L. Hernando, A. Mendiburu and J. A. Lozano, "A Tunable Generator of Instances of Permutation-Based Combinatorial Optimization Problems," in **IEEE Transactions on Evolutionary Computation**, vol. 20, no. 2, pp. 165-179, April 2016, https://doi.org/10.1109/TEVC.2015.2433680

A Tunable Generator of Instances of Permutation-based Combinatorial Optimization Problems

Leticia Hernando[1], Alexander Mendiburu[2], and Jose A. Lozano[1]

[1]Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, 20018 San Sebastián, Spain
[2]Intelligent Systems Group, Department of Computer Architecture and Technology, University of the Basque Country UPV/EHU, 20018 San Sebastián, Spain

Abstract

In this paper we propose a tunable generator of instances of permutation-based Combinatorial Optimization Problems. Our approach is based on a probabilistic model for permutations, called the Generalized Mallows model. The generator depends on a set of parameters that permits the control of the properties of the output instances. Specifically, in order to create an instance, we solve a linear programing problem in the parameters, where the restrictions allow the instance to have a fixed number of local optima and the linear function encompasses qualitative characteristics of the instance. We exemplify the use of the generator by giving three distinct linear functions that produce three landscapes with different qualitative properties. After that, our generator is tested in two different ways. Firstly, we test the flexibility of the model by producing instances similar to benchmark instances. Secondly, we account for the capacity of the generator to create different types of instances according to the difficulty for population-based algorithms. We study the influence of the input parameters in the behavior of these algorithms, giving an example of a property that can be used to analyze their performance.

Keywords— C ombinatorial Optimization Problems, instance generator, Generalized Mallows model, permutation space, local optima.

1 Introduction

In the optimization arena, the evaluation of algorithms is usually measured by means of benchmark problems. However, algorithms show different behaviors

for different problems, or even for different instances of the same problem. So commonly, when trying to study their performance, assumptions are needed to be made on the algorithm itself, the problem to which it is applied or the specific instance of the problem. Thus, having information about the characteristics of the problem instances at hand would be really useful for improving the design of algorithms or to identify the best performing algorithm from a toolbox. Due to the difficulty of having real instances whose properties we know a priori at our disposal, generators are designed in order to provide a large set of instances with different characteristics. Therefore, a tunable generator of optimization problem instances that depends on a reduced number of parameters able to control the properties of the instance is an important and useful tool for this purpose.

In spite of the works comparing and proposing algorithms to solve optimization problems, the portfolio of models that generate customized optimization problem instances is not so extensive. Most of the proposals found in the literature for this topic are for the continuous domain. In [1] a software framework that generates multimodal test functions for optimization in the continuous field was presented. In [2, 3, 4, 5] the authors proposed a continuous search space generator based on a mixture of Gaussians. A proposal in the discrete domain, and particularly for binary spaces, can be found in [6], where a NK landscape generator is described. We also find a proposal of a generator of instances in the dynamics optimization field: the moving peaks benchmark [7].

Particularly, in the space of permutations, we find generators of instances [8, 9, 10]. Unfortunately, these generators lack the flexibility to generate instances with controlled properties. Most of them are based on populating a cost matrix (depending on the problem they focus on) by taking random values, or taking random points in a square and calculating the distances between each pair of points. The complexity of the instances varies according to the interval (values and variances) used during the random sampling. On the other hand, as pointed out in [10], there are proposals in the literature that create instances where the optimum is known. Nevertheless, this is the only information provided, and there is no clue about how easy or difficult it is to solve the instance. In this same work, the authors state that the toughest technical challenges are finding (or generating) suitable test instances, and assessing how close heuristic algorithms come to the optimum.

Related to this, we present a generator for permutation-based Combinatorial Optimization Problems (COPs). Our work is inspired by [2], where the authors proposed a generator of optimization problem instances in the continuous domain. We adapt the concepts that they used to those that are similar but in the permutation space. For example, instead of working with Gaussian density functions we work with Generalized Mallows distributions. This distribution is an exponential probability model that depends on a consensus permutation and spread parameters (analogous to the mean and the variance in the Gaussian function). In a previous work [11], we presented a very preliminary version of our model using the Mallows distribution. Here, we study the influence of the parameters that the generator uses when creating the instances. We also

provide some clues on how to tune them with the aim of controlling the properties of the instances. We pay special attention to a particular property, the number of local optima, and set restrictions in the parameters so as to have a predefined number of them in the resultant instance. In order to obtain a solution for these restrictions in the parameters, the generator is seen as a linear programming problem, and a linear function is defined that promotes obtaining qualitative properties in the generated instance. In our framework, the cost of generating the instances, as well as the cost of evaluating each solution of the search space, is dominated by the number of local optima. In order to test our generator, we take into account properties that generators of problem instances in optimization should have when the resultant instances are used to evaluate Evolutionary Algorithms [2], [12], [13], [14]. Specifically, we carry out experiments to evaluate two important properties: the flexibility of the generator, and its ability to create instances of very different complexity levels for common metaheuristics. To test the first property, we adjust different parameters of our generator. We considered it interesting to check the ability of the generator to provide, for small problem sizes, instances almost identical to those found in well-known benchmarks. We compare the instances by means of the sizes of the attraction basins of the local optima. To evaluate the second property of the generator, we generate instances playing with the size of the attraction basins for the global and local optima, in addition to varying the location of the different local optima. Then, three different algorithms, a local search, an estimation of distribution algorithms, and a genetic algorithm are applied to observe their behavior when solving the different types of instances. According to the results, the generator arises as a very useful tool to perform coarse as well as fine grain analysis of optimization algorithms, as the practitioner can observe the algorithms under controlled scenarios (instances). Derived from the experiments, an interesting and novelty property in the performance of the EDA and the GA is observed.

The rest of the paper is organized as follows. In Section 2 we explain the Mallows and the Generalized Mallows models, which are the basis of our generator. In Section 3 the most common distances used for the Generalized Mallows model are detailed. Our generator is introduced in Section 4 and additional details are provided in Section 5. In Section 6 we present three examples of how to tune the parameters involved in the model to create instances with different properties. The experimentation is shown in Section 7, firstly, testing the flexibility of the model by comparing created instances with instances of common benchmarks, and secondly, comparing and analyzing properties in the performance of different metaheuristics when applying them to instances created with our generator with different input parameters. Finally, the conclusions are presented in Section 8.

2 Mallows and Generalized Mallows models

2.1 Mallows model

The Mallows model [15] is an exponential probability model for permutations based on a distance. We denote by Ω the set of permutations of size n, and define a permutation $\sigma \in \Omega$ as a bijection of the set of integers $\{1, \ldots, n\}$ onto itself. A permutation is understood as an order of the items $\{1, 2, \ldots, n\}$, this is: $\sigma = (\sigma(1)\sigma(2)\cdots\sigma(n))$, where $\sigma(i) \in \{1, 2, \ldots, n\}$ is the item in the i-th position and $\sigma(i) \neq \sigma(j), \forall i \neq j$. A special permutation which is worth mentioning is the identity permutation, $e = (123\cdots n)$ which maps each item i to position i. By composing two permutations σ and π of n items, we obtain a new permutation $\sigma\pi$ such that $\sigma\pi(j) = \sigma(\pi(j))$. Specifically, the inverse σ^{-1} of σ is the permutation such that $\sigma\sigma^{-1} = e$. Note that the composition between a permutation and the identity is the same permutation: $\sigma e = e\sigma = \sigma$.

The Mallows distribution is specified by two parameters: the consensus permutation $\sigma_0 \in \Omega$ and the spread parameter $\theta \in \mathbb{R}$. Hence, the probability assigned to each $\sigma \in \Omega$ is:

$$p(\sigma|\sigma_0, \theta) = \frac{1}{Z(\theta)} e^{-\theta d(\sigma, \sigma_0)}$$

where $Z(\theta) = \sum_{\sigma' \in \Omega} e^{-\theta d(\sigma', \sigma_0)}$ is a normalization term and $d(\sigma, \sigma_0)$ is a distance between σ and the consensus permutation σ_0.

Different values for the parameter θ produce different distributions. For instance, when $\theta = 0$ it is the uniform distribution. However, when $\theta > 0$, then σ_0 is the permutation with the highest probability. The rest of permutations $\sigma \in \Omega - \{\sigma_0\}$ have probability inversely exponentially proportional to θ and their distance to σ_0. In this sense, the Mallows distribution with $\theta > 0$ is usually considered analogous to the Gaussian distribution on the space of permutations. In Figure 1 we represent the probability assigned to the permutations of size $n = 5$ for different θ values considering the Kendall-tau metric (this distance is introduced in Section 3.1). In the X axis we represent the permutations symmetrically distributed according to their distance to the consensus permutation σ_0, in order to see the analogy with the Gaussian distribution. We observe that the larger the value of θ, the more peaked the distribution becomes around the consensus permutation.

2.2 Generalized Mallows model

The Mallows model is a simple yet efficient probability model for permutations. However, the fact that it uses just a single spread parameter limits its flexibility. In this sense, it assigns the same probability to all the permutations that are at the same distance from the consensus permutation. That is, for any distance d:

$$\forall \sigma \neq \sigma' \in \Omega, \text{ s.t. } ?d(\sigma, \sigma_0) = d(\sigma', \sigma_0), \text{ then } p(\sigma) = p(\sigma').$$

In [16], an extension of this model was proposed, called the Generalized Mallows (GM) model. For this model, we need to work with a distance which

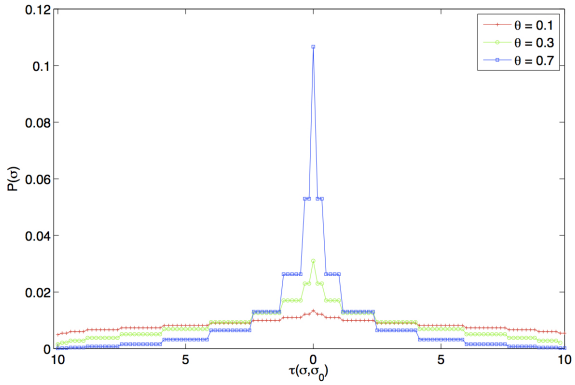

Figure 1: Probability assigned to permutations by three Mallows functions centered at σ_0 and with $\theta = 0.1, 0.3$ and 0.7. In the X axis the permutations are distributed symmetrically according to their distance to σ_0.

is able to be decomposed $d(\sigma_1, \sigma_2) = \sum_{s=1}^{n-1} d_s(\sigma_1, \sigma_2)$, in such a way that each d_s is related with the s-th item (position) of the permutation. Therefore, a different spread parameter $\theta^s \in \mathbb{R}$ can be associated to each of the elements d_s that is involved in the decomposition of the distance, so that instead of using one spread parameter, a vector of them is defined. In this distribution, the probability assigned to any permutation $\sigma \in \Omega$ is:

$$p(\sigma|\sigma_0, \theta^1, \ldots, \theta^{n-1}) = \frac{1}{Z(\theta^1, \ldots, \theta^{n-1})} e^{-\sum_{s=1}^{n-1} \theta^s d_s(\sigma, \sigma_0)},$$

where $Z(\theta^1, \ldots, \theta^{n-1}) = \sum_{\sigma' \in \Omega} e^{-\sum_{s=1}^{n-1} \theta^s d_s(\sigma', \sigma_0)}$.

