Information Sciences 397–398 (2017) 137–154

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Multiobjective decomposition-based Mallows Models estimation of distribution algorithm. A case of study for permutation flowshop scheduling problem

Murilo Zangari[a,*], Alexander Mendiburu[b], Roberto Santana[c], Aurora Pozo[a]

[a] Computer Science Department, Federal University of Paraná (UFPR), Brazil. PO 19081, ZIP Code: 81531-970, Curitiba, Brazil
[b] Intelligent Systems Group, Department of Computer Architecture and Technology, University of the Basque Country (UPV/EHU), Paseo Manuel de Lardizabal 1, 20080 San Sebastián, Guipúzcoa, Spain
[c] Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country (UPV/EHU), Paseo Manuel de Lardizabal 1, 20080 San Sebastián, Guipúzcoa, Spain

ARTICLE INFO

Article history:
Received 20 July 2016
Revised 30 January 2017
Accepted 15 February 2017
Available online 20 February 2017

Keywords:
Estimation of distribution algorithm
Mallows models
Multi-objective optimization
Decomposition-based
Permutation optimization problems
Flowshop scheduling problem

ABSTRACT

Estimation of distribution algorithms (EDAs) have become a reliable alternative to solve a broad range of single and multi-objective optimization problems. Recently, distance-based exponential models, such as Mallows Model (MM) and Generalized Mallows Model (GMM), have demonstrated their validity in the context of EDAs to deal with permutation-based optimization problems. The aim of this paper is two-fold. First, we introduce a novel general multi-objective decomposition-based EDA using Kernels of Mallows models (MEDA/D-MK framework) for solving multi-objective permutation-based optimization problems. Second, in order to demonstrate the validity of the MEDA/D-MK, we have applied it to solve the multi-objective permutation flowshop scheduling problem (MoPFSP) minimizing the total flow time and the makespan. The permutation flowshop scheduling problem is one of the most studied problems of this kind due to its fields of application and algorithmic challenge. The results of our experiments show that MEDA/D-MK outperforms an improved MOEA/D variant specific tailored for minimizing makespan and total flowtime. Furthermore, our approach achieves competitive results compared to the *best-known* approximated Pareto fronts reported in the literature for the benchmark considered.

1. Introduction

Several real-world problems can be stated as multi-objective optimization problems (MOPs), which have two or more objectives to be optimized at the same time. Very often these objectives conflict with each other. If there is no preference a priori between the objectives, then a single solution can not optimize all the objectives simultaneously. Besides, there exist a set of optimal solutions representing a *trade-off* between the objectives. The concept of Pareto dominance [40] is frequently used to define a set of non-dominated solutions. In the most MOPs, it is very time consuming (if not impossible) to obtain the complete set of optimal solutions [13,35].

* Corresponding author.
E-mail addresses: murilo.zangari@gmail.com (M. Zangari), alexander.mendiburu@ehu.es (A. Mendiburu), roberto.santana@ehu.es (R. Santana), aurora@inf.ufpr.br (A. Pozo).

http://dx.doi.org/10.1016/j.ins.2017.02.034
0020-0255/

Since the early nineties, much effort has been devoted to developing Multi-objective Evolutionary Algorithms (MOEAs) for solving MOPs. MOEAs aim at finding a set of representative Pareto-optimal solutions in a single run. One particular class of difficult problems is that in which the feasible solutions are defined on a space of permutations. Different permutation-based optimization problems reflect several real-world problems in different fields of application, such as engineering, computer science, finance, industry, etc. Behind the combinatorial nature of the solutions, every permutation-based problem can present its particular challenges, which need to be faced [6].

Over time, several approaches (including exact methods, heuristics, and metaheuristics) have been proposed to deal with permutation-based optimization problems [36,51]. Metaheuristics such as Simulated Annealing (SA) [4], Genetic Algorithms (GA) [19], Tabu Search (TS) [18] have shown their potentiality to solve permutation problems [51].

Recently, Estimation of Distribution Algorithms (EDAs) [26,38] have become a reliable approach to solve single-objective permutation problems [6,7,9,47,48]. Based on machine learning techniques, at each generation, EDAs learn a probabilistic model associated with the set of most promising solutions found so far, trying to express the interrelation between the variables of the problem explicitly. New solutions are obtained by sampling the probabilistic model learned. The sampled solutions are then used to update the population. The algorithm stops when a particular criterion is met, such as a maximum number of generations.

Moreover, EDAs have been placed in the context of multi-objective optimization [41]. Regarding the strategies used to maintain the set of non-dominated solutions (such as Pareto-based), multi-objective EDAs (MoEDAs) are similar to MOEAs, but they incorporate learning and sampling steps instead of genetic operators. Over the years, several MoEDAs have been presented in the literature [3,23,33,41,52,55].

Besides the useful application of EDAs to different domains and real-world problems [5,30,34], they were only partially successful when applied to permutation-based optimization problems [6]. This question was properly addressed in [9], where probabilistic models defined directly on the space of permutations were introduced in the context of EDAs for single-objective problems.

In that work [9], a *distance-based exponential probability model* called Mallows Model (MM) was included in the framework of EDAs, and later used to solve single-objective permutation-based problems. The MM is considered analogous to the Gaussian probability distribution over the space of permutations. Ceberio et al. [7] proposed the use of the Generalized Mallows Model (GMM) in the context of EDAs (GM-EDA), and it outperformed the state-of-the-art algorithms for the permutation flowshop scheduling problem (PFSP) minimizing the total flow time. In addition, Kernels of Mallows Models [10] and Mixtures of Mallows Models [11] were proposed to deal with multi-modal problems.

While the application of probabilistic models defined on permutations has shown its potential for single-objective optimization problems, they have not been tested for MOPs. Thus, in this paper, we propose a multi-objective Mallows Model EDA. Our framework involves two main ingredients: (i) the well-known Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) framework [54], which decomposes a MOP into a number of scalar single-objective optimization subproblems, and evolves them simultaneously in a cooperative manner using the concept of neighborhood. The approach has been useful for solving several MOPs [46]; and (ii) the use of Kernels of Mallows Models [10], which are able to model heterogeneous (multi-modal) populations. In Mallows Kernel EDA, a model is learned for every selected solution.

The main motivation behind combining these approaches is that, while the decomposition-based method deals with the search in different regions of the objective space, the learning and sampling steps of the Kernels of Mallows Models can efficiently produce promising solutions through the generations. It is worth noting that this is the first time that a MoEDA based on the use of a Mallows Model is proposed. The algorithm is called Multi-objective Decomposition-based Mallows Kernel EDA (MEDA/D-MK framework).

Additionally, in order to test the potentiality of MEDA/D-MK, we applied it to the multi-objective permutation flowshop scheduling problem (MoPFSP) minimizing the makespan and the total flow time. The problem has been proved to be *NP-hard*, and it is one of the most studied permutation optimization problems due to its significance in both academic and real-world application [51]. Two components are incorporated into MEDA/D-MK for MoPFSP to enhance its performance: (i) the constructive $LR(n/m)$ algorithm [31] to initialize the population, and (ii) a controlled perturbation procedure aiming to escape from local optima.

We have used 110 benchmark instances [37] for the experimental studies. We have compared MEDA/D-MK to an improved MOEA/D variant (which applies genetic operators, specially tailored for makespan and total flowtime) [1,12]. We have implemented a unified framework where both algorithms can be instantiated. Moreover, the only difference between them is the way in which the *variation* (reproduction) step is performed. The results indicate that MEDA/D-MK outperforms MOEA/D in a significant way. Furthermore, we have compared our approach to a set of approximated Pareto fronts (*PFs*) produced by state-of-the-art algorithms [16,37]. The results show that MEDA/D-MK achieves competitive results, mainly for the large-scale problems.

The remainder of the paper is organized as follows. The next section presents the core concepts of multi-objective optimization and the decomposition approach. Section 3 introduces the proposed MEDA/D-MK framework and describes the Mallows Model in detail. In Section 4, we present the target problem and the MEDA/D-MK algorithm configuration for MoPFSP. We present the experimental study in Section 5. Finally, in Section 6, we present the conclusions and future work.

2. Background

2.1. Multi-objective optimization

A general MOP, for a discrete domain and the minimization of the objectives, can be defined as follows [13]:

$$\begin{aligned} &minimize\ F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_q(\mathbf{x})) \\ &subject\ to\ \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

where $\mathbf{x}$ is a solution composed by n decision variables, Ω is the discrete *decision space*, $F(\mathbf{x})$ consists of q objective functions $(f_1(\mathbf{x}), \ldots, f_q(\mathbf{x}))$, and $\mathbb{R}^q$ is the q-dimensional *objective space* where $F(\mathbf{x}) : \Omega \rightarrow \mathbb{R}^q$. Every solution in the *decision space* gives a certain *point* in the *objective space*, which determines a quality of this solution in terms of the objective function values [13]. Pareto optimality [40] is used to define the set of optimal solutions.

Pareto optimality [40]: Let $\mathbf{x}, \mathbf{y} \in \Omega$, $\mathbf{x}$ is said to *dominate* $\mathbf{y}$ if and only if $f_l(\mathbf{x}) \leq f_l(\mathbf{y})$ for all $l \in \{1, .., q\}$ and $f_l(\mathbf{x}) < f_l(\mathbf{y})$ for at least one l. A solution $\mathbf{x}^* \in \Omega$ is called *Pareto optimal* if there is no other $\mathbf{x} \in \Omega$ which *dominates* $\mathbf{x}^*$. The set of all the *Pareto* optimal solutions is called the *Pareto set (PS)* and the solutions mapped in the *objective space* are called *Pareto front (PF)*, i.e., $PF = \{F(\mathbf{x}) | \mathbf{x} \in PS\}$. In many real-life applications, the *PF* is of great interest to decision makers for understanding the trade-off nature of the different objectives and selecting their preferred final solution.

While the focus of this paper is on the solution of permutation-based MOPs using MOEAs, MOPs have been also addressed using other approaches such as mathematical programming [44] and multiple criteria decision making [53].

The primary goal of MOEAs is to produce a number of non-dominated solutions that approximate to the true *PF* in a single run. Different methods to maintain a population of non-dominated solutions have been used. Popular classes of MOEAs include [28]: (i) *Pareto* dominance-based; (ii) Indicator-based and (iii) Decomposition-based.

