MOEA/DEP: An Algebraic Decomposition-based Evolutionary Algorithm for the Multiobjective Permutation Flowshop Scheduling Problem

No Author Given

No Institute Given

Abstract. Algebraic evolutionary algorithms are an emerging class of meta-heuristics for combinatorial optimization based on strong mathematical foundations. In this paper we introduce a decomposition-based algebraic evolutionary algorithm, namely MOEA/DEP, in order to deal with multiobjective permutation-based optimization problems. As a case of study, MOEA/DEP has been experimentally validated on a multiobjective permutation flowshop scheduling problem (MoPFSP). In particular, the makespan and total flowtime objectives have been investigated. Experiments have been held on a widely used benchmark suite, and the obtained results have been compared with respect to the state-of-the-art Pareto fronts for MoPFSP. The experimental results have been analyzed by means of two commonly used performance metrics for multiobjective optimization. The analysis clearly shows that MOEA/DEP reaches new state-of-the-art results for the considered benchmark.

Keywords: Algebraic Evolutionary Algorithms, Multiobjective Optimization, Permutation Flowshop Scheduling Problem

1 Introduction and Related Work

Multiobjective optimization is receiving a growing level of interest in the evolutionary computation community (see for example [18, 22, 23]). Indeed, several real-world problems can be modeled using two or more objectives to be optimized simultaneously. Since the objectives may contrast each other, a population of somehow good (i.e., Pareto optimal) solutions is generally required by a decision maker. Hence, the innate ability of evolutionary algorithms to deal with populations of solutions explains their wide and successful application in multiobjective optimization.

Recently, an algebraic framework has been successfully proposed as a general method to transform popular continuous meta-heuristics, such as Differential Evolution (DE) [13] and Particle Swarm Optimization (PSO) [2], to powerful algebraic evolutionary algorithms for combinatorial optimization. In particular, DEP (Differential Evolution for Permutations) has obtained state-of-the-art results on single-objective Permutation Flowshop Scheduling Problems (PFSPs) [13, 14].

In this paper, basing on the widely known decomposition-based MOEA/D framework [21], we propose an algebraic evolutionary algorithm for multiobjective permutation-based optimization problems. Our algorithm, namely MOEA/DEP, is mainly based on the algebraic differential mutation operator [13, 3]. As a case of study, we apply MOEA/DEP to a multiobjective PFSP (MoPFSP) [11] considering the makespan and total flowtime as objectives.

In MoPFSP, a set $J = \{1, \ldots, n\}$ of n jobs has to be scheduled on a set $M = \{1, \ldots, m\}$ of m machines. M is provided with a fixed order and each job has to visit, in the given order, all the machines. The processing time of job $j \in J$ on machine $i \in M$ is given by $p_{i,j}$. Every machine can process only one job at a time. Preemption and job-passing are not allowed. Hence, MoPFSP requires to find a permutation $\pi = \langle \pi(1), \ldots, \pi(n) \rangle$ of the n jobs that minimizes both the makespan (MS) and total flowtime (TFT) objective functions, defined, respectively, as

$$f_{MS} = c(m, \pi(n)), \tag{1}$$

$$f_{TFT} = \sum_{j=1}^{n} c(m, \pi(j)), \tag{2}$$

where, in both definitions, $c(i, \pi(j))$ is the completion time of the job $\pi(j)$ on the i-th machine and it is recursively computed as

$$c(i, \pi(j)) = p_{i,\pi(j)} + \max\{c(i, \pi(j-1)), c(i-1, \pi(j))\}, \tag{3}$$

when $i > 1$ and $j > 1$, while the terminal cases are:

$$\begin{aligned} c(1, \pi(1)) &= p_{1,\pi(1)}, \\ c(1, \pi(j)) &= p_{1,\pi(j)} + c(1, \pi(j-1)), \\ c(i, \pi(1)) &= p_{i,\pi(1)} + c(i-1, \pi(1)). \end{aligned} \tag{4}$$

A literature review for MoPFSP has been proposed in [11], where the performances of 23 algorithms have been compared. According to the authors, when MS and TFT are considered as objectives, the best performing algorithm is a multiobjective variant of the simulated annealing algorithm [19]. These results were later improved by RIPG [12] and TP+PLS [9]: both algorithms are mainly based on local search procedures hybridized with other heuristic mechanisms. However, to the best of our knowledge, current state-of-the-art results are those recently obtained in [20] by an estimation of distribution algorithm based on Mallows models, namely MEDA/D-MK. Therefore, we have experimentally compared our proposal with the reference Pareto fronts provided by the authors of [20].

