A Distance-Based Ranking Model Estimation of Distribution Algorithm for the Flowshop Scheduling Problem

Josu Ceberio, Ekhine Irurozki, Alexander Mendiburu, and Jose A. Lozano, *Member, IEEE*

***Abstract*—The aim of this paper is two-fold. First, we introduce a novel general estimation of distribution algorithm to deal with permutation-based optimization problems. The algorithm is based on the use of a probabilistic model for permutations called the generalized Mallows model. In order to prove the potential of the proposed algorithm, our second aim is to solve the permutation flowshop scheduling problem. A hybrid approach consisting of the new estimation of distribution algorithm and a variable neighborhood search is proposed. Conducted experiments demonstrate that the proposed algorithm is able to outperform the state-of-the-art approaches. Moreover, from the 220 benchmark instances tested, the proposed hybrid approach obtains new best known results in 152 cases. An in-depth study of the results suggests that the successful performance of the introduced approach is due to the ability of the generalized Mallows estimation of distribution algorithm to discover promising regions in the search space.**

***Index Terms*—Estimation of distribution algorithms, generalized Mallows model, permutation flowshop scheduling problem, permutations-based optimization problems.**

I. Introduction

SINCE Johnson published his work on the two-machine flowshop in 1954 [1], numerous papers have dealt with the flowshop scheduling problem. Early research on this topic was highly theoretical and focused mainly on exact methods. In 1976, Garey *et al.* [2] proved that this problem is NP-complete for more than two machine instances, and thus, stated the difficulty of achieving optimal solutions. In later decades, the problem was extensively addressed in the literature [3], [4]. A simplified version of this problem is the permutation flowshop scheduling problem (PFSP) [4], where the goal is to find the optimal sequence of n jobs that must be processed in m machines. Even though the PFSP was formulated in the 1950s, it is still a meaningful research topic due to its strong engineering background. Initially, the most common optimization criterion in the PFSP was the total completion time of the jobs, also called makespan. However, the minimization of the total flow time (TFT) has captured the attention of the scientific community since it is more relevant than the makespan for the current dynamic production environments. The TFT measures the sum of the times that each job remains in the flowshop. Thus, minimizing the TFT can lead to the stable utilization of resources, rapid turn-around of jobs, and minimization of work-in-progress inventory costs [5].

The literature contains evidence of a wide variety of strategies that approach the PFSP with respect to the TFT criterion (PFSP-TFT). Exact algorithms accurately work out the optimal solution. However, they are computationally costly and, with the exception of small size examples (instances up to 10–15 jobs), exact algorithms cannot be used in a reasonable time span. Examples of such methods are branch and bound approaches [6], [7]. Besides the exact methods, heuristic procedures such as constructive heuristics [5], [8]–[13] and composite heuristics [10], [14]–[16] have also been proposed. Unfortunately, such heuristics have shown a poor performance when solving PFSP instances of 50 jobs or more.

Advances in metaheuristic optimization supplied the research community with new techniques to tackle combinatorial optimization problems. As proof of these advances, we found a wide variety of strategies in the literature for solving the PFSP, such as local search [17]–[19], simulated annealing [20], tabu search [21]–[23], genetic algorithms (GAs) [24]–[28], particle swarm optimization [29]–[32], ant colony optimization [33]–[36], or estimation of distribution algorithms (EDAs) [37]. Currently, the literature points to the hybrid approaches, such as those published by Costa *et al.* [variable neighborhood search 4 - (VNS_4)] [19] and Xu *et al.* [asynchronous genetic algorithm (AGA)] [26], as the cutting-edge algorithms for optimizing the PFSP-TFT. In the first case, the authors propose the combination of a VNS with a constructive algorithm called LR(n/m) [10]. In the second case, a GA is hybridized with VNS. Additionally, to guide the initial population, LR(n/m) is also used.

In this paper, we go deeper into the development of EDAs, proposing a new EDA for permutation-based optimization problems and testing its performance on the PFSP-TFT.

Manuscript received June 18, 2012; revised October 11, 2012 and February 26, 2013; accepted April 24, 2013. Date of publication April 29, 2013; date of current version March 28, 2014. This work was supported in part by the Basque Government's programs Saiotek and Research Groups 2013–2018 under Grant IT-609-13, the Ministry of Science and Technology under Grant TIN2010-14931, COMBIOMED network in computational biomedicine (Carlos III Health Institute), and the European Commission's NICaiA Project under Grant PIRSES-GA-2009-247619. The work of Josu Ceberio was supported by a grant from the Basque Government. The work of Ekhine Irurozki was supported by the MICINN under Grant BES-2009-029143.

The authors are with the Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Gipuzkoa 20018, Spain (e-mail: josu.ceberio@ehu.es; ekhine.irurozki@ehu.es; alexander.mendiburu@ehu.es; ja.lozano@ehu.es).

Digital Object Identifier 10.1109/TEVC.2013.2260548

EDAs are a type of population-based optimization algorithm that, at each generation, learn a probabilistic model from a set of promising solutions. The new offspring is then obtained by sampling the probabilistic model.

Recently, they have become a strong alternative for solving optimization problems from diverse domains [38]–[43]. Apart from the work of Jarboui [37], other EDA-based approaches have been also proposed for solving similar permutation-coded scheduling problems [44]–[46]. However, a recently published review on EDAs for permutation-based optimization problems [47] claimed that the performance of the existing EDAs could be further improved. The authors argue that most of the existing EDAs were originally designed for solving integer or real-valued domain problems, and were later adapted to deal with permutation-based problems. Among the EDAs specifically designed for permutation-based problems, Tsutsui [48], [49] introduced two of the most successful proposals: the edge histogram-based sampling algorithm (EHBSA) and node histogram-based sampling algorithm (NHBSA). In that review [47], the experimental study demonstrated that these two algorithms were the most competitive EDAs for solving permutation-based problems. Nonetheless, in [47] the authors suggest that implementing specific probability models on rankings could improve the performance of EDAs on permutation-based problems.

Along the same research line, [50] introduced the Mallows EDA, the first preliminary attempt to apply a probabilistic model that estimates an explicit probability distribution in the domain of permutations. The Mallows model is a distance-based exponential probabilistic model considered analogous to the Gaussian probability distribution over the space of permutations. The results reported in [50] showed that Mallows EDA is able to outperform EHBSA and NHBSA for the PFPS with the makespan criterion.

In this paper, we go a step further by proposing a generalization of the Mallows model [51] in the framework of EDAs. The resulting algorithm, called generalized Mallows (GM) EDA (GM-EDA), introduces the first serious attempt to apply parametric models for permutation domains, such as the GM model, for solving permutation-based problems with EDAs.

In order to demonstrate the validity of our proposal, we approach the PFSP-TFT with a hybrid version of the GM-EDA. The hybrid GM-EDA is a two-stage algorithm. In the first stage, it runs the GM-EDA, and the resulting individual is used as the initial solution for the VNS employed in the second stage. Experimental results show that the hybrid GM-EDA is competitive with the state-of-the-art algorithms, since it is able to obtain new best known results in 152 instances of the 220 tested.

The remainder of the paper is organized as follows. In the following section, the formulation of the PFSP is given. In Section III, the GM model is described in detail. Section IV introduces the GM-EDA. In Section V, the performance of the GM-EDA over some PFSP instances is studied. Our hybrid proposal for solving the PFSP and its application to several benchmark instances is introduced afterward, in Section VI. We carry out a thorough discussion of the results in Section VII. Finally, some conclusions and ideas for future work are presented in Section VIII.

II. Problem Formulation

In the permutation flowshop scheduling problem [52], [53], n jobs ($i = 1, \ldots, n$) have to be scheduled on m machines ($j = 1, \ldots, m$) in such a way that a criterion is minimized. A job consists of m operations and the jth operation of each job must be processed on machine j for a given specific processing time without interruption. The processing times are fixed, nonnegative values and every job is available at time zero. At a given time, a job can start on the jth machine when its $(j-1)$th operation has finished on machine $(j-1)$, and machine j is free.

As introduced previously, our objective is to find a sequence of jobs such that the total flow time is minimized. In this case, the sequence of n jobs is codified as a permutation function σ from $\{1, \ldots, n\}$ onto $\{1, \ldots, n\}$, where $\sigma(i)$ is the rank assigned to job i, $i = 1, \ldots, n$, and $\sigma^{-1}(i)$ denotes the job at rank i in σ (referred also as $\sigma\langle i\rangle$). For instance, the sequence $\sigma = (2\ 3\ 1)$ denotes that job 1 is ranked in the second position ($\sigma(1) = 2$), job 2 is ranked third, and job 3 is ranked first. An alternative description of a ranking is given by the associated ordering, denoted in this case as $\sigma^{-1} = \langle 3\ 1\ 2\rangle$.

Equation (1) expresses mathematically the concept of TFT for a ranking σ of jobs, where $c_{\sigma\langle i\rangle,m}$ stands for the completion time of job $\sigma\langle i\rangle$ at machine m

$$F(\sigma) = \sum_{i=1}^{n} c_{\sigma\langle i\rangle,m}. \tag{1}$$

Being that $p_{\sigma\langle i\rangle,j}$ is the processing time required by job $\sigma\langle i\rangle$ in machine j, the completion time of job $\sigma\langle i\rangle$ on machine j can be recursively calculated as

$$c_{\sigma\langle i\rangle,j} = \begin{cases} p_{\sigma\langle i\rangle,j} & i = j = 1 \\ p_{\sigma\langle i\rangle,j} + c_{\sigma\langle i-1\rangle,j} & i > 1,\ j = 1 \\ p_{\sigma\langle i\rangle,j} + c_{\sigma\langle i\rangle,j-1} & i = 1,\ j > 1 \\ p_{\sigma\langle i\rangle,j} + \max\{c_{\sigma\langle i-1\rangle,j},\ c_{\sigma\langle i\rangle,j-1}\} & i > 1,\ j > 1 \end{cases} \tag{2}$$

III. Generalized Mallows Model

Until the 1950s, probability models for permutations were not extensively developed. The psychology domain was the first that demanded specific models to efficiently manage probability distributions over permutations. Early works proposed Thurstonian [54] and uniform models. At the end of the decade, a probabilistic model for permutations called the Mallows [55] model was introduced. The Mallows model is a probabilistic model that has been considered as analogous over permutations to the Gaussian distribution: they are both exponential unimodal models defined by two parameters, the mean and the variance.

Despite having been proposed in the late 1950s, it is a lively model that has recently received the attention of the statistical and machine learning communities, and has been

applied to problems from diverse fields, such as recommender systems [56], multilabel classification [57], [58], data modeling [59], clustering [60], and computer vision [61].

Under this model, the probability value of every permutation $\sigma \in \mathbb{S}_n$ (where $\mathbb{S}_n$ stands for the set of $n!$ permutations of n items) depends on just two parameters: a spread parameter θ and the distance to a central permutation σ_0, which is calculated by a particular metric $D(\sigma, \sigma_0)$. Formally, the Mallows model is defined as

$$P(\sigma) = \psi(\theta)^{-1} exp(-\theta D(\sigma, \sigma_0)) \tag{3}$$

where $\psi(\theta)$ is a normalization constant.

When θ equals 0, (3) assigns equal probability to every permutation σ in $\mathbb{S}_n$. The larger the value of θ, the more peaked the distribution becomes around the central permutation (we do not consider the situation $\theta < 0$). The second parameter is the central ranking σ_0. This model assigns every permutation $\sigma \in \mathbb{S}_n$ a probability that decays exponentially with respect to its distance to this central permutation σ_0. In this way, $P(\sigma)$ can be considered as an exponential nonuniform distance-based ranking model.