Notice that, the Mallows model can be seen as a particular case of the GM model, where $\theta^s = \theta, \forall s$. So, from now on, we will refer to the GM model taking the Mallows model as a specific case of it, and we will consider that $\theta^s \in \mathbb{R}^+, \forall s$.

3 Distances

As seen in the previous section, the GM model makes use of a distance or metric. The metrics utilized in this paper are the Kendall-tau and the Cayley distances, as they are the most commonly jointly used with this probabilistic model [17, 18, 19, 20, 21].

We define a distance or metric between two permutations σ_1 and σ_2 [22] as a function

$$\begin{array}{rccc} d: & \Omega \text{x} \Omega & \longrightarrow & \mathbb{R} \\ & (\sigma_1, \sigma_2) & \longmapsto & d(\sigma_1, \sigma_2) \end{array}$$

that fulfills the following properties $\forall \sigma_1, \sigma_2, \sigma_3 \in \Omega$:

1. Non-negativity: $d(\sigma_1, \sigma_2) \geq 0$ (with equality iff $\sigma_1 = \sigma_2$).
2. Symmetry: $d(\sigma_1, \sigma_2) = d(\sigma_2, \sigma_1)$.
3. Triangle inequality: $d(\sigma_1, \sigma_2) \leq d(\sigma_1, \sigma_3) + d(\sigma_3, \sigma_2)$.

Specifically, the two metrics considered in this paper also fulfill the left-invariant property:

$$d(\sigma_1, \sigma_2) = d(\sigma_3\sigma_1, \sigma_3\sigma_2).$$

3.1 Kendall-tau distance

The most commonly used distance in the GM model is the Kendall-tau [17], [23], [24]. This metric measures the minimum number of adjacent swaps between two permutations. Formally, given two permutations σ_1 and σ_2, we write this metric as:

$$d_K(\sigma_1, \sigma_2) = \sum_{r \prec_{\sigma_1} s} \mathbb{1}_{[s \prec_{\sigma_2} r]} \tag{1}$$

where $r \prec_\sigma s$ means that the item r precedes s in the permutation σ, and $\mathbb{1}_{[\cdot]}$ denotes the indicator function. As it fulfills the left-invariant property, we can write indistinctly $d_K(\sigma_1, \sigma_2)$ or $d_K(\sigma_2^{-1}\sigma_1, e)$. The Kendall-tau distance can be decomposed into the following form:

$$d_K(\sigma_1, \sigma_2) = d_K(\sigma_2^{-1}\sigma_1, e) = \sum_{s=1}^{n-1} V_s(\sigma_2^{-1}\sigma_1, e) \tag{2}$$

where $V_s(\sigma, e)$ $(s = 1, 2, ..., n-1)$ is the number of items $r > s$ that precede the item s in σ, that is:

$$V_s(\sigma, e) = \sum_{r > s} \mathbb{1}_{[r \prec_\sigma s]}.$$

Notice, that we define each $V_s(\sigma, e)$ with respect to the identity permutation, so that when decomposing the Kendall-tau distance between two permutations σ_1 and σ_2, we just need to consider the distance between the composition $\sigma_2^{-1}\sigma_1$ and e. From now, we simplify the notation by writing $V_s(\sigma)$ instead of $V_s(\sigma, e)$.

For example, lets suppose that we have two permutations of size $n = 4$: $\sigma_1 = (4132)$ and $\sigma_2 = (1342)$. Thus, $\sigma_2^{-1} = (1423)$ and $\sigma_2^{-1}\sigma_1 = (3124)$. Therefore, $d_K(\sigma_1, \sigma_2) = d_K(\sigma_2^{-1}\sigma_1, e) = \sum_{s=1}^{n-1} V_s(\sigma_2^{-1}\sigma_1) = 1 + 1 + 0 = 2$.

As the terms V_s (for $s = 1, 2, ..., n-1$) are bounded: $0 \leq V_s \leq (n - s)$, the maximum distance between two permutations that can be reached by this metric is $(n-1) + (n-2) + ... + 1 = \frac{n(n-1)}{2}$.

When using the Kendall-tau metric in the GM model, we consider a different spread parameter θ^s associated to each V_s. So that the model depends on a

consensus permutation $\sigma_0 \in \Omega$ and the spread parameters $\boldsymbol{\theta} = (\theta^1, \cdots, \theta^{n-1}) \in (\mathbb{R}^+)^{n-1}$. The normalization term $Z(\boldsymbol{\theta}) = Z(\theta^1, \ldots, \theta^{n-1})$ has the following closed form that does not depend on σ_0:

$$Z(\boldsymbol{\theta}) = \prod_{s=1}^{n-1} \left[\frac{1 - e^{-(n-s+1)\theta^s}}{1 - e^{-\theta^s}} \right].$$

Therefore the probability assigned to each permutation $\sigma \in \Omega$ can be written as:

$$p(\sigma|\sigma_0, \boldsymbol{\theta}) = \prod_{s=1}^{n-1} \left[\frac{1 - e^{-\theta^s}}{1 - e^{-(n-s+1)\theta^s}} \right] e^{-\sum_{s=1}^{n-1} \theta^s V_s(\sigma_0^{-1}\sigma)}.$$

3.2 Cayley distance

The Cayley distance $d_C(\sigma_1, \sigma_2)$ between two permutations σ_1 and σ_2 measures the minimum number of swaps (not necessarily adjacent) that are needed to transform σ_1 into σ_2. A concept closely related to the Cayley distance is the number of cycles in the permutations. A cycle is a subset $\{i_1, i_2, ..., i_r\}$ of the set of the items of the permutation such that

$$\sigma(i_1) = i_2 \ , \ \sigma(i_2) = i_3 \ , \ \ldots \ , \ \sigma(i_r) = i_1.$$

As previously mentioned, when we calculate $d_C(\sigma, e)$ we should count the number of swaps to transform σ into e. Note that, in this case, every swap involves two items of the same cycle. So, the minimum number of swaps needed to sort the r items of the same cycle is $r - 1$. This means that the Cayley distance between σ and e is n minus the number of cycles.

Taking this definition into account, we can decompose the Cayley distance between two permutations σ_1 and σ_2:

$$d_C(\sigma_1, \sigma_2) = d_C(\sigma_2^{-1}\sigma_1, e) = \sum_{s=1}^{n-1} X_s(\sigma_2^{-1}\sigma_1, e) \tag{3}$$

where

$$X_s(\sigma, e) = \begin{cases} 0 & \text{if } s \text{ is the largest item in its cycle in } \sigma \\ \\ 1 & \text{otherwise} \end{cases}$$

In this case, we also define $X_s(\sigma, e)$ with respect to the identity permutation e. So, similarly to the case of the Kendall-tau distance, we also simplify the notation using $X_s(\sigma)$.

For instance, considering again $\sigma_1 = (4132)$ and $\sigma_2 = (1342)$, we have $d_C(\sigma_1, \sigma_2) = d_C(\sigma_2^{-1}\sigma_1, e) = \sum_{s=1}^{n-1} X_s(\sigma_2^{-1}\sigma_1) = 1 + 1 + 0 = 2$.

The minimum number of cycles in a permutation is 1. This means that the maximum distance between two permutations that can be given by this metric is $n - 1$.

The GM distribution uses a different spread parameter θ^s for each X_s seen in (3). Considering the vector of spread parameters $\boldsymbol{\theta} = (\theta^1, \cdots, \theta^{n-1}) \in (\mathbb{R}^+)^{n-1}$, a closed form for the normalization term $Z(\boldsymbol{\theta})$ is also found [19]:

$$Z(\boldsymbol{\theta}) = \prod_{s=1}^{n-1}(1 + (n-s)e^{-\theta^s}).$$

Finally, the probability assigned to each $\sigma \in \Omega$ under the Cayley metric is defined by:

$$p(\sigma|\sigma_0, \boldsymbol{\theta}) = \frac{1}{\prod_{s=1}^{n-1}(1 + (n-s)e^{-\theta^s})} e^{-\sum_{s=1}^{n-1}\theta^s X_s(\sigma_0^{-1}\sigma)}.$$

4 The generator of instances: Overview

In [2] the authors proposed a generator of instances of optimization problems in the continuous domain based on a mixture of Gaussian distributions where each local optimum of the landscape corresponds with the mean of a Gaussian distribution. The generated instances are such that the function value of a solution is defined by the maximum value that any of the Gaussian components associates to that solution. Inspired by that work, we present a generator of instances of permutation-based COPs. The generator is founded on a mixture of GM distributions.

The general idea of our approach to generate an instance is to choose m consensus permutations $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, and m spread vectors of parameters $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m\}$ to create m GM distributions with them:

$$\begin{gathered} p_1(\sigma|\sigma_1, \boldsymbol{\theta}_1) \\ p_2(\sigma|\sigma_2, \boldsymbol{\theta}_2) \\ \vdots \\ p_m(\sigma|\sigma_m, \boldsymbol{\theta}_m) \end{gathered}$$

where

$$p_i(\sigma|\sigma_i, \boldsymbol{\theta}_i) = p_i(\sigma|\sigma_i, \theta_i^1, \theta_i^2, ..., \theta_i^{n-1}) = \frac{e^{-\sum_{s=1}^{n-1}\theta_i^s d_s(\sigma,\sigma_i)}}{Z(\boldsymbol{\theta}_i)}.$$

We also consider weights $\{w_1, w_2, \ldots, w_m\}$, $(w_i > 0, \forall i)$ associated to each of the GM distributions. From now on, we will assume that we generate instances of maximization problems. In this sense, the objective function value of a solution (permutation) of the instance is defined as the maximum value reached by all the GM distributions multiplied by the corresponding associated weight:

$$f(\sigma) = \max_{1 \leq i \leq m} \{w_i p_i(\sigma|\sigma_i, \theta_i^1, \theta_i^2, \ldots, \theta_i^{n-1})\}. \quad (4)$$

Thus, this model has $(n+1)m$ parameters: $(n-1)m$ spread parameters, m consensus permutations, and m weights. Notice that we need to evaluate m GM models in order to assign an objective function value to each solution of the search space.

5 The generator of instances: Particularities

Our aim is to provide a generator able to create instances with different properties. In order to do that, we should tune the parameters of our generator appropriately. This is done by following two complementary tasks; one quantitative and one qualitative. In the quantitative approach we add constraints to the parameters in order to have a predefined number of local optima. These constraints are linear in the weights of the GM components that define the instance. The qualitative approach consists of the definition of a linear function on the weights that tries to promote certain characteristics of the instance such as the relative size of the attraction basin of the global optimum versus the local optima. Different linear functions can create different types of instances.

As a result of this process, each instance of the generator is created by solving a linear programming problem in the weights.

5.1 Controlling the local optima in the instance

In this section, we study the constraints we should add in order to ensure that all the consensus permutations of the GM models are the local optima in the generated instance. These local optima are defined when considering a neighborhood determined by the solutions at distance one. Logically, the particular distance used coincides with that used in the GM models (Kendall-tau or Cayley).