Since the last decade, algorithms such as MOGLS [22] and a wide range of algorithms based on the MOEA/D framework [46,54] have been proposed for solving different continuous and combinatorial MOPs with two and three objectives successfully. The MOEA decomposition approach for MOPs is suitable for multi-modal problems, exploring different regions of the search space. Some works in the literature have shown its efficiency regarding the quality of the outcome, compared to some Pareto-based algorithms such as NSGA-II and SPEA2 [1,24,54]. Therefore, here we deal only with decomposition based approaches.

2.2. The decomposition approach

The decomposition approach uses a scalar aggregation function and a set of weight vectors to decompose a MOP into a number of single objective optimization subproblems.

Weighted sum and *Tchebycheff* are two of the most used scalar aggregation functions [35] in the MOEA decomposition method.

Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_q)^T$ be a weight vector, where $\sum_{l=1}^{q} \lambda_l = 1,\ \lambda_l \geq 0$ for all $l = \{1, \ldots, q\}$.

Weighted Sum Approach: The optimal solution to the following scalar single-optimization problem is defined as:

$$\begin{aligned} &minimize\ g^{ws}(\mathbf{x}|\boldsymbol{\lambda}) = \sum_{l=1}^{q} \lambda_l f_l(\mathbf{x}) \\ &subject\ to\ \mathbf{x} \in \Omega \end{aligned} \tag{2}$$

Tchebycheff approach: The optimal solution to the following scalar single-optimization problem is defined as:

$$\begin{aligned} &minimize\ \ g^{tc}(\mathbf{x}|\boldsymbol{\lambda}, \mathbf{z}^*) = \max_{1 \leq l \leq q} \{\lambda_l |f_l(\mathbf{x}) - z_l^*|\} \\ &subject\ to\ \mathbf{x} \in \Omega \end{aligned} \tag{3}$$

where $\mathbf{z}^* = (z_1^*, \ldots, z_q^*)^T$ is the reference point, i.e., $z_l^* = \min\{f_l(\mathbf{x}) | \mathbf{x} \in \Omega\}$ for each $l = \{1, \ldots, q\}$.

In the standard MOEA/D [54], a MOP is decomposed into N scalar aggregation scalar subproblems, and each one is associated with a weighted vector $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^N\}$. A neighborhood relation among the subproblems is defined to evolve the population in a collaborative manner. The neighborhood for each subproblem k, is defined as $B(k) = \{\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^T\}$ where T is the neighborhood size. The distance between any two weight vectors is defined according to the Euclidean distance between them. The relationship of the neighbor subproblems is used for the *selection* of parent solutions and the *replacement* (also called *update mechanism*). The size of the neighborhood T plays a vital role in MOEA/D to exchange information among the subproblems [46].

3. The framework: multi-objective decomposition-based Kernel of Mallows Model EDA

In this section, we present one of the aims of this paper: the proposal of a general multi-objective EDA combining the decomposition-based method and the probabilistic models defined directly in the space of permutations for solving

permutation-based MOPs efficiently. To the knowledge of the authors, this is the first time that an algorithm of this kind has been proposed.

In the following, we describe the MEDA/D-MK framework. Subsequently, we present the Mallows Model.

3.1. The main framework

Let $\boldsymbol{\sigma}$ denote a permutation solution and $\sigma(i)$ represent the ith position (variable) of the permutation.

A simple way to incorporate probabilistic models into decomposition-based MOEAs is to maintain a probabilistic model for each scalar subproblem k, using the $B(k)$ as the selected population. Thus, in the MEDA/D-MK framework, each subproblem k is associated to the weighted vector $\boldsymbol{\lambda}^k$ and $Pop = \{\sigma^1, \ldots, \sigma^N\}$ is the set of N current solutions. Algorithm 1 presents

Algorithm 1: MEDA/D-MK framework.

```
Initialize N uniformly distributed weight vectors, and then generate N initial permutation solutions Pop = {σ^1, ..., σ^N}
randomly or by a problem-specific procedure;
Compute every F(σ^k)
Initialize EP with the non-dominated solutions from Pop
while a termination condition is not met do
    for each subproblem k ∈ 1, ..., N do
        σ_0^k, θ^k ← Learn a Mallows Model using B(k);
        σ_s^k ← Sample a new solution from P^k(σ);
        F(σ_s^k): Compute its objective quality functions;
        Update Pop: Pop = UpdateNeighbors(σ_s^k, n_r);
        Optionally, apply a perturbation in σ^k after a number of generations without it be improved;
    end for
    Update the EP: UpdateEP(Pop, EP)
end while
Return Pop, EP
```

the MEDA/D-MK pseudo-code. In the following, we describe its steps. In addition, Table 1 summarizes the parameters required by the algorithm.

1. **Initialization:** First, the N uniform distributed weight vectors λ^1, .., λ^N are set. Usually, as in [54], the weight vectors are generated using a control parameter H, where an element of each weight vector (λ_l^k) is one of the $\{0/H, 1/H, \ldots, H/H\}$ and $C_{H+q-10}^{q-1} = N$ denotes the number of weight vectors, being q the number of objectives. Next, the Euclidean distance between any two weight vectors λ^k, λ^j is computed. For each subproblem k, the set of neighbors $B(k)$ is set with the T closest neighbors. The initial population $Pop = (\sigma^1, \ldots, \sigma^N)$ is generated in a random way or by a problem-specific method. Then, their corresponding fitness functions $F(\sigma^1), \ldots, F(\sigma^N)$ are computed. The external *Pareto EP* is initialized with the non-dominated solutions from the initial population *Pop*.
2. **Learning and Sampling:** The framework, at each generation, for each subproblem k, learns a Mallows Model by estimating the parameters θ^k and σ_0^k (described in Section 3.2). Next, it samples a new solution σ_s^k and computes its objective function $F(\sigma_s^k)$.
3. **Update the population:** Next, the sampled solution σ_s^k is used to update the current *Pop*. For each index $r \in B(k)$, if $g(\sigma_s^k|\lambda^r) \leq g(\sigma^r|\lambda^r)$, then σ^r is replaced by σ_s^k. A new solution σ_s^k can update a maximum of n_r solutions. Furthermore, different updating schemes can be used, such as the global update, where a new solution can update any solution in *Pop*.
4. A drawback of most approaches dealing with multi-objective permutation optimization problems is the lack of diversity, i.e., the search ability to explore different regions of the search space while moving towards the *PF*. Therefore, option-

Table 1
Summary of the input parameters for the Algorithm 1.

Parameter	Description
H	Control parameter to generate the weight vectors as in [54]
N	Number of subproblems
λ^k	The weight vector associated to the subproblem k
T	Neighborhood size
θ	The spread parameter from Mallows Models
σ_0	The central permutation from Mallows Models
n_r	maximum replacements by a new solution

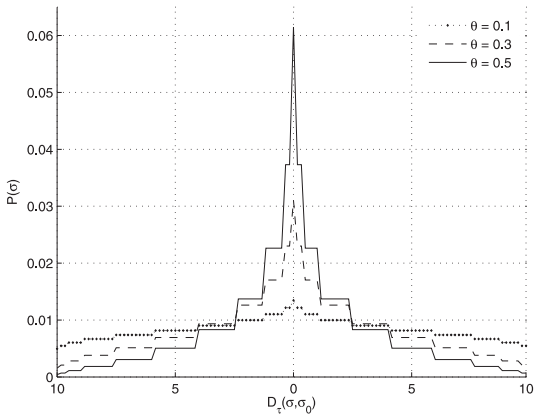


Fig. 1. The Mallows Model exponential probability distribution of a permutation σ with a spread parameter θ under a distance-metric D [9].

ally, a destruction (perturbation) method can be performed in σ^k if it does not change after a maximum number of generations. This scheme is performed to escape from local optima and, thus, to control the diversity of the population.

5. **Update the *EP*:** Finally, the *EP* is updated with the non-dominated solutions from *Pop* according to the Pareto dominance, i.e., if no solution in *EP* dominates $F(\sigma^k)$, add $F(\sigma^k)$ to *EP* and remove from it all the solutions dominated by $F(\sigma^k)$.
6. **Stop condition:** When the stop condition is met, the algorithm returns *Pop* and *EP*.

In the following, we describe the Mallows Model and the *learning* and *sampling* mechanisms incorporated into our framework.

3.2. Mallows Model

The Mallows Model [32] is a distance-based exponential probability model defined over permutation spaces. 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 two parameters: a spread parameter θ, and the distance to a central permutation σ_0, which is calculated by a distance-metric D. Formally, the Mallows Model is defined as

$$P(\sigma) = \frac{e^{-\theta D(\sigma,\sigma_0)}}{\psi(\theta)} \tag{4}$$

where $\psi(\theta)$ denotes the normalization constant.

Fig. 1 presents different exponential probabilities ($P(\sigma)$) according to different spread parameter θ values and a distance $D(\sigma, \sigma_0)$. When $\theta > 0$, the central permutation σ_0 is the permutation with the highest probability value, and the probability of the other $n! - 1$ permutations exponentially decreases with the distance to σ_0. The larger the θ, the sharper the distribution around σ_0. When $\theta = 0$, a uniform distribution is obtained, i.e., the equation assigns equal probability to every permutation σ in $\mathbb{S}_n$.

The most common distance metrics used with the Mallows Model are the Kendall's$_\tau$ distance [9,17] and the Cayley distance [20]. Ceberio et al. [8], reported that Cayley distance [20] has achieved the best results for the single-objective PFSP using the total flow time as the optimization criterion. Therefore, we only describe the Cayley distance in this paper. For additional information about the Kendall-τ and Ulam distance the reader is referred to [9] and [21] respectively.

The Cayley distance $D_c(\sigma, \pi)$, counts the minimum number of swaps (not necessary adjacent) that have to be performed to transform σ into π. When the reference permutation is the identity, $e = 123\ldots n$, the number of swaps equals n minus the number of *cycles* of σ. A cycle is understood as a closed walk of elements in the permutation such that for every $1 \leq i, j \leq n$ for which $\sigma(i) = j$ is true, then i and j are in the same cycle. In addition, the Cayley distance fulfills the *right-invariant* property, which means that $D(\sigma, \pi) = D(\sigma\gamma, \pi\gamma)$ 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 $\gamma = \pi^{-1}$, $D(\sigma, \pi)$ can be simplified as $D(\sigma\pi^{-1}, e)$ (also denoted as $D(\sigma\pi^{-1})$).

The Cayley distance $D_c(\sigma)$ can be decomposed as the sum of $n - 1$ terms: $D_c(\sigma, \pi) = \sum_{j=1}^{n-1} X_j(\sigma\pi^{-1})$ where $X_j(\sigma\pi^{-1}) = 0$ if j is the largest item in its cycle in $\sigma\pi^{-1}$, and 1 otherwise.