The rest of the paper is organized as follows. Background concepts about multiobjective optimization and the popular MOEA/D framework are provided in Section 2. Section 3 briefly recalls the algebraic framework for combinatorial evolutionary algorithms. MOEA/DEP is introduced in Section 4, while the experimental analysis is provided in Section 5. Finally, conclusions are drawn in Section 6 where some future lines of research are also depicted.

2 Multiobjective Optimization and MOEA/D Framework

Without loss of generality, a Multiobjective Optimization Problem (MOP) can be represented as k objective functions $f_1, \ldots, f_k$ to minimize, each one defined on a *decision space* X, i.e., $f_i : X \to \mathbb{R}$ for all $i \in \{1, \ldots, k\}$.

Given two solutions $x, y \in X$, x *dominates* y, denoted by $x \prec y$, if and only if: $f_i(x) \leq f_i(y)$ for all $i \in \{1, \ldots, k\}$, and $f_i(x) < f_i(y)$ for at least one i. If the two solutions x and y are such that neither $x \prec y$ nor $y \prec x$, then they are *incomparable*.

A solution $x \in X$ is *Pareto optimal* if there exists no other solution $y \in X$ such that $y \prec x$. The *Pareto set* (PS) is the set of all the Pareto optimal solutions for a given MOP. It is easy to show that the solutions in PS are incomparable. The images of the PS solutions in the *objective space* are called *Pareto front* (PF), i.e., $PF = \{(f_1(x), \ldots, f_k(x)) | x \in PS\}$.

The goal of an algorithm for multiobjective optimization is to find a set of incomparable solutions that approximates as much as possible the Pareto front of the given MOP. Often, diversity in the objective space is also taken into account in order to provide a good coverage of the true Pareto front.

The population-based nature of Evolutionary Algorithms (EAs) makes them particularly suited in solving MOPs, hence a variety of multiobjective EAs (MOEAs) have been proposed [7, 18, 22]. The most popular classes of MOEAs are: (i) domination-based, (ii) indicator-based, and (iii) decomposition-based. In this paper we focus on the decomposition-based approach and, in particular, on the MOEA/D framework originally proposed in [21].

MOEA/D decomposes a MOP into several single objective subproblems such that every population individual optimizes its own subproblem. The population evolves and interacts by means of variation and replacement operators, while the approximated Pareto set is maintained as an external archive of incomparable solutions which are updated every time a new solution is generated.

MOEA/D requires a set of weight vectors and an aggregation function in order to generate N single objective subproblems and their neighborhoods.

The N weight vectors $\lambda_1, \ldots, \lambda_N$ are k-dimensional (where k is the number of objectives), their components are non-negative and sum-up to 1. With the aim of maintaining a certain degree of diversity, the weight vectors need to be evenly spread. In [21], the setting of N and $\lambda_1, \ldots, \lambda_N$ is controlled by a parameter H, thus that $\lambda_1, \ldots, \lambda_N$ are all the possible weight vectors whose components take a value from $\left\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H}\right\}$ and sum-up to 1. Hence, N and H are related by $N = \binom{H+k-1}{k-1}$.

The aggregation function has the form $g : X \times \mathbb{R}^k \to \mathbb{R}$. The two most popular choices for g are the weighted sum g^{ws} and the Tchebycheff function g_z^{tc} that, given a solution $x \in X$ and a weight vector $\lambda \in \mathbb{R}^k$, are respectively defined as

$$g^{ws}(x|\lambda) = \sum_{i=1}^{k} \lambda_i f_i(x), \tag{5}$$

$$g_z^{tc}(x|\lambda) = \max_{1 \leq i \leq k} \{\lambda_i |f_i(x) - z_i|\}, \tag{6}$$

where $z \in \mathbb{R}^k$ is the reference point in the objective space that it is ideally set such that $z_i = \min\{f_i(x)|x \in X\}$ for all $i \in \{1, \ldots, k\}$. Practically, it is dynamically adjusted with the best objective values encountered so far in the evolution.

N population individuals, one for each weight vector, are deployed. The i-th individual is formed by: a candidate solution x_i, a weight vector λ_i, its fitness is given by $g(x_i|\lambda_i)$, and its neighborhood B_i is defined as $B_i = \{i_1, \ldots, i_T\}$, where, for all $j \in B_i$, λ_j is among the T closest vectors to λ_i (by considering Euclidean distance).

