Undefined 0 (2016) 1–8
IOS Press

A discrete differential evolution algorithm for multi-objective permutation flowshop scheduling

Marco Baioletti [a], Alfredo Milani [a] and Valentino Santucci [a]
[a] *Department of Mathematics and Computer Science, University of Perugia, Via Vanvitelli, 06123 Perugia Italy*
E-mail: {marco.baioletti,alfredo.milani}@unipg.it, valentino.santucci@dmi.unipg.it

Abstract. The more practical and interesting versions of the permutation flowshop scheduling problem (PFSP) have a variety of objective criteria to be optimized simultaneously. Multi-objective PFSP is also a relevant combinatorial multi-objective optimization problem. In this paper we propose a multi-objective evolutionary algorithm for PFSP by extending the previously proposed discrete differential evolution (DE) scheme for single-objective PFSP. This is the first application of the algebraic-based discrete DE to multi-objective problems. The algorithm is extended by adopting a variety of crossover and multi-objective selection operators. Among these, the multi-objective α-selection is a novelty of this work and can be decoupled from DE and used also in other evolutionary algorithms. The other crossover and selection operators have been taken from the existing literature and, where required, have been adapted to the problem at hand. An experimental evaluation has been conducted on all the three bi-objective PFSPs among the makespan, total flowtime and total tardiness criteria. The results show that the proposed approach is competitive with respect to the state-of-the-art algorithms.

Keywords: differential evolution, permutation flowshop scheduling, multi-objective optimization, combinatorial optimization

1. Introduction

The Permutation Flowshop Scheduling Problem (PFSP) is an important type of scheduling problem which has many applications in manufacturing and large scale product fabrication. In this problem there are n jobs $J_1, \ldots, J_n$ and m machines $M_1, \ldots, M_m$. Each job J_i is composed by m operations $O_{i1}, \ldots, O_{im}$. The generic operation O_{ij} can be executed only by the machine M_j and its given processing time is p_{ij}. Moreover, the execution of any operation cannot be interrupted (no pre-emption) and job passing is not allowed, i.e., the jobs must be executed using the same order in every machine. The goal of PFSP is to find the optimal job permutation $\pi = \langle \pi(1), \ldots, \pi(n) \rangle$ with respect to a given objective function. Three important criteria are to minimize the total flowtime (TFT), the makespan (MS) and the total tardiness (TT), respectively defined by:

$$TFT(\pi) = \sum_{i=1}^{n} C(\pi(i), m) \quad (1)$$

$$MS(\pi) = \max_{i=1,\ldots,n} C(\pi(i), m) = C(\pi(n), m) \quad (2)$$

$$TT(\pi) = \sum_{i=1}^{n} \max\{C(\pi(i), m) - d_{\pi(i)}, 0\} \quad (3)$$

where $C(h, j)$ is the completion time of the operation O_{hj} and is computed by the following recursive equation

$$C(\pi(i), j) = p_{\pi(i),j} + \max\{C(\pi(i-1), j), C(\pi(i), j-1)\}$$

for $i, j \geq 1$, while the terminal cases are $C(\pi(0), j) = C(i, 0) = 0$. In Equation (3), for each job h, also a given delivery date d_h is considered. Other criteria, not considered in this article, include: earliness, number of tardy jobs, idle time and completion time variance [46].

The minimization of each one of these criteria is computationally hard. Indeed, both the TFT and TT problems are NP-hard for $m \geq 2$, while the MS minimization becomes NP-hard when $m > 2$ [18].

Many PFSP single-optimization algorithms exist, either exact or approximate, for instance: heuristic techniques, local searches or evolutionary algorithms [17]. Anyway, in this paper we investigate the PFSP problem as a multi-objective optimization problem, in which the goal is to find a set of job permutations which are good enough with respect to two or more contrasting criteria, i.e., a set of Pareto optimal solutions.

Given k objective functions $f_1, \ldots, f_k$, a solution x dominates a solution x' (denoted by $x \prec x'$) if $f_i(x) \leq f_i(x')$ for $i = 1, \ldots, k$, and there exists at least one index $j \in \{1, \ldots, k\}$ such that $f_j(x) < f_j(x')$. A solution x is Pareto optimal if there exists no other solution x' such that $x' \prec x$. The Pareto set is the set of all the Pareto optimal solutions. If two solutions x and x' are such that neither $x \prec x'$ nor $x' \prec x$, then x and x' are incomparable.

Since the Pareto set is in general very large, the goal is to find an approximation of this set, i.e., a set composed by incomparable solutions which is as close as possible to the Pareto set. One of the most promising approaches to solve multi-objective optimization problems is to use evolutionary algorithms [8]. Indeed, evolutionary algorithms evolve a population of elements and among them find the Pareto optimal solutions.

In the context of multi-objective PFSP, many approaches have been proposed. The surveys [26,46] describe and compare a large number of algorithms for PFSP with all the three possible combinations of two objectives among TFT, MS and TT. The approaches mostly used are based on some metaheuristics, like Genetic Algorithms, Simulated Annealing, Tabu Search, Iterated Local Search, and Evolutionary Strategy.

In this paper we describe an algorithm for multi-objective optimization which is based on Differential Evolution for Permutation (DEP) [33]. DEP is a discrete differential evolution algorithm which directly operates on the permutations space and hence is well suited for permutation optimization problems like PFSP. Indeed, in [6,33,34,35], it was shown that DEP reaches state-of-the-art results with respect to total flowtime and makespan single objective optimization. Here, DEP has been extended in order to handle multi-objective problems. This paper extend the preliminary work presented in [37] by proposing the adoption of a variety of crossover and selection operators. Therefore, a deeper experimental investigation, with respect to [37], is provided. The experimental results show that the performance obtained are comparable with the state-of-the-art algorithms.

The rest of the paper is organized as follows. The second section is devoted to describe the related work. The third section describes the classical Differential

Evolution algorithm. Its extension to the permutations space and multi-objective PFSP is introduced in the fourth and fifth sections. An experimental investigation of the proposed approach is provided in the sixth section, while conclusions are drawn at the end of the paper.

2. Related work

There exist many approaches to solve the permutation flowshop scheduling problem in a multiobjective setting.

Here we describe some of the most performing algorithms according to [26], used also in our comparison. We use the same names for each algorithm as in [26].

The most effective algorithm is MOSA, a simulated annealing method for multiobjective problem described by Varadharajan and Rajendran in [44]. In this algorithm, the initial population is composed with two solutions generated by means of a heuristic function. These sequences are evolved by three improvement schemes and are later used, alternatively, as the solution of the simulated annealing method. MOSA tries to obtain nondominated solutions through the implementation of a simple probability function that tries to generate solutions on the Pareto optimal front. The probability function is varied in such a way that the entire objective space is covered uniformly obtaining as many nondominated and well-dispersed solutions as possible.

Arroyo and Armentano proposed in [3] a genetic local search algorithm, called MOGALS_arroyo, with some particular features. A method to preserve the population diversity is used. Moreover, the selection is elitist (a subset of the current Pareto front is directly copied to the next generation), the solutions are enhanced by a multiobjective local search. The concept of Pareto dominance is used to assign fitness (using the nondominated sorting procedure and the crowding measure both proposed for the NSGAII [12]) to the solutions and in the local search procedure.

A Genetic Algorithm, called PESA, was proposed by Corne et al. in [10]. PESA uses an external population (EP) and an internal population (IP) to find a well-spread Pareto front. A selection and replacement procedure are based on the degree of crowding. A simple genetic scheme is used for the evolution of IP, while EP contains the nondominated solutions found. The size of the EP is upper bounded and a hypergrid-based operator eliminates the individuals in the more crowded zones.

PESAII, described in [9], is an enhancement of PESA, obtained by adopting a different selection method. In this algorithm, instead of assigning a selective fitness to an individual, it is assigned to the hyperboxes in the objective space which are occupied by at least one element. During the selection process, the hyperbox with the best fitness is selected, and an individual is chosen at random among all inside the selected hyperbox.

