

Regarding the process of counting permutations under the Cayley distance, we also include a function to count the number of permutations with n items and `num.cycles` cycles.

```
> count.perms.cycles(perm.length = 6, num.cycles = 4)

[1] 85
```

The following functions are related to the Hamming distance. Recall that an unfixed point is a position of the permutation σ such that $\sigma(i) \neq i$, while a fixed point is a position such that $\sigma(i) = i$. A derangement is a permutation with no fixed point. `PerMallows` includes functions to count the number of permutations of n items with exactly `fixed` fixed points, to count the number of permutations with at least `unfixed` unfixed points and to count the number of permutations with no fixed points.

```
> count.perms.fixed.points(perm.length = 6, fixed = 4)

[1] 15

> count.perms.unfixed.points.gtet(perm.length = 6, unfixed = 2)

[1] 504

> count.derangements(perm.length = 6)

[1] 265
```

The generation of random permutations at a prescribed distance is supported by the `r.dist.d` function, which generates `n` permutations of n items at distance `dist.value` for a particular metric.

```
> r.dist.d(n = 4, perm.length = 5, dist.value = 3,
+     dist.name = "ulam")

     [,1] [,2] [,3] [,4] [,5]
[1,]    3    5    4    1    2
[2,]    1    5    4    3    2
[3,]    4    3    2    5    1
[4,]    4    5    2    3    1
```

Regarding the Cayley distance, the `r.perms.cycles` function generates uniformly at random permutations of n items with `cycles` cycles.

```
> r.perms.cycles(n = 3, perm.length = 6, cycles = 5)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4    2    3    1    5    6
[2,]    3    2    1    4    5    6
[3,]    1    2    3    4    6    5
```

Similarly, `r.derangement` generates `n` permutations with no fixed point.

```
> r.derangement(n = 5, perm.length = 6)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    6    4    1    5    3    2
[2,]    5    4    2    6    3    1
[3,]    2    5    4    6    1    3
[4,]    4    6    2    5    3    1
[5,]    3    6    4    1    2    5
```

6.2 Distributions on permutations

In this section we show how to deal with the Mallows and Generalized Mallows models. We show functions for inference, learning and sampling both models.

Bear in mind that Mallows model (MM) can be used with every metric for permutation considered in this paper, Cayley, Kendall's-τ, Hamming and Ulam. On the other hand, the Generalized Mallows model (GMM) can only be used with Cayley, Kendall's-τ and Hamming[1]. Remember that for distributions on permutations on `perm.length` items the dispersion parameter vector $\boldsymbol{\theta}$ has `perm.length`-1 terms when the distance name is either Cayley or Kendall's-τ, and `perm.length` when the distance is Hamming.

[1] Although strictly speaking the GMM can not be used with the Hamming distance, we denote the similar Weighted Hamming Mallows model as GMM for the sake of clarity.

Parameter fitting

The estimation of the parameters of the MM and GMM is done with separate functions, `lmm` and `lgmm` respectively. Both need as arguments the sample to fit. The metric to use, an initial guess for the consensus permutation and the estimation methods are optional methods. The latter argument can be `approx` (by default) for the approximate learning and `exact` for the exhaustive one.

```
> data("data.3")
> my.mm <- lmm(sample = data.3,
+     sigma_0_ini = identity.permutation(6),
+     dist.name = "cayley", estimation = "exact")
> my.gmm <- lgmm(sample = data.3, sigma_0_ini = c(2, 1, 6, 5, 3,
+     4), dist.name = "cayley", estimation = "approx")
> my.mm
$mode
[1] 1 2 3 4 5 6

$theta
[1] 0.8364089

> my.gmm
$mode
[1] 1 2 3 4 5 6

$theta
[1] 1.4087672 0.2876821 0.6931472 1.0986123 0.6190392
```

If the auxiliary files to save the SYT in the disk have been generated with the function `generate.aux.files`, learning the parameters of the Ulam distance can be done by reading the files instead of computing the necessary data.

```
> lmm(sample = data.3, dist.name = "ulam", disk = TRUE)
$mode
[1] 2 5 1 3 4 6

$theta
[1] 0.8900056
```

Generating from the model

Although there are two separate functions for sampling MM and GMM (`rmm` and `rgmm`), they are used in a similar way. The required arguments for the MM (GMM) are the number of permutations to be generated, `n`, the parameters of the distribution, that is σ_0 and θ ($\boldsymbol{\theta}$), the name of the distance used and the sampling algorithm (`distances`, `multistage` or `gibbs`).

```
> rmm(n = 5, sigma0 = my.mm$mode, theta = my.mm$theta,
+     dist.name = "kendall", sampling.method = "distances")

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2    3    1    4    5    6
[2,]    1    2    3    4    5    6
[3,]    3    2    1    6    4    5
[4,]    1    3    2    4    6    5
[5,]    3    1    4    6    2    5

> rgmm(n = 5, sigma0 = my.gmm$mode, theta = my.gmm$theta,
+     dist.name = "kendall", sampling.method = "multistage")

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    2    3    6    5
[2,]    1    2    3    4    5    6
[3,]    2    1    3    5    4    6
[4,]    3    2    1    4    6    5
[5,]    2    1    3    4    5    6
```