The distance metric with which the Mallows model is traditionally coupled in the literature (and is used in this manuscript) is Kendall's-τ. Denoted as $D_\tau(\sigma, \pi)$, it measures the number of pairs of items for which σ and π have opposing ordering. An equivalent definition for Kendall's-τ is the minimum number of adjacent transpositions needed to bring σ^{-1} into π^{-1}

$$D_\tau(\sigma, \pi) = |\{(i, j) : i < j, (\sigma(i) < \sigma(j) \wedge \pi(i) > \pi(j)) \vee (\pi(i) < \pi(j) \wedge \sigma(i) > \sigma(j))\}|.$$

Kendall's-τ has the right invariant property since $D_\tau(\sigma, \pi) = D_\tau(\sigma\gamma, \pi\gamma)$ for every permutation $\gamma \in \mathbb{S}_n$, where $\sigma\gamma$ stands for the composition of the permutations σ and γ, and is defined as $\sigma\gamma(i) = (\sigma \circ \gamma(i)) = \sigma(\gamma(i))$. Thus, by right invariance, when $\gamma = \pi^{-1}$, $D_\tau(\sigma, \pi)$ can be simplified as $D_\tau(\sigma\pi^{-1}, e)$ (also denoted as $D_\tau(\sigma\pi^{-1})$), where e stands for the identity permutation $e = (1\,2\,3\,...\,n)$.

The analogies between the Gaussian and the Mallows distributions can be seen by looking at Fig. 1. In this figure, every permutation in $\sigma \in \mathbb{S}_5$ (the set of 120 permutations of five items) is placed on the x-axis and their probability values for different θ are plotted. The central permutation σ_0 is that with the highest probability value in the middle of the x-axis. For representation purposes, all the individuals at the same distance from σ_0 have been equally divided into two groups and drawn on the x-axis in each side of the central permutation.

Among the many proposals, the GM model [51] is the most popular extension of the Mallows model. This extension requires the metric to be decomposed into $n-1$ terms in such a way that it can be expressed as $D(\sigma, \sigma_0) = \sum_{j=1}^{n-1} S_j(\sigma, \sigma_0)$, where $S_j(\sigma, \sigma_0)$ is related with the j^{th} position of the permutation. The central ranking σ_0 in the GM model is also the permutation with the highest probability mass. However, instead of a single spread parameter θ, the GM model uses

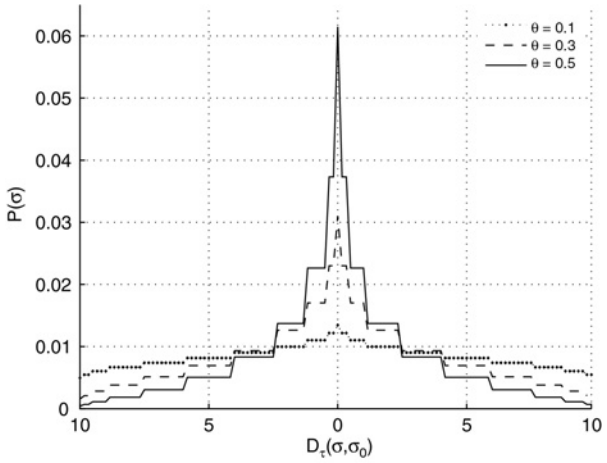

Fig. 1. Probability value for each permutation in $\mathbb{S}_5$ for different values of the spread parameter θ.

$n-1$ spread parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_{n-1})$, each θ_j affecting a particular position j of the permutation. The GM model is thus given by

$$P(\sigma) = \psi(\boldsymbol{\theta})^{-1} exp(\sum_{j=1}^{n-1} -\theta_j S_j(\sigma, \sigma_0)). \tag{4}$$

Because of the right invariance property, $S_j(\sigma, \sigma_0)$ can be rewritten as $S_j(\sigma\sigma_0^{-1}, e) = S_j(\sigma\sigma_0^{-1})$. In the particular case of the Kendall's-τ distance, the S_j terms are denoted by V_j, and are defined as $V_j(\pi) = \sum_{i=j+1}^{n} I[\pi(j) > \pi(i)]$, where $I[cond]$ equals one when the condition $cond$ equals true, and 0 otherwise. Basically, $V_j(\pi)$ is the number of positions of the permutation on the right of j with values smaller than $\pi(j)$. It follows from the definition that $V_j(\pi)$ ranges from 0 to $n - j$ for $1 \leq j < n$.

Taking into account that $D_\tau(\pi) = \sum_{j=1}^{n-1} V_j(\pi)$, (4) can be written as

$$P(\sigma) = \psi(\boldsymbol{\theta})^{-1} exp(\sum_{j=1}^{n-1} -\theta_j V_j(\sigma\sigma_0^{-1})) \tag{5}$$

for every $\sigma \in \mathbb{S}_n$. It is worth noting that one can uniquely determine any permutation π with the $n-1$ integers $V_1(\pi), ..., V_{n-1}(\pi)$ in which the Kendall's-τ distance $D_\tau(\pi)$ decomposes.

Under the uniform distribution, the $V_j(\pi)$ variables that define a permutation are independent and, as a consequence [51], the probability distribution of the random variables $V_j(\sigma\sigma_0^{-1})$ under the GM model given by (5) can be written as

$$P(V_j(\sigma\sigma_0^{-1}) = r_j) = \frac{exp(-\theta_j r_j)}{\psi_j(\theta_j)} \qquad r_j \in \{0, ..., n-j\}. \tag{6}$$

Furthermore, the normalization constant $\psi(\boldsymbol{\theta})$, which if calculated naïvely requires $n!$ sums, can be simplified as the product of $n-1$ terms

$$\psi(\boldsymbol{\theta}) = \prod_{j=1}^{n-1} \psi_j(\theta_j) = \prod_{j=1}^{n-1} \frac{1 - exp(-\theta_j(n-j+1))}{1 - exp(-\theta_j)}. \tag{7}$$

The most important consequence of (6) and (7) is that, when the GM model considers the Kendall's-τ distance, it can be expressed as a multistage ranking model [62]. This means that the process of constructing a permutation has $n-1$ stages and the probabilities at a given stage depend only on the current stage, i.e., at stage j the probability $P(V_j(\pi))$. Equivalently, the value of $V_j(\pi)$, is not affected by the decisions made at previous stages, and in the same way this decision will not affect posterior stages. This property implies that the probability distribution of a given permutation σ with $\boldsymbol{V}(\sigma\sigma_0^{-1}) = (r_1, \ldots, r_{n-1})$ can be factorized as

$$P(\sigma) = \prod_{j=1}^{n-1} P(V_j(\sigma\sigma_0^{-1}) = r_j). \tag{8}$$

A. Learning and Sampling the GM model

In order to introduce the GM model into the framework of EDAs, it is necessary to be able to learn the parameters of the model and sample new solutions from it in an efficient way. In this section we detail how we carry out both processes, which are intended to maintain a tradeoff between time complexity and accuracy.

The most common way of learning the parameters of probability distributions is by means of the maximum likelihood estimation (MLE) principle. The MLE parameters, given a sample of N permutations $\{\sigma_1, \ldots, \sigma_N\}$, are the parameters σ_0 and $\boldsymbol{\theta}$ that maximize the likelihood function (we consider its logarithm)

$$ln\ L(\sigma_1, ..., \sigma_N|\theta, \sigma_0) = ln\ P(\sigma_1, ..., \sigma_N|\boldsymbol{\theta}, \sigma_0) = -N\sum_{j=1}^{n-1}[\theta_j \bar{V}_j - ln(\psi_j(\theta_j))] \tag{9}$$

where

$$\bar{V}_j = \frac{1}{N}\sum_{i=1}^{N} V_j(\sigma_i\sigma_0^{-1}). \tag{10}$$

There are several heuristics and exact methods to simultaneously obtain the parameters $\boldsymbol{\theta}$ and σ_0 [63], [64]. However, for the sake of time efficiency (this process is repeated at each generation inside the EDA), we have divided the learning process into two stages, dealing first with the calculation of the central ranking σ_0 and then with the spread parameters $\boldsymbol{\theta}$.

Maximizing the log-likelihood with respect to σ_0 given a sample of N permutations $\{\sigma_1, \ldots, \sigma_N\}$ is equivalent to minimizing $\sum_{j=1}^{n-1} \theta_j \bar{V}_j(\sigma_i\sigma_0^{-1})$. If all θ_j were equal (the case of the Mallows model) this problem would be the same as that of minimizing the sum of the distances with every permutation in the sample, which is a well known NP-hard problem [65], referred to as consensus ranking. The first stage of the learning process consists of computing an approximation to the consensus ranking by using the Borda algorithm [66], which is carried out in $O(n \times N)$. Borda is a simple yet accurate algorithm that calculates the average permutation of a given sample, which is, in fact, a consistent estimator of the consensus ranking [62]. Its pseudo-code is given in Algorithm 1. In [67] several methods for the consensus ranking problem are compared, concluding that Borda is one of the most efficient algorithms.

Algorithm 1 Borda Algorithm for Calculating $\hat{\sigma}_0$

Input: A sample of N permutations $\{\sigma_1, \ldots, \sigma_N\}$
For j 1 **to** n
  $\pi(j) = \sum_{i=1}^{N} \sigma_i(j)/N$;
end For
$visited = \{\}$;
For j 1 **to** n
  $min\_i \leftarrow \arg\min_i\{\pi(i)|i \notin visited\}$;
  $\hat{\sigma}_0(min\_i) = j$;
  $visited \leftarrow visited \cup min\_i$;
end For
Output: $\hat{\sigma}_0$

Once the consensus ranking $\hat{\sigma}_0$ is approximated, the second stage of the learning process consists of computing the dispersion parameters θ_j for the previously computed $\hat{\sigma}_0$. The maximum likelihood estimators of the dispersion parameters given the central ranking are given by solving

$$\bar{V}_j = \frac{n-1}{exp(\theta_j) - 1} - \frac{n-j+1}{exp(\theta_j(n-j+1)) - 1}. \tag{11}$$

Equation (11) does not have a closed form expression but can be numerically solved by standard iterative algorithms for convex optimization. In this case, we use the Newton–Raphson algorithm. Since the number of iterations performed by this algorithm depends on the initial solution assigned, its time complexity cannot be exactly determined. However, when the initial solution is a good approximation, $O((\log\ h)F(h))$ is close to the real time-complexity. $F(h)$ denotes the cost of calculating the $f(x)/f'(x)$ term with a h-digit precision in the Newton–Raphson algorithm (note that $h << n$).[1] Since we have $n-1$ spread parameters, the overall time-complexity of estimating the θ_j parameters is $O((n-1) \times (\log\ h)F(h))$ [68].

In relation to the sampling process, in (8) we showed that the probability distribution defined by the GM model can be factorized into $n-1$ terms. From a computational point of view, this decomposition turns out to be a very efficient way of sampling the probability distribution, a necessary condition in the context of EDAs. The process of generating the new offspring followed by drawing the $\boldsymbol{V}(\sigma_s\sigma_0^{-1})$ vectors with the distribution in (8) and (6), $1 \leq s \leq N-1$. Then, by applying the algorithm proposed by Meila *et al.* [63], the corresponding permutation $(\sigma_s\sigma_0^{-1})^{-1}$ is obtained (see pseudocode in Algorithm 2). Finally, each permutation σ_s is obtained by inverting and composing with the consensus ranking σ_0, $((\sigma_s\sigma_0^{-1})^{-1})^{-1}\sigma_0 = \sigma_s\sigma_0^{-1}\sigma_0 = \sigma_s$. The overall sampling procedure of a solution is carried out in $O(n^2)$.

In order to illustrate the conversion from $\boldsymbol{V}(\pi)$ to π^{-1}, Fig. 2 introduces an example.

The sampling step in the EDA takes time $O(n^2 \times (M-1))$, because $M-1$ the number of new solutions sampled at each generation. Since the complexity of the sampling step is higher than that of the learning step, Borda-$O(n \times N)$ and

[1]The precision parameter h is set to 10^{-4}.