Another genetic algorithm, called PGA_ALS, was proposed by Pasupathy et al [30]. This algorithm initially generates four good solutions that are introduced in a random population. PGA_ALS evolves the internal population using a Pareto ranking-based procedure similar to that used in NSGAII. A crowding procedure is also proposed and used as a secondary selection criterion. The nondominated solutions are kept in an external archive, and two different local searches are then applied to half of the stored solutions for improving the quality of the returned Pareto front.

In [2] a multiobjective tabu search method called MOTS is described. Several paths of solutions in parallel are explored, and each path has its own tabu list. The initial solutions is generated using a heuristic. A local search method is applied to the set of current solutions to generate several new solutions. A clustering procedure ensures that the size of the current solution set remains constant. An external archive is used to memorize all the nondominated solutions found during the execution.

For the description of the other algorithms see the survey [26]. Many of them are Genetic Algorithm enhanced by some particular features such as multiobjective selection operators. Other approaches can be also found in [46].

3. The Differential Evolution algorithm

In this section we provide a short introduction to Differential Evolution (DE) algorithm. For more details see [31]. Differential Evolution (DE) is a powerful population-based evolutionary algorithm [25,38] for optimizing non-linear and even non-differentiable real functions in $\mathbb{R}^n$. The main peculiarity of DE is to exploit the distribution of the differences between solutions in order to probe the search space.

DE initially generates a random population of NP candidate solutions $x_1, \ldots, x_{NP}$ uniformly distributed

in the solutions space. At each generation, DE performs a mutation and a crossover operation in order to produce, for each target individual x_i in the current population, a trial vector u_i. Each target vector is then replaced in the next generation by the associated trial vector if and only if the trial is better than the target. This process is iteratively repeated until a stop criterion is met, for instance a given amount of fitness evaluations has been performed, or after a certain amount of time.

The differential mutation is the core operator of DE and generates a mutant vector v_i for each target individual x_i. The most used mutation scheme is "rand/1" and it is defined as

$$v_i = x_{r_0} + F \cdot (x_{r_1} - x_{r_2}) \tag{4}$$

where r_0, r_1, r_2 are three random integers in $[1, NP]$ mutually different among them and with respect to i. x_{r_0} is called base vector, $x_{r_1} - x_{r_2}$ is the difference vector, and $F \in [0, 1]$ is the scale factor parameter, while $+$, $-$ and $\cdot$ are the usual vectorial operations in $\mathbb{R}^n$. In [31] it is argued that the differential mutation confers to DE the ability to automatically adapt the mutation step size and orientation to the fitness landscape at hand.

After the mutation, a crossover operator generates a population of NP trial vectors u_i by recombining each mutant v_i and its corresponding target x_i. The most used crossover operator is the binomial one that builds the trial vector u_i taking some components from x_i and some components from v_i according to the crossover probability $CR \in [0, 1]$.

Finally, in the selection phase, the next generation population is usually selected by a one-to-one tournament among x_i and u_i, for $1 \leq i \leq NP$. The fittest enters the population for the next generation.

Many variants for mutation, crossover and selection have been proposed, see for instance [31].

Even if the mutation is designed for numerical vectors, DE can be also applied to combinatorial optimization problems by means of particular techniques.

One of the most used approach is to encode elements of the discrete search space S as numerical vectors. These vectors are evolved by the usual mutation and crossover operators. In the selection phase, when the elements have to be evaluated, the vectors are decoded back to elements of S, since the function f to be optimized is defined on S. The main problem of this approach is that in many combinatorial optimization problems, any reasonable decoding function $d : \mathbb{R}^n \to S$ is highly non-injective, due to the different cardinalities of the domain and the range of the function d. In this way, many different vectors correspond to the same element $x \in S$, and then large plateaus are likely to be present in the search space. This is notoriously known to degrade the performance of a search algorithm (see for instance [19]).

4. The Algebraic Evolution algorithm

In a series of papers [5,6,33,34,35,36] we have defined a discrete differential evolution algorithm which works directly on a discrete search space represented as a finitely generated group.

A group G is a set endowed with an internal binary operation $\circ$ which is associative, there exists a neutral element $e \in G$ such that $x \circ e = e \circ x = x$ for all $x \in G$, and each $x \in G$ has a corresponding inverse element x^{-1} such that $x \circ x^{-1} = x^{-1} \circ x = e$. A group G is finitely generated if there exists a subset $H \subseteq G$, whose elements are called *generators*, such that for each $x \in G$ there exists a finite sequence $h_1, \ldots, h_k$ of elements of H with the property that $h_1 \circ \cdots \circ h_k = x$. The sequence $h_1, \ldots, h_k$ is called a decomposition of x and it is not generally unique. Anyway, let $|x|$ denote the length of a minimal decomposition.

The differential mutation operator (4) can be defined in a finitely generated group G by interpreting the vector operations $+$, $-$ and $\cdot$ as operations inside G. In particular the addition $x + y$ can be replaced by $x \circ y$, while the subtraction $x - y$ can be replaced by $y^{-1} \circ x$. In this way, addition and subtraction are related with the same property as in $\mathbb{R}^n$, indeed, $y + (x - y)$ corresponds to $y \circ (y^{-1} \circ x)$ and both are equals to x.

The multiplication of x by a coefficient $F \in [0, 1]$ can be interpreted in the following way: take a minimal decomposition $h_1, \ldots, h_k$ of x (hence $k = |x|$) and compute $h_1 \circ \cdots \circ h_l$, where $l = \lceil F \cdot k \rceil$.

Since a minimal decomposition of x may not be unique, the latter operation is not uniquely defined. Anyway, since its use is limited to Differential Evolution, which is a stochastic algorithm, our proposal is to generate a random minimal decomposition of x. In order to avoid biases in the search process, it is strictly required to use a fair generator of random decomposition, which can sample a decomposition of x as uniformly as possible.

It is worth to notice that, using this interpretation, each $x \in G$ can be seen either as an element of the search space, or as the difference between two ele-

ments of the search space, similarly to what happens for vectors in $\mathbb{R}^n$.

A particular group which has been studied in our previous work is $\mathcal{S}(n)$, i.e., the group of the permutations of the set $\{1, \ldots, n\}$, where $\circ$ is the usual permutations composition operator. The resulting differential evolution algorithm, called Differential Evolution for Permutation (DEP), has been tested on some combinatorial optimization problems based on permutations. Notably, DEP have obtained state-of-the-art results in permutation flowshop scheduling problems [33,34,35] and very good performance in other problems [5,6].

The group $\mathcal{S}(n)$ has several generating sets. In our works we have focused the attention on the set of adjacent swaps, i.e. the permutations which swaps two consecutive positions i and $i+1$, for $1 \leq i < n$.

In [33,35] we have described an algorithm which finds a random minimal decomposition of a permutation in terms of adjacent swaps. Since this algorithm is a generalization of the Bubble-Sort, it has been called RandBS (Randomized Bubble-Sort).

Hence, the key component of the algorithm DEP, i.e., the differential mutation, produces a mutant ν_i for each population individual π_i in the following way

1. Find r_0, r_1, r_2 different to i and to each other
2. $\delta \leftarrow \pi_{r_2}^{-1} \circ \pi_{r_1}$
3. $S \leftarrow RandBS(\delta)$
4. $k \leftarrow len(S)$
5. $l \leftarrow \lceil F \cdot k \rceil$
6. $\nu_i \leftarrow \pi_{r_0}$
7. for $j = 1, \ldots, l$ apply S_j to ν_i

For the sake of completeness, all the previous proposals of DEP apply a purposely defined selection scheme and a permutation-based crossover.

5. Discrete Differential Evolution for Multi-Objective Optimization

In this section we describe the algorithm Multi-Objective Differential Evolution for Permutations (MODEP) which directly evolves a population of NP permutations $\{\pi_1, \ldots, \pi_{NP}\}$. With respect to the classical DE, important variations have been made to the mutation, crossover and selection operators. Moreover, an additional archive of non-dominated solutions, i.e., ND, is introduced to maintain the evolved Pareto front.