The solutions $x_1, \ldots, x_N$ are randomly initialized and the non-dominated ones enter the external archive EP. The populations $x_1, \ldots, x_N$ and EP are iteratively evolved by means of genetic variation and replacement operators. For each individual i, a new solution x'_i is generated by using genetic operators, such as crossover and mutation, designed for the representation at hand. Usually, x'_i is obtained by recombining one or more solutions from $\{x_j|j \in \{i\} \cup B_i\}$. Then, x'_i replaces x_i if fitter in the i-th subproblem, i.e., when $g(x'_i|\lambda_i) < g(x_i|\lambda_i)$. Similarly, neighbors replacement is performed by comparing x'_i with its neighbors, i.e., for each index $j \in B_i$, x'_i replaces x_j if $g(x'_i|\lambda_j) < g(x_j|\lambda_j)$. Moreover, x'_i is also used to update the solutions in EP. At the end of the evolution, the approximated Pareto set is given by EP.

3 Algebraic Differential Evolution for Permutations

As described in [13], the design of the Algebraic Differential Evolution (ADE) resembles that of the classical DE. The population $\{x_1, \ldots, x_N\}$ of N candidate solutions is iteratively evolved by means of the three operators of differential mutation, crossover and selection. Differently from numerical DE, ADE addresses combinatorial optimization problems whose search space is representable by finitely generated groups. Since crossover and selection schemes for combinatorial spaces are widely available in literature, the main focus is on the Differential Mutation (DM) operator. DM is widely recognized as the key component of DE [16] and, in its most common variant, generates a mutant v according to

$$v \leftarrow x_{r_0} \oplus F \odot (x_{r_1} \ominus x_{r_2}) \tag{7}$$

where $x_{r_0}, x_{r_1}, x_{r_2}$ are three randomly selected population individuals, while $F \in [0, 1]$ is the DE scale factor parameter. In numerical DE, the operators $\oplus, \ominus, \odot$ are the usual vector operations of $\mathbb{R}^n$, while, in ADE, their definitions are formally derived using the underlying algebraic structure of the search space.

The triple $(X, \circ, G)$ is a finitely generated group representing a combinatorial search space if:

- X is the discrete set of solutions;

- $\circ$ is a binary operation on X satisfying the group properties, i.e., closure, associativity, identity (e), and invertibility (x^{-1}); and
- $G \subseteq X$ is a finite generating set of the group, i.e., any $x \in X$ has a (not necessarily unique) minimal-length decomposition $\langle g_1, \ldots, g_l \rangle$, with $g_i \in G$ for all $i \in \{1, \ldots, l\}$, and whose evaluation is x, i.e., $x = g_1 \circ \cdots \circ g_l$.

For the sake of clarity, the length of a minimal decomposition of x is denoted with $|x|$.

Using $(X, \circ, G)$ we can provide the formal definitions of the operators $\oplus, \ominus, \odot$. Let $x, y \in X$ and $\langle g_1, \ldots, g_k, \ldots, g_{|x|} \rangle$ be a minimal decomposition of x, then

$$x \oplus y := x \circ y, \tag{8}$$

$$x \ominus y := y^{-1} \circ x, \tag{9}$$

$$F \odot x := g_1 \circ \cdots \circ g_k, \text{ with } k = \lceil F \cdot |x| \rceil \text{ and } F \in [0, 1]. \tag{10}$$

The algebraic structure on the search space naturally defines neighborhood relations among the solutions. Indeed, it induces a colored digraph whose vertices are the solutions in X and two generic solutions $x, y \in X$ are linked by an arc with color $g \in G$ if and only if $y = x \circ g$. Therefore, a simple one-step move can be directly encoded by a generator, while a composite move can be synthesized as the evaluation of a sequence of generators (a path on the graph). In analogy with $\mathbb{R}^n$, the elements of X can be dichotomously interpreted both as solutions (vertices on the graph) and as displacements between solutions (colored paths on the graph). As detailed in [13], this allows to provide a rational interpretation to the discrete DM of definition (7). The key idea is that the difference $x \ominus y$ is the evaluation of the generators on a shortest path from y to x.

Clearly, $\oplus$ and $\ominus$ do not depend on the generating set, thus they are uniquely defined. Conversely, $\odot$ relies on the chosen generating set and, also fixing it, a minimal decomposition is not unique in general. Therefore, $\odot$ is implemented as a stochastic operator, thus requiring a randomized decomposition algorithm for the finitely generated group at hand.