If the auxiliary files have been generated with the function to store the SYT in disk with the `generate.aux.files` function, the distances sampling for the Ulam distance can rely on them. Again, the `disk` parameter must be set to `TRUE`.

```
> rmm(n = 3, sigma0 = identity.permutation(6),
+     theta = 1, dist.name = "ulam", disk = TRUE)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4    3    1    5    6    2
[2,]    1    2    5    4    6    3
[3,]    6    1    3    2    5    4
```

Probability

The probability of a permutation for a given MM (GMM) is calculated with the `dmm` (`dgmm`) function. The arguments of `dmm` (`dgmm`) are the permutation, the consensus permutation σ_0, dispersion parameter θ ($\boldsymbol{\theta}$) and the name of the distance used, which by default is Kendall's-τ.

```
> dmm(perm = my.mm$mode, sigma0 = my.mm$mode,
+     theta = my.mm$theta, dist.name = "ulam")

[1] 0.01228094

> dmm(perm = sigma, sigma0 = my.mm$mode,
+     theta = my.mm$theta, dist.name = "hamming")

[1] 0.00200106
```

```
> dgmm(perm = my.gmm$mode, sigma0 = my.gmm$mode,
+     theta = my.gmm$theta, dist.name = "kendall")

[1] 0.05945101

> dgmm(perm = sigma, sigma0 = my.gmm$mode, theta = my.gmm$theta,
+     dist.name = "kendall")

[1] 0.0003483458
```

In this way, the previous functions can be used to calculate the log-likelihood of a sample of permutations as follows.

```
> log.prob <- apply(data.3, MARGIN = 1, FUN = function(x) {
+     log(dmm(x, my.mm$mode, my.mm$theta, dist.name = "cayley"))
+ })
> sum(log.prob)

[1] -123.8265

> log.prob <- apply(data.3, MARGIN = 1, FUN = function(x) {
+     log(dgmm(x, my.gmm$mode, my.gmm$theta,
+     dist.name = "cayley")) })
> sum(log.prob)

[1] -122.1056
```

`PerMallows` includes the computation of the marginal distribution for both MM and GMM under the Hamming distance. The set of fixed and unfixed points is represented in the distance decomposition vector, so $H_j(\sigma) = 0$ means that j is a fixed point, $H_j(\sigma) = 1$ that j is an unfixed point and any other value of $H_j(\sigma)$ means that j is unknown. The following computes the marginal distribution of those permutations having fixed points at positions 2 and 4 and unfixed points at position 1.

```
> marginal(h = c(1, 0, -1, 0, -1), theta = c(1.1, 2, 1, 0.2, 0.4))

[1] 0.0808545
```

`PerMallows` includes functions to compute the expectation of the distance (resp. its decomposition vector) under the MM (resp. GMM) under the Hamming distance. The former computes the expectation of the distance under an MM of a given dispersion parameter and number of items. The latter, computes the expectation of the distance decomposition vector given a multidimensional dispersion parameter.

```
> expectation.mm(theta = 3, perm.length = 9, "ulam")

[1] 1.812466

> expectation.gmm(c(1.1, 2, 1, 0.2, 0.4), "cayley")

[1] 0.6246747 0.3512144 0.5246331 0.6208475 0.4013123 0.0000000
```

6.3 Conclusions

The `PerMallows` package includes the algorithms developed in this dissertation. The package has been illustrated in the present chapter, including a whole battery of functions, from those related to distributions on permutations to those related just to permutations and distances.

The learning and sampling functions are the main topic of this dissertation and, as such, they have received special attention. We provide easy-to-use functions for these complex functions which, by making the code freely accessible, can be modified and extended.

By packaging the most useful functions for permutations, we expect the package to be a useful starting point for the novice researchers on permutations in order to get familiar to permutations spaces, specially, those coming from other domains of the statistical field.

7

Conclusions

In this chapter we summarize the contents of this dissertation and highlight the main contributions. Afterwards, the publications derived from this thesis are enumerated. We conclude the chapter by exposing some future work lines.

Summary

This thesis is devoted to the distance-based probability models for permutations and extensions thereof, namely the Mallows (MM), Generalized Mallows (GMM) and Weighted Mallows (WMM) models. The distances considered have been the Kendall's-τ, Cayley, Hamming and Ulam. The combination of models and distances has lead to seven different models, namely (1) MM under Kendall's-τ, (2) GMM under Kendall's-τ, (3) MM under Cayley, (4) GMM under Cayley, (5) MM under Hamming, (6) WMM under Hamming and (7) MM under Ulam distance. Our main concern has been the development of efficient learning and sampling algorithms for the mentioned distributions.