To simplify our description, let us restrict to the case of two objective functions f_1 and f_2. A population of NP permutations $\{\pi_1, \ldots, \pi_{NP}\}$ is randomly generated at the beginning. At each iteration, a secondary population of trial elements $\{\upsilon_1, \ldots, \upsilon_{NP}\}$ is generated by means of the mutation and crossover operators. Then, a selection operator selects the next population from the two populations $\{\pi_1, \ldots, \pi_{NP}\}$ and $\{\upsilon_1, \ldots, \upsilon_{NP}\}$.

The pseudo-code of MODEP is depicted in Alg. 1.

Algorithm 1 MODEP

```
1:  Initialize Population
2:  Update ND
3:  while num_fit_eval ≤ max_fit_eval do
4:      for i ← 1 to NP do
5:          ν_i ← DifferentialMutation(i)
6:          υ_i^(1), υ_i^(2) ← Crossover(π_i, ν_i)
7:          Update ND
8:          υ_i ← SelectChild(υ_i^(1), υ_i^(2))
9:      end for
10:     π ← Selection (π, υ)
11: end while
```

The differential mutation has been already described in Section 4, in the rest of this section we will describe the other algorithmical components.

5.1. Crossover

In the Algebraic Differential Evolution, also the crossover operator has to be changed according to the discrete search space. Indeed, the crossover operators used in the classical Differential Evolution are designed for vectors (or strings) and cannot be used for other combinatorial objects like the permutations.

In particular, permutations can be seen as special strings with an "all different" constraint, and there exist many crossover operators designed for them. Among these operators, we have selected:

- edge recombination (ER) [45] which builds a trial whose pairs of adjacent items are inherited and assorted as much as possible from the parent solutions;
- partially_matched (PM) [16] where two random cut-points are selected, the middle region in the first parent is copied to the trial, the items in the middle region of the second parent are placed in the trial by mapping them through the first parent, and the remaining items are filled as in the second parent;

- uniform_like (UL) [43] that scans the items from left to right and, for each position, randomly chooses one of the parents' items, but, if the item has been already used before, the entry is left empty and randomly filled later;
- position_based (PB) [42] which builds a trial by copying a randomly selected subset of items from one parent and by inserting the remaining items using the order of their appearance in the other parent;
- order_based (OB) [42] that is a slight variation of PB where the order of items in the position selected in one parent is imposed on the corresponding position in the other parent;
- order (O) [14] which is similar to PB except that the randomly selected items belong to an interval of positions;
- order2 (O2) [1] that is like O with the variation that the items of the second parent are circularly inserted into the offspring starting from the position immediately after the chosen interval;
- maximal_preservative (MP) [27] which, similarly to ER, tries to preserve as much as possible the pairs of adjacent items in both parents;
- cyclic (CY) [14] that identifies a number of so-called cycles between the two parents, then, the trial is filled by alternating the cycles from the two parents;
- cyclic2 (CY2) [1] which works as CY, but starting from a randomly selected position.

All of these operators can be applied in two ways: $v_i^{(1)} \leftarrow CROSS(\pi_i, \nu_i)$ and $v_i^{(2)} \leftarrow CROSS(\nu_i, \pi_i)$.

Both children are generated and evaluated, also to update the ND set. Then, only one trial is kept for the selection phase. Indeed, $v_i^{(1)}$ and $v_i^{(2)}$ are compared with respect to both f_1 and f_2. If $v_i^{(1)} \prec v_i^{(2)}$, then the trial v_i is $v_i^{(1)}$. Analogously, if $v_i^{(2)} \prec v_i^{(1)}$, then the trial v_i is $v_i^{(2)}$. When $v_i^{(1)}$ and $v_i^{(2)}$ are incomparable, then one of them is randomly selected.

5.2. Selection

In the classical Differential Evolution with a single objective, the selection phase is simple because there are no cases of incomparability between the trial and the target vectors. Having many objective functions, this situation has to handled. Therefore, we have implemented 5 selection operators: α-selection, DEMO-selection, GDE2-selection, NDSA-selection and Crowding-selection. In particular, α-selection is a novelty of this work and it extends to the multi-objective case the selection scheme previously presented in [33].

5.2.1. α-selection

The new multi-objective α-selection operator chooses the new population element π_i' between the old element π_i and the trial ν_i by using $\prec$ as comparison.

If $\pi_i \prec \nu_i$, then π_i' becomes π_i, i.e., π_i remains in the population. Otherwise, if $\nu_i \prec \pi_i$ or it is equal to π_i, then ν_i replaces π_i in the next generation population. However, if π_i and ν_i are incomparable, then we use a probabilistic method somehow similar to the α-selection described in [33,35].

Suppose first that $f_1(\nu_i) < f_1(\pi_i)$ but $f_2(\nu_i) \geq f_2(\pi_i)$. Then, π_i' becomes ν_i with probability $\max\{0, \alpha_2 - \Delta_i^{(2)}\}$, otherwise it retains the old element π_i, where

$$\Delta_i^{(2)} = \frac{f_2(\nu_i) - f_2(\pi_i)}{f_2(\pi_i)}$$

is the relative worsening of ν_i with respect to π_i according to f_2.

Analogously, if $f_1(\nu_i) \geq f_1(\pi_i)$ and $f_2(\nu_i) < f_2(\pi_i)$. Then, π_i' becomes ν_i with probability $\max(0, \alpha_1 - \Delta_i^{(1)})$, where

$$\Delta_i^{(1)} = \frac{f_1(\nu_i) - f_1(\pi_i)}{f_1(\pi_i)}.$$

The rationale behind this selection operator is that ν_i enters the population if it dominates or is equal to π_i or, with a small probability, if it is not too worse than π_i in one of the objective functions, while it is better than π_i in the other objective function. Moreover, note that the probability of accepting a slightly worsening population element linearly shades from α_h, when $\Delta_i^{(h)} = 0$, to 0, when $\Delta_i^{(h)} = \alpha_h$, for $h = 1, 2$.

Therefore, the parameters α_h regulates how worse ν_i can be in order to be accepted in the new population: if $\alpha_1 = \alpha_2 = 0$ only better elements (in the Pareto sense) can replace old elements in the population. Hence, α-selection generalizes the classic one-to-one Pareto selection by providing a parametrized mechanism to break ties.

5.2.2. NDSA-selection

The NDSA-selection operates on the overall population $\{\pi_i\}_{i=1}^{NP} \cup \{\nu_i\}_{i=1}^{NP}$ by the well known Non Dominated Sorting Algorithm [13], which computes

the dominance depth for each element and a crowding distance for all the elements at the same dominance depth. Elements are compared with respect the dominance depth, and ties are broken considering the crowding distance. This order relation is denoted by $\prec_{NDSA}$ and it is used to sort the overall population. The best NP elements are selected for the next generation.

5.2.3. GDE2-selection

The GDE2-selection is based on the selection operator used in General Differential Evolution 2 algorithm [23]. After having computed, as in the previously described operator, the Pareto level and the crowding distance for all the element of the overall population, each trial ν_i is compared with the corresponding target π_i with respect to $\prec_{NDSA}$ relation and the better of them is kept in the next population.

5.2.4. DEMO-selection

The DEMO-selection is taken from Differential Evolution for Multi-Objective Optimization algorithm [32]. Again, the dominance depth and the crowding distance are computed for all the elements of the overall population. But conversely from GDE2-selection, each trial ν_i is compared, using the $\prec_{NDSA}$ relation, with $\pi_{j(i)}$, which is the element in the previous population closest to ν_i. The distance we have used is the well known *footrule-distance* between permutations, also called *position-based distance* in [39]. This choice of the distance measure is motivated by its low computational complexity of $\Theta(n)$ [39].