Under this distance, the normalization constant $\psi(\theta)$, is formalized as:

$$\psi(\theta) = \prod_{j=1}^{n-1} \psi_j(\theta) = \prod_{j=1}^{n-1} (n - j)\exp(-\theta) + 1 \tag{5}$$

For the incorporation of the MM as a probabilistic model into EDAs, it is necessary to define effective learning and sampling methods. In the following, the general methods for these steps are presented.

3.2.1. Learning

Given a set of permutations, which works as a selected population, the learning process consists of estimating the consensus (central) permutation (σ_0) and the spread parameter (θ) for MM (or ($\boldsymbol{\theta}$) for GMM). Different methods have been reported to estimate σ_0 and θ in the context of EDAs [7,8,10,11].

Usually, this process is approached via maximum likelihood estimation (MLE). However, the time required for an exact learning scales factorially with the number of variables [7]. To deal with this shortcoming, Ceberio et al. [8], carried out an approximated learning mechanism. The estimated consensus permutation (σ_0) is calculated as the *set median permutation*, which is the permutation that minimizes the sum of the distances to the rest of the permutations in the sample. Once σ_0 is estimated, the MLE for the spread parameter θ for MM (or $\boldsymbol{\theta}$ for GM) is computed. The expression for this parameter is obtained by equaling to zero the derivate of the likelihood.

$$X_j = \frac{j}{j + \exp(\theta_j)} \tag{6}$$

Ceberio et al. [8] have solved this equation through the Newton–Raphson algorithm [7]. Formally, after the learning step, the Mallows Model under the Cayley distance can be defined as follows:

$$P(\sigma) = \psi(\theta)^{-1} \exp\left(\sum_{j=1}^{n-1} -\theta X_j(\sigma\sigma_0^{-1})\right) \tag{7}$$

for every $\sigma \in \mathbb{S}_n$.

3.2.2. Sampling

The sampling step aims to obtain new solutions from the model learned (Eqs. (6) and (7)) given by the probability of any σ according to the $n-1$ binary variables $X_1(\sigma), \ldots, X_{n-1}(\sigma)$ in which the distance is decomposed. The probability of each $X_j(\sigma)$ follows:

$$P(X_j(\sigma\sigma_0^{-1}) = 1) = \frac{(n-j)\exp(-\theta)}{\psi_j(\theta)} \tag{8}$$

Then, the sampling procedure follows a random method to generate a permutation σ given $X(\sigma)$ [20]. Ceberio et al. [8,11] described the sampling process according to three steps. First, a random binary vector $X(\sigma\sigma_0^{-1})$ is sampled using Eq. (8). Second, the associated $\sigma\sigma_0^{-1}$ is calculated according to the techniques described in [20]. Finally, by *right invariance*, the final permutation σ is obtained. The complexity of this procedure is $\mathcal{O}(n^2)$.

3.3. Kernels of Mallows Models

Despite the success of MM and GMM in the context of EDAs [7], they are unimodal models, i.e., they can provide accurate distributions when the population of the problem is *homogeneous* (low sparsity between the selected solutions). However, these models are not flexible enough to accurately model populations with solutions that are very sparse regarding the distance metric considered under the model. To overcome this drawback, mixtures of distance-based probability models [27] have been used to deal with multi-modal problems [10,11].

In [10], Kernels of Mallows Models EDA was proposed, where a kernel is defined for each selected solution, using this solution as its central permutation. The spread parameter θ is predefined to control the exploration/exploitation of the kernel. Then, new solutions are sampled from each kernel under a distance metric D.

This strategy is suitable for the multi-objective case, since most of the MOPs are multi-modal due to the number of Pareto optimal solutions that they may contain when the objectives are conflicting. Moreover, one of the primary goals of MOEAs is to maintain a diverse population for finding solutions that cover the entire true *PF*.

Therefore, in the MEDA/D-MK framework, each subproblem i is associated to a Mallows Kernel. Consequently, N Mallows Kernel models are kept (i.e., estimate a σ_0^k and a θ^k) at every generation. The interaction between the subproblems is kept by the update step, where a new sampled solution can update its current parent solution, but also its neighborhood.

4. A case of study: MEDA/D-MK for MoPFSP

In order to demonstrate the validity of MEDA/D-MK, we have applied it to solve the MoPFSP minimizing total flowtime and makespan. Thus, in the following, we describe the problem and report the recent approaches proposed to deal with it.

4.1. The flowshop scheduling problem

The objective of the PFSP is to find a permutation that optimizes a particular criterion. The PFSP schedules n jobs with given processing times on m machines, where the sequence of processing a job on all machines is identical and unidirectional for each job. The problem has the following assumptions [51]: (i) All jobs are available at time zero, (ii) each job can only be processed on, at most, one machine at the same time, (iii) each machine can process only one job at a time,

(iv) different jobs have the same processing order on all machines, and (v) the set-up times of the jobs on machines are sequence-independent and are included in the processing times.

A feasible permutation solution is denoted as $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ where $\sigma(i)$ represents the job to be processed in the ith position. Let p_{ij} denote the processing time for job i on machine j. C_{ij} is the completion time of job i on machine j. The completion time of a job i in the last machine m is indicated as C_{im}, and it is recursively calculated as follows:

$$\begin{aligned} C_{11} &= p_{11} \\ C_{i1} &= C_{(i-1)1} + p_{i,1}, \;\; i = 2, \ldots, n \\ C_{1j} &= C_{1(j-1)} + p_{1,j}, \;\; j = 2, \ldots, m \\ C_{im} &= max(C_{(i-1)j}, C_{i(j-1)}) + p_{ij}, \;\; i = 2, \ldots, n \;\; and \;\; j = 2, \ldots, m \end{aligned} \tag{9}$$

Usually, the completion time of a job (C_{im}) is also indicated as $C_i(\sigma)$. A scheduling problem can naturally involve multiple objectives to be optimized at the same time. Most literature referring to the PFSP focuses on a group of objectives based on the completion time, e.g., the minimization of the total flow time ($TFT(\sigma)$) and the minimization of the makespan ($C_{max}(\sigma)$). Besides, other groups of objectives have also been studied, such as the objectives based on due dates (tardiness) [36,51].

The makespan represents the time needed to process all the jobs. Minimizing makespan is related to the increase of throughput and machine utilization, and it is formulated as follows:

$$C_{max}(\sigma) = \max\{C_i(\sigma)\}, \quad i = 1, \ldots, n \tag{10}$$

The flow time of a job (F_i) is the time elapsed between its release time and its completion time. As the release time is always 0, therefore, $F_i(\sigma) = C_i(\sigma)$. The total flow time is formulated as follows:

$$TFT(\sigma) = \sum_{i=1}^{n} C_i(\sigma), \quad i = 1, \ldots, n \tag{11}$$

Different metaheuristics have been proposed to solve PFSP regarding the optimization of the *TFT*, such as, Variable Neighborhood Search (VNS) [14], Asynchronous Genetic Local Search (AGA) [49], General Mallows Model EDA (GM-EDA) [7], Algebraic Differential Evolution Algorithm (DEP) [42].

As in several real-world problems, PFSP can be defined as a multi-objective optimization problem. If no decision making preference is known a priori, all the objectives are considered equally important. A multi-objective PFSP can be stated as:

$$\begin{aligned} &minimize \;\; F(\sigma) = (C_{max}(\sigma), TFT(\sigma)), \\ &\qquad\qquad subject \; to \; \sigma \in \Omega \end{aligned} \tag{12}$$

4.2. MoPFSP: related Work

Different algorithms have been proposed for solving the MoPFSP. In the following, we discuss the related works.

Minella et al. [36] reviewed and evaluated a total of 23 algorithms including both flowshop-specific and general multi-objective optimization approaches on 110 benchmark test instances. These PFSP instances are those of Taillard [45] with the addition of due dates (which are used for the total tardiness criterion). The best approaches that they identified were those based on simulated annealing and genetic local search. The benchmark is available at their web page repository http://soa.itit.es.

Several works have used this Taillard benchmark to evaluate their approaches [16,29,37]. Consequently, we have also used it to evaluate our proposal.

Later, Minella et al. [37] introduced an efficient metaheuristic algorithm based on an Iterated Greedy technique. The algorithm called restarted iterated Pareto greedy (RIPG), combines (i) an efficient initialization of the population, (ii) the management of the approximated Pareto front, and (iii) a specially tailored local search. In all computational tests, and performance indicators, the proposed RIPG yields better results (many times in a significant way) than other 5 re-implemented (or adapted) state-of-the-art methods for the following pairs of objectives: (i) makespan and total flowtime and (ii) makespan and total tardiness. Additionally, their best-known results are available at the same repository containing the benchmark. As we will show later, we include their *best-known* results in our comparison study.

In the same year, Dubois–Lacoste et al. [16] proposed a hybrid algorithm, called TP+PLS, which combines a two-phase local search and a Pareto local search. The authors tested it for different bi-objective combinations (makespan, total flowtime, total tardiness), showing a good performance. Their best-known results are also included in our comparison study.

Algorithms based on the MOEA/D framework have already been applied to solve MoPFSP. Chang et al. [12], proposed the application of a MOEA/D variant, using the two-point crossover and the insert-based moving mutation, specifically tailored for minimizing makespan and total flowtime. Their results showed that the algorithm outperformed the Pareto-based algorithms NSGAII and SPEA2 for the set of test instances proposed by Ishibuchi et al. [22]. Moreover, Ke and Zhang [1] hybridized MOEA/D and Tabu Search (TS). Their primary motivation was to use TS to help MOEA/D escape from local optimal solutions. As the TS has a higher computational cost, it was only applied to good solutions. The results showed that MOEA/D-TS outperformed the MOEA/D for the test instances proposed by Ishibuchi et al. [22]. As we will present later, we include a

MOEA/D variant based on the same genetic operators used by Alhindi and Zhang [1], Chang et al. [12] in our experimental study.

The authors in [29] proposed a decomposition-based multi-objective local search algorithm (MOLSD). First, they used an efficient PFSP heuristic procedure to initialize the population. Next, the algorithm executes a Pareto local search embedded with a *heavy* perturbation procedure, followed by a single insert-based local search and a restarted method to escape from local optima. The results reported showed that MOLSD outperformed some state-of-the-art algorithms, such as RIPG [37]. Unfortunately, their *best-known* approximated *PFs* are not available for comparison.