The algebraic Differential Evolution for Permutations (DEP) [13] is an implementation of ADE for the search space of permutations. Indeed, the permutations of the set $\{1, \ldots, n\}$, together with the usual permutation composition, form the so-called Symmetric group $\mathcal{S}(n)$. Since $\mathcal{S}(n)$ is finite, it is also finitely generated. Different choices for the generating set are possible. The most common and practical ones are:

- *ASW*, which is based on *adjacent swap moves* and it is formally defined as

$$ASW = \{\sigma_i : 1 \leq i < n\}, \tag{11}$$

  where σ_i is the identity permutation with the items i and $i + 1$ exchanged. Hence, given a generic $\pi \in \mathcal{S}_n$, the composition $\pi \circ \sigma_i$ swaps the i-th and $(i + 1)$-th items of π.

- *EXC*, which is based on *exchange moves* and it is formally defined as

$$EXC = \{\epsilon_{ij} : 1 \leq i < j \leq n\}, \tag{12}$$

  where ϵ_{ij} is the identity permutation with the items i and j exchanged. Hence, given a generic $\pi \in \mathcal{S}_n$, the composition $\pi \circ \epsilon_{ij}$ swaps the i-th and j-th items of π.
- *INS*, which is based on *insertion moves* and it is formally defined as

$$INS = \{\iota_{ij} : 1 \leq i, j \leq n\}, \tag{13}$$

  where ι_{ij} is the identity permutation where the item i is shifted to position j. Hence, given a generic $\pi \in \mathcal{S}_n$, the composition $\pi \circ \iota_{ij}$ shifts the i-th item in π to position j.

A minimal decomposition for a generic permutation $\pi \in \mathcal{S}_n$ can be obtained by ordering the items in π using a sorting algorithm that only performs moves from the chosen set, i.e., adjacent swaps for *ASW*, generic exchanges for *EXC*, and insertions for *INS*. The sequence of generators corresponding to the moves performed during the sorting process is annotated, then the sequence is reversed and each generator is replaced with its inverse [13].

Therefore, randomized decomposers for *ASW*, *EXC*, and *INS* have been implemented by means of generalized and randomized variants of, respectively, the bubble-sort, selection-sort, and insertion-sort algorithms [13, 3]. They have been called *RandBS*, *RandSS*, *RandIS* and each one exploits a different algebraic property of permutations. Since (the ordered permutation) e is the only permutation with 0 inversions, *RandBS* iteratively decreases the number of inversions by swapping two suitable adjacent items. *RandSS* exploits the fact that e is the only permutation with n cycles in its cycle decomposition, thus it iteratively increases the number of cycles by exchanging two items belonging to the same cycle. *RandIS* considers that e is the only permutation with a (unique) longest increasing subsequence (LIS) of length n, thus it iteratively extends a LIS by shifting a suitable item in a suitable position.

For further implementation details, proofs of correctness and complexity we refer the interested reader to [13] and [3].

4 MOEA/DEP

In this section we introduce MOEA/DEP: an algebraic evolutionary algorithm for permutation-based multiobjective optimization problems. MOEA/DEP implements the MOEA/D framework, described in Section 2, by relying on the algebraic operators for the permutations space described in Section 3.

Given a MOP represented by k objective functions such as $f_i : \mathcal{S}_n \to \mathbb{R}$ for all $i \in \{1, \ldots, k\}$, MOEA/DEP decomposes the MOP into N single objective subproblems and simultaneously minimizes them by evolving a population of N permutations $\pi_1, \ldots, \pi_N \in \mathcal{S}_n$ that are also used to build up the approximated Pareto set in the archive *EP*.

The algorithmic scheme of MOEA/DEP is depicted in Algorithm 1. Various initializations are performed at lines 1–4, then the population is evolved in the generational loop of lines 5–23. Every iteration can be divided in two parts: offsprings generation (lines 6–12), and population update (lines 13–22). When a given termination criterion is met, MOEA/DEP returns the approximated Pareto set EP.