5.2.5. Crowding-selection

A simple variation of DEMO-selection is that, instead of using $\prec_{NDSA}$, it is possible to compare elements with respect to $\prec$. Therefore, in this operator only when $\nu_i \prec \pi_{j(i)}$, the trial enters in the new population instead of the corresponding target. Note that in case of incomparability, the target remains in the population.

5.3. Pareto Front

The algorithm keeps updated the approximated Pareto front ND, which contains all the non-dominated elements ever generated. Initially ND contains all the non-dominated population elements created during the random initialization. Then, at each generation, all the couples of children $v_i^{(1)}$ and $v_i^{(2)}$ are used to update ND. A new element v enters ND if it is not dominated by any element of ND. Moreover, all the elements of ND which are dominated by v are removed.

6. Experimental Results

MODEP has been implemented in C++ and tested over the commonly adopted benchmark suite composed by the Taillard's instances for the flowshop-scheduling problem and the additional due times given in [26]. These instances are divided in 11 groups of 10 instances basing on the numbers of jobs n and machines m. The values of n are in the set $\{20, 50, 100, 200\}$, while m lies in $\{5, 10, 20\}$. The combination $(n = 200, m = 5)$ is not present, while, as done in [26], also the combination $(n = 500, m = 20)$ has been removed from the original Taillard set. The processing times p_{ij} of each instance have been randomly generated in $\{1, \ldots, 99\}$, while the due date of each job J_i is generated by multiplying the value $\sum_{j=1}^{m} p_{ij}$ by a random factor in $[1, 4]$. Three combinations of objectives have been considered: (MS, TFT), (MS, TT), and (TFT, TT).

Every algorithm execution has been performed using $2000 \cdot n \cdot m$ fitness evaluations. For each execution, the obtained Pareto front (corresponding to ND) has been analyzed by computing two performance indices: the hypervolume I_H and the unary multiplicative epsilon I_ϵ^1. These indices are the ones suggested in [26],

I_H is computed as the area delimited by the solutions of ND, after each coordinate has been normalized in $[0, 1]$, and the reference point $(1.2, 1.2)$. The normalization process is the same adopted in [26].

I_ϵ^1 compares the obtained ND with the best known Pareto front B and is computed as

$$I_\epsilon^1 = \max_{x \in B} \min_{y \in ND} \max_{j=1,2} \frac{f_j(y)}{f_j(x)}.$$

The best known Pareto front for each instance are obtained from [26]. Moreover, it worths to note that every single Pareto front has not necessarily been obtained from a single algorithm, indeed, in [26] the authors have merged the results of 23 different algorithms.

The values for indices I_H and I_ϵ^1 reported in the following have been computed by averaging over the multiple executions.

The rest of the section is organized as follows. In Section 6.1 a calibration of the DEP components and parameters is performed. The calibrated DEP configurations are tested in Section 6.2 by performing an experimental analysis and comparing them with the best known Pareto fronts of every problem instance considered. Finally, in Section 6.3, the experimental results are aggregated and compared with those of the state-of-the-art algorithms for multi-objective PFSPs.

Table 1
Calibration Results produced by *irace*

(MS, TT)	(TFT, TT)	(MS, TFT)
$(DEMO, OB, 100)$	$(CROWD, O2, 100)$	$(GDE2, CY2, 100)$
$(DEMO, O2, 50)$	$(DEMO, O2, 50)$	$(CROWD, O2, 100)$
$(DEMO, OB, 50)$	$(CROWD, OB, 50)$	$(GDE2, O2, 50)$
$(DEMO, O2, 100)$	$(CROWD, O2, 50)$	$(DEMO, CY, 100)$

6.1. Calibration of MODEP

Since we have implemented 5 different selection operators (α-selection has also two parameters α_1 and α_2), 10 crossover operators and the algorithm has also the population size parameter NP, we have decided to tune MODEP by using the *irace* software package [24].

Indeed, *irace* (Iterated Race for Automatic Algorithm Configuration) is able to automatically select the best configurations of a parametric algorithm by comparing the performance of all the variants on a set of given instances. *irace* implements the iterated racing procedure and adopt statistical tests to iteratively discard algorithm configurations. It is implemented as an *R* package and produces in output the list of the four best configurations. A budget of 2000 executions, each one of $2000 \cdot n \cdot m$ fitness evaluations, has been set for the *irace* calibration. All the other *irace* parameters have been set to their default values.

In order to avoid the "overtuning" issue, we have decide to randomly generate new calibration instances different from the testing instances. Therefore, 8 new instances have been randomly generated using the same method adopted for the Taillard's instances. The instance sizes considered are: $(n = 20, m = 10)$, $(n = 20, m = 20)$, $(n = 50, m = 10)$, $(n = 50, m = 20)$, $(n = 100, m = 10)$, $(n = 100, m = 20)$, $(n = 200, m = 10)$ and $(n = 200, m = 20)$.

The set of values for the population size NP has been discretized to $\{50, 100, 150\}$, while all the proposed crossover and selection operators have been considered. *irace* has been separately run on the three objective combinations (MS, TT), (TFT, TT), and (MS, TFT) by comparing the configuration results with respect to the hypervolume measure I_H.

Table 1 provides, for each objectives combination, the best four MODEP configurations produced by the *irace* calibration. The dashed lines indicate where there are significant differences (in the statistical sense) among the configurations.

All the significantly best configurations in every objective combinations have been selected for the testing phase. Both for the (MS, TT) and (TFT, TT) cases, there is a single MODEP configuration that significantly outperforms the others, respectively $(DEMO, OB, 100)$ and $(CROWD, O2, 100)$. Instead, in the (MS, TFT) case, *irace* has not been able to provide a sharp suggestion between the first two configurations two configurations $(GDE2, CY2, 100)$ and $(CROWD, O2, 100)$, so both have been selected for the testing phase.

Note that all the four selected configurations are somehow different regarding the selection and crossover operators, but all of them adopt the same population size $NP = 100$. This fact is not surprising because the three optimization problems, although sharing the same data, could have different fitness landscapes and each of them would need an ad-hoc configuration. Nevertheless, it is interesting the indication about the population size. Finally, the 12 best configurations reported in Table 1 consider 3 selection operators and 4 crossover operators, thus ruling out the remaining 2 selection and 6 crossover schemes.

Table 2
MODEP (DEMO,OB,100) results for (MS, TT) objectives

n	m	I_H	I_ϵ^1
20	5	1.415	1.012
20	10	1.411	1.015
20	20	1.388	1.021
50	5	1.380	1.042
50	10	1.371	1.055
50	20	1.349	1.074
100	5	1.357	1.067
100	10	1.351	1.076
100	20	1.326	1.094
200	10	1.284	1.083
200	20	1.263	1.095

6.2. Experimental Test of MODEP

MODEP has been run 10 times for each instance and the adopted stopping criterion is the maximum number of evaluations, which has been set to $2000 \cdot n \cdot m$ using, for each combination of objectives, the configuration(s) selected by *irace* and described in Section 6.1. The performance measures employed are the aforementioned hypervolume I_H and the unary multiplicative epsilon I_ϵ^1. All the results have been aggregated for instance size n, m and the average measures are provided in Tables 2, 3, 4, and 5, respectively for the objectives combinations: (MS, TT), (TFT, TT), (MS, TFT) with the first MODEP configuration, and (MS, TFT) with the second MODEP configuration.

Table 3
MODEP (CROWD,O2,100) results for (TFT, TT) objectives

n	m	I_H	I_ϵ^1
20	5	1.417	1.003
20	10	1.421	1.009
20	20	1.401	1.011
50	5	1.391	1.051
50	10	1.368	1.053
50	20	1.365	1.068
100	5	1.370	1.061
100	10	1.351	1.072
100	20	1.346	1.093
200	10	1.338	1.087
200	20	1.298	1.105

Table 4
MODEP (GDE2,CY2,100) results for (MS, TFT) objectives

n	m	I_H	I_ϵ^1
20	5	1.411	1.007
20	10	1.402	1.011
20	20	1.370	1.012
50	5	1.388	1.047
50	10	1.370	1.047
50	20	1.355	1.074
100	5	1.369	1.065
100	10	1.350	1.093
100	20	1.282	1.116
200	10	1.268	1.098
200	20	1.252	1.116

Regarding (MS, TT), as shown by Table 2, MODEP works well on this problem and the values of the second index I_ϵ^1 (whose optimal value is 1) are quite good, also the values for I_H (whose optimal value is 1.44) are good compared to those reported in [26]. It is worth to notice that I_H and I_ϵ^1 get worse results as n increases and, fixing n, as m increases.