The authors in [51] presented a more recent review of the approaches designed to deal with the MoPFSP and the benchmarks available in the literature. They reviewed 86 articles and reported that the most investigated metaheuristic methods have been genetic operators (GA), Tabu Search (TS) and Simulated Annealing (SA). Moreover, most of these algorithms use a PFSP-specific constructive method to initialize the population. The authors pointed out that one trend in this area is the combination of efficient heuristic methods to generate new hybrid algorithms in order to take advantage of the different approaches. The framework proposed in this paper follows this trend.

4.3. MEDA/D-MK for MoPFSP

In order to apply MEDA/D-MK for solving MoPFSP efficiently, we have defined some ad-hoc algorithm components.

The objectives $TFT(\sigma)$ and $C_{max}(\sigma)$ have different scales. Therefore, as in [12], we have used normalization values for *Weighted sum* and *Tchebycheff*. So, the vector of objective function values $F(\sigma)$ is normalized (using the *max-min* approach) before computing $g(\sigma|\boldsymbol{\lambda})$. However, when the objective function values are normalized, $\mathbf{z}^*$ does not guarantee a lowest reference value. To deal with this issue, each z_l is multiplied (decreased) by a factor α. For the bi-objective PFSP, we set $\alpha = 0.6$ in accordance to Chang et al. [12]. The adapted *Weighted sum* and *Tchebycheff* scalarizing functions for MoPFSP are as follows:

Weighted Sum:

$$\begin{aligned} &minimize \quad g^{ws}(\sigma|\boldsymbol{\lambda}, \mathbf{z}^*, \mathbf{w}^*) = \sum_{l=1}^{q} \lambda_l \frac{f_l(\sigma) - \alpha\ z_l^*}{w_l^* - z_l^*} \\ &\qquad\qquad subject\ to\ \sigma \in \Omega \end{aligned} \tag{13}$$

Thebycheff:

$$\begin{aligned} &minimize \quad g^{tc}(\sigma|\boldsymbol{\lambda}, \mathbf{z}^*, \mathbf{w}^*) = \max_{1\le l\le q} \left\{ \lambda_l \frac{f_l(\sigma) - \alpha\ z_l^*}{w_l^* - z_l^*} \right\}; \\ &\qquad\qquad subject\ to\ \sigma \in \Omega \end{aligned} \tag{14}$$

where $\mathbf{z}^*$ is the reference point, i.e., the minimum (best) values found so far for each objective value, and $\mathbf{w}^*$ is the maximum (worst) values found so far for each objective value.

1. **Initialization step**: First, the $LR(n/m)$ procedure constructs a single solution (σ_{lr}). Next, one subproblem is arbitrarily chosen and initialized with σ_{lr}. Second, the algorithm randomly chooses $(N/2) - 1$ subproblems and initializes them with a light perturbed σ_{lr} solution. The perturbation on σ_{lr} is based on $n/10$ insert-based moves. This strategy allows the initial population to maintain some characteristics of σ_{lr} and also can find other promising solutions. Next, the remaining $N/2$ subproblems are initialized randomly (i.e., without a heuristic) to guarantee that the convergence and some diversity is maintained at the initial *Pop*. Finally, the *EP* is initialized with the non-dominated solutions from *Pop*.
2. **Learning step**: The central permutation σ_0^k is set to its current solution σ^k. In [10], to enhance the search ability of the model, an adaptive interval of values for θ is defined. In their strategy, at every generation, if the best solution was not improved, then θ was increased. Otherwise, θ was set to the lower bound value. However, in our preliminary experiments, this strategy did not achieve successful results. Besides, a fixed θ value at every generation obtained good results. The experimental study to assign different probabilities is presented in Section 5.7. It is worth noting that, in this particular implementation, no learning is completed, as the central permutation is set to its current solution, and the value of the θ parameter is fixed (provided by the practitioner).
3. **Sampling step:** The new solution σ_s^k is sampled according to Eq. (7) and the steps described in Section 3.2.2. Moreover, in our preliminary study, it was observed that one cause for early convergence of the population was the sampling of the solutions already present in the population. Thus, the sampling step has been enhanced by the addition of two procedures:
   (a) A single insert-based movement is applied to σ_s^k with a low probability.
   (b) A validation procedure is performed to avoid that the same permutation appears many times in the neighborhood, which is detrimental for diversity. So, if the sampled solution $\sigma_s^k \in B(k)$ then, the algorithm discards σ_s^k, and tries to sample a new one until it samples a solution $\sigma_s^k \notin B(k)$ or reaches a maximum number of trials (T). If T is reached, the algorithm maintains the last sampled solution anyway.
4. **Update *Pop*** : The new sampled solution is used to update *Pop*. The update procedure follows the scheme described in the general framework.

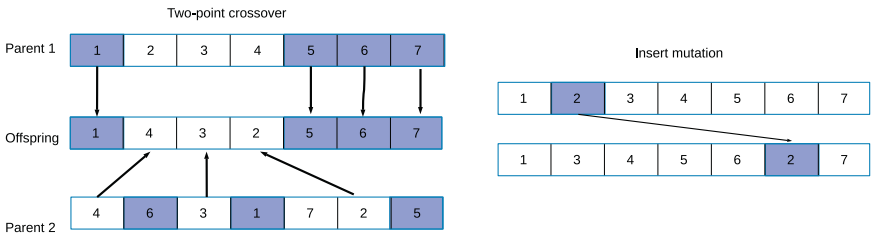


Fig. 2. Two-point crossover and insert mutation operators.

5. **Shaking procedure:** To lead the algorithm to escape from local optima and control the diversity of the population, MEDA/D-MK is improved with a controlled shaking procedure that can be performed at any subproblem k. The procedure works as follows: If σ^k has not been enhanced (updated) after a predefined number of generations ($count^k$), σ^k is updated by receiving n_{sh} random insert-based movements, even if it computes a worse $F(\sigma^k)$. Then, the *count* for a new perturbation in the subproblem k is reset to 0.

Moreover, the *EP* is updated as described in the general framework.

Regarding the input parameters, besides those summarized in Table 1, the MEDA/D-MK for MopFSP includes: 1) the factor α for the adapted *Weighted Sum* and *Tchebycheff*, 2) the maximum number of consecutive generations without an improvement in the subproblem k before executing the shaking procedure ($count^k$), and 3) the number of random insert-based movements (n_{sh}) to perturb the solution.

5. Experimental studies

The aim of the experimental studies is to evaluate the performance of MEDA/D-MK for MoPFSP compared to the state-of-the-art approaches. In this analysis, we: (i) evaluate if, within the MOEA/D framework, using the Mallows Model is preferable to genetic operators, and (ii) check if our approach is competitive compared to the state-of-the-art results reported by the scientific community.

5.1. Algorithms and the PFSP benchmark

Murata and Ishibuchi [39] evaluated various genetic operators for makespan and total flowtime. Their results showed that the two-point crossover and the insert-based mutation (see Fig. 2) achieved the best results. Thus, as done in [1,12] to the MOEA/D variants for MoPFSP, we have incorporated these operators in our MOEA/D instantiation. It is worth noting that our MOEA/D variant also incorporates the constructive PFSP initialization and the shaking procedure.

MEDA/D-MK and MOEA/D have been implemented into the same framework in C++ using Ubuntu 16.04. The execution time analysis was performed on a PC with Intel Xeon E5-620 2.4 GHz processor and 12GB memory.

The Taillard PFSP benchmark [37] has been widely employed in the literature as a benchmark to evaluate optimization algorithms. Each processing time p_{ij} is generated from a random uniform distribution in the range [0, 99]. The benchmark is composed by 110 test instances having different combinations of number of jobs $n = \{20, 50, 100, 200\}$ and number of machines $m = \{5, 10, 20\}$. The instances are grouped in 11 different combinations (scales) $n \times m$ (20×5, 20×10, 20×20, 50×5, 50×10, 50×20, 100×5, 100×10, 100×20, 200×10 and 200×20) containing 10 instances each.

Usually, if the source code is not available, the researchers have to re-implement the state-of-the-art algorithm(s) to compare them to the proposed approach. However, it is not easy to make a fair re-implementation, and sometimes they achieve different results regarding those previously reported, as it was stated in [36]. The other option is to use the *best-known* approximated *PFs* provided by the research community for comparison.

5.2. Performance metrics and statistical test

Two performance metrics are used: (i) The well-known Pareto-compliant indicator *Hypervolume* (*HV*) [50,56], and (ii) the coverage indicator (*C-metric*) [43].

Let P^* be a set of uniformly distributed *Pareto* optimal solutions along the *true PF* in the objective space, and P be an approximated set to the *true PF* obtained by an algorithm considered.

Hypervolume (*HV*): Let $z^r = (z_1^r, \ldots, z_m^r)^T$ be a reference point in the objective space that is dominated by all Pareto-optimal objective vectors. The *HV* measures the size (volume) of the objective space dominated by the solutions in P and bounded by z^r:

$$HV(P) = VOL\Big(\bigcup_{\mathbf{x}\in P} [f_1(\mathbf{x}), z_1^r] \times \cdots \times [f_m(\mathbf{x}), z_m^r]\Big) \tag{15}$$

In our experiments, the obtained approximated *PFs* (P) are normalized between [0, 1] using the *max-min* normalization. Therefore, each objective function is normalized as:

$$f_{li}(\mathbf{x}) = \frac{f_{li}(\mathbf{x}) - \min f_l(\mathbf{x})}{\max f_l(\mathbf{x}) - \min f_l(\mathbf{x})},\ i = 1, .., h \text{ and } l = 1, .., q \tag{16}$$

where q is the number of objectives, and h is composed by all *PFs* considered according to the compared algorithms and runs ($h = |algorithms \times runs|$). The *HV* reference point is set to $z^r = (1.01, 1.01)$. The higher the $HV(P)$, the better the approximation of P to the true *PF*.

The *C-metric* measures the "degree" of dominance of a Pareto front over another, i.e., the proportion of the solutions in B that are dominated by at least one solution in A. $C(A,B) = 1$ means that all solutions in B are dominated by some solution in A, while $C(A,B)=0$ means that no solution in B is dominated by a solution in A.

$$C(A, B) = \frac{|\{u \in B | \exists v \in A : v \text{ dominates } u\}|}{|B|} \tag{17}$$

also, $C(A, B)$ is not necessarily equal to $1 - C(B, A)$.