Notice that by our definition of an instance, no other permutation of the search space, apart from the consensus permutations, can be a local optimum (local maximum). This means that with our generator we know that the number of local optima is always smaller or equal to the number m of GM components. However, it is possible that a consensus permutation is buried by a GM component centered at any other permutation, and thus, this permutation will not be a local optimum. Hence, the key point is to ensure that all the consensus permutations are, indeed, local optima for our instance. In this way, we will be able to create instances with exactly m local optima.

In what follows, we will assume that it is not possible to find two local optima with the same objective function value. Therefore, the distance between two local optima needs to be higher than one, so the consensus permutations need to satisfy:

$$d(\sigma_i, \sigma_j) \geq 2, \forall i \neq j. \tag{5}$$

We remark that, although we work under this assumption, the generator would be able to face with a situation in which $f(\sigma_i) = f(\sigma_j)$, $i \neq j$.

By the definition of local maximum, the following constraint has to be fulfilled for σ_i to be a local optimum:

$$f(\sigma_i) > f(\sigma), \quad \forall \sigma \in \Omega \quad s.t. \quad d(\sigma, \sigma_i) = 1. \tag{6}$$

The following Lemma 5.1 shows a necessary condition for a consensus permutation to be a local optimum. In Theorem 5.2 we give the necessary and sufficient condition in the parameters to guarantee that all consensus permutations are local optima in the instance. The proofs of Lemma 5.1 and Theorem 5.2 are given in the appendices 9 and 10, respectively.

Lemma 5.1. *Let's consider an instance I created with our generator and let σ_i be the consensus permutation of the i-th GM model used in it. If σ_i is a local optimum in the generated instance, that is:*

$$f(\sigma_i) > f(\sigma), \quad \forall \sigma \in \Omega \quad \textit{such that} \quad d(\sigma, \sigma_i) = 1,$$

then the objective function value of σ_i is reached by the i-th GM model, i.e.:

$$\begin{aligned} f(\sigma_i) &= \max_{1 \leq j \leq m} \{w_j p_j(\sigma_i | \sigma_j, \boldsymbol{\theta}_j)\} = w_i p_i(\sigma_i | \sigma_i, \boldsymbol{\theta}_i) \\ &= w_i \frac{e^{-\sum_{s=1}^{n-1} \theta_i^s d_s(\sigma_i, \sigma_i)}}{Z(\boldsymbol{\theta}_i)} = \frac{w_i}{Z(\boldsymbol{\theta}_i)} \end{aligned} \tag{7}$$

Theorem 5.2. *A consensus permutation σ_i of any of the GM models involved in our generator is a local optimum in the generated instance if and only if:*

$$\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma, \sigma_j)}, \forall j \neq i, \tag{8}$$

$\forall \sigma$ *such that* $d(\sigma, \sigma_i) = 1$.

In conclusion, if the parameters fulfill the restrictions in (8), we know that all the consensus permutations of the GM models of our generator are local optima in the instance. By fixing the values of θ_j^s in the inequalities in (8), the restrictions are linear in the weights, and therefore they can be solved with a linear programing problem. However, we can find inconveniences when working with the restrictions in (8). On the one hand, we could have numerical problems. For example, for high permutation sizes n, the values $e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma, \sigma_j)}$ will be close to 0 (as the distances could be considerably high), and, due to precision limits, when handling such values in the resolution of the linear programming problem, the results could be stored as 0.00. Therefore, the desired restrictions are not fulfilled and finally, the consensus permutations are not local optima in the instance. On the other hand, the number of constraints needed to be fulfilled is considerably high: $m(m-1)(n-1)$ and $m(m-1)n(n-1)/2$, in the case of the Kendall-tau and the Cayley distances, respectively.

With the aim of avoiding the numerical instability of (8) and in order to reduce the number of restrictions, we show in the following Theorem 5.3 sufficient conditions to fulfill (8). In fact, we find a significant decrease in the number of restrictions, as they are a total of m, independently of the distance used. This reduction in the number of restrictions does not imply, however, limitations in the flexibility of the generator to create different types of instances. The proof of the Theorem 5.3 is shown in the appendix 11.

Theorem 5.3. *Let $\{w_1, \ldots, w_m\}$, $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m\}$ and $\{\sigma_1, \ldots, \sigma_m\}$ be the weight values, the spread parameters and the consensus permutations, respectively, of the m GM components used in our generator to create an instance I. If the following constraints are fulfilled:*

$$\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_{i+1}}{Z(\boldsymbol{\theta}_{i+1})}, \quad i = 1, \ldots, m-1 \tag{9}$$

$$\left[2 - e^{-\left(\min_{j,s}\{\theta_j^s\}\right)}\right] \frac{w_m}{Z(\boldsymbol{\theta}_m)} > \frac{w_1}{Z(\boldsymbol{\theta}_1)} \tag{10}$$

then, the restrictions in (8) are satisfied.

In summary, if restrictions (9) and (10) are fulfilled, we ensure that the consensus permutations of the GM distributions are the local optima of the instance when considering the neighborhood defined by the distance used in the probabilistic models. Note that, by the Lemma 5.1, these constraints also imply that the objective function value of a consensus permutation is reached by the GM component centered at itself: $f(\sigma_i) = \frac{w_i}{Z(\boldsymbol{\theta}_i)}$. Thus, the restrictions in (9) mean that: $f(\sigma_i) > f(\sigma_{i+1})$. So, σ_1 is the local optimum with the highest objective function value, that is, the global optimum. Although the restrictions in (9) impose an order in the local optima regarding their objective function value, this can be assumed without loss of generality. That is, one can previously sort the consensus permutations in the desired order.

5.2 The linear programming problem

Once the number of local optima has been established, in order to customize the properties of an instance, we use a linear function in the weights associated to the GM models. This function carries out qualitative properties of the instance. In the following section we use three different functions to exemplify three different kinds of landscapes regarding the sizes of the attraction basins of the local optima. Of course, the careful choice of the location of the different local optima (consensus permutations), as well as the choice of the spread parameters, help in the generation of the instances with the requested properties.

In general terms, the generator of instances of COPs based on permutations can be described as a 5-tuple $(n, m, \Sigma, \Theta, G)$, where:

- $n \in \mathbb{N}$ is the size of the permutation.

- $m \in \mathbb{N}$ is the number of GM functions (number of local optima of the instance).
- $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ is the set of consensus permutations of size n.
- $\Theta = \begin{pmatrix} \theta_1^1 & \cdots & \theta_1^{n-1} \\ \cdots & \cdots & \cdots \\ \theta_m^1 & \cdots & \theta_m^{n-1} \end{pmatrix} \in (\mathbb{R}^+)^{m \times (n-1)}$
  is the matrix with the spread parameters of the m GM distributions.
- G is the linear objective function of the linear programming problem.

See Algorithm 1 to observe the steps that the generator follows. Notice that in the linear programming problem (step 5 in the algorithm), we add the constraint $w_m > 0$, just to force the weights to be positive: if $w_m > 0$, then $w_i > 0, \forall i$, as (9) implies $\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_m}{Z(\boldsymbol{\theta}_m)}$. We also force $w_1 < k \in \mathbb{R}^+$, so as to upper bound the values of the weights. Moreover, as we previously remarked, if the user desires to work with two local optima σ_i and σ_{i+1} with equal fitness function values, inequality $\frac{w_i}{Z(\boldsymbol{\theta}_i)} = \frac{w_{i+1}}{Z(\boldsymbol{\theta}_{i+1})}$ is needed to be used here, instead of $\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_{i+1}}{Z(\boldsymbol{\theta}_{i+1})}$.

We would like to remark that the $\boldsymbol{\theta}$ values should be chosen taking into account the type of distance used and the permutation size n, in order to avoid numerical problems when calculating the normalization terms $Z(\boldsymbol{\theta})$. We suggest using these values in the intervals $[ln(\frac{n-1}{2}), 3ln(n-1)]$ and $[2ln(\frac{n-1}{3}), 6ln(n-1)]$, for the Kendall-tau and the Cayley distances, respectively.

The source code (in R-project) of the generator is available in the website[1].

5.3 Complexity of the generator

We can distinguish two kinds of complexity associated with the generator: (i) the computational complexity of generating a particular instance, and (ii) the cost of evaluating the objective function value of a solution in the generated instance.

The first one is dominated by the solution of the linear programming problem. In the literature, we can find powerful algorithms whose cost, in the worst case, depends on the number of variables, which in our problem is m: $\{w_1, \ldots, w_m\}$.

Given an output instance of the generator, when assigning the objective function value to any solution of the search space (ii), we should evaluate in the worst case m components $w_i p_i(\sigma|\sigma_i, \boldsymbol{\theta}_i)$ in order to look for the highest value. Each of these evaluations implies the calculation of the $n-1$ terms of the decomposition of the distance (equalities (2) and (3)). In the case of the Kendall-tau distance, the cost of calculating these terms is $O(n^2)$, whereas

[1] http://www.sc.ehu.es/ccwbayes/members/leticia/GeneratorOfInstances/code.html

Algorithm 1 Algorithm to generate instances of permutation-based Combinatorial Optimization Problems

1. Set n
2. Set m
3. Choose the consensus permutations: $\sigma_1, \sigma_2, ..., \sigma_m$, such that: $d(\sigma_i, \sigma_j) \geq 2, \forall i \neq j$.
4. Choose the spread parameters:

$$\Theta = \begin{pmatrix} \theta_1^1 & \cdots & \theta_1^{n-1} \\ \cdots & \cdots & \cdots \\ \theta_m^1 & \cdots & \theta_m^{n-1} \end{pmatrix}$$

5. Solve the linear programming problem in the weights $\{w_1, \ldots, w_m\}$:

$$min/max\{G(w_1, ..., w_m)\}$$

subject to

$$\begin{aligned} \frac{w_i}{Z(\boldsymbol{\theta}_i)} &> \frac{w_{i+1}}{Z(\boldsymbol{\theta}_{i+1})}, \quad i = 1, \ldots, m-1 \\ \frac{w_1}{Z(\boldsymbol{\theta}_1)} &< \left[2 - e^{-\left(\min\limits_{i,s}\{\theta_i^s\}\right)}\right] \frac{w_m}{Z(\boldsymbol{\theta}_m)} \\ w_m &> 0 \\ w_1 &< k \quad (k \in \mathbb{R}^+) \end{aligned}$$

6. $\forall \sigma \in \Omega$ define the objective function value as:

$$f(\sigma) = \max_{1 \leq i \leq m} \left\{ \frac{w_i}{Z(\boldsymbol{\theta}_i)} e^{-\sum\limits_{s=1}^{n-1} \theta_i^s d_s(\sigma, \sigma_i)} \right\}$$

for the Cayley distance it is $O(n)$. Therefore, in the worst case, the cost in the evaluation of each solution is $O(mn^2)$ and $O(mn)$ for the Kendall-tau and the Cayley distances, respectively. Commonly, in real instances of COPs, we find that the number of local optima is considerably higher than the size of the permutations [25]. So that we will be willing to generate instances with $m >> n$. Thus, the cost of evaluating the solutions is principally conditioned by the number of GM components used in the generator.

6 Creating Instances Using our Generator

In this section we present three examples of how to generate instances with different properties. These three types of instances are associated with three linear functions and three careful choices of the σ_i and θ_i^s, $\forall i, s$. Basically, we design instances for three different scenarios.