Algorithm 1 MOEA/DEP scheme

1: Initialize N evenly distributed weight vectors $\lambda_1, \ldots, \lambda_N$
2: For all $i \in \{1, \ldots, N\}$, compute the B_i neighborhood of size T
3: Initialize N permutations (solutions) $\pi_1, \ldots, \pi_N$
4: Initialize EP with the non-dominated solutions in $\{\pi_1, \ldots, \pi_N\}$
5: **while** termination condition is not met **do**
6: **for** each subproblem $i \in \{1, \ldots, N\}$ **do**
7: Set $C_i \leftarrow \{1, \ldots, N\} \setminus \{i\}$ with probability q, or B_i otherwise
8: Randomly select three distinct indices r_0, r_1, r_2 from C_i
9: Set the "base permutation" $\rho_i \leftarrow \pi_{r_0}$ and perturbate it with probability pm
10: Generate the mutant $\nu_i \leftarrow \rho_i \oplus F \odot (\pi_{r_1} \ominus \pi_{r_2})$ with the chosen $gset$
11: Generate the offspring $\pi'_i \leftarrow CrossOver(\pi_i, \nu_i, CR)$
12: **end for**
13: **for** each subproblem $i \in \{1, \ldots, N\}$ **do**
14: Evaluate $f_1(\pi'_i), \ldots, f_k(\pi'_i)$ and update EP
15: Update π_i:
16: **if** $g(\pi'_i|\lambda_i) < g(\pi_i|\lambda_i)$, **then** $\pi_i \leftarrow \pi'_i$
17: Update neighbors (no more than U):
18: set $u \leftarrow 0$
19: **while** $C_i \neq \emptyset$ and $u < U$ **do**
20: $j \leftarrow$ select and remove a random index from C_i
21: **if** $g(\pi'_i|\lambda_j) < g(\pi_j|\lambda_j)$, **then** $\pi_j \leftarrow \pi'_i$ and $u \leftarrow u + 1$
22: **end for**
23: **end while**
24: **return** EP

Detailed descriptions of the initialization, offsprings generation, and population update are provided in, respectively, Sections 4.1, 4.2, and 4.3, while a short description of the employed crossover operators is provided in Section 4.4.

4.1 Initialization

The initializations performed at lines 1–4 have been implemented following the original MOEA/D scheme [21] described in Section 2. Moreover, the initial individuals $\pi_1, \ldots, \pi_N$ are sampled uniformly random among all the permutations in $\mathcal{S}_n$, though, optionally, one of them can be initialized using a heuristic function for the problem at hand (e.g., the Liu-Reeves constructive heuristic for PFSP [10]).

4.2 Offsprings generation

At lines 6–12, starting from the current individuals $\pi_1, \ldots, \pi_N$, a population of offsprings $\pi'_1, \ldots, \pi'_N$ is generated by means of the differential mutation (DM) and crossover operators.

Two variants with respect to the original MOEA/D and DM schemes have been considered:

1. A (possibly) *extended neighborhood*: at lines 7–8, the three individuals undergoing DM are randomly selected from the whole population with probability q, or from the neighbors in B_i with probability $1 - q$.
2. A *pre-mutation* operation: at line 9, the so-called "base individual" of DM undergoes a preliminary small perturbation with probability pm. The perturbation is implemented as a single random insertion move, i.e., $\rho_i \leftarrow \rho_i \circ \iota$, where ι is randomly selected from INS.

Both the extended neighborhood and the pre-mutation allow to mitigate the loss of diversity in the population. Indeed, since $\pi \odot F(\pi \ominus \pi) = \pi$ for any $\pi \in \mathcal{S}_n$ and $F \in [0, 1]$, the DM operator cannot generate novel genotypes once the pool from which it selects its input individuals is converged to a super-genotype. Hence, the extended neighborhood mitigates this problem by also allowing the selection from the whole population. Moreover, if also the whole population is converged to a single-genotype, the pre-mutation allows anyway to make small moves.

Basing on the chosen generating set $gset \in \{ASW, EXC, INS\}$, at line 10, the mutant ν_i is generated by means of the algebraic DM operator described in Section 3.

Finally, at line 11, an offspring π'_i is obtained by recombining the mutant ν_i with the original individual π_i. The parameter CR may, or may not, be considered basing on the chosen crossover operator. See Section 4.4 for their definitions.

4.3 Population update

At lines 13–22, both the main population $\pi_1, \ldots, \pi_N$ and the external archive EP are updated by considering the newly generated offsprings $\pi'_1, \ldots, \pi'_N$.

EP is updated using a non-dominance criterion. A new solution π' enters EP if and only if there exists no $\rho \in EP$ such that $\rho \prec \pi'$. Once π' is added to EP, every $\rho \in EP$ such that $\pi' \prec \rho$ is removed from EP. Hence, at any moment of the evolution, EP contains the set of incomparable solutions that are not dominated by any of the solutions generated so far.