The results of the optimization of (TFT, TT) are shown in Table 3 and are similar to those for (MS, TFT). Again the decreasing behaviour of I_H and I_ϵ^1 with respect to n and m is present.

Finally, the results of the optimization of (MS, TFT) are shown in Tables 4 and 5, for the selected configurations $(GDE2, CY2, 100)$ and $(CROWD, O2, 100)$, respectively. In this case the results show that no configuration is always better than the other. Anyway, the same behaviour observed in the two previous tables occurs also here.

Conclusively, we can say that the experimental results obtained for the performance measures considered are close to their optimal theoretical values (1.44 for I_H and 1 for I_ϵ^1), thus indicating very good performances of the proposed algorithm.

Table 5
MODEP (CROWD,O2,100) results for (MS, TFT) objectives

n	m	I_H	I_ϵ^1
20	5	1.411	1.007
20	10	1.404	1.011
20	20	1.368	1.011
50	5	1.398	1.050
50	10	1.369	1.048
50	20	1.353	1.074
100	5	1.367	1.063
100	10	1.354	1.091
100	20	1.285	1.115
200	10	1.270	1.097
200	20	1.251	1.111

6.3. Comparison with State-of-the-art Algorithms

In this section, we provide a comparison of MODEP with the 23 state-of-the-art schemes considered in [26]. All the algorithms used for comparison are briefly explained or referenced in Section 2.

In order to make the comparison we aligned our results to those of [26] by considering the same reference points for I_H and the same reference Pareto fronts for I_ϵ^1. We decided to compare our results with those of [26] obtained with the largest stopping criterion of $200 \cdot n \cdot m/2$ milliseconds. Though our stopping criterion is expressed in fitness evaluations we have verified that the running time in seconds is no worse than 115% of that of [26]. Moreover, we think that the budget of evaluations is a more fair criterion for future comparisons.

The new rankings among the multi-objective PFSP algorithms are provided in Tables 6, 7 and 8 respectively for (MS, TT), (TFT, TT) and (MS, TFT). The results are ordered by the hypervolume values I_H. Furthermore, we can observe that, almost always, as the hypervolume decreases, the epsilon indicator increases. Indeed, the indicators are contradictory in very few cases, thus indicating that the rankings provided in the tables are somehow meaningful.

The tables clearly show that MODEP is the second best performing algorithm (over 23) on all the three objectives combinations. In particular, in the (MS, TT) and (MS, TFT) cases (see Tables 6 and 8) the distance from the most performing scheme, namely MOSA_Varadharajan [44], is not worse than 0.003

Table 6
State-of-the-art Comparison for (MS, TT) objectives

Rank	Algorithm	I_H	I_ϵ^1
1	MOSA_Varadharajan [44]	1.357	1.056
2	**MODEP (DEMO,OB,100)**	1.354	1.058
3	MOGALS_Arroyo [3]	1.301	1.080
4	PESA [10]	1.291	1.081
5	PESAII [9]	1.288	1.084
6	PGA_ALS [30]	1.255	1.118
7	MOTS [2]	1.235	1.129
8	MOGA_Murata [29]	1.195	1.138
9	CMOGA [28]	1.181	1.143
10	NSGAII [12]	1.165	1.155
11	SPEA [47]	1.164	1.155
12	CNSGAII [11]	1.162	1.154
13	ϵ-NSGAII [22]	1.115	1.182
14	$(\mu + \lambda)$-PAES [21]	1.110	1.183
15	ϵ-MOEA [11]	1.061	1.216
16	PAES [21]	1.028	1.230
17	MOSA_Suresh [41]	0.955	1.303
18	SA_Chakravarty [7]	0.870	1.410
19	PILS [15]	0.866	1.363
20	A-IBEA [48]	0.599	1.514
21	ENGA [4]	0.559	1.542
22	SPEAII [20]	0.522	1.549
23	NSGA [40]	0.490	1.593
24	B-IBEA [48]	0.417	1.637

Table 7
State-of-the-art Comparison for (TFT, TT) objectives

Rank	Algorithm	I_H	I_ϵ^1
1	MOSA_Varadharajan [44]	1.375	1.036
2	**MODEP (CROWD,O2,100)**	1.369	1.056
3	MOGALS_Arroyo [3]	1.324	1.061
4	PGA_ALS [30]	1.315	1.074
5	MOTS [2]	1.309	1.067
6	PESA [10]	1.263	1.088
7	PESAII [9]	1.262	1.087
8	$(\mu + \lambda)$-PAES [21]	1.200	1.121
9	MOGA_Murata [29]	1.198	1.120
10	CMOGA [28]	1.188	1.125
11	NSGAII [12]	1.176	1.134
12	CNSGAII [11]	1.171	1.135
13	SPEA [47]	1.157	1.141
14	ϵ-NSGAII [22]	1.144	1.148
15	ϵ-MOEA [11]	1.111	1.164
16	PAES [21]	1.066	1.197
17	MOSA_Suresh [41]	1.048	1.216
18	PILS [15]	0.900	1.334
19	SA_Chakravarty [7]	0.584	1.498
20	SPEAII [20]	0.445	1.576
21	ENGA [4]	0.397	1.635
22	A-IBEA [48]	0.339	1.670
23	NSGA [40]	0.338	1.686
24	B-IBEA [48]	0.338	1.671

units in both the measures, while, all the other 22 competitors are sharply distant from MODEP.

In order to have a better understanding of the results produced by MODEP, Figures 1, 2 and 3 show the plot of a MODEP run against the best known Pareto front from [26] on a few problem instances. The shown instances are the first ones of the $(n = 100, m = 5)$ and $(n = 200, m = 20)$ groups for the (MS, TT) and (MS, TFT) objectives (Figures 1 and 3), while, for the (TFT, TT) case (Figure 2), due to the small cardinality of the best known Pareto front on the previous instances, the first instance of the groups $(n = 50, m = 20)$ and $(n = 100, m = 20)$ have been chosen. In order to analyze the plots, we recall that the best known Pareto front provided in [26] are a merge of solutions obtained by 23 different algorithms, thus, potentially, no single algorithm is able to generate the Pareto front provided.