Regarding the sampling algorithms, we have considered three different variants:

- **Gibbs sampler** This algorithm is very popular in the statistical community. It is a randomized algorithm that allows obtaining samples from an approximation of the distribution of interest. Gibbs has been used to sample from every model considered in this dissertation. It is, in every case, the quickest algorithm for the generation of a sample of fixed size. However, since it is an approximate algorithm, the samples generated have much lower quality than the rest of the algorithms.
- **Multistage/Chain sampler** These two algorithms fall into same category because, from an abstract point of view, both perform in a similar way: they both generate a permutation by first, randomly generating a distance decomposition vector and then, u.a.r. generating a permutation among those consistent with such decomposition vector. However, the strategies followed by each of them differ substantially. The Multistage sampler exploits the factorization of the models. It has been used

to sample from the MM and GMM under both Kendall's-τ and Cayley. The Chain sampler is based on the well known chain rule for probability distributions which expresses the joint distribution as a product of conditional distributions. It has been considered for the MM and WMM under the Hamming distance. Since it is not an approximate algorithm, it is very accurate. Moreover, it is stable for small sample sizes.

- **Distances sampler** This algorithm exploits the combinatorial nature of permutations to generate samples from the MM but can not generate from a GMM or WMM. The key idea is that a MM assigns equal probability value to permutations at equal distance. It is a valid algorithm for any right invariant distance such as the four considered in this dissertation. This algorithm generates a permutation from the MM relying on (i) counting the number of permutations at each distance and (ii) u.a.r. generating permutations at a given distance. The generated samples are accurate and the quality does not strongly depend on the sample size since it is not an approximate algorithm. The number of permutations at each distance quickly increases with n being unfordable to store the sequence, in general, for n larger than 150.

The learning process has been approached via maximum likelihood estimation of the parameters of the distributions for every case. The learning strategies fall into one of these two classes:

- **Exact learning** The exact learning for the parameters of the distribution of a given sample is considered for the MM and GMM under the Cayley distance and for the MM under the Hamming distance.
- **Approximate learning** With the exception of the MM under the Hamming distance, the problem of learning the exact MLE for the parameters of a given sample is either known or supposed to be a NP-complete problem. The problem of approximating the MLE for the parameters of a sample of permutations is carried out by, first, approximating the central permutation and secondly, calculating the dispersion parameter for the estimated distribution. The approximate learning is performed for every model considered in this dissertation with the exception of the MM under the Hamming distance.

Methodological contributions in a nutshell

The next lines try to clarify the contributions of this thesis. We follow the structure of this dissertation, which is organized regarding the distance for permutations considered in the model, to summarize the methodological contributions of this thesis.

- **Kendall's-τ** A method to u.a.r. generate permutations at a given distance -equivalently, with a given number of inversions- is introduced for the first time. The Distances sampler has been introduced. The conclusions drawn

from the experimental evaluation are the use of the Multistage algorithm for both MM and GMM. Moreover, a good performance of the approximate learning process is noted.

- **Cayley** A method to u.a.r. generate permutations at a given distance is introduced for the first time. Two methods to u.a.r. generate permutations consistent with a given distance decomposition vector are introduced. The relations of the MM and GMM under the Cayley distance with other models in the literature are studied. We also give the expression for the expected distance decomposition vector under both MM and GMM.
  Regarding the sampling process, we have introduced the Gibbs, Multistage and Distances sampling algorithms described above. In view of the experimental results, we recommend the use of Distances to sample from a MM and the Multistage to sample from a GMM.
  The exact learning for the MM and GMM, which follows a branch and bound strategy, has been run with samples of permutations of 25 items. For situations where the exact algorithm is too slow (larger permutations or for almost uniform samples) we encourage the use of the approximate algorithm we propose.
- **Hamming** This is the first time that a model analogous to the GMM is used, the WMM. It is shown how to efficiently compute the normalization constant. Moreover, it can be adapted to compute the joint and the conditional probabilities of both MM and WMM. An efficient formula to compute the expected distance and distance decomposition vector under both MM and WMM is introduced. All these computations are based on the efficient computation of the elementary symmetric polynomial of a set of variables and their derivatives.
  The Gibbs, Chain and Distances sampling algorithms described above are introduced for the generation from the MM and WMM. An optimized implementation of the Chain algorithm results on a very efficient algorithm in terms of time. The experimental results show that the Distances sampler provides the best trade-off between time and accuracy for sampling the MM while the Chain is our recommendation for generating from the WMM.
  A polynomial time algorithm to recover the MLE for the parameters of a sample of permutations from a MM is introduced. For the WMM, we introduce an algorithm for the approximate MLE for the parameters. We prove the approximate consensus permutation to be an asymptotically unbiased estimator of the real one. In view of the results, we conclude that the approximate algorithm is as accurate as well as quick alternative.
- **Ulam** We introduce an algorithm based on the standard young tableaux to operate with the combinatorial and statistical problems related to the Ulam distance. This problems include counting and u.a.r. generating permutations at a given distance as well as the efficient computation of the normalization constant and the expression of the expected distance, both for the MM.