Algorithm 2 Conversion from $\boldsymbol{V}(\pi)$ to π^{-1}

1: **Input:** A vector of $V(\pi)$
2: Create an empty list π^{-1}
3: Insert n in position 0
4: **For** $j = n - 1 : 1$ in decreasing order
5: Insert j in position V_j of π^{-1}
6: **end For**
7: **Output:** π^{-1}

Algorithm 3 Generalized Mallows EDA

1: $D_0 \leftarrow$ Generate M individuals at random.
2: t $\leftarrow 1$
3: **Do** $\{$
4: $D_{t-1} \leftarrow$ Evaluate individuals.
5: $D_{t-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{t-1}.
6: $\hat{\sigma}_0 \leftarrow$ Estimate the consensus ranking from D_{t-1}^{Se}.
7: $\hat{\boldsymbol{\theta}} \leftarrow$ Estimate the spread parameters from D_{t-1}^{Se} and $\hat{\sigma}_0$.
8: $D_t \leftarrow$ Sample $M - 1$ individuals from $p_t(\sigma) = p(\sigma | D_{t-1}^{Se}, \hat{\sigma}_0, \hat{\boldsymbol{\theta}})$ to create the new population.
9: $\}$**Until** (Stopping criterion is met)

Input: $\boldsymbol{V}(\pi) = (2,\ 0,\ 1)$

insert	$n = 4$	in	0	$\rightarrow$	$\pi^{-1} = (\mathbf{4})$
insert	$j = 3$	in	$V_3(\pi) = 1$	$\rightarrow$	$\pi^{-1} = (4\ \mathbf{3})$
insert	$j = 2$	in	$V_2(\pi) = 0$	$\rightarrow$	$\pi^{-1} = (\mathbf{2}\ 4\ 3)$
insert	$j = 1$	in	$V_1(\pi) = 2$	$\rightarrow$	$\pi^{-1} = (2\ 4\ \mathbf{1}\ 3)$

Output: $\pi^{-1} = (2\ 4\ 1\ 3)$

Fig. 2. Given the $V(\pi)$ vector, permutation π^{-1} is built.

Newton–Raphson- $O((n-1) \times (\log\ h)F(h))$, then the overall time complexity of a generation of the GM-EDA is dominated by this term and is $O(n^2 \times (M-1))$.

IV. Generalized Mallows EDA

GM-EDA is basically a classical EDA where the probabilistic model used is the GM model introduced in the previous section. In Algorithm IV the general outline of the GM-EDA is described.

The behavior of the GM-EDA not only depends on the general parameters of a classical EDA (population size, selection mechanism, stopping criterion, etc.) but also on the specific parameters of the probabilistic model used. In this particular case, the vector $\boldsymbol{\theta}$ is the set of parameters that determines the shape of the GM model, and thus, it is closely related to the behavior of the EDA. Therefore, before applying the algorithm in a general setting, we consider it very valuable to analyze its behavior in relation to these parameters.

In order to understand such behavior, we analyze the evolution of the GM model through the generations. To do that, we carried out a preliminary experiment studying the average value of the vector $\boldsymbol{\theta}$ at each generation. The first instances of Taillard's PFSP benchmark of size 20, 50, and 100 introduced in Section V are used. The general parameters of the GM-EDA are also described in the next section.

According to the results shown in Fig. 3, in the initial generations the average $\boldsymbol{\theta}$ values are close to 0 for all the instances, denoting a diverse population, which is reasonable taking into account the random initialization procedure. As the generations move forward, the average $\boldsymbol{\theta}$ values increase rapidly, reaching values of up to 500. In the smallest size instances—those with 20 or 50 jobs—the average $\boldsymbol{\theta}$ values converge much faster than for instances of 100 jobs. However, in all the cases, the results indicate that, after a certain amount of generations, the convergence of the population accelerates dramatically.

With the aim of understanding the meaning of the average $\boldsymbol{\theta}$ values seen in Fig. 3, we studied the probability assigned to the consensus ranking σ_0 in relation to $\boldsymbol{\theta}$ (the GM model considers $n-1$ spread parameters, but for the sake of simplicity, we set the same value for all θ_j). According to the results shown in Fig. 4, for $\boldsymbol{\theta}$ values below 2, the probability of σ_0 is close to 0, thus encouraging an exploration stage. However, once a threshold is reached, the probability assigned to σ_0 increases quickly, leading the algorithm to an exploitation phase.

In the view of the results in Fig. 4, when θ takes values higher than 10 the probability of σ_0 is almost 1, and Fig. 3 shows that in the first few generations of the EDA, such a threshold is easily reached. In order to prevent the EDA from getting stuck, and to enhance the exploration ability of the model, we decided to set an upper bound value θ_{upper} to the spread parameters so that a trade off between exploitation and exploration is guaranteed.

V. Generalized Mallows EDA for PFSP

As mentioned previously, in order to analyze the performance of the new GM-EDA, we apply our proposal to an extensively studied problem in the literature, the permutation flowshop scheduling problem with respect to the total flowtime criterion. Although several algorithms have been presented in the literature, in order to obtain a real feedback, we compare our approach with the best two state-of-the-art algorithms: AGA [26] and VNS$_4$ [19].

AGA hybridizes a GA with VNS. The algorithm starts initializing the population randomly, with the exception of one solution, which is calculated with the constructive algorithm LR(n/m) [10]. At each iteration, $N/2$ pairs of individuals in the population (N being the population size) are selected uniformly at random and submitted to an asynchronous crossover operator. The asynchronous crossover operator consists of three steps: 1) an *enhanced VNS* (eVNS) is applied to the parent individuals; 2) a twice two-point crossover (TTP) is performed with the resulting solutions of the eVNS; and 3) the individuals obtained from the crossover are again submitted to the eVNS. The asynchronous behavior of the operator is due to the number of improvements permitted in the local search procedures in the eVNS, the second call being much more powerful than the first. The best two individuals obtained from the crossover operator are added to the new population. Additionally, when all the individuals in the population have the same TFT value, the population is restarted randomly, keeping only the best solution in the new population.

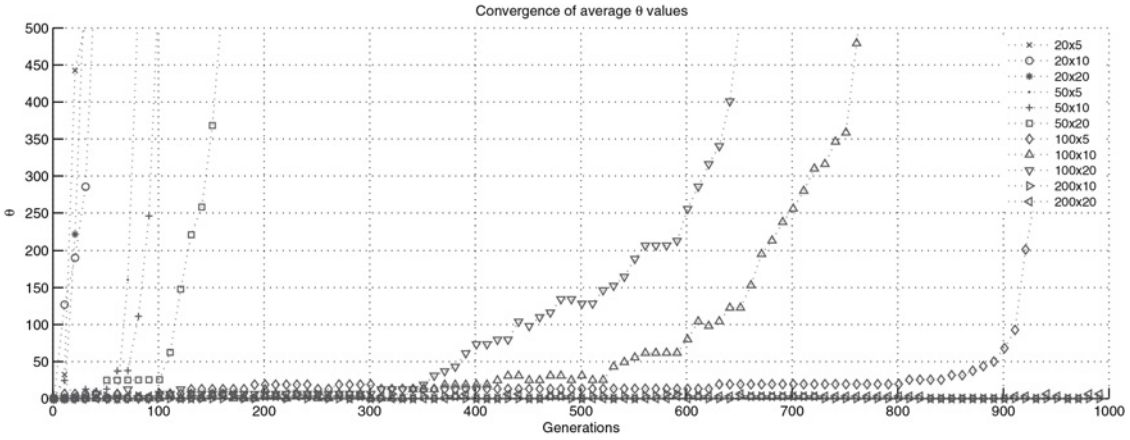

Fig. 3. Convergence of average $\boldsymbol{\theta}$ values for the first instances of the 20 × 5, 20 × 10, 20 × 20, 50 × 5, 50 × 10, 50 × 20, 100 × 5, 100 × 10, and 100 × 20 configurations of Taillard's benchmark.

The recently published paper of Costa *et al.* [19] introduces an exhaustive analysis of different variations of the VNS, concluding that VNS_4 (the version proposed in [19]) is the most successful. VNS_4 explores solutions alternating between the insert neighborhood (considered as the main neighborhood) and the interchange neighborhood. The insert operator considers all those solutions as its neighbors that can be obtained by moving a job into another position. The swap operator considers those solutions as its neighbors that can be obtained by swapping a pair of jobs. When a local optima for both neighborhoods is reached, a shake procedure is performed to escape from that area of the search space. The shake consists of random insert movements of 14 jobs in the individual. In addition, VNS_4 also starts from an initial solution provided by the constructive algorithm LR(n/m) [10].

As it has not been possible to obtain the original codes of AGA and VNS_4 from the authors, we have implemented both proposals. The implementation of AGA and VNS_4 was made honestly by following the indications of the respective papers.[2] All the algorithms were coded in C++ programming language and the experimentation was conducted on a cluster of 20 nodes, each of them equipped with two Intel Xeon X5650 CPUs and 48 GB of memory.

Regarding the experimentation benchmark, AGA, VNS_4, and GM-EDA were tested using Taillard's [69] benchmark. This benchmark gathers 120 instances with the following configurations of n jobs × m machines: 20 × 5, 20 × 10, 20 × 20, 50 × 5, 50 × 10, 50 × 20, 100 × 5, 100 × 10, 100 × 20, 200 × 10, 200 × 20, and 500 × 20 (ten instances of each).

A. Parameter Settings

The parameters employed by GM-EDA are the following.

1) Population size is set to $10n$.
2) Truncation selection mechanism is applied by choosing the best 10% of individuals (n).
3) $10n-1$ individuals are sampled at each new generation.

[2]Both programs, as well as supplementary material can be obtained from http://www.sc.ehu.es/ccwbayes/members/jceberio/GMEDA.html.

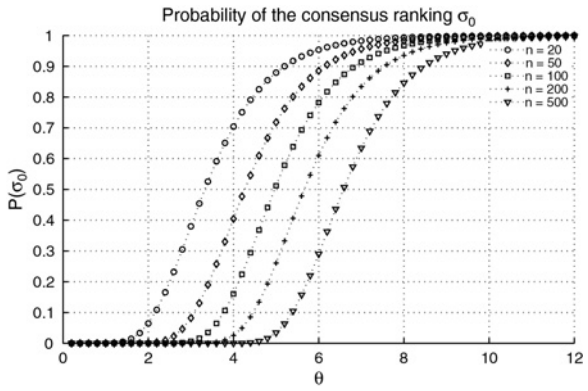

Fig. 4. Probability assigned to σ_0 for different $\boldsymbol{\theta}$ and n values.

4) Elitism criteria is followed.
5) Restart mechanism: as seen in the literature, the elitism criteria and truncation selection implemented in this approach may lead the algorithm to loss diversity in the population. In order to prevent GM-EDA from getting stuck, a restart mechanism is applied when the fitness of all individuals in the population is the same. The restart mechanism generates a new population by applying a shake procedure over the best individual found so far. The shake procedure consists of five random insert movements of jobs around their current position (the five preceding and five succeeding locations are considered). The seed individual used is not added to the population in order to avoid super-individual effects.
6) The upper bound value for $\boldsymbol{\theta}$ was calculated by testing which value of θ_{upper} makes the GM-EDA perform the best over the first instance of each $n \times m$ configuration. Different θ_{upper} between $\theta_{\min}$ and $\theta_{\max}$ in steps of 0.1 were studied [the average value of ten repetitions of the GM-EDA was calculated (see Appendix)].
7) As regards the stopping criterion, AGA authors proposed setting an execution time of $n \times m \times 0.4$ s, as suggested in many other previous works. As this value is strongly related to the hardware used for the experiments and to the quality of the code, we think that the total number

of evaluations is a much better criterion. In order to set a maximum number of evaluations, we run AGA for each instance for $n \times m \times 0.4$ s, recording the number of evaluations (see Appendix). These values were used throughout the experiments as the stopping criterion for GM-EDA, AGA, and VNS$_4$.

The parameters of AGA and VNS$_4$ have been set following the indications of their respective papers (with the exception of the stopping criterion).