Due to the stochastic behavior of the optimization algorithms, 10 runs are performed for each test instance and algorithm configuration. Next, the non-parametric Friedman's statistical test [15] is applied to the performance assessment results (qualitative values). The Friedman's test is a test which can be used to carry out multiple comparisons among all methods. It allows us to detect differences considering the global set of algorithms. The null hypothesis for Friedman's test is that samples of different groups (in our case, algorithms) have been selected from a population having equal medians. If the Friedman's test rejects the null hypothesis, the *Nemenyi post-hoc* test is applied to determine if there are statistical differences between each pair of algorithms. Finally, the algorithms are ranked based on the number of times that each algorithm statistically outperforms the others.

5.3. Parameters setting

We have conducted an experimental parameter setting to justify the design choices for MEDA/D-MK and MOEA/D. In the following, the parameter settings are described according to this preliminary study. Later, we present the influence of the spread parameter and the PFSP-specific components.

For the genetic operators, the neighborhood size (T) defines the range of neighbor solutions that can be selected as the parents p_1 and p_2. In accordance to [10, 1], we set $T_s = 10$, probability of crossover $P_c = 1.0$ and probability of the single insert mutation $P_m = 0.5$. Moreover, for the MEDA/D-MK sampling step (described in Section 4.3), the probability to apply a unique insert-based movement in σ_s^k is also $P_m = 0.5$. The remaining of the general parameters setting are:

1. *Number of subproblems* (N)*:* A large number of subproblems means a higher computational cost because it increases the number of fitness evaluations at each generation. However, a small number of subproblems can deteriorate the algorithm search ability to explore the different regions of the objective space. Therefore, we set $N = 100$ as it is a reasonable number of weight vectors (subproblems) for solving bi-objective problems.
2. *Update neighborhood:* We have used a scheme named *range sensitive global update*, which means that, according to the Euclidean distance, a new sampled solution σ_s^k tries to update the current population from the closest neighbor (i.e., from its index k) to the farther neighbor solution.
3. *Maximum replacements* (n_r)*:* Each sampled solution (σ_s^k) can update a maximum of $n_r = 2$ subproblems.
4. *Spread parameter* (θ)*:* It is set to assign a probability of 0.8 to the central permutation (σ_0^k) at every generation.
5. *Stopping criterion:* The algorithms stop when a maximum number of evaluations is reached. For each problem, the maximum number of evaluations depends on the problem size as $MaxGen = n \times 1000$.

Moreover, we track the outcome of the algorithms at each $(n \times 1000)/100$ generations by inspecting the current *PF*, i.e., 10 approximated *PFs* are provided and evaluated in each run. In this way, we can evaluate the algorithm's behavior throughout the evolution.

5.4. Comparison to the MOEA/D variant

In this section, we have compared MEDA/D-MK to MOEA/D variant using the *Weighted sum* and *Tchebycheff* approaches. Firstly, we have provided the average normalized *HV* results obtained by MOEA/D and MEDA/D-MK for each one of the 110 Taillard test instances and ranked their results according to the Kruskall–Wallis statistical test [15]. This table is presented in a supplementary document[1]. The results show that MEDA/D-MK using *Weighted Sum* achieves the best rank results with a significant difference in 78 of the 110 test instances.

Next, the final outcomes are grouped for each instance configuration $n \times m$ (10 instances each). Thus, the analyses of results were conducted according to 100 independent runs using the two quality indicators and the non-parametric

[1] Available at https://github.com/MuriloZangari/supplementary_results_mopfsp.

Table 2
Average *HV* values obtained by MEDA/D-MK and MOEA/D using the *Weighted Sum* and *Tchebycheff* for each group of test instances *jobs* × machines. The best-ranked results is highlighted in boldface.

Instance	*Weighted sum*		*Tchebycheff*	
	MOEA/D^w	MEDA/D-MK^w	MOEA/D^t	MEDA/D-MK^t
20× 5	0.8087	**0.8728**	0.7804	**0.8413**
20× 10	0.8074	**0.8757**	0.7387	0.8289
20× 20	0.8021	**0.8577**	0.7151	0.7985
50× 5	**0.8678**	**0.9242**	0.7617	0.8381
50× 10	0.7449	**0.8378**	0.5818	0.6642
50× 20	0.7531	**0.8544**	0.6150	0.6914
100× 5	0.8578	**0.8892**	0.7423	0.8151
100× 10	0.7550	**0.8161**	0.5764	0.6350
100× 20	0.7319	**0.8344**	0.5037	0.5491
200× 10	**0.8011**	**0.8258**	0.5885	0.6219
200× 20	0.7278	**0.8101**	0.4497	0.4816

Table 3
C-metric values between A = MEDA/D-MK against MOEA/D and the *best-known* reference approximated *PF*. The algorithm that covered more solutions is highlighted in boldface.

Instance	B = MOEA/D		B = *reference sets*	
	C(A,B)	C(B,A)	C(A,B)	C(B,A)
20× 5	**0.54**	0.03	0.12	**0.15**
20× 10	**0.40**	0.06	0.08	**0.12**
20× 20	**0.25**	0.07	0.09	**0.11**
50× 5	**0.96**	0.02	**0.79**	0.11
50× 10	**0.91**	0.07	**0.94**	0.02
50× 20	**0.83**	0.12	**0.98**	0.02
100× 5	**0.98**	0.02	**0.78**	0.02
100× 10	**0.96**	0.01	**0.82**	0.04
100× 20	**0.82**	0.15	**1.00**	0.00
200× 10	**0.72**	0.16	**0.83**	0.00
200× 20	**0.85**	0.13	**1.00**	0.00

Friedman statistical test (at 5% significance level) to check if the results obtained have a statistical difference. Then, the *post-hoc* test *Nemenyi* is applied to check (if they have achieved significant differences) which one(s) produced the best results. Table 2 presents the average *HV* values obtained by MEDA/D-MK and MOEA/D using the *Weighted sum* and the *Tchebycheff* scalarizing functions. For each group of instances, the best-ranked result, according to the statistical test, is highlighted in boldface. Table 3 shows the results of the *C-metric* obtained by the MEDA/D-MK against MOEA/D.

The results in Table 2 show that *Weighted sum* outperforms *Tchebycheff*. Moreover, MEDA/D-MK with *Weighted Sum* (referenced as MEDA/D-MK^w) outperforms MOEA/D^w with a significant difference in 9 of the 11 groups of instances. The groups of test instances for which MEDA/D-MK^w does not outperform MOEA/D^w with a significant difference are 50 × 5 and 200 × 10. As Mietthen [35] argued, the *Weighted Sum* approach is good at convex (concave) bi-objective problems, while Tchebycheff approach is useful when the problem is non-convex. This is also confirmed in our results, in which*Weighted Sum* outperforms *Tchebycheff*. So, for the remaining experimental studies, we have used only the *Weighted Sum*.

Moreover, it is evident from Table 3 that, according to the *C-metric*, the approximated *PFs* from MEDA/D-MK dominates a large percentage of solutions from the approximated *PFs* obtained by MOEA/D in all the groups of instances.

It is explicit from Fig. 3 that the *sampling* step defined in MEDA/D-MK is more time-consuming compared to the conventional genetic operators. The difference increases as the problem scale increases, since the sampling step from MEDA/D-MK is $\mathcal{O}(n^2)$. This behavior is typical for EDAs, which are slower than GAs and other similar approaches since, usually, they are more complex but, generally, they are able to produce solutions of better quality. Moreover, in the supplementary document,[2] we provide the average computational time for each one of the 110 test instances.

5.5. Discussion: Mallows Model vs. genetic operators:

In MOEA/D, $B(k)$ defines the range of solutions that can be chosen as parent solutions. It is easy to prove that (at least for MOPs with concave (convex) *PF* shapes) the solutions of two subproblems have a higher level of similarity as their

[2] Available at https://github.com/MuriloZangari/supplementary_results_mopfsp.

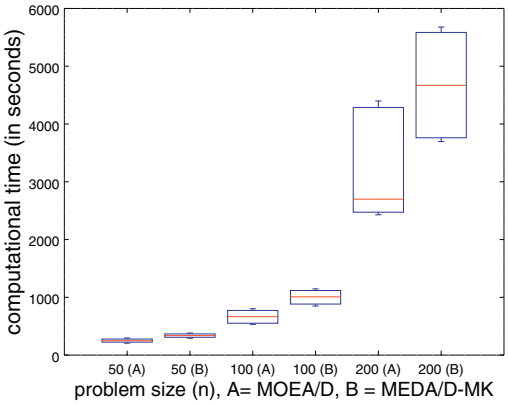


Fig. 3. Box plot of the average computational time (in seconds) used by MOEA/D (A) and MEDA/D-MK (B) for the merged group of instances with 50, 100 and 200 jobs, after $n \times 1000$ generations.

weight vectors are closer. In Mallows Models, the θ parameter determines the probabilities of sampling a solution close to the central permutation σ_0. Thus, θ has a similar behavior to $B(k)$, which can influence the search regarding the sparsity (distance) between the offspring and their parents.

Besides, MM and the genetic operators are different regarding the type of variation/movements that they perform: the Cayley distance is based on the minimum number of swaps (not necessarily adjacent) that have to be carried out to transform σ into π, and the genetic operator is based on the two-point crossover.

It is also known that, unlike other EAs, EDAs can provide models expressing the regularities of the problem structure [25], being this property one of the primary motivations of using EDAs instead of other EAs.

5.6. Comparison to reference sets

We devote this section to analyze our approach in the context of the state-of-the-art approaches/results for the benchmark considered. As it has not been possible to obtain the source code of the best-performing approaches (requested to the authors), we have used the *best-known* approximated *PFs* reported by their respective authors. The *best-known* reference sets reported by [37][3] were produced by joining all the approximated *PFs* achieved by 7 re-implemented (or adapted) state-of-the-art methods, including their best-performing proposed algorithm RIPG, different parameters configurations, and several runs. Dubois–Lacoste et al. [16][4] provided the reference sets for each algorithm evaluated, including their best-performing algorithm TP+PLS. In order to have comparable sets, we have followed the same procedure, composing our reference sets by merging the outcomes from MEDA/D-MK.

It must be noted that the results shown in this section must be interpreted carefully, since the reference sets were obtained with different stopping conditions and merging different runs/algorithms. Therefore, we have compared MEDA/D-MK to the reference sets throughout the generations. This comparison allow us to observe the dynamics of the MEDA/D-MK during the search.