The sampling algorithms include the Gibbs and Distances samplers. We suggest the use of the latter despite its complexity due to an impressive gain of performance with regard to the approximate Gibbs sampler.
The learning process, which is carried out in an approximate way, is shown to produce accurate parameters.

All the contributions have been coded in `c++` for the maximum efficiency. For the sake of usability, all the functions considered have been packaged and an `R` interface has been included. In order to be a complete and useful framework for dealing with permutations and distance-based probability models the common operations are also implemented. The resulting package can be downloaded from `http://cran.r-project.org/web/packages/PerMallows/index.html`.

7.1 Publications of this thesis

In this section, we present the publications and submissions produced during this thesis. First we present the publications directly derived from the thesis and then we present the collaborations with other researchers relative to this thesis.

- Irurozki, E., Calvo, B. & Lozano, J. A. (2014). An R package for permutations, Mallows and Generalized Mallows models. Submitted to *Journal of Statistical Software*, July, 2014.
- Irurozki, E., Calvo, B. & Lozano, J. A. (2014). Sampling and learning the Mallows and Generalized Mallows models under the Cayley distance. Submitted to *Methodology and Computing in Applied Probability*, march 2014.
- Irurozki, E., Calvo, B. & Lozano, J. A. (2014). Sampling and learning the Mallows and Weighted Mallows models under the Hamming distance. Submitted to *Journal of Statistical Planning and Inference*, July 2014
- Irurozki, E., Calvo, B. & Lozano, J. A. (2014). Sampling and learning the Mallows model under the Ulam distance. Technical Report.

The next contribution belongs to the context of learning distributions on permutation spaces. However, the approach taken is a completely different to those in this thesis and it has not been included in the present dissertation for the sake of consistency.

- Irurozki, E., Calvo, B. & Lozano, J. A (2011). Learning Probability Distributions over Permutations by Means of Fourier Coefficients. In Butz, Cory, Lingras & Pawan (editors), Advances in Artificial Intelligence. Springer Berlin / Heidelberg.

Some of the methods introduced in this thesis have been applied in the optimization domain. As a result, the following collaborations have been published.

- Ceberio, J., Irurozki, E., Mendiburu, A. & Lozano, J. A (2014). Extending Distance-based Ranking Models in Estimation of Distribution Algorithms. In In Proceedings of the 2014 IEEE Congress on Evolutionary Computation.
- Ceberio, J., Irurozki, E., Mendiburu, A. & Lozano, J. A. (2014). A Distance-based Ranking Model Estimation of Distribution Algorithm for the Flowshop Scheduling Problem. IEEE Transactions on Evolutionary Computation, 18(2).
- Ceberio, J., Irurozki, E., Mendiburu, A. & Lozano, J. A. (2012). A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Progress in Artificial Intelligence, 1(1), 103-117.

7.2 Future work

In this section we describe the future research directions of this thesis. Besides the direct application of any of the models to real data for analysis, we consider two different interesting approaches. The first consists on adapting the models for data similar to permutations. The second one considers extensions of the probability models.

Extending the data

In this thesis we have restricted, with few exceptions, to complete permutation data. However, we consider the research lines that result of extending the permutation data to the following data:

- Partial permutations: a subset of the items have been placed in the permutations. However, for the remaining positions there is no information. It is sometimes seen as the set of permutations that contains every permutation in which the unknown positions are set with the unset items in every possible way.
- Multi-permutations: a permutation in which repetitions are allowed. This structure considers a tupple of length m of an alphabet of $n < m$ symbols so the symbols are repeated. Distances, basic for the distance-based models, have been defined for multi-permutations [24] and they are claimed to be fundamental in areas such as Cryptanalysis [110].
- Permutations with ties: in the preference domain it is very common to have preference information among groups of items but not inside the groups. For example, it is very popular to rate objects, such as films in a film database, with stars, 1 to a bad film, 5 to a good one. If a user has rated film A with 4 stars and films B and C with 2 stars, implies a preference of film A over B. However, no preference relation among films B and C can be derived. This preference relation for groups of items is usually modeled with a tied permutation.

- Signed permutation group: a natural of extension of the Symmetric group in which the items have one of two possible signs, positive or negative. A signed permutation is an ordered set of the first n natural numbers such that each number can be positive or negative. Distances have been defined for signed permutations [11] and also it is possible to find references on adaptations of distance-based models to signed permutations [7].

Extending the models

When modeling a vast population it is reasonable to assume that it is not homogeneous but rather consisting on a set of homogeneous sub-populations. As we stated, the distance-based models are considered the analogous to the Gaussian and, as such, it can be the basis for more complex models such as mixtures of Mallows models and their extensions. Mixture models can therefore led to multi-modal distributions that can accurately represent real permutation datasets. Examples in the ranking domain of mixtures can be found in the literature [89], [78]. Moreover, an interesting approach is to mix models considering different distances, similar to than in [78]. In this context, the proposed sampling and learning methods can be of great interest.