B. Taillard's Benchmark Results

Each algorithm–instance pair is run 20 times. The performance measure employed in our study is the average relative percentage deviation (ARPD)

$$ARPD = \left(\sum_{i=1}^{20} \frac{(\text{Algorithm}_i - \text{Best}) \times 100}{\text{Best}} \right) /20$$

with Algorithm$_i$ being the solution found by the algorithm in the ith repetition and Best the best known solution for the instance under study. The ARPD results of the executions are summarized in Table I. Results in bold denote the algorithm that obtained the lowest relative error among the compared approaches. The results marked as (*) denote that, at least once, the best result (also shown in the table) has been achieved. The best results in italics denote new best known solutions.

As can be observed in Table I, the efficiency of the algorithms varies with respect to the size of the problem. For instances of size 20, AGA and VNS$_4$ obtain the optimal results in all the cases. GM-EDA, meanwhile, achieves the optimal result in 22 instances of 30. When comparing instances of 50, 100, 200, and 500, AGA clearly outperforms VNS$_4$ and GM-EDA.[3]

Although the results of GM-EDA for the medium-large instances are noncompetitive, it is worth noting the ability of GM-EDA to achieve optimal solutions for the smallest size instances. Referring to the literature in the field, hybrid approaches are generally the best performing proposals when facing real problems, since they sum the abilities of the combined algorithms.

VI. Hybrid Generalized Mallows EDA

In this section, we introduce the Hybrid GM-EDA (HGM-EDA). The HGM-EDA is proposed as a two-stage algorithm, which in the first stage executes the GM-EDA previously introduced and in the second stage applies a VNS to the solution provided by GM-EDA. Since VNS is the most preferred algorithm seen in the literature with which hybridize, in the following section we introduce the specific design of the VNS that we propose for hybridization.

[3]The results reported by Costa *et al.* [19] do not match those observed in this experimentation. The authors in their paper report better results for VNS$_4$ and worse results for AGA.

Algorithm 4 Variable Neighborhood Search (VNS)

Input: The best solution obtained by GM-EDA, σ_{best}.
  $\sigma^C \leftarrow \sigma_{best}$
Do {
Do {
  $\sigma^S \leftarrow LocalOpt(N_S, \sigma^C)$
  $\sigma^I \leftarrow \arg\min_{\sigma \in N_I(\sigma^S)} f(\sigma)$
  $\sigma^C \leftarrow \sigma^I$
} **Until** (*no improvement*) /* a local optimum is found */
If ($f(\sigma^C) < f(\sigma_{best})$) **Then**
  $\sigma_{best} \leftarrow \sigma^C$
Else
  $\sigma^C \leftarrow \sigma_{best}$
  $Shake(\sigma^C)$
} **Until** (*MaxEvaluations are reached*)
Output: σ_{best}

A. Variable Neighborhood Search

The VNS approach that we propose in this paper explores solutions combining the two most used neighborhoods in the literature: the interchange neighborhood and the insert neighborhood.

The interchange neighborhood (N_S) considers that two permutations are neighbors if one is obtained by interchanging two elements in the other. It is formally defined as

$$N_S(\sigma) = \{\sigma' | \sigma'_k = \sigma_k, \forall k \neq i, j, \sigma'_i = \sigma_j, \sigma'_j = \sigma_i\}. \quad (12)$$

On the other hand, two solutions are neighbors under the insert neighborhood (N_I) if one is obtained by moving an element of the other solution to a different place. It is formally defined as

$$\sigma' \in N_I(\sigma) \Leftrightarrow \exists i, j \in \{1, \ldots, n\} \text{ with } i \neq j \text{ s.t.}$$

$$\begin{cases} (\sigma'_k = \sigma_k,\ k < i \wedge k > j) \wedge, \\ (\sigma'_k = \sigma_{k+1},\ i \leq k < j) \wedge, & \text{when } i < j \\ \sigma'_j = \sigma_i, \\ \\ (\sigma'_k = \sigma_k,\ k < i \wedge k > j) \wedge, \\ (\sigma'_k = \sigma_{k-1},\ i < k \leq j) \wedge, & \text{when } i > j \\ \sigma'_i = \sigma_j. \end{cases} \quad (13)$$

The VNS performs as follows (see pseudo-code in Algorithm 4). In a first step, a greedy local search procedure is carried out using the interchange neighborhood ($LocalOpt(N_S, \sigma^C)$) until a local optimum is reached (σ^S). In the next step, the algorithm chooses the best neighbor of σ^S in N_I.

The algorithm repeats this procedure until a local optimum is found for both neighborhoods. If the obtained solution σ^C is better than that found by the VNS so far, the best solution (σ_{best}) is updated. Next, a shake procedure, similar to that employed in the restart mechanism, is applied (this time ten jobs are shifted instead of five). The whole process is repeated until the maximum number of evaluations is reached.

TABLE I

ARPD RESULTS FOR TAILLARD'S BENCHMARK INSTANCES. RESULTS IN BOLD DENOTE THE APPROACH WITH THE LOWEST ARPD. RESULTS MARKED * DENOTE THE OPTIMUM OR BEST VALUE THAT WAS ACHIEVED AT LEAST ONCE AMONG THE 20 REPETITIONS PERFORMED. BEST RESULTS IN ITALICS DENOTE NEW BEST KNOWN SOLUTIONS

Instance	Best	AGA	VNS_4	GM-EDA	VNS	HGM-EDA	Instance	Best	AGA	VNS_4	GM-EDA	VNS	HGM-EDA
20×05	14033	**0.00** *	**0.00** *	0.18 *	**0.00** *	**0.00** *	100×05	*253713*	0.25	1.21	0.82	0.39	**0.19** *
	15151	**0.00** *	**0.00** *	0.48	**0.00** *	**0.00** *		*242777*	**0.22**	1.71	1.00	0.38	0.27 *
	13301	**0.00** *	**0.00** *	0.50 *	**0.00** *	**0.00** *		*238180*	**0.18** *	1.45	0.80	0.27	0.22
	15447	**0.00** *	**0.00** *	0.43 *	**0.00** *	**0.00** *		*227889*	**0.17** *	1.29	0.78	0.23	0.20
	13529	**0.00** *	**0.00** *	0.21 *	**0.00** *	**0.00** *		*240589*	**0.21**	1.29	0.80	0.31	0.23 *
	13123	**0.00** *	**0.00** *	0.08 *	**0.00** *	**0.00** *		232936	0.22	1.41	0.80	0.31	**0.17** *
	13548	**0.00** *	**0.00** *	0.79	**0.00** *	**0.00** *		*240669*	**0.15** *	1.34	1.00	0.34	0.34
	13948	**0.00** *	**0.00** *	0.18 *	**0.00** *	**0.00** *		*231428*	**0.13**	1.63	0.90	0.25	0.19 *
	14295	**0.00** *	**0.00** *	0.18 *	**0.00** *	**0.00** *		*248481*	0.23	1.48	0.87	0.29	**0.20** *
	12943	**0.00** *	**0.00** *	0.46 *	**0.00** *	**0.00** *		*243360*	**0.15** *	1.40	0.96	0.29	0.24
20×10	20911	**0.00** *	**0.00** *	0.45 *	**0.00** *	**0.00** *	100×10	*299431*	**0.32**	1.52	1.69	0.51	0.33 *
	22440	**0.00** *	**0.00** *	0.54 *	**0.00** *	**0.00** *		*274593*	**0.59**	1.57	2.07	0.87	0.68 *
	19833	**0.00** *	**0.00** *	0.31 *	**0.00** *	**0.00** *		*288630*	**0.34** *	1.53	1.71	0.59	0.35
	18710	**0.00** *	**0.00** *	0.75	**0.00** *	**0.00** *		*302105*	**0.32**	1.60	1.89	0.56	0.35 *
	18641	**0.00** *	**0.00** *	0.35	**0.00** *	**0.00** *		*285340*	0.39	1.42	1.73	0.58	**0.32** *
	19245	**0.00** *	**0.00** *	0.77	**0.00** *	**0.00** *		*270817*	**0.30**	1.64	1.70	0.62	0.33 *
	18363	**0.00** *	**0.00** *	0.48 *	**0.00** *	**0.00** *		*280649*	**0.23**	1.44	1.51	0.51	0.26 *
	20241	**0.00** *	**0.00** *	0.47 *	**0.00** *	**0.00** *		*291665*	**0.35**	1.62	1.88	0.74	0.46 *
	20330	**0.00** *	**0.00** *	0.27 *	**0.00** *	**0.00** *		*302624*	**0.36** *	1.46	1.76	0.64	0.41
	21320	**0.00** *	**0.00** *	0.24 *	**0.00** *	**0.00** *		*292230*	**0.30** *	1.65	1.50	0.52	0.32
20×20	33623	**0.00** *	**0.00** *	0.65 *	**0.00** *	**0.00** *	100×20	*367267*	0.57	1.47	2.03	0.69	**0.44** *
	31587	**0.00** *	**0.00** *	0.29 *	**0.00** *	**0.00** *		*374032*	**0.31** *	1.19	1.80	0.56	0.34
	33920	**0.00** *	**0.00** *	0.04 *	**0.00** *	**0.00** *		*371417*	0.47	1.31	1.93	0.62	**0.36** *
	31661	**0.00** *	**0.00** *	0.28 *	**0.00** *	**0.00** *		*373822*	0.53 *	1.30	1.86	0.67	**0.38**
	34557	**0.00** *	**0.00** *	0.26	**0.00** *	**0.00** *		*370459*	0.42	1.20	1.77	0.56 *	**0.32**
	32564	**0.00** *	**0.00** *	0.30 *	**0.00** *	**0.00** *		*372768*	0.51	1.46	2.17	0.79	**0.41** *
	32922	**0.00** *	**0.00** *	0.61	**0.00** *	**0.00** *		*374483*	0.42	1.54	1.90	0.75	**0.35** *
	32412	**0.00** *	**0.00** *	0.52	**0.00** *	**0.00** *		*385456*	0.46	1.41	1.96	0.66	**0.43** *
	33600	**0.00** *	**0.00** *	0.56 *	**0.00** *	**0.00** *		*376063*	0.43 *	1.32	1.82	0.66	**0.33**
	32262	**0.00** *	**0.00** *	0.41 *	**0.00** *	**0.00** *		*379899*	**0.48**	1.29	2.05	0.64	0.49 *
50×5	64803	**0.05** *	0.78	0.79	0.16	0.12 *	200×10	*1047662*	0.48	1.25	1.19	0.89	**0.17** *
	68062	**0.06** *	0.88	0.94	0.13	0.12		*1036042*	0.91	1.52	1.46	1.04	**0.30** *
	63162	**0.19** *	1.21	1.34	0.35	0.38		*1047571*	0.48	1.44	1.12	0.79	**0.14** *
	68226	**0.17**	1.12	1.27	0.28	0.22 *		*1032095*	0.52	1.40	1.13	0.65	**0.21** *
	69392	**0.09** *	0.87	0.89	0.15 *	0.15		*1037053*	0.62	1.29	1.32	0.87	**0.13** *
	66841	**0.10** *	0.80	0.82	0.23 *	0.18 *		*1006650*	0.50	1.36	1.39	0.98	**0.19** *
	66258	**0.03**	0.74	0.95	0.11 *	0.07 *		*1053390*	0.89	1.60	1.17	1.11	**0.18** *
	64359	**0.05**	0.89	0.97	0.22	0.23 *		*1046246*	0.50	1.38	1.26	0.87	**0.13** *
	62981	**0.09** *	0.83	0.81	0.14 *	0.14 *		*1025145*	0.63	1.43	1.11	0.98	**0.10** *
	68898	**0.07** *	1.05	0.93	0.23 *	0.22		*1031176*	0.78	1.67	1.29	1.09	**0.20** *
50×10	87207	**0.33**	1.12	2.10	0.42	0.38 *	200×20	*1226879*	0.63	1.35	1.59	1.11	**0.26** *
	82820	**0.22** *	1.09	2.45	0.57	0.60 *		*1241811*	0.86	1.46	1.45	1.20	**0.33** *
	79987	**0.23** *	1.07	1.84	0.35 *	0.36		*1266153*	0.84	1.41	1.33	1.00	**0.24** *
	86581	**0.17** *	0.90	1.83	0.28	0.32		*1237053*	0.96	1.55	1.43	1.46	**0.29** *
	86450	**0.14** *	0.90	2.01	0.42	0.38		*1223551*	0.84	1.51	1.64	1.15	**0.25** *
	86637	**0.13** *	0.77	1.55	0.29	0.29		*1225254*	1.01	1.52	1.52	1.29	**0.29** *
	88866	**0.25** *	0.89	1.97	0.51	0.48		*1241847*	0.64	1.26	1.27	1.07	**0.25** *
	86824	**0.19** *	0.94	2.03	0.33	0.36		*1240820*	1.10	1.57	1.57	1.17	**0.36** *
	85526	**0.29**	1.11	2.10	0.49	0.42 *		*1229066*	1.12	1.58	1.48	1.25	**0.27** *
	88077	**0.09** *	0.76	2.00	0.45	0.45		*1247156*	0.92	1.38	1.44	1.14	**0.28** *
50×20	125831	**0.10** *	0.65	1.76	0.30	0.39 *	500×20	6708053	**0.11** *	0.35	8.90	2.09	2.02
	119259	**0.04** *	0.51	1.58	0.32 *	0.22 *		6829668	**0.25** *	0.38	8.58	1.85	1.94
	116459	**0.19** *	0.73	2.24	0.56	0.44 *		6747387	**0.24** *	0.41	8.46	2.04	2.04
	120712	**0.22** *	0.61	1.92	0.40	0.34		6787054	**0.26** *	0.45	8.75	1.96	1.89
	118184	**0.40**	0.86	2.30	0.50	0.52 *		6755257	**0.39**	0.41 *	8.72	2.01	1.92
	120703	**0.19** *	0.62	1.78	0.42	0.35		6751496	**0.19** *	0.42	8.58	2.15	2.13
	122962	**0.38**	0.71	2.10	0.47	0.47 *		6708860	**0.27** *	0.45	9.15	2.06	2.05
	122489	**0.16**	0.75	2.24	0.45	0.55 *		*6769821*	**0.31** *	0.58	8.62	2.24	2.09
	121872	**0.17**	0.76	1.79	0.40	0.37 *		6720474	**0.15** *	0.46	8.69	1.99	1.91
	124064	**0.23** *	0.90	1.95	0.45	0.42		6767645	**0.19** *	0.44	8.51	2.12	2.00