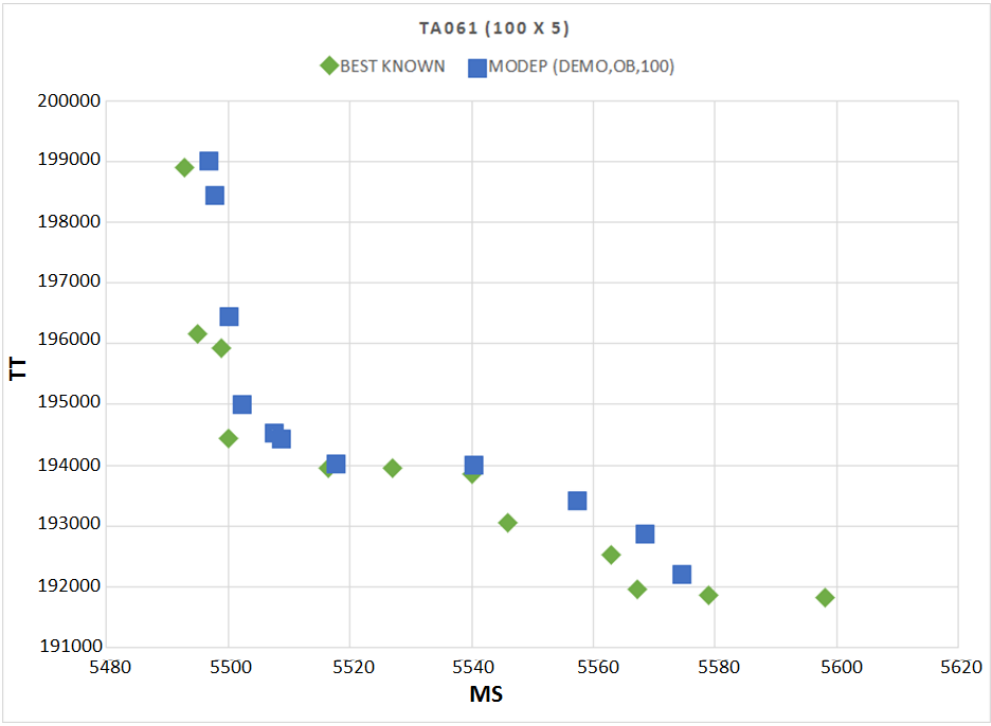
The plots show that the Pareto fronts obtained by MODEP are very close to the best known ones. In particular, in some cases, as for example in Figures 3a and 3b, some MODEP solutions are not dominated from the known ones and, more interestingly, they dominates some known solutions. Therefore, new best known Pareto front can be obtained by merging the previous ones with the solutions provided by MODEP.

Summarizing, in the light of the results provided we can conclude that MODEP is among the state-of-the-art algorithms for the multi-objective PFSP problems.

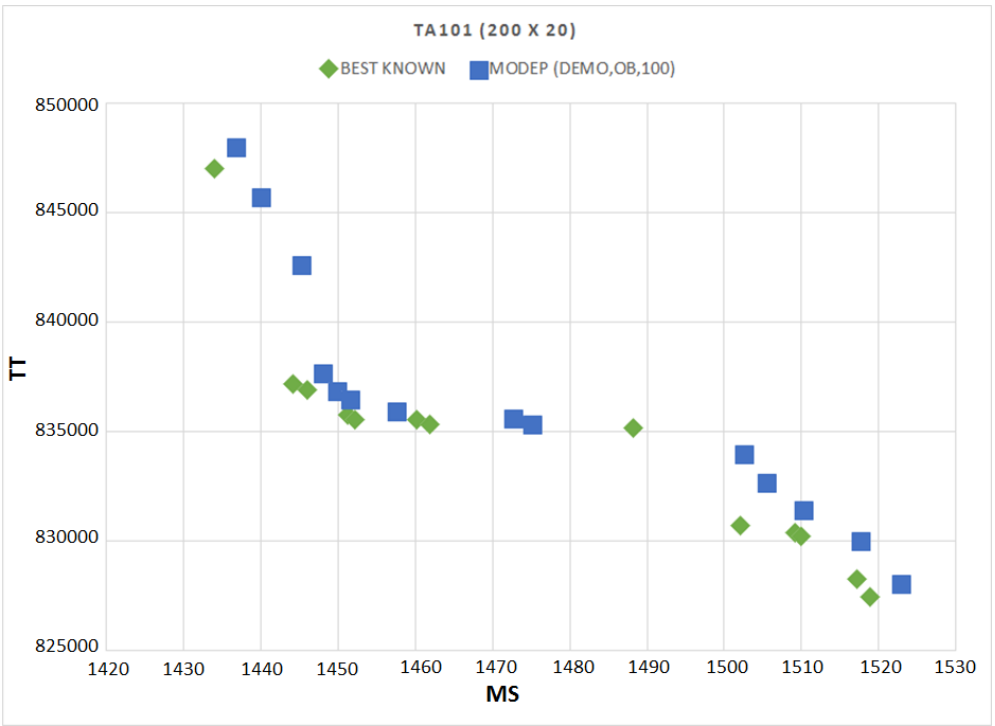
7. Conclusion and Future Work

In this paper we have described an algorithm for optimization of multi-objective permutation flowshop scheduling problems using the algebraic differential evolution approach.

This is the first application of the algebraic-based discrete DE to multi-objective problems. A large variety of crossover and selection operators available in literature have been implemented in the algorithm. Among these, the multi-objective α-selection is a novelty of this work and can be decoupled from DE and used also in other evolutionary algorithms.