Kernels are a highly interesting tool which has already been used in the ranking domain [76], [83]. One of the most problematic aspects of kernels is the underlying algebra. Our concern of the efficient sampling and learning algorithms may be of great utility for the kernel context.

We consider that a fruitful field can come from the combination of classification with permutation data which generalizes the ranking interpretation. In this way, we can use a specific distance based model for specific domains.

Classification in the preference scenario has given raise to a discipline called preference learning. The distance-based models have been considered in this scenario as the probability model over the permutation data. The distances used have been considered are those natural in the ranking domain, i.e. Kendall-s-τ, Spearman's-ρ and Spearman's footrule.

References

1. A. D. Gordon. A measure of agreement between rankings. *Biometrika*, 66(1):7–15, 1979.
2. Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. Parimutuel betting on permutations. In *Internet and Network Economics*, volume 5385, pages 126–137, Shanghai, China, December 17-20, 2008, 2008.
3. David Aldous and Persi Diaconis. Longest increasing subsequences: from parience sorting to the Baik-Deift-Johansson Theorem. *Society*, 36(4):413–432, 1999.
4. Alnur Ali and Marina Meila. Experiments with Kemeny ranking: What works when? *Mathematical Social Sciences*, 64(1):28–40, 2012.
5. Alexandr Andoni and Huy L Nguyen. Near-Optimal Sublinear Time Algorithms for Ulam Distance. In Moses Charikar, editor, *Symposium on Discrete Algorithms, SODA*, pages 76–86. SIAM, 2010.
6. Jörg Arndt. *Generating Random Permutations*. PhD thesis, Australian National University, 2010.
7. Raman Arora and Marina Meila. Consensus Ranking with Signed Permutations. In *AISTATS*, volume 31 of *JMLR Proceedings*, pages 117–125. JMLR.org, 2013.
8. Richard Arratia, A D Barbour, and Simon Tavaré. *Logarithmic combinatorial structures: a probabilistic approach*. EMS Monographs in Mathematics. European Mathematical Society (EMS), Zürich, 2003.
9. Richard Arratia and Simon Tavare. The Cycle Structure of Random Permutations. *The Annals of Probability*, 20(3):1567–1591, 1992.
10. B Babington Smith. Discussion on professor Ross's paper. *Journal of the Royal Statistical Society1*, 12:153–162, 1950.
11. David A Bader, Bernard M E Moret, and Mi Yan. A Linear-Time Algorithm for Computing Inversion Distance between Signed Permutations with an Experimental Study. In *Algorithms and Data Structures Symposium (WADS)*, volume 2125 of *Lecture Notes in Computer Science*, pages 365–376. Springer, 2001.
12. Martin Bader. The transposition median problem is NP-complete. *Theoretical Computer Science*, 412(12-14):1099–1110, 2011.

13. Jinho Baik, Percy Deift, and Kurt Johansson. On the distribution of the length of the longest increasing subsequence of random permutations. *Journal of the American Mathematical Society*, 12(4):1119–1178, 1999.
14. Frank B Baker and Michael R Harwell. Computing Elementary Symmetric Functions and Their Derivatives: A Didactic. *Applied Psychological Measurement*, 20(2):169–192, 1996.
15. Jason Bandlow. An Elementary Proof of the Hook Formula. *The Electronic Journal of Combinatorics*, 15(1), 2008.
16. 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(2):157–165, April 1989.
17. Paul Beame, Eric Blais, and Dang-Trinh Huynh-Ngoc. Longest Common Subsequences in Sets of Permutations. page 9, April 2009.
18. M Beckman and T C Koopmans. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.
19. Nadja Betzler, Jiong Guo, Christian Komusiewicz, and Rolf Niedermeier. Average parameterization and partial kernelization for computing medians. *Journal of computer and system science*, 77(4):774–789, 2011.
20. David M Blei and Peter Frazier. Distance dependent Chinese restaurant processes. In Johannes Fürnkranz and Thorsten Joachims, editors, *International Conference on Machine Learning (ICML)*, pages 87–94. Omnipress, 2010.
21. David M Blei, Thomas L Griffiths, and Michael I Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM*, 57(2), 2010.
22. Miklós Bóna. *Combinatorics of permutations*. Discrete mathematics and its applications. Chapman & Hall/CRC Press, Boca Raton, FL, London, 2004.
23. J. Borda. *Memoire sur les Elections au Scrutin.* Histoire de l'Academie Royal des Sciences., 1781.
24. Sarit Buzaglo, Eitan Yaakobi, Tuvi Etzion, and Jehoshua Bruck. Systematic Codes for Rank Modulation. November 2013.
25. Ioannis Caragiannis, Ariel D Procaccia, and Nisarg Shah. When Do Noisy Votes Reveal the Truth? In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC '13, pages 143–160, New York, NY, USA, 2013. ACM.
26. Josu Ceberio, Ekhine Irurozki, Alexander Mendiburu, and Jose A Lozano. A Review of Distances for the Mallows and Generalized Mallows Estimation of Distribution Algorithms. *Journal of Computational Optimization and Applications*, 2014.
27. Josu Ceberio, Ekhine Irurozki, Alexander Mendiburu, and Jose A Lozano. Extending Distance-based Ranking Models in Estimation of Distribution Algorithms. In *IEEE Congress on Evolutionary Computation*, page Accepted for publication, 2014.
28. Josu Ceberio, Alexander Mendiburu, and Jose A Lozano. Introducing The Mallows Model on Estimation of Distribution Algorithms. In *International Conference on Neural Information Processing (ICONIP)*, number 23-25, Shanghai, 2011.
29. Moses Charikar and Robert Krauthgamer. Embedding the Ulam metric into l1. *Theory of Computing*, 2(1):207–224, 2006.