In the first scenario, we try to generate easy to optimize instances. In this sense, our goal is to make the attraction basin of the global optimum σ_1 as large as possible. Our second scenario contemplates difficult instances and the function we optimize tries to make the attraction basin of the global optimum as small as possible. Our last setting generates instances in the middle of the two previous examples, where we attempt to have all the attraction basins of the local optima (including the global optimum) of similar sizes. Obviously, these are just some illustrative examples but, by properly choosing the linear function and tuning the rest of parameters, the generator would be able to provide other kinds of instances.

6.1 Easy instances: Global optimum with a large attraction basin

In this example, we tune the parameters with the aim of obtaining an instance with a large attraction basin for the global optimum. As this scenario describes a general situation, we do not refer to any specific distance. We assume that $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ are the consensus permutations in our generator, so they are the local optima in the instance, where σ_1 is the global optimum.

First, we choose the linear function to optimize in the linear programming problem. In this scenario, we propose maximizing the difference between the objective function value of σ_1 (global optimum) and the objective function value of σ_2 (local optimum with the highest value after σ_1):

$$G_{\mathrm{M}axGO} = max\left\{\frac{w_1}{Z(\boldsymbol{\theta}_1)} - \frac{w_2}{Z(\boldsymbol{\theta}_2)}\right\}.$$

By means of this linear function, we promote that those permutations σ that are close to σ_1 will receive an objective function value given by the first GM component, i.e.: $f(\sigma) = \max_{1\leq i\leq m}\{w_i p_i(\sigma|\sigma_i, \boldsymbol{\theta}_i)\} = w_1 p_1(\sigma|\sigma_1, \boldsymbol{\theta}_1)$. Thus, with the help of the rest of the parameters, we can easily force as many solutions as possible to be in the attraction basin of σ_1.

The local optima can be chosen paying special attention to the distances between them. For example, it is recommendable to choose σ_1 far from the rest of the local optima, in order to contribute to obtaining a large attraction basin for it. In doing this, we provoke a large number of solutions appear closer to σ_1 than to the rest of local optima.

The parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_m$ are also essential when trying to control the properties of the generated instance. In order to provide an easy and intuitive example, let's assume that we are in the situation $\theta_i^j = \theta_i^k = \theta_i, \forall j \neq k$. So, we are considering the Mallows model. Under this assumption, we are assigning the same objective function value to all the solutions that are at the same distance from the consensus permutation. This situation gives us intuition about how to choose the spread parameters in order to model the different attraction basins of the local optima. If we want the global optimum to have a large attraction basin, we should choose $\theta_1 << \theta_i$, $i \geq 2$, because, as we saw in Section 2.1, the larger the value of θ the more peaked the Mallows function. Hence, with a small θ those solutions close to σ_1 have an objective function value similar to σ_1, and therefore the possibilities of having them in the attraction basin of the global optimum are very high.

6.2 Difficult instances: Global optimum with a small attraction basin

We propose, in this scenario, to minimize the difference between the objective function value of the global optimum σ_1 and its neighbors. When looking for the objective function value of the neighbors we need to calculate the values of the elements involved in the decomposition of the distance between those neighbors and all the local optima of the instance: $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$. However, this calculation would imply a high computational cost. Therefore, we can simplify this idea taking into account the following function to minimize:

$$G_{\mathrm{M}inGO} = min \left\{ \frac{w_1}{Z(\boldsymbol{\theta}_1)} - \frac{1}{m-1} \sum_{i=2}^{m} \frac{w_i}{Z(\boldsymbol{\theta}_i)} e^{-\max_j \{\theta_i^j\}(d(\sigma_1, \sigma_i)+1)} \right\}.$$

Notice that the second element in the subtraction is the average of lower bounds for the values assigned to those neighbor solutions given by the GM components centered at the local optima which are different from the global optimum. Thus, it is a lower bound for the objective function values of the neighbors of σ_1, and therefore, minimizing this difference, we would be minimizing the desired difference. The aim of choosing this linear function is to help the neighbor solutions of σ_1 to have their objective function value assigned by the GM component centered at other local optimum different to σ_1, and thus, to belong to a different attraction basin.

For this scenario, σ_1 can be chosen as a consensus permutation that is close

to, at least, one local optimum. For example, this local optimum could be σ_2, i.e., the local optimum with the highest objective function value after σ_1. If σ_2 is near σ_1, there are less possibilities of having a large number of solutions in the attraction basin of σ_1, because we can provoke (also taking into account the rest of parameters) that the solutions belong to the attraction basin of σ_2.

As we explained in the example above, choosing $\theta_i^j = \theta_i^k$, $\forall j \neq k$ we provide a more intuitive example. So, under this context, by denoting θ_i as the spread parameter of the i-th Mallows model ($\theta_i = \theta_i^1 = \ldots = \theta_i^{n-1}$), the larger the value of θ_1 in comparison to the rest of θ_i, $i > 1$, the smaller the size of the attraction basin of the global optimum. Moreover, as an example of a more difficult scenario, we try to obtain the size of the attraction basin of σ_2 as large as possible by choosing $\theta_2 << \theta_i$ ($i \neq 2$).

6.3 Local optima with similar sizes of attraction basins

In this last example, we are interested in creating instances where all the local optima (included the global optimum) have attraction basins similar in size.

Assuming that σ_1 is the global optimum, and σ_m is the local optimum with the lowest objective function value, the considered function to minimize in this case, is the difference between the objective function values of these two permutations (and implicitly, minimize the difference between the objective function values of all the local optima):

$$G_{\mathrm{S}imAB} = \min \left\{ \frac{w_1}{Z(\boldsymbol{\theta}_1)} - \frac{w_m}{Z(\boldsymbol{\theta}_m)} \right\}.$$

In this sense, the landscape created with this linear function is flatter than in the other two examples.

Taking into account that we want to obtain local optima with attraction basins of similar sizes, these solutions should be uniformly distributed in the search space. So that we could choose them in order for them to be in a situation as similar to the following one as possible: $d(\sigma_i, \sigma_j) \approx d, \forall i, j$, where d is the largest possible value.

The values of all θ_i^j should be similar for all the elements of the decomposition of the distance, and also similar for all the GM components. That is, with $\theta_i^s = \theta$, $\forall i, s$, the Mallows models centered at all the consensus permutations have the same shape. This implies that we do not force any of the local optima to have more solutions in their attraction basins than the rest of the local optima.

7 Experiments

A generator of instances of combinatorial optimization problems is a useful tool when analyzing, comparing, and evaluating the behavior of different optimization algorithms. In this section, we show the influence of the input parameters in the resultant instance, and present several use cases of our generator with the aim of demonstrating its importance and applicability.

First, we want to show that the generator is highly flexible. For this purpose, we find it interesting to check if we are able to generate instances similar to those available in well-known benchmarks. Thus, the first part of the experimentation is devoted to this goal, working with small problem sizes. Secondly, we demonstrate that, by tuning the input parameters, the user can create different instances, placing emphasis on coarse grain characteristics (such as the attraction basin sizes of the local optima) as well as on fine grain characteristics (distribution of the local optima). This is a highlighting characteristic of our generator, as it allows to conduct a deeper analysis of the behavior of optimization algorithms.

7.1 Flexibility of the generator

In order to create an artificial instance similar to an instance that can be found in well-known benchmarks (from now on we will refer to them as benchmark instances), we carry out the following steps:

i) Calculate, for the benchmark instance, the local optima and the sizes of their attraction basins.

ii) Use these values to fix three input parameters of our generator: n (permutation size), m (number of local optima) and Σ (set of consensus permutations).

iii) Choose one of the linear functions to optimize from those defined in Section 6 (G_{MaxGO}, G_{MinGO} and G_{SimAB}), taking into account the characteristics of the benchmark instance. Of course, the generator allows the user to define other linear functions.

iv) Carry out a search over the space of Θ parameters to minimize the difference between the artificial instance and the benchmark instance. Note that each time a set of Θ parameters is tested, the linear programming problem needs to be solved.

A key point in the previous process is to define a way to measure the similarity between the artificial and the benchmark instance (step iv). For example, one option could be to compare the objective function values of the solutions, trying to create an artificial instance with the same absolute objective function values as the given benchmark instance. However, most of the local and population-based algorithms only use the relative value of the solutions instead of the exact function values. Therefore, we could rank the solutions according to their objective function value in the artificial and the benchmark instances, comparing both rankings. Nevertheless, considering that we generate the instances with locality criteria in mind, we have used the difference between the attraction basin sizes of the artificial and the benchmark instances as the similarity measure.

We generate instances with properties similar to those of two well known COPs: the Permutation Flowshop Scheduling Problem and the Linear Ordering

Problem. The Flowshop Scheduling Problem can be stated as follows: there are b jobs to be scheduled in c machines. A job consists of c operations and the j-th operation of each job must be processed on machine j for a specific processing time without interruption. We consider that the jobs are processed in the same order on different machines. The objective of the PFSP is to find the order in which the jobs have to be scheduled on the machines, minimizing the total flow time. In the Linear Ordering Problem (LOP), given a matrix $B = [b_{ij}]_{n \times n}$ of numerical entries, we have to find a simultaneous permutation σ of the rows and columns of B, such that the sum of the entries above the main diagonal is maximized (or equivalently, the sum of the entries below the main diagonal is minimized).

We work with 5 instances of the PFSP obtained from the well-known benchmark proposed by Taillard[2], and 5 instances of the LOP, obtained from the xLOLIB benchmark [26]. The size of the original instances has been reduced to $n = 8$, that is, in the instances of the PFSP, we consider 8 jobs and 5 machines, and in the LOP instances, the size of the matrices is 8x8. The instances used are available in the website[3]. The reason for choosing a small permutation size is to keep, for each candidate Θ (step iv), the cost of calculating the attraction basin sizes of the local optima affordable.

We consider two different neighborhoods: 2-exchange (N_{2-exch}) and adjacent swap (N_{Adj}). These neighborhoods can be defined using the Cayley and the Kendall-tau distances, respectively. Particularly, the 2-exchange neighborhood considers that two solutions are neighbors if they differ from one swap (Cayley distance is one), whereas they are neighbor solutions under the adjacent swap neighborhood if they differ from one adjacent swap (Kendall-tau distance is one). Using a deterministic greedy local search algorithm, we start from each solution of the search space with the aim of finding the number of local optima and their attraction basin sizes. We denote by $\mathcal{B}(\sigma_i)$ the attraction basin of σ_i in the benchmark instance, while $\hat{\mathcal{B}}(\sigma_i)$ refers to the attraction basin of σ_i in the artificial instance created with our generator. The error ϵ_i is, therefore, $\epsilon_i = ||\mathcal{B}(\sigma_i)| - |\hat{\mathcal{B}}(\sigma_i)||$. We remark that, when the local optima of a benchmark instance are calculated using N_{2-exch} (N_{adj}), the artificial instance is generated using the Cayley distance (Kendall-tau metric) in the GM models.