Fig. 4 shows the average *HV* values obtained by MEDA/D-MK throughout the generations (10 plots) compared to the average *HV* values obtained from the reference sets (constant lines) for the group of instances. According to the results, we can make the following remarks:

- In general, MEDA/D-MK is able to keep evolving throughout all the generations. However, the algorithm slowly improves the results after 70% of the generations (except for the group of instances 20×20). If source codes of the algorithms were available, it would be interesting to analyze the ability of each algorithm to evolve throughout generations.
- The reference sets from Dubois–Lacoste et al. outperform the reference sets from Minella et al. in all the cases. The difference increases as the problem scale increases.
- MEDA/D-MK achieves competitive results. Our approach easily outperforms the reference sets from Minella et al. in all the cases, except for the 20×5. Also, it outperforms the reference sets from Dubois–Lacoste et al. for many of the instances containing 20 jobs, and for almost all the instances of size 200×10 and 200×20. The HV values for every particular instance are provided in the supplementary material.
- This type of probabilistic models, like the Mallows Model we introduced, have a higher complexity than those of its competitors. However, as it has been demonstrated in other works on EDAs [7], even if they are usually slower than other approaches, they are able to *continue evolving* and thus obtain better results.

[3] Minella et al.: http://soa.iti.es.

[4] Dubois-Lacoste et al.: http://iridia.ulb.ac.be/~jdubois/pfsp_refsets.htm.

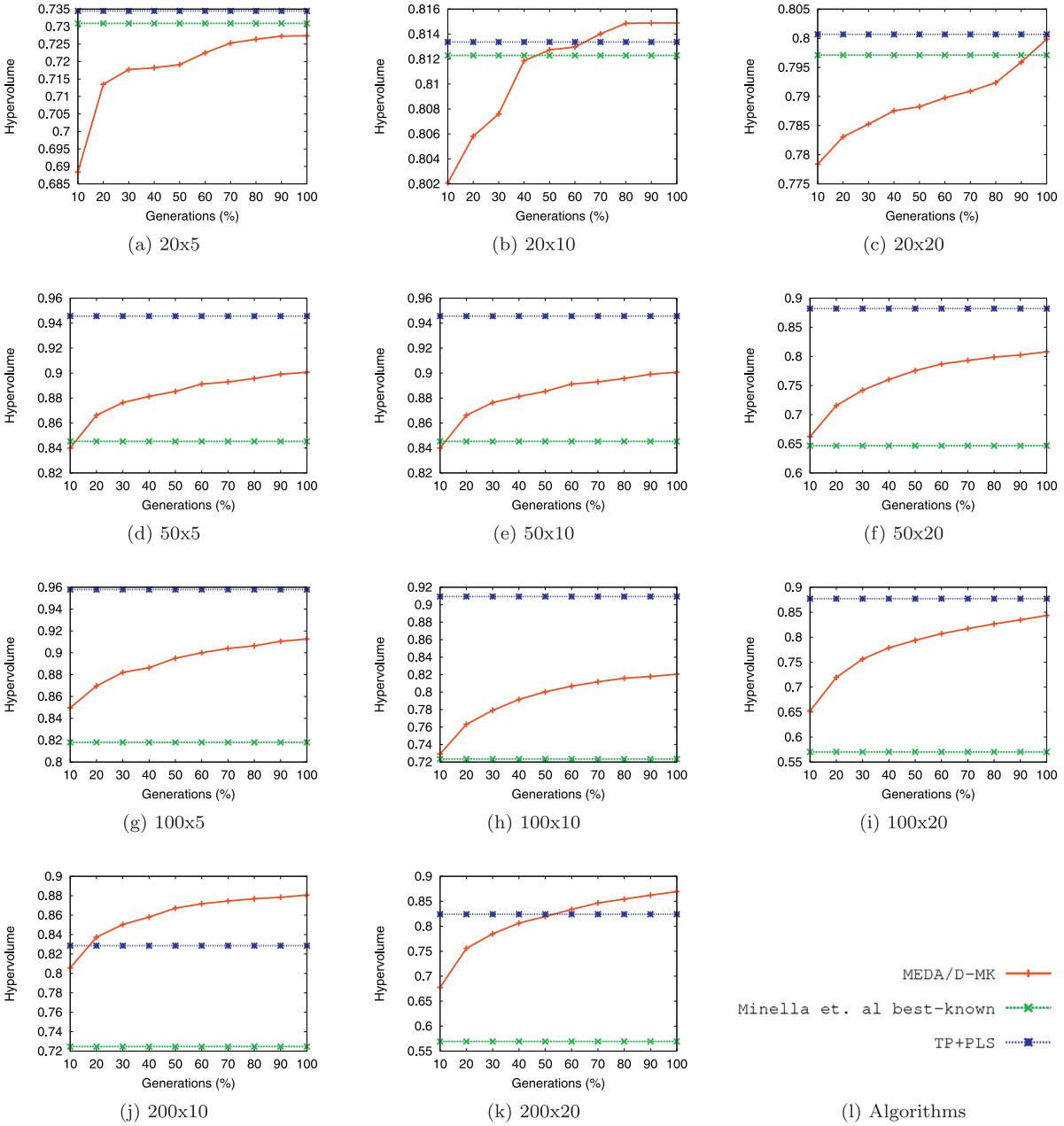


Fig. 4. Average *HV* values obtained by MEDA/D-MK throughout the generations (every %10) compared to the reference sets from the literature (those from Minella et. al [37] and TP+PLS [16]) for the 11 groups of instances (a)–(k).

To complement the information presented in Fig. 4, Table 4 presents a comparison (by pairs) based on the final *HVs* obtained by the three different approaches for every (110) test instances. In this table, for each group of instances, we count how many times one approach outperforms the other (or not). Regarding that, each group contains 10 instances. Looking at the results, it can be seen that MEDA/D-MK and TP+PLS systematically outperform the results provided by Minella's best-known sets. Regarding MEDA/D-MK and TP-PLS, the later shows better results for the groups with 50 and 100 jobs, while the former behaves better for 20×10, 200×10, and 200×20.

Finally, Fig. 5 presents the plots of the final approximated *PFs* obtained by the algorithms for four representative large-scale test instances (Ta071, Ta082, Ta091 and Ta101). From this figure, we can observe that the *PFs* are concave, and that the cardinality of the fronts increases with the number of machines. As expected, the best approximated *PFs* are those obtained by MEDA/D-MK and TP+PLS. In general, TP+PLS finds more solutions that cover the extremes regions of the PFs because of

Table 4
Comparison between pairs of algorithms according to the *HV* measure. For each pair of algorithms × group of instances, a cell indicates three values: 1) the number of times that the final *HV* of approach *A* is better than that of approach *B*, 2) the number of draws, 3) the number of times that the *HV* of approach *B* is better than that of approach *A*). Reference sets: ***A*= MEDA/D-MK, *B*= Minella et. al best-known, *C*= Dubois-Lacoste** et al.**(TP+PLS)**.

Instances	*A-draw-B*	*A-draw-C*	*C-draw-B*
20× 5	2-5-3	1-3-6	6-2-2
20× 10	6-3-1	6-3-1	4-2-4
20× 20	6-1-3	5-3-2	6-1-3
50× 5	9-0-1	0-0-10	10-0-0
50× 10	10-0-0	0-0-10	10-0-0
50× 20	10-0-0	0-0-10	10-0-0
100× 5	9-0-1	3-1-6	10-0-0
100× 10	10-0-0	1-0-9	10-0-0
100× 20	10-0-0	2-0-8	10-0-0
200× 10	10-0-0	9-0-1	10-0-0
200× 20	10-0-0	8-0-2	10-0-0
Total	**92-9-9**	**35-10-65**	**96-5-9**

Table 5
The θ values used to assign the respective probabilities $P(\sigma_0) = \{0.5, 0.6, 0.7, 0.8\}$ for the different problem sizes (n).

n	$P(\sigma_0) = 0.5$	$P(\sigma_0) = 0.6$	$P(\sigma_0) = 0.7$	$P(\sigma_0) = 0.8$
20	5.60	5.90	6.30	6.80
50	7.48	7.78	8.20	8.60
100	8.90	9.20	9.60	10.01
200	10.3	10.60	11.00	11.41

its search behavior, in which the first phase of the method applies a single-solution algorithm to find high-quality solutions for each objective function separately.

To ease the comparison to other methods, we have compiled our *best-known* reference approximated *PFs*. Each reference set was produced by merging all the MEDA/D-MK results regarding all runs and all parameter configurations. Our 110 *best known* reference sets are available on-line.[5]

5.7. Influence of the MEDA/D-MK components

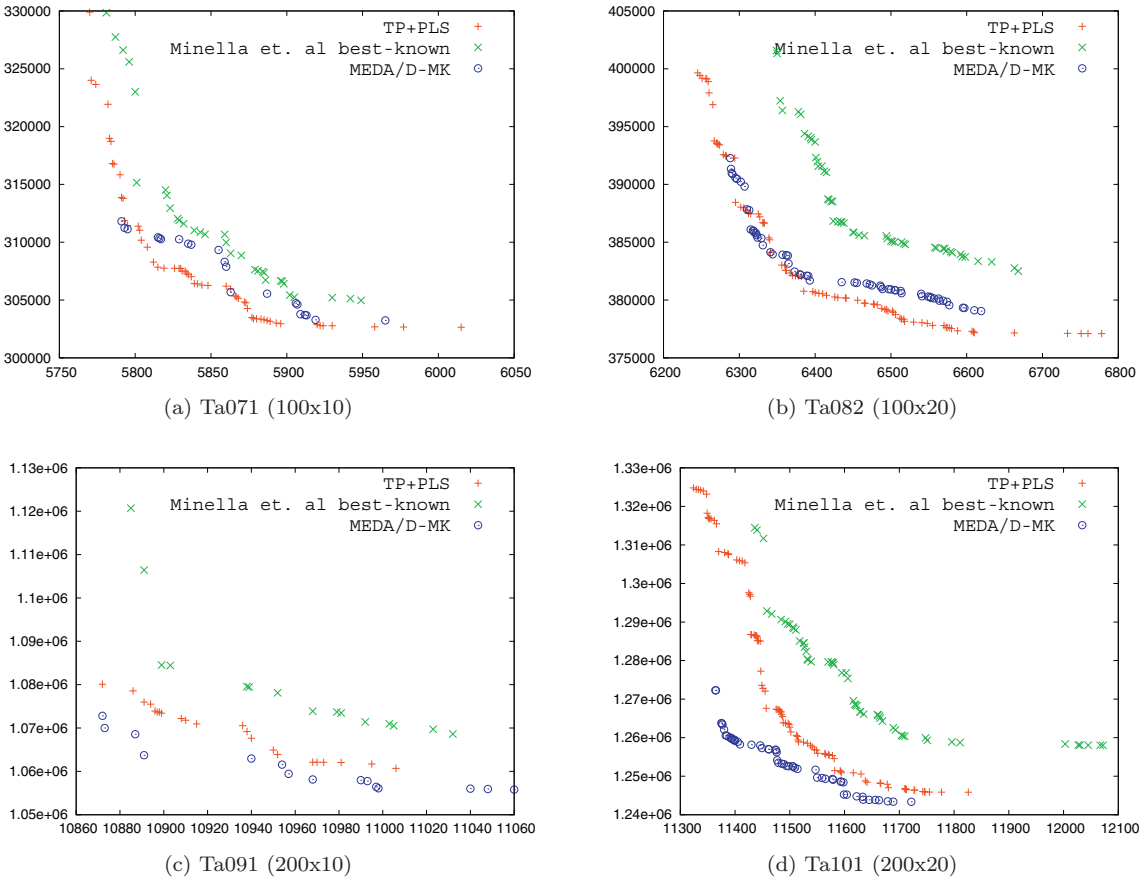
In this section, in order to further provide an understanding MEDA/D-MK components and the parameters used, we present the study of (i) the spread parameter θ, (ii) the constructive algorithm $LR(n/m)$ to initialize the population, and (iii) the controlled shaking procedure used to escape from the local optima.