30. Weiwei Cheng and Eyke Hüllermeier. A New Instance-Based Label Ranking Approach Using the Mallows Model. In *Advances in Neural Networks (ISNN)*, volume 5551 of *Lecture Notes in Computer Science*, pages 707–716. Springer Berlin Heidelberg, 2009.
31. Weiwei Cheng and Eyke Hullermeier. A Simple Instance-Based Approach to Multilabel Classification Using the Mallows Model. In *Workshop Proceedings of Learning from Multi-Label Data*, pages 28–38, Bled, Slovenia, 2009.
32. Douglas Edward Critchlow, Michael A Fligner, and Joseph S Verducci. Probability Models on Rankings. *Journal of Mathematical Psychology*, 35:294–318, 1991.
33. Villo Csiszár. Conditional independence relations and log-linear models for random matchings. *Acta Mathematica Hungarica*, 122(1-2):131–152, July 2008.
34. Villo Csiszár. On L-decomposability of random orderings. *Journal of Mathematical Psychology*, 53(4):294–297, 2009.
35. C de la Higuera and F Casacuberta. Topology of strings: median string is NP-complete. *Theoretical Computer Science*, 230(1-2):39–48, 2000.
36. Michael Deza and Tayuan Huang. Metrics on permutations, a survey. *Journal of Combinatorics, Information and System Sciences*, 23:173–185, 1998.
37. Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2009.
38. Persi Diaconis. *Group representations in probability and statistics*. Institute of Mathematical Statistics, 1988.
39. Persi Diaconis. A Generalization of Spectral Analysis with Application to Ranked Data. *The Annals of Statistics*, 17(3):949–979, 1989.
40. Persi Diaconis. The Markov chain Monte Carlo revolution. *Bulletin of the American Mathematical Society*, 46(2):179–205, November 2008.
41. Persi Diaconis and R L Graham. Spearman's Footrule as a Measure of Disarray. 1976.
42. Persi Diaconis and Susan Holmes. A Bayesian Peek into Feller Volume I. *Sankhya: The Indian Journal of Statistics, Series A (1961-2002)*, 64(3), 2002.
43. Jean-Paul Doignon. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika*, 69(1):33–54, 2004.
44. Peter Donnelly. Partition structures, Polya urns, the Ewens sampling formula, and the ages of alleles. *Theoretical Population Biology*, 30(2):271–288, 1986.
45. Nicholas M Ercolani and Daniel Ueltschi. Cycle structure of random permutations with cycle weights. pages 1–21, 2011.
46. Warren J Ewens. The sampling theory of selectively neutral alleles. *Theoretical Population Biology*, 3(1):87–112, 1972.
47. Farzad Farnoud, Vitaly Skachek, and Olgica Milenkovic. Error-Correction in Flash Memories via Codes in the Ulam Metric. *IEEE Transactions on Information Theory*, 59(5):3003–3020, 2013.
48. William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968.
49. Valentin Féray. Asymptotics of some statistics in Ewens random permutations. *Electronic Journal of Probability*, 18(76), 2013.
50. Marcelo Fiori, Pablo Sprechmann, Joshua T Vogelstein, Pablo Musé, and Guillermo Sapiro. Robust Multimodal Graph Matching: Sparse Coding Meets Graph Matching. In *Advances in Neural Information Processing Systems (NIPS)*, pages 127–135, 2013.