In order to choose the linear function to optimize in the linear programming problem (step iii), we take into account the proportion of the attraction basin size of the global optimum with respect to the size of the search space: $\frac{|\mathcal{B}(\sigma_1)|}{|\Omega|}$. According to preliminary experiments, we choose in our generator one of the objective functions: $G_{\mathrm{M}axGO}$, $G_{\mathrm{M}inGO}$ and $G_{\mathrm{S}imAB}$ (explained in Section 6), if such a proportion is higher than $\frac{0.6}{m}$, lower than $\frac{0.4}{m}$, or between these two values, respectively.

In Tables 1 and 2 we provide, for PFSP and LOP respectively, the differences between the sizes of the attraction basins of the local optima of each benchmark

[2]http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir /ordonnancement.html

[3]http://www.sc.ehu.es/ccwbayes/members/leticia/GeneratorOfInstances /Instances.html

Table 1: Results obtained for the errors in the attraction basin sizes of the local optima for the 5 instances of the PFSP, for the Cayley and the Kendall-tau distances.

	Inst	m	$\frac{\epsilon_1}{\lvert\Omega\rvert}$	$\bar{\epsilon} = \frac{1}{m}\sum_{i=1}^{m}\frac{\epsilon_i}{2\lvert\Omega\rvert}$	$\frac{1}{m-1}\sum_{i=1}^{m}\left(\frac{\epsilon_i}{2\lvert\Omega\rvert} - \bar{\epsilon}\right)^2$
Kendall	1	296	0.007465	0.001312	0.000004
	2	319	0.005903	0.000771	0.000001
	3	424	0.003571	0.000732	0.000000
	4	469	0.008656	0.000575	0.000001
	5	655	0.004439	0.000418	0.000000
Cayley	1	10	0.000273	0.004058	0.000031
	2	12	0.005357	0.007252	0.000073
	3	24	0.000570	0.004384	0.000026
	4	14	0.005456	0.005308	0.000042
	5	22	0.001811	0.003102	0.000006

instance and the instance generated with our model. The first five rows in each table refer to the results when using the N_{adj} neighborhood in the local search and the Kendall-tau distance in our generator. The last five rows show the results for the same five instances when we utilize the N_{2-exch} neighborhood for solving them and the Cayley distance in our generator. In the first column we provide the number of the instance (1-5), and in the second column we indicate the number of local optima obtained in each case. The third column shows the proportion of solutions of the search space that are in the attraction basin of the global optimum but should not be there, or that should be there but they are not, that is: $\frac{||\mathcal{B}(\sigma_1)|-|\hat{\mathcal{B}}(\sigma_1)||}{|\Omega|}$. The fourth and the fifth columns indicate the average and the variance of the proportion of solutions of the search space that are not in the corresponding attraction basin. Notice that we divide by $2|\Omega|$, because if a solution is not in its corresponding attraction basin, the error is counted twice: in the attraction basin that it is in and in the one that it should be in.

We observe from Tables 1 and 2 that the artificial instances generated are almost identical in terms of sizes of attraction basins of the local optima. The average error found in the sizes of the attraction basins of all the local optima is lower than 0.14% in the PFSP instances when using the adjacent neighborhood (Kendall-tau distance), and lower than 0.73% when applying the 2-exchange neighborhood (Cayley distance). In the LOP instances, the errors are lower than 0.12% and 0.81%, for the adjacent and the 2-exchange neighborhoods (Kendall-tau and Cayley distances), respectively. Moreover, we observe that these errors are uniformly distributed among the attraction basins of the local optima, as the variances are very small: less than $0.73 \cdot 10^{-4}$ in all cases and almost zero in some of them.

For illustrative purposes, we plot the sizes of the attraction basins of the

Table 2: Results obtained for the errors in the attraction basin sizes of the local optima for the 5 instances of the LOP, for the Cayley and the Kendall-tau distances.

	Inst	m	$\frac{\epsilon_1}{\lvert\Omega\rvert}$	$\bar{\epsilon} = \frac{1}{m}\sum_{i=1}^{m}\frac{\epsilon_i}{2\lvert\Omega\rvert}$	$\frac{1}{m-1}\sum_{i=1}^{m}\left(\frac{\epsilon_i}{2\lvert\Omega\rvert} - \bar{\epsilon}\right)^2$
Kendall	1	435	0.003423	0.001168	0.000002
	2	720	0.003993	0.000700	0.000001
	3	895	0.003795	0.000536	0.000001
	4	920	0.004886	0.000542	0.000001
	5	3737	0.001265	0.000111	0.000000
Cayley	1	28	0.002307	0.005175	0.000032
	2	27	0.001910	0.008091	0.000037
	3	19	0.002679	0.003836	0.000015
	4	24	0.005704	0.005958	0.000013
	5	319	0.004812	0.000877	0.000001

local optima for two pairs of benchmark and artificial instances, when using the Cayley distance (2-exchange neighborhood). Particularly, Figure 2 shows the fifth instance of the PFSP and Figure 3 shows the third instance of the LOP[4]. For both figures, in the X axis the local optima are indicated sorted by their objective function value. That is, the local optimum number 1 is the global optimum, and the local optimum number m is the local optimum with the lowest objective function value. As can be observed, the sizes of the attraction basins of the local optima for the benchmark and artificial instances are almost identical. Notice that, when an attraction basin of a local optimum σ_i is larger or smaller than the attraction basin of the local optimum σ_{i+1} in the benchmark instance, it also happens in almost all of the cases of the artificial instances. Thus, we can conclude that the generator is flexible enough to create instances as complex as the instances found in common benchmark problems.

7.2 Tuning the parameters of the generator

We will show that our generator allows the user to control coarse grain features of the instances, such as the sizes of the attraction basins of the local optima, as well as to characterize the instances in a more detailed way. In this section we work with the Mallows (not Generalized) models.

7.2.1 Choosing the linear function G to optimize

As described previously, the generator allows the practitioner to define the linear function G to optimize. In this case, without loss of generality, we use the three functions introduced in Section 6: G_{MaxGO}, G_{MinGO} and G_{SimAB}. Our goal

[4]The remaining figures can be found in the website: http://www.sc.ehu.es /ccwbayes/members/leticia/GeneratorOfInstances/figures.html

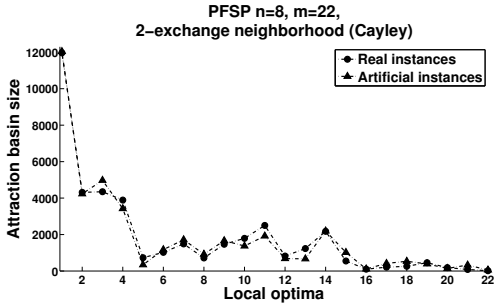

Figure 2: Sizes of the attraction basins of the 22 local optima found in the fifth instance of the PFSP using the 2-exchange neighborhood. We represent those sizes found in the real instance with a circle, and the sizes obtained in the artificial instance with a triangle.

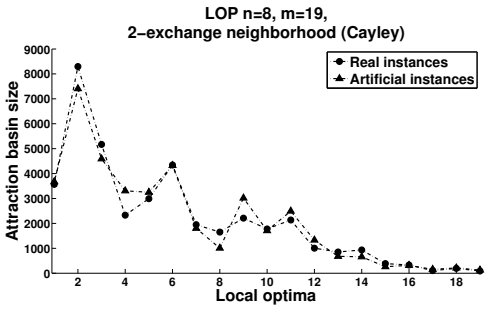

Figure 3: Sizes of the attraction basins of the 19 local optima found in the third instance of the LOP using the 2-exchange neighborhood. We represent those sizes found in the real instance with a circle, and the sizes obtained in the artificial instance with a triangle.

is to create instances with different attraction basin sizes for the local optima. Based on this feature, one can expect that the easiest instances will be those with a large attraction basin size for the global optimum. Regarding the remaining parameters, we use four permutation sizes $n = \{30, 40, 50, 100\}$, two different numbers of local optima $m = \{10^4, 10^5\}$, and two distances (Kendall-tau and Cayley). The consensus permutations (local optima) Σ are taken uniformly at random. The range of values for θ has been chosen following the recommendations given in Section 5.2 to avoid numerical errors. We specify them according to the desired type of instance. For each combination we create 10 instances.

- *MaxGO-Instances*: In this set, we want to promote a radical difference between the sizes of the attraction basins of the global optimum and the rest of the local optima, with the aim of obtaining the attraction basin of the global optimum as large as possible. We fix the value of θ_1:
  - $\theta_1 = ln(\frac{n-1}{3})$ for the Kendall-tau distance,
  - $\theta_1 = 2ln(\frac{n-1}{3})$ for the Cayley distance.

  The values of $\theta_i, i \neq 1$ are chosen uniformly at random in the intervals:
  - $I_K = [ln(n-1), 2ln(n-1)]$ for the Kendall-tau distance,
  - $I_C = [3ln(n-1), 4ln(n-1)]$ for the Cayley distance.

  Note that, in this sense, the value of θ_1 is always smaller than the rest of θ_i. The local optima are sorted such that $d(\sigma_2, \sigma_1) \geq d(\sigma_3, \sigma_1) \geq ... \geq d(\sigma_m, \sigma_1) \geq 2$. Therefore, the closest local optimum to the global optimum is the one that has the minimum objective function value: σ_m.

- *MinGO-Instances*: This set of instances is generated with the objective of having a small attraction basin of the global optimum and a large attraction basin of the closest local optimum. The values of $\theta_i, i \neq \{1, 2\}$, are chosen uniformly at random in the intervals I_K and I_C for the Kendall-tau and the Cayley distances, respectively, as in the *MaxGO-Instances*. The values for θ_1 and θ_2 are chosen as the highest and the lowest. So, we choose:
  - $\theta_1 = 3ln(n-1)$ and $\theta_2 = ln(\frac{n-1}{3})$ for Kendall-tau,
  - $\theta_1 = 6ln(n-1)$ and $\theta_2 = 2ln(\frac{n-1}{3})$ for Cayley.

  The local optima are sorted such that $2 \leq d(\sigma_2, \sigma_1) \leq d(\sigma_3, \sigma_1) \leq ... \leq d(\sigma_m, \sigma_1)$.

- *SimAB-Instances*: In this set of instances, the values of $\theta_i, \forall i$, are chosen uniformly at random from the intervals I_K and I_C for the Kendall-tau and the Cayley distances, respectively, as in the previous two examples. Notice that this time, we do not make any distinction between θ_1, θ_2 and the rest of θ_i. The consensus permutations are also taken at random without taking into account the distances between them.

In order to confirm our suspicion, that is, that the difficulty of the problem will be conditioned by the size of the attraction basin of the global optimum, we apply a local search (LS) algorithm, an Estimation of Distribution Algorithm (EDA) and a Genetic Algorithm (GA) to the three sets of instances proposed above. The algorithms are run 20 times for each instance, and we take the best solution reached by each algorithm for each repetition. The stopping criterion in the algorithms is the evaluation of $1000n^2$ solutions. Below, we detail the three algorithms used:

- **LS:** We use a random multi-start hill climbing approach where the best solution found in the neighborhood is chosen at each step. The neighborhoods used are: N_{adj} and N_{2-exch} for the instances where the Kendal-tau and the Cayley distances, respectively, were used in the Mallows models.