B. Experimentation

In order to study the performance of HGM-EDA, we repeat the previous experimentation.

1) *Parameter Tuning:* Since HGM-EDA is a two stage algorithm, the number of evaluations assigned to each stage is set to half of the maximum allowed. In addition, the number of restarts allowed for GM-EDA has been limited to $10n$ (these parameters have been decided without any previous experiment). In the case that the criterion on restarts is fulfilled before half of the evaluations have run, the remaining evaluations will be performed by the VNS as well. The rest of the parameters are set as described in Section V.

2) *Taillard's Benchmark Results:* We evaluate AGA, VNS_4, GM-EDA, VNS (the version used in the hybrid approach), and HGM-EDA algorithms over the instances of Taillard's benchmark. Each *algorithm - instance* pair is run 20 times. The best result and ARPD values are summarized in Table I.

According to those results, from the 120 Taillard's benchmark instances tested, HGM-EDA finds the best TFT solution for 85 instances (for which 48 are new best known solutions), AGA for 70 and VNS_4 for 31. Regarding the ARPD results, AGA is the best in 88 instances, HGM-EDA in 62, and VNS_4 in 30 out of 120 instances.

In order to state whether there exist statistical differences among the results, we applied a nonparametric Friedman's test to the best TFT results obtained by the AGA, VNS_4, and HGM-EDA algorithms among the 20 repetitions for each $n \times m$ configuration. A level $\alpha = 0.05$ of significance was set. The statistical test reported significant differences between the algorithms for some configuration types. Therefore, a post-hoc method was used to carry out all pairwise comparisons and determine which algorithms stand out from the rest of the approaches. In particular, Shaffer's static procedure is used, as suggested for such cases in [70]. Again, the significance level was fixed to $\alpha = 0.05$. The statistical analysis confirmed the following points.

1) No significative difference was found for instances of size 20 between AGA, VNS_4, and HGM-EDA.
2) AGA and HGM-EDA perform better than VNS_4 for instances with 50 and 100 jobs. However, no statistical differences were found to distinguish between AGA and HGM-EDA.
3) HGM-EDA is the best algorithm for solving instances of 200 jobs, followed by AGA, with VNS_4 in third place.
4) Finally, AGA and VNS_4 outperform HGM-EDA for instances of size 500. However, the test does not determine which of them is better.

To demonstrate that the contribution of GM-EDA is essential for an efficient performance of the hybrid approach, VNS was also evaluated independently in this experiment. The obtained results prove that, when run separately, each stage of the hybrid approach, GM-EDA and VNS, does not perform as well as when they are run combined.

3) *Random Benchmark Results:* From the previous experiments, it can be observed that HGM-EDA performs better as the size of the problem enlarges. However, for problems of size 500, HGM-EDA does not work as expected. In order to analyze this behavior, we decided to create new instances to fill the gap between 200 and 500. 100 new instances were proposed with the following configurations: 250 $\times$ 10, 250 $\times$ 20, 300 $\times$ 10, 300 $\times$ 20, 350 $\times$ 10, 350 $\times$ 20, 400 $\times$ 10, 400 $\times$ 20, 450 $\times$ 10, and 450 $\times$ 20 (ten instances for each configuration).[4] The processing times of the jobs were generated uniformly at random (integer values between 1 and 100 as reported by Taillard [69]).

We executed 20 independent runs. θ_{upper} and the maximum number of evaluations for the stopping criterion were calculated following the same procedure explained for Taillard's benchmark (see the Appendix).

According to the results (see Table II), HGM-EDA obtains the best TFT solution for 67 instances and AGA for 33. On the other hand, VNS_4 was not successful in any of the cases. Regarding the ARPD, HGM-EDA is the best in 61 instances out of 100 and AGA in 39.

In order to state whether there exist statistical differences among the algorithms, we applied the same statistical test as before, obtaining the following results.

1) HGM-EDA is the best algorithm for solving instances of 250, 300, and 350 jobs, followed by AGA, with VNS_4 in third place.
2) AGA and HGM-EDA outperform VNS_4 for instances of 400 $\times$ 10. However, no statistical differences were found to distinguish between AGA and HGM-EDA.
3) For configurations 400 $\times$ 20 and 450 $\times$ 10, AGA is statistically the best, followed by VNS_4, and HGM-EDA in third position.
4) Finally, AGA and VNS_4 outperform HGM-EDA for instances of 450 $\times$ 20. However, the test does not determine which of them is better.

C. Additional Experimentation

As previously mentioned, in this paper we considered optimizing PFSP with respect to the TFT due to its complexity and relevance for the current production environments. However, taking into account that the makespan is also an interesting criterion when optimizing the PFSP, we decided to carry out some additional executions with GM-EDA and HGM-EDA for the Taillard's PFSP instances with the makespan criterion. The results were summarized in a short report and uploaded to the website as additional material.

Considering that the complexity of the proposed algorithm is higher than that of its competitors, we carried out some additional experiments using the execution time as the stopping criterion. A report with the results and some conclusions was uploaded at the Website as additional material.

VII. Discussion

In the previous section, we saw that HGM-EDA is a strong alternative to optimize PFSP. However, the unexpected performance observed for the largest instances deserves a

[4]Both Taillard's benchmark and random benchmark can be obtained from http://www.sc.ehu.es/ccwbayes/members/jceberio/GMEDA.html.

TABLE II

ARPD RESULTS FOR TAILLARD'S BENCHMARK INSTANCES. RESULTS IN BOLD DENOTE THE APPROACH WITH THE LOWEST ARPD. RESULTS MARKED * DENOTE THE OPTIMUM OR BEST VALUE THAT WAS ACHIEVED AT LEAST ONCE AMONG THE 20 REPETITIONS PERFORMED

Instance	Best	AGA	VNS_4	HGM-EDA	Instance	Best	AGA	VNS_4	HGM-EDA
250×10	1566623	0.65	1.53	**0.15** *	350×20	3462591	1.03	1.11	**0.23** *
	1589117	0.89	1.56	**0.25** *		3472888	0.70	0.97	**0.41** *
	1633818	0.64	1.48	**0.22** *		3448696	0.79	0.98	**0.20** *
	1603824	1.03	1.57	**0.23** *		3487674	0.34	0.85	**0.33** *
	1652808	0.60	1.26	**0.19** *		3422898	0.85	1.11	**0.36** *
	1568866	0.38	0.65	**0.14** *		3451601	0.71	1.10	**0.39** *
	1592812	0.86	1.43	**0.16** *		3482315	0.52	0.92	**0.31** *
	1552918	0.66	1.46	**0.17** *		3448931	0.93	1.25	**0.41** *
	1594956	0.76	1.30	**0.13** *		3471376	1.04	1.12	**0.34** *
	1592563	0.91	1.41	**0.19** *		3432934	1.00	1.29	**0.35** *
250×20	1847471	0.71	1.40	**0.21** *	400×10	3933307	**0.18** *	0.51	0.81
	1892483	1.01	1.38	**0.17** *		3910048	**0.18**	0.71	0.36 *
	1879796	0.61	1.13	**0.17** *		3934557	**0.10** *	0.52	0.53
	1852134	0.90	1.32	**0.18** *		3875265	0.62	1.23	**0.36** *
	1836747	0.76	1.37	**0.28** *		3932228	**0.15**	0.34	0.43 *
	1859562	0.86	1.27	**0.21** *		3913491	**0.18** *	0.42	0.98
	1876186	0.99	1.38	**0.23** *		3865710	**0.19**	0.47	0.47 *
	1850963	0.87	1.46	**0.27** *		3918728	**0.34**	0.73	0.80 *
	1859994	0.98	1.41	**0.18** *		3903109	**0.38**	0.69	0.62 *
	1837633	0.89	1.25	**0.22** *		3934891	**0.12**	0.44	0.56 *
300×10	2241165	0.73	1.32	**0.09** *	400×20	4465942	**0.34** *	0.48	1.11
	2221209	0.81	1.47	**0.17** *		4442588	**0.26** *	0.57	1.36
	2239369	0.79	1.34	**0.16** *		4419467	**0.25** *	0.49	0.97
	2216046	0.57	1.05	**0.19** *		4450298	**0.42** *	0.67	1.32
	2255074	0.75	1.17	**0.13** *		4380600	**0.32** *	0.66	1.29
	2259913	0.76	1.31	**0.15** *		4460906	**0.23** *	0.50	1.02
	2244410	0.77	1.36	**0.18** *		4472922	**0.50** *	0.89	1.20
	2203072	0.84	1.17	**0.21** *		4438057	**0.16** *	0.49	1.19
	2245325	0.83	1.22	**0.21** *		4419300	**0.15** *	0.52	1.20
	2239058	0.61	1.23	**0.08** *		4402417	**0.19** *	0.48	1.02
300×20	2586158	1.01	1.39	**0.27** *	450×10	4870929	**0.26** *	0.70	1.67
	2569577	0.91	1.21	**0.29** *		4884041	**0.16** *	0.48	1.75
	2588574	0.96	1.39	**0.23** *		4930649	**0.19** *	0.59	0.98
	2584806	0.73	1.23	**0.15** *		4933807	**0.02** *	0.06	1.48
	2601077	0.91	1.28	**0.20** *		4934094	**0.12** *	0.27	1.66
	2605560	1.06	1.47	**0.32** *		4922342	**0.09** *	0.51	1.37
	2593450	0.88	1.10	**0.21** *		4931140	**0.09** *	0.50	1.43
	2556756	0.99	1.55	**0.35** *		4933524	**0.11** *	0.55	1.37
	2583311	1.13	1.32	**0.27** *		4906973	**0.13** *	0.66	1.39
	2625824	0.75	1.03	**0.24** *		4976526	**0.11** *	0.57	1.63
350×10	3051090	0.69	1.28	**0.15** *	450×20	5532585	**0.30** *	0.39	1.78
	2996234	0.56	0.86	**0.11** *		5545906	**0.33** *	0.62	1.76
	2977597	1.26	1.34	**0.24** *		5584602	**0.28** *	0.41	1.36
	3026374	0.87	1.37	**0.18** *		5588641	**0.20** *	0.22	1.61
	3031433	0.84	1.49	**0.16** *		5556343	**0.26** *	0.52	1.73
	3023478	0.72	1.32	**0.17** *		5507299	**0.26** *	0.73	2.02
	3037375	0.68	1.20	**0.14** *		5534081	**0.41** *	0.60	1.85
	3078709	0.70	0.88	**0.14** *		5580460	**0.20** *	0.43	1.60
	3029894	0.82	1.35	**0.17** *		5558982	**0.37** *	0.30	1.49
	3014352	0.71	1.26	**0.21** *		5559057	**0.15** *	0.06	1.28

thorough study. In this section we go deeper into this topic and propose methods to improve the efficiency of HGM-EDA.