51. Michael A Fligner and Joseph S Verducci. Distance based ranking models. *Journal of the Royal Statistical Society*, 48(3):359–369, 1986.
52. Michael A Fligner and Joseph S Verducci. Multistage Ranking Models. *Journal of the American Statistical Association*, 83(403):892, 1988.
53. Michael A Fligner and Joseph S Verducci. *Probability Models and Statistical Analyses for Ranking Data*. Springer, 1993.
54. Deborah S Franzblau and Doron Zeilberger. A Bijective Proof of the Hook-Length Formula. *Journal of Algorithms*, 3(4):317–343, 1982.
55. Johannes Furnkranz and Eyke Hullermeier. Preference Learning and Ranking [Special Issue]. *Machine Learning*, 93(2-3), 2013.
56. Ira M Gessel and Christophe Reutenauer. Counting permutations with given cycle structure and descent set. *Journal of Combinatorial Theory, Series A*, 64(2):189–215, 1993.
57. Kenneth Glass and Chi-Keung Ng. A Simple Proof of the Hook Length Formula. *The American Mathematical Monthly*, 111(8):700–704, 2004.
58. Curtis Greene, Albert Nijenhuis, and Herbert S. Wilf. A Probabilistic Proof of a Formula for the Number of Young Tableaux of a Given Shape. *Advances in Mathematics*, 31:104–109, 1979.
59. John Guiver and Edward Snelson. Bayesian inference for Plackett-Luce ranking models. In *International Conference on Machine Learning (ICML)*, ICML '09, pages 377–384. ACM, 2009.
60. Jayanti Gupta and Paul Damien. Conjugacy class prior distributions on metric-based ranking models. *Journal of the Royal Statistical Society Series B*, 64(3):433–445, 2002.
61. Godfrey Harold Hardy and Srinivasa Aiyangar Ramanujan. Asymptotic formulae in combinatory analysis. *Journal London Mathematical Society*, 17:75–115, 1918.
62. Farzad Farnoud Hassanzadeh and Olgica Milenkovic. Multipermutation Codes in the Ulam Metric for Nonvolatile Memories. *IEEE Journal on Selected Areas in Communications*, 32(5):919–932, 2014.
63. D P Helmbold and M K Warmuth. Learning Permutations with Exponential Weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.
64. Leticia Hernando, Alexander Mendiburu, and Jose A Lozano. Generating Customized Landscapes in Permutation-based Combinatorial Optimization Problems. In *Learning and Intelligent OptimizatioN Conference (LION 7)*, Catania, Italy, 2013.
65. Jonathan Huang. Fourier Theoretic Probabilistic Inference over Permutations. *Journal of Machine Learning Research*, 10:997–1070, 2009.
66. Eyke Hüllermeier and Johannes Fürnkranz. On predictive accuracy and risk minimization in pairwise label ranking. *Journal of Computer and System Sciences*, 76(1):49–62, 2010.
67. David R Hunter. MM Algorithms for Generalized Bradley-Terry Models. *The Annals of Statistics*, 32(1):384–406, 2004.
68. Ekhine Irurozki, Borja Calvo, and Jose A Lozano. Learning Probability Distributions over Permutations by Means of Fourier Coefficients. In *Canadian Conference on AI*, volume 6657 of *Lecture Notes in Computer Science*, pages 186–191. Springer, 2011.
69. Ekhine Irurozki, Borja Calvo, and Jose A Lozano. Sampling and learning the Mallows and Generalized Mallows models under the Cayley distance. Technical report, University of the Basque Country, 2014.

70. Ekhine Irurozki, Borja Calvo, and Jose A Lozano. Sampling and learning the Mallows and Weighted Mallows models under the Hamming distance. Technical report, University of the Basque Country, 2014.
71. Ekhine Irurozki, Borja Calvo, and Jose A Lozano. Sampling and learning the Mallows model under the Ulam distance. Technical report, University of the Basque Country, 2014.
72. R Jonker and A Volgenant. A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems. *The Netherlands Computing*, 38(4):325–340, 1987.
73. Paul Kidwell, Guy Lebanon, and William S Cleveland. Visualizing incomplete and partially ranked data. *IEEE transactions on visualization and computer graphics*, 14(6):1356–63, 2008.
74. Alexandre Klementiev, Dan Roth, and Kevin Small. Unsupervised Rank Aggregation with Distance-Based Models. *Learning*, pages 472–479, 2008.
75. Donald E Knuth. Permutations, matrices and generalized Young tableaux. *Pacific Journal of Mathematics*, 34:709–727, 1970.
76. Risi Kondor. Ranking with kernels in Fourier space. *Technology*, (1), 2008.
77. Pedro Larrañaga and Jose A Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.
78. Paul H Lee and Philip L H Yu. Mixtures of weighted distance-based models for ranking data with applications in political studies. *Computational Statistics & Data Analysis*, 56(8):2486–2500, 2012.
79. Tyler Lu and Craig Boutilier. Learning Mallows Models with Pairwise Preferences. *Learning*, 2011.
80. Duncan Luce R. *Individual Choice Behavior*. Wiley, New York, 1959.
81. C L Mallows. Non-null ranking models. *Biometrika*, 44(1-2):114–130, 1957.
82. Bhushan Mandhani and Marina Meila. Tractable Search for Learning Exponential Models of Rankings. *Journal of Machine Learning Research*, 5:392–399, 2009.
83. Yi Mao and Guy Lebanon. Non-Parametric Modeling of Partially Ranked Data. *Journal of Machine Learning Research*, 9:2401–2429, 2008.
84. John I Marden. *Analyzing and Modeling Rank Data*. Chapman & Hall, 1995.
85. Albert Maydeu-Olivares. Limited information estimation and testing of Thurstonian models for paired comparison data under multiple judgment sampling. *Psychometrika*, 66(2):209–227, 2001.
86. Alberto Maydeu-Olivares. Thurstonian modeling of ranking data via mean and covariance structure analysis. *Psychometrika*, 64(3):325–340, 1999.
87. Alberto Maydeu-Olivares. Limited information estimation and testing of Thurstonian models for preference data. *Mathematical Social Sciences*, 43(3):467–483, 2002.
88. Peter McCullagh. Random Permutations and Partition Models. In Miodrag Lovric, editor, *International Encyclopedia of Statistical Science*, pages 1170–1177. Springer Berlin Heidelberg, 2011.
89. Marina Meila and Harr Chen. Dirichlet Process Mixtures of Generalized Mallows Models. In *Uncertainty in Artificial Intelligence (UAI)*, pages 285–294, 2010.
90. Marina Meila, Kapil Phadnis, Arthur Patterson, and Julian Bilmes. Consensus ranking under the exponential model. In *Uncertainty in Artificial Intelligence (UAI)*, pages 285–294, Corvallis, Oregon, 2007.