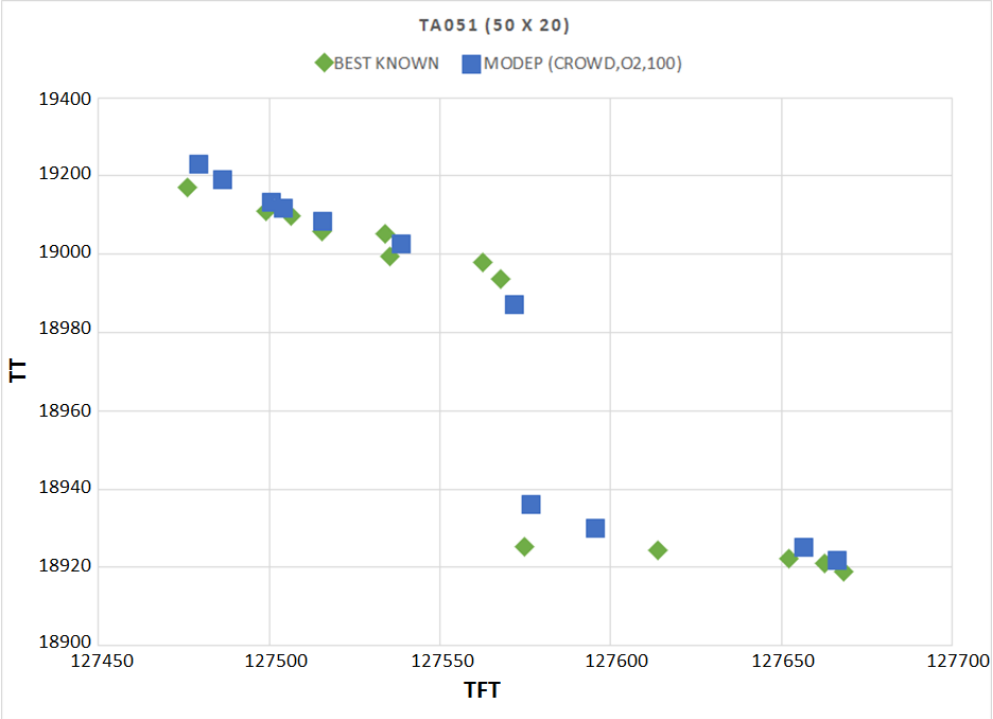


(a) Obj. (MS, TT) - Inst. Ta061 $(n = 100, m = 5)$

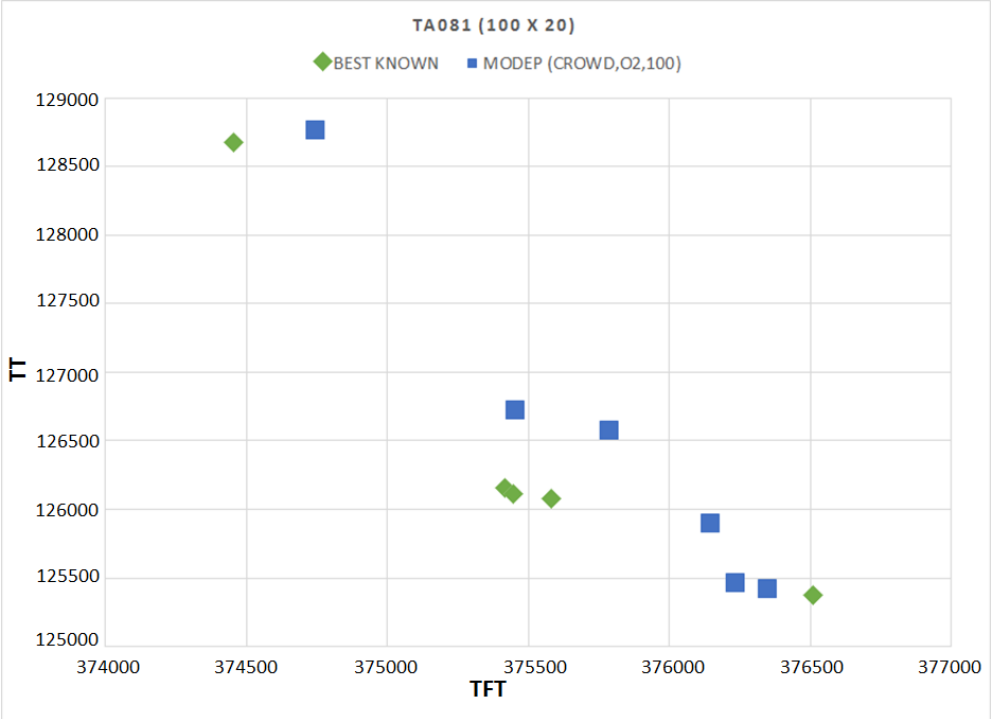


(b) Obj. (MS, TT) - Inst. Ta101 $(n = 200, m = 20)$

Fig. 1. Pareto Fronts Comparisons on (MS, TT) - MODEP vs Best Known

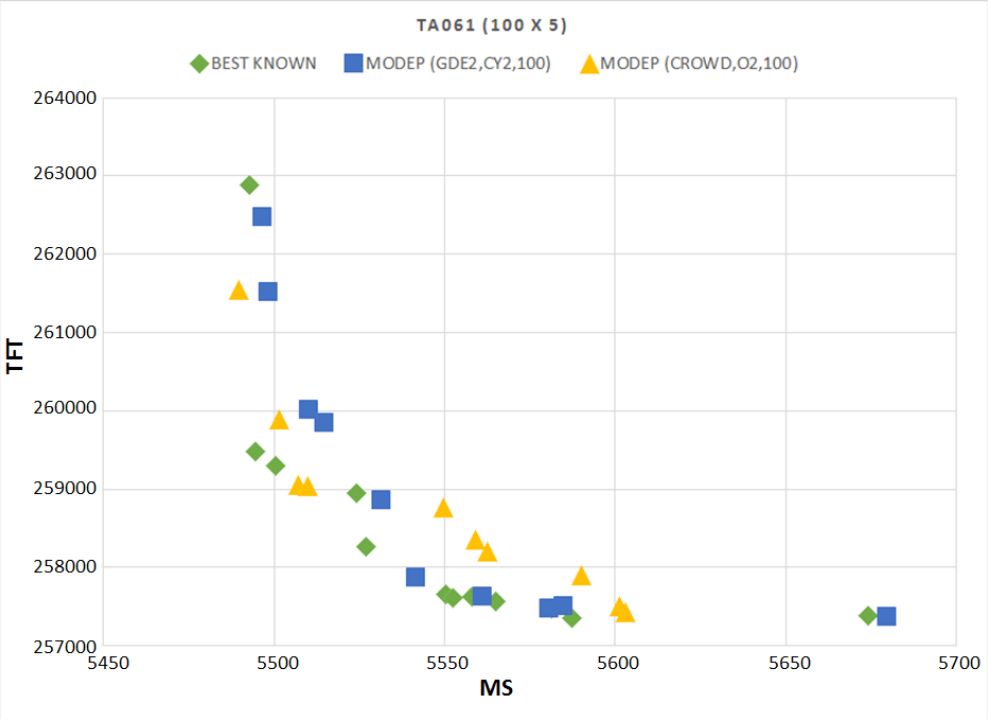


(a) Obj. (TFT, TT) - Inst. Ta051 $(n = 50, m = 20)$

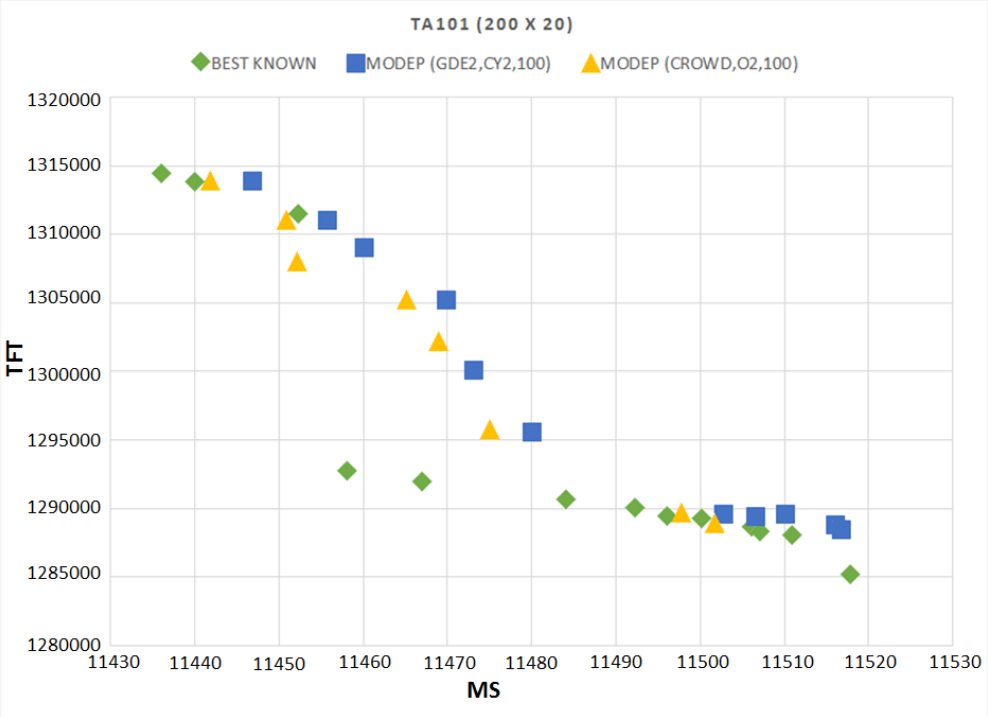


(b) Obj. (TFT, TT) - Inst. Ta081 $(n = 100, m = 20)$

Fig. 2. Pareto Fronts Comparisons on (TFT, TT) - MODEP vs Best Known



(a) Obj. (MS, TFT) - Inst. Ta061 $(n = 100, m = 5)$



(b) Obj. (MS, TFT) - Inst. Ta101 $(n = 200, m = 20)$

Fig. 3. Pareto Fronts Comparisons on (MS, TFT) - MODEP vs Best Known

Table 8
State-of-the-art Comparison for (MS, TFT) objectives

Rank	Algorithm	I_H	I_ϵ^1
1	MOSA_Varadharajan [44]	1.350	1.061
2	**MODEP (CROWD,O2,100)**	1.348	1.062
3	**MODEP (GDE2,CY2,100)**	1.345	1.062
4	MOGALS_Arroyo [3]	1.336	1.064
5	MOTS [2]	1.312	1.083
6	PESA [10]	1.275	1.091
7	PGA_ALS [30]	1.272	1.122
8	PESAII [9]	1.268	1.093
9	MOGA_Murata [29]	1.176	1.146
10	CMOGA [28]	1.167	1.149
11	CNSGAII [11]	1.133	1.168
12	NSGAII [12]	1.128	1.173
13	SPEA [47]	1.122	1.173
14	$(\mu + \lambda)$-PAES [21]	1.077	1.196
15	ϵ-NSGAII [22]	1.077	1.199
16	ϵ-MOEA [11]	1.040	1.226
17	PAES [21]	0.980	1.252
18	MOSA_Suresh [41]	0.922	1.314
19	PILS [15]	0.861	1.367
20	SA_Chakravarty [7]	0.781	1.431
21	ENGA [4]	0.477	1.604
22	SPEAII [20]	0.458	1.600
23	B-IBEA [48]	0.425	1.631
24	A-IBEA [48]	0.424	1.631
25	NSGA [40]	0.409	1.658

The experimental results show that this method is promising and reaches results which are comparable to the state-of-the-art algorithms. Since the best MODEP configurations in the different objectives combinations are all different, as a future work, we would like to explore the possibility of designing a meta MODEP which automatically chooses the selection and crossover, for instance by a self-adapting scheme. Another possible line of research is to extend our investigation to other objective functions, like earliness, number of tardy jobs, etc.