The spread parameter is directly related to the shape of the exponential probabilistic model [20]. Thus, we have evaluated different θ values in order to assign the following probabilities to the central permutation $P(\sigma_0) = \{0.5, 0.6, 0.7, 0.8\}$ that depends on the number of variables (see Eq. (7)).

Table 5 presents the θ values for the different problem sizes and their corresponding probabilities of sampling $P(\sigma_0)$. Table 6 shows that, regarding the *HV* indicator, MEDA/D-MK with the highest $P(\sigma_0)$ achieves the best results. Besides, for $P(\sigma_0) = 0.7$ and $P(\sigma_0) = 0.8$ the difference is not significant in 10 of the 11 cases. It is worth noting that we have used the strategy that avoids the central permutation to be sampled to prevent many copies in the population. A higher $P(\sigma_0)$ means that the probability of sampling a solution close to the σ_0 is higher (low sparsity between permutation solutions), which means that the algorithm conducts an exploitation in each kernel. The other components/ingredients of the framework may promote the exploration in different areas of the search space.

Table 7 shows that both MEDA/D-MK and MOEA/D achieve better results using the $LR(n/m)$ initialization (referenced as MOEA/D_{LR} and MEDA/D-MK_{LR}), mainly for the larger test instances. Overall, the MEDA/D-MK_{LR} produces the best results. For the smallest instances $n = \{20, 50\}$, MEDA/D-MK does not achieve a significant difference with and without the constructive initialization. There are two possible explanations for this behavior: the $LR(n/m)$ initialization can quickly approximate the solutions to the true *PF*, or the search behavior of the algorithms is not efficient enough to evolve the population for the smallest problem scales. We should further investigate this aspect.

[5] Available at https://github.com/MuriloZangari/supplementary_results_mopfsp.



(a) Ta071 (100x10) (b) Ta082 (100x20)

(c) Ta091 (200x10) (d) Ta101 (200x20)

Fig. 5. Illustrative plots of the final approximated *PFs* by MEDA/D-MK, Minella et. al best-known and TP+PLS for (a) Ta071, (b) Ta081, (c) Ta091, and (d) Ta101 instances.

Table 6
Average *HV* values obtained by MEDA/D-MK using different $P(\sigma_0)$ values for the group of instances.

instance	$P(\sigma_0) = 0.5$	$P(\sigma_0) = 0.6$	$P(\sigma_0) = 0.7$	$P(\sigma_0) = 0.8$
20 × 5	**0.7536**	**0.7661**	**0.7584**	**0.7575**
20 × 10	**0.8255**	**0.8261**	**0.8300**	**0.8297**
20 × 20	**0.7907**	**0.7910**	**0.7969**	**0.7958**
50 × 5	**0.9057**	**0.9048**	**0.8998**	**0.9008**
50 × 10	0.7785	0.7793	**0.8026**	**0.7879**
50 × 20	0.7737	0.7713	**0.8065**	**0.8074**
100 × 5	**0.8203**	**0.8191**	**0.8308**	**0.8007**
100 × 10	**0.7786**	**0.7907**	**0.7955**	**0.7989**
100 × 20	0.7006	**0.7599**	**0.7645**	**0.7655**
200 × 10	0.7678	**0.7822**	**0.7923**	**0.7972**
200 × 20	0.6543	0.6531	0.6782	**0.6883**

The shaking procedure has two additional parameters to be set. To avoid the complexity of the parameters setting, we set them according to the number of jobs (n). After n generations without σ^k being improved, the procedure executes the perturbation based on (n/10) insert-based movements.

Table 8 shows that MEDA/D-MK with the shaking procedure (MEDA/D-MK$_{SH}$) produces the best *HV* results with a significant difference in all the cases except for the smallest ones, 20 × 5, and 20 × 10. Overall, the results show that the search needs a balance between the exploitation and exploration to keep converging to the true *PF* while it is able to escape from local optima.

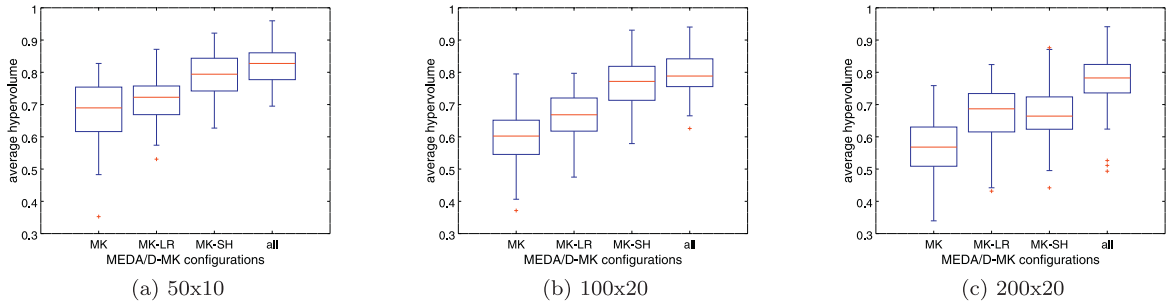
Fig. 6 complements these results. The *LR*(n/m) initialization has a major positive effect as the problem scale increases. Moreover, the shaking procedure has a more positive influence compared to the constructive *LR*(n/m) initialization, and both components applied together are really efficient.

Table 7
Average *HV* values obtained by MOEA/D and MEDA/D-MK with and without the constructive algorithm $LR(n/m)$ to initialize the population.

Instance	MOEA/D	MOEA/D$_{LR}$	MEDA/D-MK	MEDA/D-MK$_{LR}$
20 × 5	**0.7342**	**0.7359**	**0.7509**	**0.7590**
20 × 10	**0.8372**	**0.8346**	**0.8467**	**0.8512**
20 × 20	0.8167	0.8086	**0.8281**	**0.8245**
50 × 5	0.8712	0.8862	**0.9079**	**0.9165**
50 × 10	0.6936	0.7369	**0.7735**	**0.7984**
50 × 20	0.6605	0.7121	**0.7751**	**0.8005**
100 × 5	0.7143	0.8444	0.8184	**0.8804**
100 × 10	0.6587	0.7751	0.7654	**0.8560**
100 × 20	0.5881	0.6528	**0.7469**	**0.7677**
200 × 10	0.5251	**0.8064**	0.6204	**0.8452**
200 × 20	0.5460	0.6852	0.6481	**0.7610**

Table 8
Average *HV* values obtained by MOEA/D and MEDA/D-MK with and without the controlled shaking procedure.

instance	MOEA/D	MOEA/D$_{SH}$	MEDA/D-MK	MEDA/D-MK$_{SH}$
20 × 5	**0.7503**	**0.7499**	**0.7658**	**0.7725**
20 × 10	**0.8339**	**0.8335**	**0.8351**	**0.8501**
20 × 20	0.8053	0.8050	0.8039	**0.8201**
50 × 5	0.8883	0.8869	0.9026	**0.9245**
50 × 10	0.7240	0.7420	0.7433	**0.8061**
50 × 20	0.6866	0.7205	0.7109	**0.8140**
100 × 5	0.7956	0.7932	0.8298	**0.8577**
100 × 10	0.7266	0.7458	0.7707	**0.8422**
100 × 20	0.6003	0.6454	0.6502	**0.7636**
200 × 10	0.7388	0.7420	0.7635	**0.7933**
200 × 20	0.5796	0.6396	0.6436	**0.7196**



(a) 50x10 (b) 100x20 (c) 200x20

Fig. 6. Boxplot of the average *HV* obtained by 4 configurations: (i) MEDA/D-MK without both components (defined as MK), (ii) MEDA/D-MK with $LR(n/m)$ (MK-LR), (iii) MEDA/D-MK with shaking procedure (MK-SH), and (iv) MEDA/D-MK with both components (all) for the group of instances (a) 50x10, (b) 100x20, and (c) 200x20.

6. Conclusion and future work

Recently, EDAs that incorporate distance-based exponential probability models over permutations have been proposed for solving different single-objective permutation optimization problems efficiently. In this paper, we have presented, for the first time, a general multi-objective estimation of distribution algorithm based on decomposition and Kernels of Mallows Models (MEDA/D-MK framework).

In order to demonstrate the viability of the proposal to solve multi-objective permutation-based problems, we have applied it on 110 MoPFSP test instances minimizing makespan and total flow time. The MEDA/D-MK for MoPFSP is enhanced with a constructive PFSP-specific procedure to initialize the solutions, and a perturbation procedure to control the exploration/exploitation for finding a diverse set of non-dominated solutions.

The experimental study shows that MEDA/D-MK outperforms the tailored MOEA/D for MoPFSP. Thus, we have showed the potentiality of using Mallows Models EDA in the context of the MOEA/D framework. Moreover, the analysis demonstrated that our approach achieved competitive results compared to the reference sets reported in the literature for the benchmark considered. For the large-scale test instances, MEDA/D-MK has significantly produced better approximated *PFs* using a reasonable number of evaluations and computational cost. Furthermore, for reproducibility purposes, we have pro-

duced our *best-known* results, being they available at https://github.com/MuriloZangari/supplementary_results_mopfsp for future comparison to other approaches.