91. N Mladenović and P Hansen. Variable neighborhood search. *Comput. Oper. Res.*, 24(11):1097–1100, 1997.
92. Carl Mueller and Shannon Starr. The Length of the Longest Increasing Subsequence of a Random Mallows Permutation. *Journal of Theoretical Probability*, 26(2):514–540, 2013.
93. Thomas Brendan Murphy and Donal Martin. Mixtures of distance-based models for ranking data. *Computational Statistics & Data Analysis*, 41(3âĂŞ4):645–655, 2003.
94. Jean-Christophe Novelli, Igor Pak, and Alexander V Stoyanovskii. A direct bijective proof of the hook-length formula. *Discrete Mathematics & Theoretical Computer Science*, 1(1):53–67, 1997.
95. J Olivares-Rodríguez C.; Oncina. A Stochastic Approach to Median String Computation. *Lecture Notes in Computer Science*, 5342:431–440, 2008.
96. Deepti Pachauri, Risi Kondor, and Vikas Singh. Solving the multi-way matching problem by permutation synchronization. In Christopher J C Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1860–1868, 2013.
97. James Petterson, Tibério S Caetano, Julian John McAuley, and Jin Yu. Exponential Family Graph Matching and Ranking. *Advances in Neural Information Processing Systems 22*, abs/0904.2:1455—-1463, 2009.
98. R L Plackett. The Analysis of Permutations. *Journal of the Royal Statistical Society*, 24(10):193–202, 1975.
99. Vladimir Popov. Multiple genome rearrangement by swaps and by element duplications. *Theoretical computer science*, 385(1-3):115–126, 2007.
100. William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.
101. H Rademacher. *On the Partition Function P(n)*. Hodgson, 1937.
102. Michel Regenwetter, A.A.J. Marley, and H. Joe. Random Utility threshold models for subset choice. *Australian Jourrul of Psychology*, 50(3):175–185, 1998.
103. Gilbert de Beauregard Robinson. On the representation of the symmetric group. *American Journal of Mathematics*, 60(3):745–760, 1938.
104. Michael Saks and C Seshadhri. Estimating the Longest Increasing Sequence in Polylogarithmic Time. In *Foundations of Computer Science, FOCS*, pages 458–467. IEEE Computer Society, 2010.
105. Craige Schensted. Longest increasing and decreasing subsequences. *Canadian Journal of Mathematics*, 13:179–191, 1961.
106. Tommaso Schiavinotto and Thomas Stützle. A Review of Metrics on Permutations for Search Landscape Analysis. *Computers & OR*, 34:3143–3153, 2007.
107. R P Stanley. *Enumerative Combinatorics*. Wadsworth Publ. Co., Belmont, CA, USA, 1986.
108. Simon Tavaré and Warren J Ewens. *Multivariate Ewens distribution*, chapter 41, pages 232–246. Wiley, 1997.
109. L L Thurstone. A law of comparative judgment. *Psychological Review*, 34(4):273–286, 1927.
110. Serge Vaudenay. On the need for multipermutations: Cryptanalysis of MD4 and SAFER. In Bart Preneel, editor, *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science*, pages 286–297. Springer Berlin Heidelberg, 1995.

111. Maksims Volkovs and Richard S Zemel. Efficient Sampling for Bipartite Matching Problems. In Peter L Bartlett, Fernando C N Pereira, Christopher J C Burges, Léon Bottou, and Kilian Q Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1322–1330, 2012.
112. Grace Yao and Ulf Böckenholt. Bayesian estimation of Thurstonian ranking models based on the Gibbs sampler. *British Journal of Mathematical and Statistical Psychology*, 52(1):79–92, 1999.