A. Why Does the Performance of HGM-EDA Decrease for the Largest Instances?

In Table II, we can see that the fitness difference between HGM-EDA and AGA increases from configuration 400 × 20 onward, which favors AGA. Since HGM-EDA is based on GM-EDA, we analyze two aspects of HGM-EDA: the performance of GM-EDA against AGA and the contribution of each stage of the hybrid approach to the optimization.

To study the performance of GM-EDA versus AGA, in Fig. 5 the ARPD between GM-EDA and AGA is introduced. This time, for each instance i within the set of instances of a given configuration, we consider the best result of the 20 runs performed by GM-EDA and AGA, called $GM\text{-}EDA_i^{\text{best}}$ and AGA_i^{best}, respectively. Then, we calculate the ARPD for each $n \times m$ configuration (note that each configuration contains a set of ten instances)

$$\text{ARPD} = \left(\sum_{\text{i=1}}^{10} \frac{(\text{GMEDA}_\text{i}^{\text{best}} - \text{AGA}_\text{i}^{\text{best}}) \times 100}{\text{AGA}_\text{i}^{\text{best}}} \right) /10. \quad (14)$$

According to the results, GM-EDA shows a regular ARPD below 0.02 with respect to AGA up to configuration 350 × 10. The low deviation described for such instances coincides with the good performance of HGM-EDA. However, the results

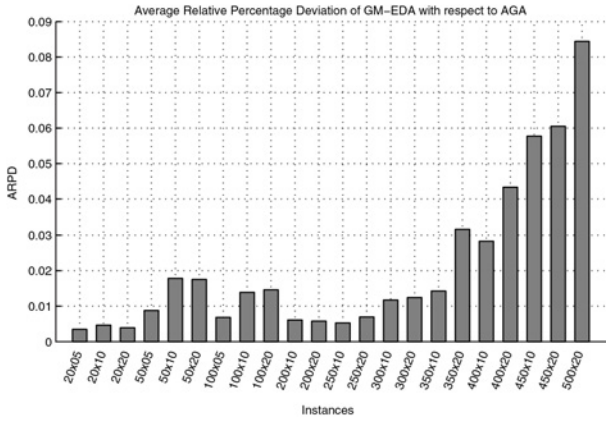

Fig. 5. Average relative percentage deviation between the best results obtained by GM-EDA and AGA in the ten instances of each configuration.

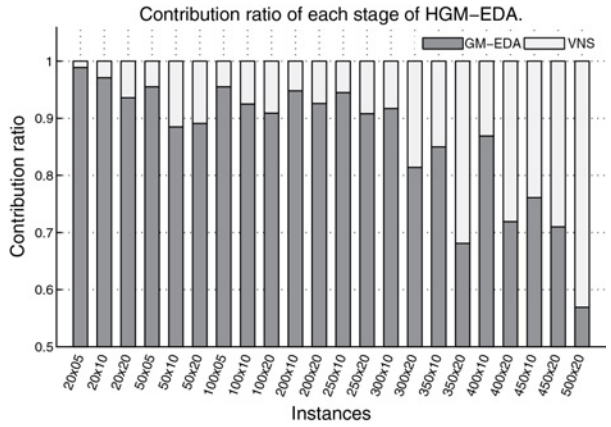

Fig. 6. Contribution ratios performed by each stage of HGM-EDA: GM-EDA and VNS. Ratios were calculated averaging results obtained from 20 independent runs over the first instance of each configuration $n \times m$. Note that the ratio axis is only shown from 0.5 to 1.

for configurations from 400 × 20 onward show a dramatic increase of ARPD. In fact, the largest differences in Fig. 5 match the poor results seen for HGM-EDA for such instances.

The results observed suggest that HGM-EDA relays its performance to the ability of GM-EDA to find promising solutions. As HGM-EDA is a two-stage algorithm, in the following lines we propose a study of the contribution ratio of each stage on the final solution. For this analysis, we consider three particular solutions from HGM-EDA: the best individual of the first population of GM-EDA (Sol_{init}), the solution returned by GM-EDA when the first stage finishes ($\text{Sol}_{\text{GMEDA}}$), and the final solution returned by the VNS in the second stage of the HGM-EDA ($\text{Sol}_{\text{final}}$). The contribution ratios (CR) are computed as $CR_{\text{GMEDA}} = (Sol_{\text{init}} - Sol_{\text{GMEDA}})/Sol_{\text{final}}$ and $CR_{\text{VNS}} = (Sol_{\text{GMEDA}} - Sol_{\text{final}})/Sol_{\text{final}}$. The ratios were calculated averaging the results obtained from 20 independent runs over the first instance of each configuration instance set. The results are shown in Fig. 6.

The plotted results support the conclusion that we had come to previously: GM-EDA is the stage of the algorithm that conditions the performance of HGM-EDA. While improvement ratios for instances between 20 and 200 jobs show a regular contribution around 0.9 for GM-EDA, the results for the instances from 400 × 20 onward are significantly lower.

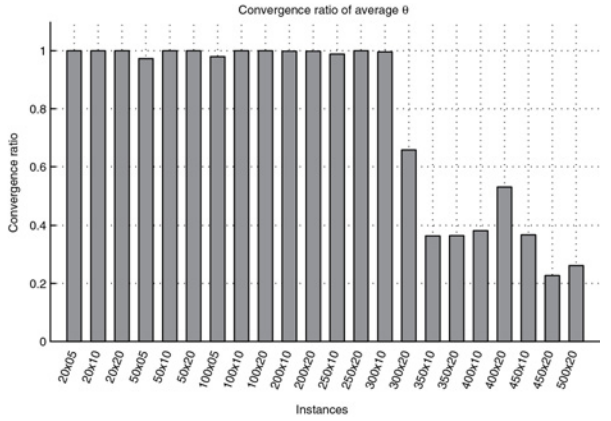

Fig. 7. Convergence ratio of average $\boldsymbol{\theta}$ values to θ_{upper} at the end of the execution. First instance of each configuration type was selected.

B. Why Does GM-EDA Become Less Competitive Than AGA for the Largest Instances?

Since the GM model determines the effect of GM-EDA, we analyze the convergence of the probabilistic model when optimizing the largest instances. As mentioned in Section IV, the vector $\boldsymbol{\theta}$ determines the exploration or intensification behavior of the GM model when optimizing. Therefore, in this new experiment, we examine the average value of the dispersion parameters $\boldsymbol{\theta}$ obtained at the end of the executions. In this case, the first instance of each configuration $n \times m$ was studied. For comparison purposes, the results are normalized to the range [0, 1] by dividing the average $\boldsymbol{\theta}$ calculated for each instance, with the θ_{upper} set in each case.

According to the results shown in Fig. 7, in instances up to 300 × 10, the average $\boldsymbol{\theta}$ converged to the θ_{upper} set. However, in larger instances, the average $\boldsymbol{\theta}$ values do not achieve values higher than 0.7 in each case. This means that the convergence of the model did not even reach 70% of the upper bound fixed to each configuration type.

This experiment suggests that the GM model learned was in an exploration phase (taking into account Fig. 4) when stopped, instead of intensifying around the promising area.

Alternatively, examining the initialization of AGA in detail, we realized that the constructive heuristic employed, LR(n/m) (also in VNS_4), has a decisive contribution when solving large-scale instances. Some preliminary experiments showed that the performance of AGA and VNS_4 was significantly worse when LR(n/m) was removed from the procedure.

In order to state what the real advantage of including LR(n/m) is, we compare the intermediate result calculated by LR(n/m) as the first stage of AGA and that of GM-EDA. We define the ARPD between LR and GM-EDA as

$$\text{ARPD} = \left(\sum_{\text{i}=1}^{10} \frac{(\text{LR}_{\text{i}} - \text{GMEDA}_{\text{i}}) \times 100}{\text{GMEDA}_{\text{i}}} \right) /10. \qquad (15)$$

(GMEDA_i denotes the average of the fitness value obtained by the GM-EDA in the 20 repetitions.) The results are shown in

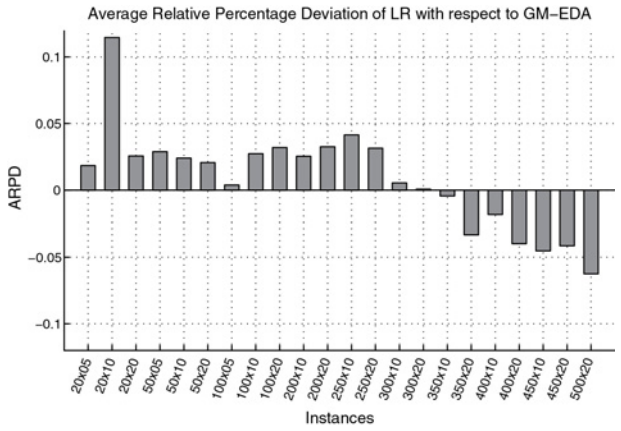

Fig. 8. Average relative percentage deviation of LR with respect to GM-EDA performed at the first stage of HGM-EDA (the average fitness value of 20 repetitions was selected). The ARPD of the ten instances of each configuration was calculated. Note that the ARPD axis is shown from -0.12 to 0.12. Instances with histograms above 0 denote that the solutions obtained by GM-EDA are, on average, better than those achieved by LR(n/m) and the opposite when the results obtained are below 0.

Fig. 8. The values used to calculate the ARPD are the average values of 20 repetitions over each instance.

The results report that the performance of GM-EDA is better than that of LR(n/m) for instances up to 400×10. However, for larger instances LR(n/m) is better. The results suggest that the poor performance seen for larger instances might also be motivated by the random initialization of the initial population of GM-EDA.

C. Guided HGM-EDA

Previous analysis suggests that the random initialization and the lack of enough evaluations could be the main reasons for the unexpected performance of HGM-EDA in the largest instances (instances from 350×20 onward). Inspired by the state-of-the-art algorithms that we compared it with, we propose a method for guiding the initialization of the first population, based on LR(n/m).

As with every constructive algorithm, LR(n/m) measures the preference to append a job to the current partial solution. For each nonscheduled job, the heuristic calculates an index value that considers the increase of the TFT when the nonscheduled job is appended to the partial solution, and the increase of the TFT to the solution by the averaged processing times of the remaining jobs. The job with the lowest index value is selected to be appended to the partial solution.