- **EDA:** We use the EDA presented in [27] for solving permutation-based problems. The authors used the Mallows distribution with the Kendall-tau distance as the probability model. As proposed by the authors, the population size is $10n$, and the n best individuals are selected for learning the probability distribution.

- **GA:** We use the GA proposed in [28]. As suggested by the authors, the population size is $20n$, a position based crossover operator (POS) and an insertion mutation operator (ISM) are used, and a tournament selection of size 2.

Tables 3, 4 and 5 show how the LS, the EDA and the GA, respectively, behave according to the type of instance they are applied to. Particularly, the tables show the percentage of the times that the best solution reached by the algorithm is the global optimum σ_1, the best local optimum σ_2, or any other local optimum. For the case of the EDA and the GA not reaching a local optimum, we have added additional information indicating which is the closest local optimum. With these results, we prove that the *MaxGO-Instances* are, indeed, easy for the LS as the global optimum is reached 100% of the times. On the other hand, we also check that the *MinGO-Instances* are instances where the global optimum is never seen. We find that in the *SimAB-Instances*, it is also difficult to find the global optimum. However, the difference between the *SimAB-Instances* and the *MinGO-Instances* is that in the *MinGO-Instances* there is a local optimum (σ_2 in all cases) that is seen in 100% of the runs, whereas in the set of *SimAB-Instances* different local optima are reached.

For the Kendall-tau distance, when applying the EDA (Table 4) and the GA (Table 5) to the *MaxGO-Instances*, the global optimum is reached, or, if not, the best solution found is always closer to the global optimum than to any other local optimum. In the *MinGO-Instances*, the global optimum is never observed, however the best solution is σ_2, or, at least, the closest local optimum to the best solution is σ_2. In the *SimAB-Instances*, σ_1 and σ_2 are never found, and other different local optima, or solutions closer to other local optima than to σ_1 and σ_2, are always seen. When applying the EDA using the Cayley distance, the

Table 3: Average percentage of the times that the best solution reached by the LS is the global optimum σ_1, the local optimum σ_2, or other different local optimum.

	Kendall-tau			Cayley		
	MaxGO-Inst	*MinGO-Inst*	*SimAB-Inst*	*MaxGO-Inst*	*MinGO-Inst*	*SimAB-Inst*
σ_1	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%
σ_2	0.00%	100.00%	0.00%	0.00%	100.00%	0.00%
σ_i, $(i \neq 1, 2)$	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%

Table 4: Average percentage of the times that the best solution reached by the EDA is (or is close to) the global optimum σ_1, the local optimum σ_2, or other different local optimum.

	Kendall-tau			Cayley		
	MaxGO-Inst	*MinGO-Inst*	*SimAB-Inst*	*MaxGO-Inst*	*MinGO-Inst*	*SimAB-Inst*
σ_1	77.54%	0.00%	0.00%	0.00%	0.00%	0.00%
σ_2	0.00%	77.87%	0.00%	0.00%	0.00%	0.00%
σ_i, $(i \neq 1, 2)$	0.00%	0.00%	76.37%	0.00%	0.00%	0.00%
closest σ_1	22.46%	0.00%	0.00%	99.50%	0.00%	0.00%
closest σ_2	0.00%	22.13%	0.00%	0.00%	99.88%	0.00%
closest σ_i $(i \neq 1, 2)$	0.00%	0.00%	23.63%	0.50%	0.12%	100.00%

best solutions found are never local optima. This is due to the fact that in the EDA we are considering the Mallows model with the Kendall-tau distance, and therefore it is difficult for this model to reach a local optimum for the 2-exchange neighborhood. However, a really high percentage of the times the best solutions found are closer to σ_1, σ_2 and other different σ_i for the *MaxGO-Instances*, the *MinGO-Instances* and the *SimAB-Instances*, respectively. This also happens to the GA when it is applied to the instances generated using the Cayley distance. However, this algorithm is able to find σ_1 and σ_2 in the *MaxGO-Instances* and the *MinGO-Instances*, respectively, at least a small percentage of the times.

Table 5: Average percentage of the times that the best solution reached by the GA is (or is close to) the global optimum σ_1, the local optimum σ_2, or other different local optimum.

	Kendall-tau			Cayley		
	MaxGO-Inst	*MinGO-Inst*	*SimAB-Inst*	*MaxGO-Inst*	*MinGO-Inst*	*SimAB-Inst*
σ_1	40%	0.00%	0.00%	8.00%	0.00%	0.00%
σ_2	0.00%	40.00%	0.00%	0.00%	13.00%	0.00%
σ_i, $(i \neq 1, 2)$	0.00%	0.00%	31.00%	0.00%	0.00%	20.00%
closest σ_1	60%	0.00%	0.00%	92.00%	0.00%	0.00%
closest σ_2	0.00%	60.00%	0.00%	0.00%	87.00%	0.00%
closest σ_i $(i \neq 1, 2)$	0.00%	0.00%	69.00%	0.00%	0.00%	80.00%

7.2.2 Influence of Σ and Θ

As seen in the previous experiments, changing the function to optimize, together with a careful ranking of randomly located local optima and choice of parameters Θ, allows us to create instances with different qualitative characteristics and hence complexities. However, it is also interesting to know the sensitivity of the generated instances to the input parameters, i.e. the set of permutations Σ, and the set of parameters Θ. It is clear that the location of the local optima can have a high impact on the complexity of an instance. Taking that into account, our generator provides complete freedom on the choice of the location of the local optima. We can devise several ways to do that, such as: choosing random permutations, using some proximity criterion between the global optimum and the local optima, or even creating a matrix with distance constraints and looking for the set of permutations that tries to fulfill such constraints.

The contribution of the set of parameters Θ is not so intuitive. They control the shape of the Mallows models involved in the generator and, thus, they have a big influence on the attraction basin of the local optima. Therefore, we consider it interesting to measure their contribution.

The experiments carried out to measure the sensitivity of the instances to the input parameters are as follows. The permutation size and the number of local optima have been fixed to $n = \{30\}$ and $m = \{10^4\}$, using the Kendall-tau and the Cayley distances in the Mallows models. The linear function considered is G_{SimAB}, as we think it is the least biased to analyze the influence of the input parameters. We have evaluated 11 values for θ_1 for each distance. Nine of these values are the points that divide the intervals I_K and I_C previously defined, in 10 identical subintervals and the other two are the extremes of the intervals. The rest of the spread parameters $\theta_i, i \neq 1$, are chosen uniformly at random in those intervals I_K and I_C.

For the location of the local optima, they have been chosen according to three different configurations[5]:

- 1^{st} configuration: Global optimum surrounded by all the local optima, as close as possible.
- 2^{nd} configuration: All the local optima are close except the global optimum that is as far from them as possible.
- 3^{rd} configuration: All the local optima, including the global optimum, are uniformly spread along the search space.

In summary, we have a total of 66 combinations: 2 (types of distance) x 3 (distribution of Σ) x 11 (values of Θ). We create 10 instances for each possible combination. As the criterion to evaluate the influence of the input parameters we have used the attraction basin size of the global optimum. For each of the generated instances we have estimated this size with the following procedure

[5]Due to space limitations, the algorithms to generate these configurations are available in the website: http://www.sc.ehu.es/ccwbayes/members/leticia /GeneratorOfInstances/configurations.html

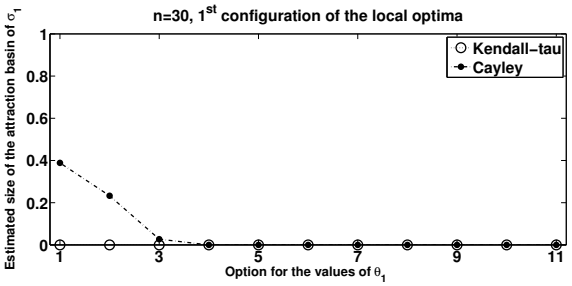

Figure 4: Estimated attraction basin sizes of σ_1 for the 1^{st} configuration.

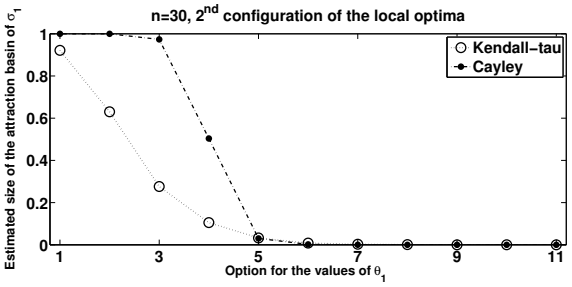

Figure 5: Estimated attraction basin sizes of σ_1 for the 2^{nd} configuration.

(note that an exact basin calculation is computationally unfeasible): we run the LS algorithm of the previous section, recording the number of times that σ_1 is seen. The proportion of times that the LS reaches σ_1 is an estimator of the proportion of its attraction basin size.

Figures 4, 5 and 6 show the results for the 1^{st}, 2^{nd} and 3^{rd} configurations of the local optima, respectively. Each point represents the average value of the attraction basin sizes of σ_1 of 10 instances. The figures mainly prove the big influence that the choice of θ_1 has in the attraction basin of σ_1. Although the localization of the local optima is relevant, (as can be seen with small values of θ_1) this influence is neglected by the peaky shape imposed in the Mallows model that generates the global optimum for medium to high values of θ_1.

7.2.3 A case study: Discovering differences between the LS, the GA and the EDA

As a case of use of our generator, we show in this section an example on how to utilize the generated instances to discover new facts about metaheuristic algorithms. Particularly, we have tested the previously defined LS, EDA and GA in all the instances generated in the previous section. Each algorithm has

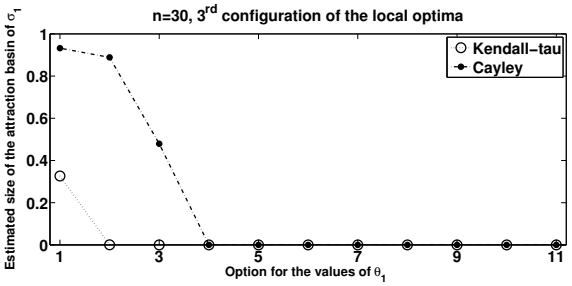

Figure 6: Estimated attraction basin sizes of σ_1 for the 3^{rd} configuration.

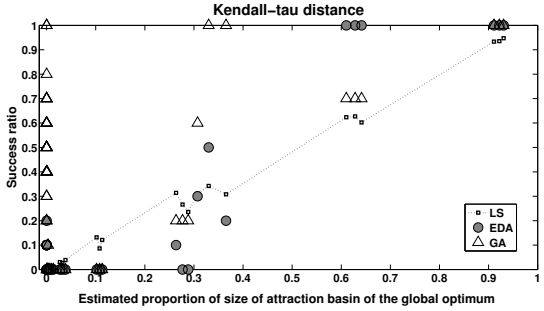

Figure 7: Performance of the LS, the EDA and the GA with respect to the estimated attraction basin size of the global optimum. Kendall-tau distance.

been run 20 times in each instance.