Finally, we would like to test our multi-objective differential evolution algorithm also in other multi-objective combinatorial optimization problems, for instance the multi-objective knapsack problem.

References

[1] Michael Affenzeller, Stefan Wagner, Stephan Winkler, and Andreas Beham. *Genetic algorithms and genetic programming: modern concepts and practical applications*. Crc Press, 2009.

[2] Vinícius Amaral Armentano and José Elias Claudio. An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem. *Journal of Heuristics*, 10(5):463–481, 2004.

[3] Jose Elias Claudio Arroyo and Vinicius Amaral Armentano. Genetic local search for multi-objective flowshop scheduling problems. *European Journal of Operational Research*, 167(3):717 – 738, 2005.

[4] Tapan P. Bagchi. *Pareto-Optimal Solutions for Multi-objective Production Scheduling Problems*, pages 458–471. Springer Berlin Heidelberg, 2001.

[5] M. Baioletti, A. Milani, and V. Santucci. Linear ordering optimization with a combinatorial differential evolution. In *2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2135–2140, 2015.

[6] Marco Baioletti, Alfredo Milani, and Valentino Santucci. An extension of algebraic differential evolution for the linear ordering problem with cumulative costs. In *Proc. of PPSN XIV - Parallel Problem Solving from Nature 2016*, pages 123–133. Springer International Publishing, 2016.

[7] Karunakaran Chakravarthy and Chandrasekharan Rajendran. A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization. *Production Planning & Control*, 10(7):707–714, 1999.

[8] Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler. Evolutionary multi-objective optimization. *European Journal of Operational Research*, 181(3):1617–1619, 2007.

[9] David W. Corne, N. R. Jerram, Joshua D. Knowles, and M. J. Oates. *PESA-II: Region-based selection in evolutionary multiobjective optimization*, pages 283–290. Morgan Kaufmann, San Francisco, 2001.

[10] David W. Corne, Joshua D. Knowles, and Martin J. Oates. *The Pareto Envelope-Based Selection Algorithm for Multiobjective Optimization*, pages 839–848. Springer Berlin Heidelberg, 2000.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[12] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[13] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002.

[14] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag Berlin Heidelberg, 2003.

[15] Martin Josef Geiger. On operators and search space topology in multi-objective flow shop scheduling. *European Journal of Operational Research*, 181(1):195–206, 2007.

[16] David E. Goldberg and Robert Jr. Lingle. Alleles loci and the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 154–159, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[17] J. Gupta and J.E. Stafford. Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711, 2006.

[18] Jatinder N.D. Gupta and Edward F. Stafford Jr. Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699 – 711, 2006.

[19] Holger H Hoos and Thomas Stützle. *Stochastic local search: Foundations & applications*. Elsevier, 2004.

[20] Mifa Kim, Tomoyuki Hiroyasu, Mitsunori Miki, and Shinya Watanabe. Spea2+: Improving the performance of the strength pareto evolutionary algorithm 2. In *International Conference on Parallel Problem Solving from Nature*, pages 742–751. Springer, 2004.

[21] Joshua D. Knowles and David W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evol. Comput.*, 8(2):149–172, 2000.

[22] Joshua B. Kollat and Patrick M. Reed. *The Value of Online Adaptive Search: A Performance Comparison of NSGAII, ϵ-NSGAII and ϵMOEA*, pages 386–398. Springer Berlin Heidelberg, 2005.

[23] Saku Kukkonen and Jouni Lampinen. An extension of generalized differential evolution for multi-objective optimization with constraints. In *Proc. of PPSN VIII - Parallel Problem Solving from Nature 2004*, pages 752–761, 2004.

[24] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[25] Alfredo Milani and Valentino Santucci. Community of scientist optimization: An autonomy oriented approach to distributed optimization. *AI Communications*, 25(2):157–172, 2012.

[26] Gerardo Minella, Rubén Ruiz, and Michele Ciavotta. A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451–471, 2008.

[27] Heinz Muhlenbein. Evolution in time and space-the parallel genetic algorithm. In *Foundations of genetic algorithms*, pages 316–337, 1991.

[28] Tadahiko Murata, Hisao Ishibuchi, and Mitsuo Gen. *Specification of Genetic Search Directions in Cellular Multi-objective Genetic Algorithms*, pages 82–95. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[29] Tadahiko Murata, Hisao Ishibuchi, and Hideo Tanaka. Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers & Industrial Engineering*, 30(4):957 – 968, 1996.

[30] T. Pasupathy, Chandrasekharan Rajendran, and R.K. Suresh. A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. *The International Journal of Advanced Manufacturing Technology*, 27(7):804–815, 2006.

[31] K.V. Price, R.M. Storn, and J.A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Berlin, 2005.

[32] Tea Robič and Bogdan Filipič. DEMO: differential evolution for multiobjective optimization. In *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, pages 520–533, 2005.

[33] V. Santucci, M. Baioletti, and A. Milani. Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. *IEEE Transactions on Evolutionary Computation*, 20(5):682–694, Oct 2016.

[34] V. Santucci, M. Baioletti, and A. Milani. Solving permutation flowshop scheduling problems with a discrete differential evolution algorithm. *AI Communications*, 29(2):269–286, 2016.

[35] Valentino Santucci, Marco Baioletti, and Alfredo Milani. A differential evolution algorithm for the permutation flowshop scheduling problem with total flow time criterion. In *Proc. of PPSN XIII - Parallel Problem Solving from Nature 2014*, pages 161–170, 2014.

[36] Valentino Santucci, Marco Baioletti, and Alfredo Milani. An algebraic differential evolution for the linear ordering problem. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2015)*, pages 1479–1480, 2015.

[37] Valentino Santucci, Marco Baioletti, and Alfredo Milani. A discrete differential evolution algorithm for multi-objective permutation flowshop scheduling. In *Proceedings of IPS 2015 - Italian Workshop on Planning and Scheduling*, pages 80–87, 2015.

[38] Valentino Santucci, Alfredo Milani, and Flavio Vella. A study on the synchronization behaviour of differential evolution and a self-adaptive extension. *Journal of Artificial Intelligence and Soft Computing Research*, 2(4):279–301, 2012.

[39] Tommaso Schiavinotto and Thomas Stützle. A review of metrics on permutations for search landscape analysis. *Computers and Operational Research*, 34(10):3143–3153, 2007.

[40] N. Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, 2(3):221–248, 1994.

[41] R. K. Suresh and K. M. Mohanasundaram. Pareto archived simulated annealing for permutation flow shop scheduling with multiple objectives. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 2, pages 712–717, 2004.

[42] Gilbert Syswerda. Schedule optimization using genetic algorithms. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, pages 332–349. Van Nostrand Reinhold, New York, NY, 1991.

[43] David M. Tate and Alice E. Smith. A genetic approach to the quadratic assignment problem. *Computers and Operations Research*, 22(1):73–83, 1995.

[44] T. K. Varadharajan and Chandrasekharan Rajendran. A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. *European Journal of Operational Research*, 167(3):772–795, 2005.

[45] L. Darrell Whitley, Timothy Starkweather, and D'Ann Fuquay. Scheduling problems and traveling salesmen: The genetic edge recombination operator. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 133–140, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[46] M.M. Yenisey and B. Yagmahan. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, 45:119–135, 2014.

[47] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[48] Eckart Zitzler and Simon Künzli. *Indicator-Based Selection in Multiobjective Search*, pages 832–842. Springer Berlin Heidelberg, 2004.