In this case, in order to guide the initialization of the first population, we propose a modification of the LR(n/m) to transform the constructive heuristic into a nondeterministic proposal. A nondeterministic proposal allows the presumable provision of a different seed individual at each repetition of the algorithm. The modification proposed replaces the top-ranking job selection by sampling a probability distribution inversely proportional to the index value of the jobs in the ranking.

In order to analyze the impact of the guided initialization and the premature stopping of GM-EDA, we compare AGA, HGM-EDA, and Guided HGM-EDA over the first instance of the configurations 400×20, 450×10, 450×20, and 500×20. The algorithms are run from one to eight times, the maximum number of evaluations specified in Appendix. The results are shown in Fig. 9.

It can be seen that Guided HGM-EDA achieves better results than HGM-EDA and outperforms the results of AGA when extending the number of evaluations employed. The successful results achieved by Guided HGM-EDA against AGA and HGM-EDA support the potential of the EDA introduced in this paper and confirm the determinant effect of implementing a guided initialization when dealing with large size instances.

TABLE III

θ_{upper} VALUES AND MAXIMUM NUMBER OF EVALUATIONS FOR TAILLARD'S BENCHMARK

Instance	$\theta_{\min}$	$\theta_{\max}$	θ_{upper}	Evaluations
20×05	1.0	3.0	1.5	182224100
20×10	1.0	3.0	1.4	224784800
20×20	1.0	3.0	1.4	256896400
50×05	2.5	5.5	3.7	220712150
50×10	2.5	5.5	2.8	256208100
50×20	2.5	5.5	3.0	275954150
100×05	3.5	6.0	4.9	235879800
100×10	3.5	6.0	3.7	266211000
100×20	3.5	6.0	4.7	283040000
200×10	4.0	6.0	5.3	272515500
200×20	4.0	6.0	5.5	287728850
500×20	4.0	7.0	4.4	260316750

TABLE IV

θ_{upper} VALUES AND MAXIMUM NUMBER OF EVALUATIONS FOR THE RANDOM BENCHMARK

Instance	$\theta_{\min}$	$\theta_{\max}$	θ_{upper}	Evaluations
250×10	4.0	7.0	5.2	267779100
250×20	4.0	7.0	4.4	284574350
300×10	4.0	7.0	4.6	273847500
300×20	4.0	7.0	5.2	284672900
350×10	4.0	7.0	6.6	278369000
350×20	4.0	7.0	7.0	286225300
400×10	4.0	7.0	5.5	275491800
400×20	4.0	7.0	5.0	283913500
450×10	4.0	7.0	4.0	277455350
450×20	4.0	7.0	6.7	269271450

VIII. CONCLUSION

In this paper, we presented a novel algorithm for dealing with permutation-based problems, called GM-EDA. GM-EDA introduces a probabilistic distance-based ranking exponential model. As this manuscript is the first serious attempt to introduce an EDA of such characteristics, in order to demonstrate the efficiency of the proposed algorithm, it was tested on the well-known permutation flowshop scheduling problem. In particular, a hybrid version of the GM-EDA was proposed (GM-EDA + VNS) to obtain state-of-the-art results.

The experiments carried out showed that the hybrid version of the GM-EDA (HGM-EDA) is the best algorithm for solving the PFSP-TFT. From 220 instances tested (Taillard's and random benchmarks), HGM-EDA obtained the best solution

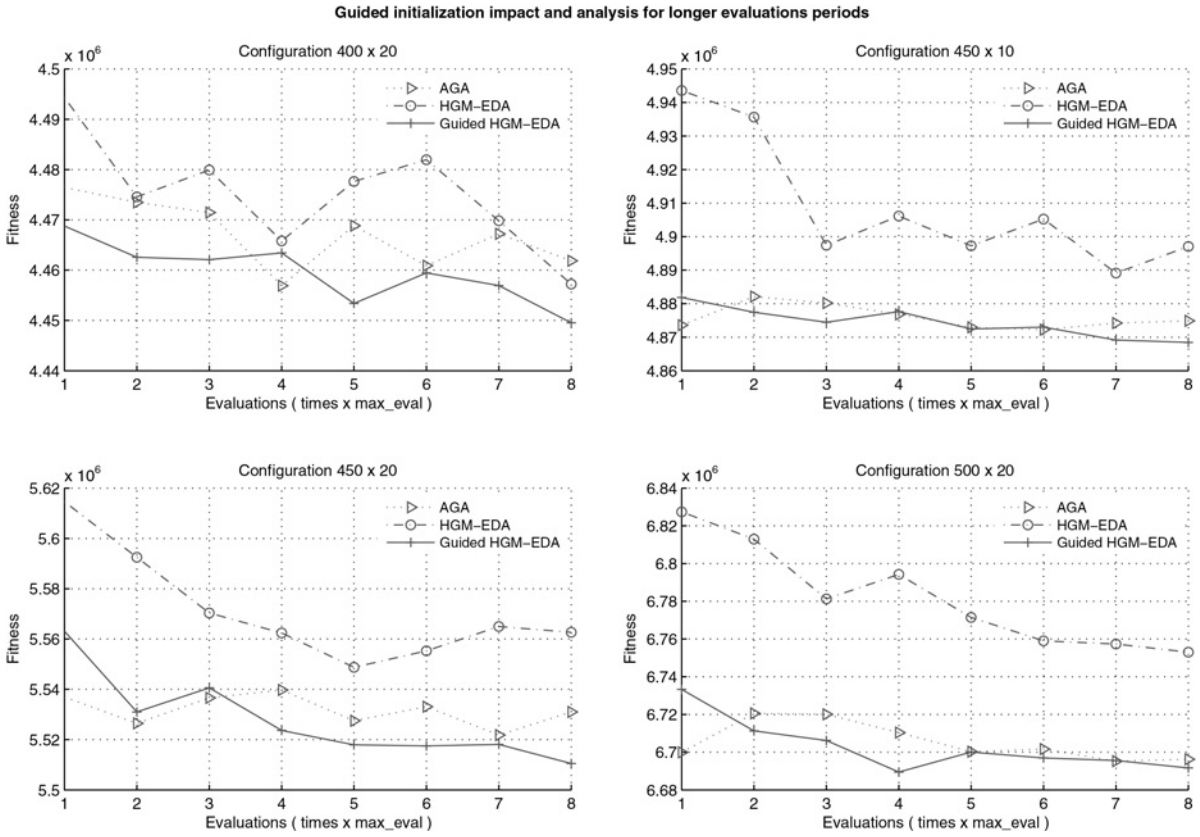

Fig. 9. Average fitness results of ten runs of AGA, HGM-EDA, and guided HGM-EDA over the first instance of the configurations 400 × 20, 450 × 10, 450 × 20, and 500 × 20. The algorithms have been run for one to eight times the maximum number of evaluations (max_*eval*) specified in the Appendix.

for 152 instances. The analysis of the results concluded that the success of HGM-EDA is due to the ability of GM-EDA to find optimal regions in the search space. Furthermore, the experiments proved that employing a heuristic for initializing the algorithm is preferable than a random initialization, mainly when solving large size instances.

Additional experiments revealed that GM-EDA and HGM-EDA are the most time-consuming algorithms among the studied approaches. However, further executions demonstrated that HGM-EDA is still very competitive compared with its competitors when they are executed with the same running time.

The incorporation of the GM model to the framework of EDAs leaves many trends that are worth extending in future work. In the following paragraphs we will briefly mention some of them.

As introduced previously, the GM model is defined by two parameters: the consensus ranking σ_0 and the vector of spread parameters $\boldsymbol{\theta}$. In Section III, we highlighted the difficulty of calculating these parameters exactly, and we proposed approximation methods that permit us to use the GM model in EDAs. However, we believe that there is still room for improvement. More advanced methods for calculating the σ_0 or self-adapting procedures for the estimation of $\boldsymbol{\theta}$ may lead GM-EDA to improve its efficiency.

Another feature of the GM model that deserves a deeper study is that related to how the unimodal property of the model behaves when solving problems/instances with many global optima. In this sense, mixtures of Mallows models [71] might be considered to optimize different areas of the landscape separately.

In addition, in this proposal, Kendall's τ distance was considered due to its decomposable properties, which allow us to factorize the GM model, and thus, learn and sample the model efficiently. However, there exist many other metrics that could be developed, such as Cayley, Ulam, or Hamming [72]. In fact, finding the best distance for approaching a certain permutation problem is challenging research.

The inclusion of the information of the instance into the GM model is another interesting issue. Due to the characteristics of the GM model, the inclusion procedure of external information into the model is relatively easy. However, finding the relevant information in the instance is a very hard task.

Regarding the approach for solving the PFSP, there are also other points that should be analyzed to obtain better results: the percentage of evaluations used in each stage of the HGM-EDA, the effect of developing more elaborate population initialization methods, the shake method used in the restart scheme, or the selection of the neighborhoods for the proposed VNS algorithm as a part of HGM-EDA.

Finally, with the aim of ratifying these initial results, we consider it interesting to test the GM-EDA on a wider set of permutation-based problems, such as the quadratic assignment problem or the linear ordering problem.

APPENDIX
EXECUTION PARAMETERS

Tables III and IV show the maximum number of evaluations and θ_{upper} values used in the experimentation for Taillard's and random benchmarks, respectively. θ_{upper} values have been obtained testing different values of θ, starting at θ_{min} and ending at θ_{max}, in steps of 0.1. Note that only the first instance of each configuration was used to look for the most adequate parameter values.

REFERENCES

[1] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Res. Logis. Quart.*, vol. 1, no. 1, pp. 61–68, 1954.

[2] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flow shop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, May 1976.

[3] S. R. Hejazi and S. Saghafianz, "Flowshop-scheduling problems with makespan criterion: A review," *Int. J. Prod. Res.*, vol. 43, no. 14, pp. 2895–2929, Feb. 2005.

[4] J. Gupta and J. E. Stafford, "Flow shop scheduling research after five decades," *Eur. J. Oper. Res.*, vol. 169, no. 3, pp. 699–711, 2006.

[5] J. M. Framinan and R. Leisten, "An efficient constructive heuristic for flowtime minimisation in permutation flow shops," *Omega*, vol. 31, no. 4, pp. 311–317, 2003.

[6] C. Wang, C. Chu, and J.-M. Proth, "A branch-and-bound algorithm for n-job two machine flow shop scheduling problems," in *Proc. INRIA/IEEE Symp. Emerging Technol. Factory Autom.*, vol. 2. Oct 1995, pp. 375–383.

[7] C.-S. Chung, J. Flynn, and O. Kirca, "A branch and bound algorithm to minimize the total flow time for m-machine permutation flowshop problems," *Int. J. Prod. Econom.*, vol. 79, no. 3, pp. 185–196, Oct. 2002.

[8] H.-S. Woo and D.-S. Yim, "A heuristic algorithm for mean flowtime objective in flowshop scheduling," *Comput. Oper. Res.*, vol. 25, no. 3, pp. 175–182, 1998.

[9] M. Nawaz, E. E. Enscore, Jr., and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[10] J. Liu and C. R. Reeves, "Constructive and composite heuristic solutions to the p//[summation operator]ci scheduling problem," *Eur. J. Oper. Res.*, vol. 132, no. 2, pp. 439–452, 2001.

[11] C. Rajendran, "Heuristic algorithm for scheduling in a flowshop to minimize total flowtime," *Int. J. Prod. Econom.*, vol. 29, no. 1, pp. 65–73, Feb. 1993.

[12] C. Rajendran and H. Ziegler, "An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs," *Eur. J. Oper. Res.*, vol. 103, no. 1, pp. 129–138, Nov. 1997.

[13] X. Li, Q. Wang, and C. Wu, "Efficient composite heuristics for total flowtime minimization in permutation flow shops," *Omega*, vol. 37, no. 1, pp. 155–164, 2009.