In Figures 7 and 8 we show the success ratio of the LS, the EDA, and the GA (average proportion of the times that the best solution reached is the global optimum) with respect to the estimated size of the attraction basin of the global optimum, for the Kendall-tau and Cayley metrics, respectively. As expected, a good correlation can be observed for the LS. The larger the attraction basin of the global optimum, the higher the probability of finding it. In contrast, it is remarkable that the behavior of the EDA and the GA is not so clearly correlated. In fact, there are cases of small attraction basins and high success ratios and vice versa. This type of scenario puts in value the usefulness of our generator: it allows us to go further, analyzing the impact of other characteristics such as the distribution of the local optima in the behavior of the algorithms. For this purpose, in Figures 9 and 10 the success ratio of the LS, the EDA, and the GA is measured related to the average distance of the global optimum to the rest of the local optima. For these two figures, 10 instances of each configuration of the local optima with similar attraction basin sizes of the global optimum have been chosen (between $1.1 \cdot 10^{-31}$ and $1.4 \cdot 10^{-25}$ for Kendall-tau and between 0.3 and 0.4 for Cayley). Due to this similarity, the behavior of the LS is almost

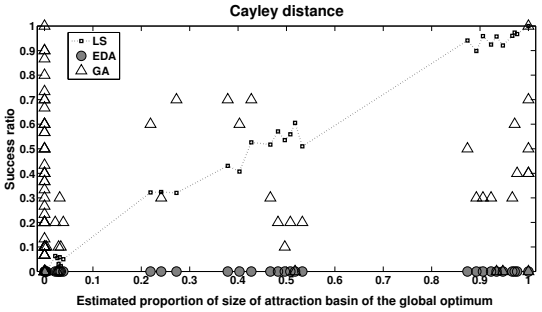

Figure 8: Performance of the LS, the EDA, and the GA with respect to the estimated attraction basin size of the global optimum. Cayley distance.

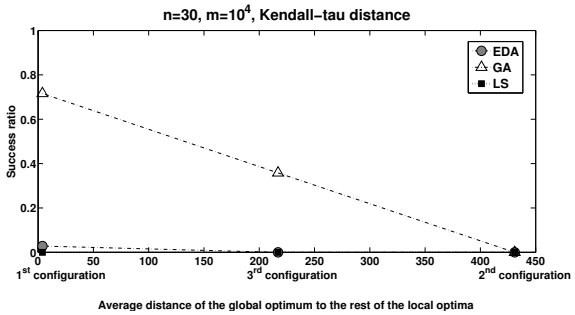

Figure 9: Performance of the LS, the EDA, and the GA according to the average distance of the local optima to the global optimum. Kendall-tau distance.

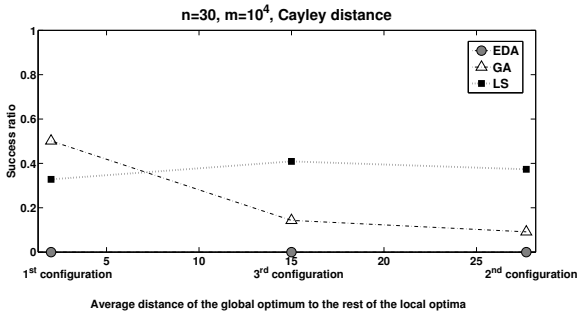

Figure 10: Performance of the LS, the EDA, and the GA according to the average distance of the local optima to the global optimum. Cayley distance.

identical, independently of the average distances between the local optima and the global optimum. However, this feature (average distance) notably affects the performance of the other two algorithms (particularly clear is the influence in the GA). In view of the experiments, we conjecture that, for the GA the average distance between the local optima and the global optimum has a higher influence on its behavior than the size of the attraction basin of the global optimum. The potential of our generator is shown, as it would allow to perform a set of experiments analyzing the effect that reducing or increasing the average distance between the local optima and the global optimum has on the GA.

Regarding the EDA, for the Kendall-tau distance, we can also conclude that, if the global optimum is close to the rest of the local optima, the probability of finding it is a bit higher (near 0.04) than if the global optimum is not so close (almost 0.00). Notice that in these instances the proportion of the size of the attraction basin of the global optimum is very small, and the EDA finds it difficult (in all the cases) to reach it. However, for the Cayley distance, despite the proportion of the size of the attraction basin of the global optimum being considerably high (between 0.3 and 0.4), the EDA is not able to reach the global optimum, not even in the case that the global optimum is close to the rest of the local optima. We can conclude that this algorithm is not suitable for this type of instances. This is another example of a conclusion that we can obtain about the performance of this EDA when it is applied to instances with the properties chosen.

8 Conclusions and future work

In the optimization field we can find several proposals of generators of instances. Above all, these generators are for the continuous domain, or for binary spaces. However, a few generators have been proposed for permutation-based COPs, and they are not able to control many properties of the generated instance. In this paper, we present a flexible generator, based on a mixture of GM models. The parameters of these GM models are input parameters for the generator, and by tuning them, the user can control quantitative as well as qualitative properties of the resultant instance, such as the attraction basin sizes, the number of local optima, or their location. We have provided the restrictions that these parameters need to fulfill to obtain quantitative properties in the generated instance. Precisely, we have given sufficient conditions to ensure that the instance has a predefined number of local optima. Moreover, we have proposed to solve the constraints in the parameters by means of a linear programming problem. For this purpose, we have added a linear function to optimize that helps to obtain qualitative properties in the instance.

We have tested two important properties of our generator: its flexibility and its ability to create instances of very different complexities for local and population-based common algorithms. To assess the first property, we have considered instances of the PFSP and the LOP. We have measured the similarity between the artificial and the benchmark instances by means of the attrac-

tion basin sizes of the local optima. We find that, for small permutation sizes (those computationally comparable), our generator is flexible enough to create instances with almost the same sizes of attraction basins of the local optima as the benchmark instances. In order to study the second property, we have created a large set of instances of very different properties (permutation size, number of local optima, size of the attraction basins of the local optima, etc.) playing with the different input parameters available. According to the results, we claim that our generator is a very useful tool for the community to analyze and improve the performance of different optimization algorithms, and therefore it supposes an innovative and relevant proposal in this arena.

As we have seen, the weakest point of our algorithm is the expensive cost of evaluating a solution, that is basically conditioned by m (the number of local optima). Therefore, in order to be able to work with a high number of local optima, a key issue would be to think of different processes that help to reduce the time complexity of one fitness evaluation. Starting by sorting, from high to low, the local optima according to their fitness value, helps significantly to reduce this time. For example, in the case of population-based algorithms, when assigning a fitness to a given solution, we could stop the process of looking for the maximum value given by the GM models, when we find a value higher than the next fitness of the consensus permutation to test. According to a preliminary analysis, with this process we reduced the time to 10%. In the case of local search algorithms, a kind of incremental evaluation could be carried out, and in order to evaluate the neighbors it would not be necessary to evaluate all the models, because their change in fitness is limited by the new distance (+1 or -1). Other techniques such as parallelism could be also applied to make the computation times affordable. It is important to delve into this aspect in order to be able to produce, with our generator, instances with a large number of local optima, for which any algorithm could find a solution in reasonable time.

9 Proof of the lemma 5.1

Let's suppose that (7) is false, that is: $\exists k \neq i$ such that

$$\begin{aligned} f(\sigma_i) &= \max_{1 \leq j \leq m} \{w_j p_j(\sigma_i|\sigma_j, \boldsymbol{\theta}_j)\} \\ &= w_k p_k(\sigma_i|\sigma_k, \boldsymbol{\theta}_k) = w_k \frac{e^{-\sum_{s=1}^{n-1} \theta_k^s d_s(\sigma_i, \sigma_k)}}{Z(\boldsymbol{\theta}_k)}. \end{aligned}$$

Then, the objective function value of the first permutation found in the shortest path between σ_i and σ_k, that is, the permutation σ such that $d(\sigma, \sigma_i) = 1$ and

$d(\sigma, \sigma_k) = d(\sigma_i, \sigma_k) - 1$, is:

$$\begin{aligned} f(\sigma) &= \max_{1 \leq j \leq m} \{w_j p_j(\sigma|\sigma_j, \boldsymbol{\theta}_j)\} \\ &\geq w_k p_k(\sigma|\sigma_k, \boldsymbol{\theta}_k) = w_k \frac{e^{-\sum_{s=1}^{n-1} \theta_k^s d_s(\sigma, \sigma_k)}}{Z(\boldsymbol{\theta}_k)}. \end{aligned}$$

Notice that

$$d(\sigma, \sigma_k) = \sum_{s=1}^{n-1} d_s(\sigma, \sigma_k) < d(\sigma_i, \sigma_k) = \sum_{s=1}^{n-1} d_s(\sigma_i, \sigma_k),$$

in fact, $d(\sigma, \sigma_k) = d(\sigma_i, \sigma_k) - 1$, so that the elements involved in the decomposition of the distance between σ and σ_k need to be equal to the elements of the decomposition of the distance between σ_i and σ_k, with the exception of one that has to be one unit lower. This is:

$$\begin{aligned} d_s(\sigma, \sigma_k) &= d_s(\sigma_i, \sigma_k), \forall s \neq t \\ d_t(\sigma, \sigma_k) &= d_t(\sigma_i, \sigma_k) - 1. \end{aligned}$$

So then,

$$\begin{aligned} \sum_{s=1}^{n-1} \theta_k^s d_s(\sigma, \sigma_k) &= \sum_{s \neq t} \theta_k^s d_s(\sigma_i, \sigma_k) + \theta_k^t [d_t(\sigma_i, \sigma_k) - 1] \\ &< \sum_{s=1}^{n-1} \theta_k^s d_s(\sigma_i, \sigma_k) \\ &\Rightarrow e^{-\sum_{s=1}^{n-1} \theta_k^s d_s(\sigma, \sigma_k)} > e^{-\sum_{s=1}^{n-1} \theta_k^s d_s(\sigma_i, \sigma_k)}, \end{aligned}$$

and therefore $w_k \frac{e^{-\sum_{s=1}^{n-1} \theta_k^s d_s(\sigma, \sigma_k)}}{Z(\boldsymbol{\theta}_k)} > w_k \frac{e^{-\sum_{s=1}^{n-1} \theta_k^s d_s(\sigma_i, \sigma_k)}}{Z(\boldsymbol{\theta}_k)}$.

Thus, we have a permutation σ such that $d(\sigma, \sigma_i) = 1$ and $f(\sigma) > f(\sigma_i)$. This proves that if (7) is not fulfilled, σ_i is not a local optimum.

10 Proof of the theorem 5.2

The definition of local optimum is given by (6), so that: $f(\sigma_i) > f(\sigma), \forall \sigma \in \Omega$ s.t. $d(\sigma, \sigma_i) = 1$. By lemma 5.1, $f(\sigma_i) = \frac{w_i}{Z(\theta_i)}$, and therefore we can rewrite (6) in the following form:

$$\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \max_{1 \leq j \leq m} \left\{ \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma, \sigma_j)} \right\},$$

$\forall \sigma \ s.t. \ d(\sigma, \sigma_i) = 1$.
Obviously, this is fulfilled for $j = i$, so that this is equivalent to:

$$\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma, \sigma_j)}, \forall j \neq i,$$

$\forall \sigma \ s.t. \ d(\sigma, \sigma_i) = 1$. Therefore, (8) is fulfilled.