Departing from the significant results reported in this paper, we state some trends for future work: (i) the analysis of other permutation distance metrics such as the Ulam distance [21], (ii) investigate the efficiency of our approach to deal with the MoPFSP involving more objectives (e.g., the total tardiness), and (iii) to solve other permutation problems, e.g., the multi-objective Quadratic Assign Problem.

In addition, components of our algorithm, such as the Mallows Models, could also be incorporated to recent methods proposed for automatically designing MOEAs. In particular, they could be included in the AutoMOEA template [2], a recent component-wise design of multi-objective algorithms, which is able to combine automatically different procedures and components, creating novel MOEA algorithms.

Acknowledgments

This work has received support from CNPq (Productivity Grant Nos. 306103/2015-0 and Program Science Without Borders Nos.: 400125/2014-5), from CAPES (Brazil Government), from the IT-609-13 program (Basque Government) and TIN2016-78365-R (Spanish Ministry of Economy, Industry and Competitiveness). We are grateful to Josu Ceberio and Ekhine Erurozki for providing their Mallows Models source code and for useful comments and suggestions.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ins.2017.02.034.

References

[1] A. Alhindi, Q. Zhang, MOEA/D with tabu search for multiobjective permutation flow shop scheduling problems, in: Proceedings of the Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 1155–1164.
[2] L.C.T. Bezerra, M.L.-I. nez, T. Stützle, Automatic component-wise design of multiobjective evolutionary algorithms, IEEE Trans. Evol. Comput. 20 (3) (2016) 403–417.
[3] P.A. Bosman, D. Thierens, Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms, Int. J. Approximate Reasoning 31 (3) (2002) 259–289.
[4] S.P. Brooks, B.J. Morgan, Optimization using simulated annealing, Statistician (1995) 241–257.
[5] A.E.I. Brownlee, M. Pelikan, J. McCall, A. Petrovski, An application of a multivariate estimation of distribution algorithm to cancer chemotherapy, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), ACM, 2008, pp. 463–464.
[6] J. Ceberio, E. Irurozki, A. Mendiburu, J.A. Lozano, A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems, Prog. Artif. Intell. 1 (1) (2012) 103–117.
[7] J. Ceberio, E. Irurozki, A. Mendiburu, J.A. Lozano, A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem, IEEE Trans. Evol. Comput. 18 (2) (2014) 286–300.
[8] J. Ceberio, E. Irurozki, A. Mendiburu, J.A. Lozano, Extending distance-based ranking models in estimation of distribution algorithms, in: Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, 2014, pp. 2459–2466.
[9] J. Ceberio, A. Mendiburu, J.A. Lozano, Introducing the Mallows model on estimation of distribution algorithms, in: Neural Information Processing, Springer, 2011, pp. 461–470.
[10] J. Ceberio, A. Mendiburu, J.A. Lozano, Kernels of Mallows Models for solving permutation-based problems, in: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, ACM, 2015, pp. 505–512.
[11] J. Ceberio, R. Santana, A. Mendiburu, J.A. Lozano, Mixtures of Generalized Mallows models for solving the quadratic assignment problem, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2015, pp. 2050–2057.
[12] P.C. Chang, S.H. Chen, Q. Zhang, J.L. Lin, MOEA/D for flowshop scheduling problems, in: Proceedings of the Congress on Evolutionary Computation (CEC), IEEE, 2008, pp. 1433–1438.
[13] C.C. Coello, G. Lamont, D. van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems., Genetic and Evolutionary Computation, second ed., Springer, Berlin, Heidelberg, 2007.
[14] W.E. Costa, M.C. Goldbarg, E.G. Goldbarg, New VNS heuristic for total flowtime flowshop scheduling problem, Expert Syst. Appl. 39 (9) (2012) 8149–8161.
[15] J. Derrac, S. Garcia, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms., Swarm Evol. Comput. 1 (1) (2011) 3–18.
[16] J. Dubois-Lacoste, M. López-Ibáñez, T. Stützle, A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems, Comput. Oper. Res. 38 (8) (2011) 1219–1236.
[17] M.A. Fligner, J.S. Verducci, Multistage ranking models, J. Am. Stat. Assoc. 83 (403) (1988) 892–901.
[18] F. Glover, Tabu search-part i, ORSA J. Comput. 1 (3) (1989) 190–206.
[19] J.H. Holland, Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, University of Michigan Press, Ann Arbor, MI, 1975.
[20] E. Irurozki, B. Calvo Molinos, J.A. Lozano Alonso, Sampling and learning the Mallows and Generalized Mallows models under the Cayley distance, Technical Report, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 2014a.
[21] E. Irurozki, B. Calvo Molinos, J.A. Lozano Alonso, Sampling and learning the Mallows and Generalized Mallows models under the Ulam distance, Technical Report, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 2014b.
[22] H. Ishibuchi, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, IEEE Trans. Evol. Comput. 7 (2) (2003) 204–223.
[23] H. Karshenas, R. Santana, C. Bielza, P. Larrañaga, Multi-objective optimization with joint probabilistic modeling of objectives and variables, in: International Conference on Evolutionary Multi-Criterion Optimization (CEC), in: Lecture Notes in Computer Science, Springer, 2011, pp. 298–312.
[24] L. Ke, Q. Zhang, R. Battiti, Hybridization of decomposition and local search for multiobjective optimization, IEEE Trans. Cybern. 44 (10) (2014) 1808–1820.
[25] P. Larrañaga, H. Karshenas, C. Bielza, R. Santana, A review on probabilistic graphical models in evolutionary computation, J. Heuristics 18 (5) (2012) 795–819.
[26] , Estimation of distribution algorithms. a new tool for evolutionary computation, in: P. Larrañaga, J.A. Lozano (Eds.), Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[27] G. Lebanon, Y. Mao, Non-Parametric modeling of partially ranked data, in: Advances in neural information processing systems, 2007, pp. 857–864.
[28] K. Li, K. Deb, Q. Zhang, S. Kwong, An evolutionary many-objective optimization algorithm based on dominance and decomposition, IEEE Trans. Evol. Comput. 19 (5) (2015) 694–716.
[29] X. Li, M. Li, Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem, IEEE Trans. Eng. Manage. 62 (4) (2015) 544–557.
[30] Z. Li, P. Liu, C. Deng, S. Guo, P. He, C. Wang, Evaluation of estimation of distribution algorithm to calibrate computationally intensive hydrologic model, J. Hydrol. Eng. 21 (6) (2016) 1–8.
[31] J. Liu, C.R. Reeves, Constructive and composite heuristic solutions to the ci scheduling problem, Eur J Oper Res 132 (2) (2001) 439–452.
[32] C.L. Mallows, NON-NULL ranking models. i, Biometrika 44 (1957) 114–130.
[33] L. Martí, J. García, A. Berlanga, J.M. Molina, Multi-objective optimization with an adaptive resonance theory-based estimation of distribution algorithm: a comparative study, in: International Conference on Learning and Intelligent Optimization, Springer, 2011, pp. 458–472.
[34] A. Mendiburu, J. Miguel-Alonso, J.A. Lozano, M. Ostra, C. Ubide, Parallel EDAs to create multivariate calibration models for quantitative chemical applications, J. Parallel Distrib. Comput. 66 (8) (2006) 1002–1013.
[35] K. Miettinen, Nonlinear multiobjective optimization, vol. 12, Springer Science & Business Media, 2012.
[36] G. Minella, R. Ruiz, M. Ciavotta, A review and evaluation of multiobjective algorithms for the flowshop scheduling problem, Eur. J. Oper. Res. 20 (3) (2008) 451–471.
[37] G. Minella, R. Ruiz, M. Ciavotta, Restarted iterated pareto greedy algorithm for multi-objective flowshop scheduling problems, Comput. Oper. Res. 38 (11) (2011) 1521–1533.
[38] H. Mühlenbein, G. Paass, From recombination of genes to the estimation of distributions i. binary parameters, in: International Conference on Parallel Problem Solving from Nature (PPSN), Springer, 1996, pp. 178–187.
[39] T. Murata, H. Ishibuchi, Performance evaluation of genetic algorithms for flowshop scheduling problems, in: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, IEEE, 1994, pp. 812–817.
[40] V. Pareto, Cours d'économie politique, vol. 1, Librairie Droz, 1964.
[41] M. Pelikan, K. Sastry, D.E. Goldberg, Multiobjective estimation of distribution algorithms, in: M. Pelikan, K. Sastry, E. Cantú-Paz (Eds.), Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, Springer, 2006, pp. 223–248.
[42] V. Santucci, M. Baioletti, A. Milani, Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion, IEEE Trans. Evol. Comput. 20 (5) (2016) 682–694.
[43] O. Schütze, X. Esquivel, A. Lara, C.A.C. Coello, Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 16 (4) (2012) 504–522.
[44] R.E. Steuer, Multiple Criteria Optimization: Theory, Computation, and Applications, Wiley, 1986.
[45] E. Taillard, Benchmarks for basic scheduling problems, Eur. J. Oper. Res. 64 (2) (1993) 278–285.
[46] A. Trivedi, D. Srinivasan, K. Sanyal, A. Ghosh, A survey of multi-objective evolutionary algorithms based on decomposition, IEEE Trans. Evol. Comput. (2016). Accpeted for publication.
[47] S. Tsutsui, Probabilistic model-building genetic algorithms in permutation representation domain using edge histograms, in: Parallel Problem Solving from Nature VIII, Lecture Notes in Computer Science, vol. 2439, Springer, 2002, pp. 224–233.
[48] S. Tsutsui, Node histogram vs. edge histogram: A comparison of probabilistic model-building genetic algorithms in permutation domains, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 1939–1946.
[49] X. Xu, Z. Xu, X. Gu, An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization, Expert Syst. Appl. 38 (7) (2011) 7970–7979.
[50] J. Yan, C. Li, Z. Wang, L. Deng, S. Demin, Diversity Metrics in Multi-objective Optimization: Review and Perspective., in: Proceedings of the 10th International Conference on Integration Technology (ICIT), IEEE, 2007, pp. 553–557.
[51] M.M. Yenisey, B. Yagmahan, Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends, Omega 45 (2014) 119–135.
[52] M. Zangari, R. Santana, M.A. Pozo, MOEA/D-GM: Using probabilistic graphical models in MOEA/D for solving combinatorial optimization problems, Technical Report, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 2015.
[53] M. Zeleny, J.L. Cochrane, Multiple criteria decision making, University of South Carolina Press, 1973.
[54] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[55] A. Zhou, Q. Zhang, G. Zhang, A multiobjective evolutionary algorithm based on decomposition and probability model, in: IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–8.
[56] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.