[14] Y. Zhang, X. Li, J. Zhu, and Q. Wang, "Composite heuristic algorithm for permutation flowshop scheduling problems with total flowtime minimization," in *Proc. 12th Int. Conf. Comput. Supported Cooperative Work Des.*, Apr. 2008, pp. 903–907.

[15] A. Allahverdi and T. Aldowaisan, "New heuristics to minimize total completion time in m-machine flowshops," *Int. J. Prod. Econom.*, vol. 77, no. 1, pp. 71–83, 2002.

[16] J. M. Framinan, R. Leisten, and R. Ruiz-Usano, "Comparison of heuristics for flowtime minimisation in permutation flowshops," *Comput. Oper. Res.*, vol. 32, pp. 1237–1254, May 2005.

[17] X. Dong, H. Huang, and P. Chen, "An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion," *Comput. Oper. Res.*, vol. 36, no. 5, pp. 1664–1669, 2009.

[18] J. K. Pawel Jan Kalczynski, "On the NEH heuristic for minimizing the makespan in permutation flow shops," *Omega*, vol. 35, no. 1, pp. 53–60, 2007.

[19] W. E. Costa, M. C. Goldbarg, and E. G. Goldbarg, "New VNS heuristic for total flowtime flowshop scheduling problem," *Expert Syst. Applicat.*, vol. 39, no. 9, pp. 8149–8161, 2012.

[20] I. Osman and C. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989.

[21] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *Eur. J. Oper. Res.*, vol. 47, no. 1, pp. 65–74, Jul. 1990.

[22] E. Nowicki and C. Smutnicki, "A fast tabu search algorithm for the permutation flow-shop problem," *Eur. J. Oper. Res.*, vol. 91, no. 1, pp. 160–175, May 1996.

[23] C. R. Reeves, "Improving the efficiency of tabu search for machine sequencing problems," *J. Oper. Res. Soc.*, vol. 44, no. 4, pp. 375–382, Apr. 1993.

[24] V. S. Vempati, C.-L. Chen, and S. F. Bullington, "An effective heuristic for flow shop problems with total flow time as criterion," *Comput. Ind. Eng.*, vol. 25, no. 1–4, pp. 219–222, 1993.

[25] M. S. Nagano, R. Ruiz, and L. A. N. Lorena, "A constructive genetic algorithm for permutation flowshop scheduling," *Comput. Ind. Eng.*, vol. 55, pp. 195–207, Aug. 2008.

[26] X. Xu, Z. Xu, and X. Gu, "An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization," *Expert Syst. Applict.*, vol. 38, no. 7, pp. 7970–7979, 2011.

[27] L.-Y. Tseng and Y.-T. Lin, "A hybrid genetic local search algorithm for the permutation flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 198, no. 1, pp. 84–92, Oct. 2009.

[28] Y. Zhang, X. Li, and Q. Wang, "Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization," *Eur. J. Oper. Res.*, vol. 196, no. 3, pp. 869–876, Aug. 2009.

[29] M. F. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 1930–1947, 2007.

[30] C.-J. Liao, C.-T. Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Comput. Oper. Res.*, vol. 34, pp. 3099–3111, Oct. 2007.

[31] B. Jarboui, S. Ibrahim, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimization for solving permutation flowshop problems," *Comput. Ind. Eng.*, vol. 54, no. 3, pp. 526–538, 2008.

[32] H. Liu, L. Gao, and Q. Pan, "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem," *Expert Syst. Appl.*, vol. 38, pp. 4348–4360, Apr. 2011.

[33] T. Stützle, "An ant approach to the flow shop problem," in *Proc. 6th Eur. Congr. Intell. Tech. Soft Comput*, 1997, pp. 1560–1564.

[34] C. Rajendran and H. Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs," *Eur. J. Oper. Res.*, vol. 155, no. 2, pp. 426–438, 2004.

[35] Y. Gajpal and C. Rajendran, "An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops," *Int. J. Prod. Econom.*, vol. 101, no. 2, pp. 259–272, Jun. 2006.

[36] Y. Zhang, X. Li, Q. Wang, and J. Zhu, "Similarity based ant-colony algorithm for permutation flowshop scheduling problems with total flowtime minimization," in *Proc. Int. Conf. Comput. Supported Cooperat. Work Des.*, 2009, pp. 582–589.

[37] B. Jarboui, M. Eddaly, and P. Siarry, "An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems," *Comput. OR*, vol. 36, no. 9, pp. 2638–2646, 2009.

[38] P. Larrañaga and J. A. Lozano, *Estimation of Distribution algorithms: A New Tool for Evolutionary Computation*. Berlin, Germany: Kluwer Academic Publishers, 2002.

[39] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, "Toward a new evolutionary computation: Advances on estimation of distribution Algorithms," in *Studies in Fuzziness and Soft Computing*. Secaucus, NJ, USA: Springer, 2006.

[40] M. Pelikan, K. Sastry, and E. Cantú-Paz, "Scalable optimization via probabilistic modeling: From algorithms to applications," in *Studies in Computational Intelligence*. Secaucus, NJ, USA: Springer Inc., 2006.

[41] R. Santana, P. Larrañaga, and J. A. Lozano, "Protein folding in simplified models with estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 418–438, Aug. 2008.

[42] Q. Zhang, J. Sun, and E. Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 192–200, Apr. 2005.

[43] Q. Zhang, J. Sun, E. Tsang, and J. Ford, "Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem," *Stud. Fuzziness Soft Comput.*, vol. 192/2006, pp. 281–292, 2006.

[44] L. Wang and C. Fang, "An effective estimation of distribution algorithm for the multimode resource-constrained project scheduling problem," *Comput. Oper. Res.*, vol. 39, no. 2, pp. 449–460, Feb. 2012.

[45] Q.-K. Pan and R. Ruiz, "An estimation of distribution algorithm for lot-streaming flow shop problems with setup times," *Omega*, vol. 40, no. 2, pp. 166–180, 2012.
[46] S.-H. Chen and M.-C. Chen, "Addressing the advantages of using ensemble probabilistic models in estimation of distribution algorithms for scheduling problems," *Int. J. Prod. Econom.*, vol. 141, no. 1, pp. 24–33, 2012.
[47] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems," *Progr. Artif. Intell.*, vol. 1, no. 1, pp. 103–117, Jan. 2012.
[48] S. Tsutsui, "Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram," in *Proc. Parallel Prob. Solv. Nat.*, 2002, pp. 224–233.
[49] S. Tsutsui, M. Pelikan, and D. E. Goldberg, "Node histogram versus edge histogram: A comparison of PMBGAs in permutation domains," Medal, Tech. Rep., 2006.
[50] J. Ceberio, A. Mendiburu, and J. A. Lozano, "Introducing the Mallows model on estimation of distribution algorithms," in *Proc. Int. Conf. Neural Inform. Process.*, vol. 7063, Nov. 2011, pp. 461–470.
[51] M. A. Fligner and J. S. Verducci, "Distance based ranking models," *J. Royal Stat. Soc.*, vol. 48, no. 3, pp. 359–369, 1986.
[52] K. Baker, *Introduction to Sequencing and Scheduling*. New York, NY, USA: Wiley, 1974.
[53] A. Allahverdi, C. Ng, T. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, 2008.
[54] L. L. Thurstone, "A law of comparative judgment." *Psych. Rev.*, vol. 34, no. 4, pp. 273–286, 1927.
[55] C. L. Mallows, "Nonnull ranking models." *Biometrika*, vol. 44, no. 1–2, pp. 114–130, 1957.
[56] T. Lu and C. Boutilier, "Learning mallows models with pairwise preferences," in *Proc. 28th Int. Conf. Mach. Learn.*, ser. ICML '11, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, Jun. 2011, pp. 145–152.
[57] W. Cheng and E. Hüllermeier, "A simple instance-based approach to multilabel classification using the mallows model," in *Proc. Workshop Learn. MultiLabel Data*, Sep. 2009, pp. 28–38.
[58] G. Lebanon and J. Lafferty, "Cranking: Combining rankings using conditional probability models on permutations," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 363–370.
[59] Y. Mao and G. Lebanon, "Nonparametric modeling of partially ranked data," *J. Mach. Learn. Res.*, vol. 9, pp. 2401–2429, 2008.
[60] M. Meila and H. Chen, "Dirichlet process mixtures of generalized Mallows models," *Uncertainty Artif. Intell.*, vol. 1, no. l, pp. 358–367, 2010.
[61] K. K. Lau, P. C. Yuen, and Y. Y. Tang, "Directed connection measurement for evaluating reconstructed stroke sequence in handwriting images," *Pattern Recogn.*, vol. 38, no. 3, pp. 323–339, 2005.
[62] M. Fligner and J. Verducci, "Multistage ranking models," *J. Amer. Stat. Assoc.*, vol. 83, no. 403, pp. 892–901, 1988.
[63] M. Meila, K. Phadnis, A. Patterson, and J. Bilmes, "Consensus ranking under the exponential model," in *Proc. 22nd Conf. Uncertainty Artif. Intell.*, Vancouver, BC, USA, Jul. 2007, pp. 285–294.
[64] B. Mandhani and M. Meila, "Tractable search for learning exponential models of rankings," *J. Mach. Learn. Res.*, vol. 5, pp. 392–399, 2009.
[65] J. Bartholdi, C. A. Tovey, and M. A. Trick, "Voting schemes for which it can be difficult to tell who won the election," *Soc. Choice Welfare*, vol. 6, no. 2, pp. 157–165, Apr. 1989.
[66] J. C. Borda, "Memoire sur les elections au scrutin," *Histoire de l'Academie Royale des Science*, 1784.
[67] A. Ali and M. Meila, "Experiments with Kemeny ranking: What works when?" *Math. Soc. Sci.*, vol. 64, no. 1, pp. 28–40, Jul. 2011.
[68] J. M. Borwein and P. B. Borwein, *Pi and the AGM: A Study in the Analytic Number Theory and Computational Complexity*. New York, NY, USA: Wiley, 1987.
[69] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, Jan. 1993.
[70] S. Garcia and F. Herrera, "An extension on statistical comparisons of classifiers over multiple data set" for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, pp. 2677–2694, Dec. 2008.
[71] T. B. Murphy and D. Martin, "Mixtures of distance-based models for ranking data," *Comput. Stat. Data Anal.*, vol. 41, no. 3–4, pp. 645–655, Jan. 2003.
[72] M. Deza and T. Huang, "Metrics on permutations: A survey," *J. Combinatorics, Inf. Syst. Sci.*, vol. 23, pp. 173–185, 1998.

Josu Ceberio received the M.Sc. degree in computer science from the University of the Basque Country, Spain, in 2009 and is pursuing the Ph.D. degree from the Department of Computer Science and Artificial Intelligence, University of the Basque Country.

His current research interests include evolutionary computation, probability models on rankings, and permutation-based problems.

Ekhine Irurozki received the M.Sc. degree in computer science from the University of the Basque Country, Spain, in 2008. She is currently a Ph.D. student of the Intelligent Systems Group at the same university.

Her current research interests include combinatorial optimization and machine learning. She is particularly interested in probability distributions on the space of permutations and is currently working on the development of new techniques for efficiently learning and sampling permutations of that kind of model.

Alexander Mendiburu received the Ph.D. degree from the University of the Basque Country, Spain, in 2006.

Since 1999, he has been a Lecturer with the Department of Computer Architecture and Technology, University of the Basque Country. His current research interests include evolutionary computation, probabilistic graphical models, and parallel computing.

Jose A. Lozano (M'05) received the Ph.D. degree from the University of the Basque Country, Spain, in 1998.

Since 2008, he has been a Full Professor with the Department of Computer Science and Artificial Intelligence, University of the Basque Country, where he leads the Intelligent Systems Group. He has edited three books and has published over 60 refereed journal papers. His current research interests include evolutionary computation, machine learning, data mining, pattern analysis, probabilistic graphical models, and bioinformatics.

Prof. Lozano is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and a member of the editorial boards of the *Evolutionary Computation Journal*, *Soft Computing*, and three other journals.