Let's suppose now that the constraints in (8) are satisfied, we will prove that, then, σ_i is a local optimum. The definition of objective function value of σ_i in our generator is the following:

$$f(\sigma_i) = \max_{1 \leq j \leq m} \left\{ \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma_i, \sigma_j)} \right\}.$$

We can rewrite this expression, distinguishing between $j = i$ and $j \neq i$, as:

$$f(\sigma_i) = \max_{j \neq i} \left\{ \frac{w_i}{Z(\boldsymbol{\theta}_i)}, \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma_i, \sigma_j)} \right\}$$

However, we know by (8) that

$$\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma, \sigma_j)}, \forall \sigma \ s.t. \ d(\sigma, \sigma_i) = 1$$

If this is fulfilled for all σ such that $d(\sigma, \sigma_i) = 1$, specifically, this is fulfilled for the first permutation σ' found in the shortest path between σ_i and σ_j, such that $d(\sigma', \sigma_i) = 1$ and $d(\sigma', \sigma_j) = d(\sigma_i, \sigma_j) - 1$. Reasoning as in the proof of the Lemma 5.1, the elements involved in the decomposition of the distance between σ' and σ_j are:

$$\begin{aligned} d_s(\sigma', \sigma_j) &= d_s(\sigma_i, \sigma_j), \forall s \neq t \\ d_t(\sigma', \sigma_j) &= d_t(\sigma_i, \sigma_j) - 1 \end{aligned}$$

and therefore

$$\begin{aligned} &\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma', \sigma_j) < \sum_{s=1}^{n-1} \theta_j^s d_s(\sigma_i, \sigma_j) \\ \Rightarrow \quad &e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma', \sigma_j)} > e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma_i, \sigma_j)}. \end{aligned}$$

So, (8) implies $\forall \sigma \ s.t. \ d(\sigma, \sigma_i) = 1$:

$$\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma_i, \sigma_j)} \Rightarrow f(\sigma_i) = \frac{w_i}{Z(\boldsymbol{\theta}_i)} > f(\sigma).$$

11 Proof of the theorem 5.3

From (9) we have:

$$\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)}, \forall i < j$$

and obviously, as $\frac{w_j}{Z(\boldsymbol{\theta}_j)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} \theta_j^s d_s(\sigma,\sigma_j)}$, $\forall \sigma$, then inequalities in (8) when $i < j$ are fulfilled. Thus, we just have to prove that (9) and (10) imply (8) when $i > j$.

We can rewrite (10) in the following form:

$$\left[1 - e^{-\left(\min_{j,s}\{\theta_j^s\}\right)}\right] \frac{w_m}{Z(\boldsymbol{\theta}_m)} > \frac{w_1}{Z(\boldsymbol{\theta}_1)} - \frac{w_m}{Z(\boldsymbol{\theta}_m)} \tag{11}$$

As it is known by (9): $\frac{w_m}{Z(\boldsymbol{\theta}_m)} \leq \frac{w_j}{Z(\boldsymbol{\theta}_j)}, \forall j$.

So, (11) implies:

$$\left[1 - e^{-\left(\min_{j,s}\{\theta_j^k\}\right)}\right] \frac{w_j}{Z(\boldsymbol{\theta}_j)} > \frac{w_1}{Z(\boldsymbol{\theta}_1)} - \frac{w_m}{Z(\boldsymbol{\theta}_m)}, \forall j$$

Moreover, $\frac{w_1}{Z(\boldsymbol{\theta}_1)} - \frac{w_m}{Z(\boldsymbol{\theta}_m)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} - \frac{w_i}{Z(\boldsymbol{\theta}_i)}, \forall i > j$
so that,

$$\left[1 - e^{-\left(\min_{j,s}\{\theta_j^s\}\right)}\right] \frac{w_j}{Z(\boldsymbol{\theta}_j)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} - \frac{w_i}{Z(\boldsymbol{\theta}_i)}, \forall i > j$$

and thus

$$\begin{aligned} \frac{w_i}{Z(\boldsymbol{\theta}_i)} &> \frac{w_j}{Z(\boldsymbol{\theta}_j)} - \left[1 - e^{-\left(\min_{j,s}\{\theta_j^s\}\right)}\right] \frac{w_j}{Z(\boldsymbol{\theta}_j)} \\ &\Rightarrow \frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\left(\min_{j,s}\{\theta_j^s\}\right)} \end{aligned} \tag{12}$$

Notice that as $d(\sigma, \sigma_j) \geq 1, \forall \sigma \neq \sigma_j$:

$$\begin{aligned} \min_{j,s}\{\theta_j^s\} &< \min_{j,s}\{\theta_j^s\} d(\sigma,\sigma_j) = \min_{j,s}\{\theta_j^s\} \sum_{s=1}^{n-1} d_s(\sigma,\sigma_j) \\ &< \sum_{s=1}^{n-1} [\theta_j^s d_s(\sigma,\sigma_j)]. \end{aligned}$$

Finally, inequality (12) implies

$$\frac{w_i}{Z(\boldsymbol{\theta}_i)} > \frac{w_j}{Z(\boldsymbol{\theta}_j)} e^{-\sum_{s=1}^{n-1} [\theta_j^s d_s(\sigma,\sigma_j)]}.$$

Acknowledgment

This work has been partially supported by the Saiotek and Research Groups 2013-2018 (IT- 609-13) programs (Basque Government), TIN2013-41272P (Spanish Ministry of Science and Innovation), COMBIOMED network in computational biomedicine (Carlos III Health Institute), CRC-Biomarkers 6-12-TK-2011-014 (Diputación Foral de Bizkaia) and NICaiA PIRSES-GA-2009-247619 Project (European Commission). Leticia Hernando holds a grant from the Basque Government.

References

[1] J. Rönkkönen, X. Li, V. Kyrki, and J. Lampinen, "A framework for generating tunable test functions for multimodal optimization," *Soft Computing*, pp. 1–18, 2010.

[2] M. Gallagher and B. Yuan, "A general-purpose tunable landscape generator," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 590–603, October 2006.

[3] B. Yuan and M. Gallagher, "On building a principled framework for evaluating and testing evolutionary algorithms: A continuous landscape generator," *In the 2003 Congress on Evolutionary Computation*, vol. 1, pp. 451–458, December 2003.

[4] R. Morgan and M. Gallagher, "Using landscape topology to compare continuous metaheuristics: A framework and case study on edas and ridge structure," *Evolutionary Computation*, vol. 20, no. 2, pp. 277–299, May 2012.

[5] ——, "When does dependency modelling help? using a randomized landscape generator to compare algorithms in terms of problem structure," in *Parallel Problem Solving from Nature, PPSN XI*, ser. Lecture Notes in Computer Science, R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, Eds. Springer Berlin / Heidelberg, 2010, vol. 6238, pp. 94–103.

[6] K. A. De Jong, M. A. Potter, and W. M. Spears, "Using problem generators to explore the effects of epistasis," in *Seventh International Conference on Genetic Algorithms*. Morgan Kaufmann, 1997, pp. 338–345.

[7] J. Branke, "Evolutionary computation in dynamic environments," *Kluwer Academic*, 2002.

[8] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. Resende, and W. R. Stewart Jr, "Designing and reporting on computational experiments with heuristic methods," *Journal of Heuristics*, vol. 1, no. 1, pp. 9–32, 1995.

[9] J. Cirasella, D. S. Johnson, L. A. McGeoch, and W. Zhang, "The asymmetric traveling salesman problem: Algorithms, instance generators, and tests," in *Algorithm Engineering and Experimentation*. Springer, 2001, pp. 32–59.

[10] R. L. Rardin and R. Uzsoy, "Experimental evaluation of heuristic optimization algorithms: A tutorial," *Journal of Heuristics*, vol. 7, no. 3, pp. 261–304, 2001.

[11] L. Hernando, A. Mendiburu, and J. A. Lozano, "Generating Customized Landscapes in Permutation-based Combinatorial Optimization Problems," in *Learning and Intelligent OptimizatioN Conference (LION 7)*, Catania, Italy, 2013.

[12] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1–2, pp. 245 – 276, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370295001247

[13] M. Gallagher, "Fitness distance correlation of neural network error surfaces: A scalable, continuous optimization problem," in *Machine Learning: ECML 2001*, ser. Lecture Notes in Computer Science, L. Raedt and P. Flach, Eds. Springer Berlin Heidelber, 2001, vol. 2167, pp. 157–166.

[14] J. H. Holland, "Building blocks, cohort genetic algorithms, and hyperplane-defined functions," *Evolutionary Computation*, vol. 8, pp. 373–391, 2000.

[15] C. L. Mallows, "Non-null ranking models," *Biometrika*, vol. 44, no. 1-2, pp. 114–130, 1957.

[16] M. Fligner and J. S. Verducci, "Multistage ranking models," *Journal of the American Statistical Anssociation*, vol. 83, no. 403, pp. 892–901, 1988.

[17] B. Mandhani and M. Melia, "Tractable search for learning exponential models of rankings," *Journal of Machine Learning Research*, vol. 5, pp. 392–399, 2009.

[18] M. A. Fligner and J. S. Verducci, "Distance based ranking models," *Journal of the Royal Statistical Society*, vol. 48, no. 3, pp. 359–369, 1986.

[19] E. Irurozki, B. Calvo, and J. A. Lozano, "Sampling and learning the Mallows and Generalized Mallows models under the Cayley distance," *Methodology and Computing in Applied Probability. Submitted.*, 2014.

[20] J. Lafferty and G. Lebanon, "Conditional models on the ranking poset," in *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, vol. 15. MIT Press, 2003, pp. 431–438.

[21] G. Lebanon and Y. Mao, "Non-parametric modeling of partially ranked data," *Journal of Machine Learning Research*, vol. 9, pp. 2401–2429, 2008.

[22] T. de Lima and M. Ayala-Rincon, "Complexity of cayley distance and other general metrics on permutation groups," in *Computing Congress (CCC), 2012 7th Colombian*, 2012, pp. 1–6.

[23] W. Cheng and E. Hullenmeier, "A simple instance-based approach to multilabel classification using the mallows model," in *Workshop Proceedings of Learning from Multi-Label Data*, Bled, Slovenia, 2009, pp. 28–38.

[24] I. Caragiannis, A. D. Procaccia, and N. Shah, "When do noisy votes reveal the truth?" in *Proceedings of the Fourteenth ACM Conference on Electronic Comerce*, New York, USA, 2013, pp. 143–160.

[25] L. Hernando, A. Mendiburu, and J. A. Lozano, "An evaluation of methods for estimating the number of local optima in combinatorial optimization problems," *Evolutionary Computation*, vol. 21, no. 4, pp. 625–658, 2013.

[26] T. Schiavinotto and T. Stützle, "The linear ordering problem: instances, search space analysis and algorithms," *Journal of Mathematical Modelling and Algorithms*, 2004.

[27] J. Ceberio, A. Mendiburu, and J. A. Lozano, "Introducing the mallows model on estimation of distribution algorithms," in *2011 International Conference on Neural Information Processing (ICONIP-2011)*, Shanghai, China, 23-25, November 2011, pp. 461–470.

[28] J. A. Aledo, J. A. Gámez, and D. Molina, "Tackling the rank aggregation problem with evolutionary algorithms," *Applied Mathematics and Computation*, vol. 222, pp. 632–644, 2013.