The main population individuals $\pi_1, \ldots, \pi_N$ are updated basing on their own single objective subproblem. Basically, the MOEA/D scheme described in Section 2 is applied with only few variants.

Since the k MOP objective functions may have different scales, as in [6] and [20], we have used a normalization approach for both the weighted sum and the Tchebycheff aggregation functions. Hence, the equations (5) and (6) are replaced with

$$g^{ws}_{z,w}(x|\lambda) = \sum_{i=1}^{k} \lambda_i \frac{f_i(x) - \alpha z_i}{w_i - z_i}, \tag{14}$$

$$g^{tc}_{z,w}(x|\lambda) = \max_{1\leq i\leq k}\left\{\lambda_i \frac{f_i(x) - \alpha z_i}{w_i - z_i}\right\}, \tag{15}$$

where each z_i and w_i are, respectively, the minimum (best) and the maximum (worst) values observed so far for the i-th objective function, while an $\alpha < 1$ guarantees that αz_i is a proper lower bound. As in [6] and [20], we set $\alpha = 0.6$.

Therefore, basing on the chosen aggregation function $g \in \{g^{ws}, g^{tc}\}$, the population is updated similarly to the general framework described in Section 2. Note that the (possibly) extended neighborhood introduced in Section 4.2 influences also the neighbors replacement. Moreover, for the sake of population diversity, a *limited neighborhood update* has been introduced, i.e., every offspring is allowed to replace no more than U neighbors (that are randomly scanned).

4.4 Crossover for permutations

In this work we have experimented two popular crossovers for permutation representations, namely, the two point crossover *TPII* adopted in [13] and the order based crossover *OBX* used in [3, 1].

Given the parents $\pi, \nu \in \mathcal{S}(n)$, both *TPII* and *OBX* select a random subset of positions $P \subseteq \{1, \ldots, n\}$ and build the offspring $\pi' \in \mathcal{S}(n)$ by setting $\pi'(i) \leftarrow \pi(i)$ for any $i \in P$, and inserting the remaining items starting from the leftmost free place of π' and following the order of appearance in ν.

The difference between *TPII* and *OBX* is that *TPII* uses an interval of contiguous positions, while, in *OBX*, P can be any subset.

Furthermore, two additional variants of *TPII* and *OBX* have been introduced in order to consider the classical DE crossover parameter $CR \in [0, 1]$. The modified variants, $TPII^{CR}$ and OBX^{CR}, constrain the size of P to $|P| = \lceil CR \cdot n \rceil$.

Therefore, in line 11 of Algorithm 1, *CrossOver* is chosen from $\{TPII, OBX, TPII^{CR}, OBX^{CR}\}$.

5 Experiments

MOEA/DEP has been experimentally investigated on the multiobjective PFSP (MoPFSP) considering the two popular objectives MS and TFT defined, respectively, in equations (1) and (2).

Experiments have been held on the widely known Taillard benchmark suite for PFSP [17] composed by 110 instances, i.e., 10 instances for each combination of $n = \{20, 50, 100, 200\}$ (number of jobs) and $m = \{5, 10, 20\}$ (number of machines), except the combination $n = 200, m = 5$.

The effectiveness of MOEA/DEP is compared with the recent state-of-the-art algorithm for MoPFSP, i.e., the estimation of distribution algorithm MEDA/D-MK introduced in [20]. In order to perform a fair comparison, the same performance metrics used in [20] have been considered, i.e., the hypervolume (HV) [4] and the coverage indicator (C-metric) [15].

The hypervolume is computed by means of a reference point $w \in \mathbb{R}^k$ in the objective space dominated by all the considered solutions. Hence, given a

set of solutions A, $HV(A)$ measures the volume of the part of objective space dominated by A and bounded by w. The exact formulation of HV is provided in [4]. Moreover, as done in [20], the Pareto fronts have been normalized using the same mechanism of equations (14) and (15), while w is set to $(1.01, 1.01)$.

The C-metric compares two solutions sets A and B by counting the relative number of solutions in B that are dominated by at least one solution in A, i.e., $C(A,B) = |\{b \in B | \exists a \in A : a \prec b\}| / |B|$. Note that C is not symmetric, thus $C(A,B)$ is usually compared with $C(B,A)$ in order to establish which is the best solutions set.

Also the same termination criterion of [20] has been adopted, i.e., every MOEA/DEP execution terminates after $n \times 1000$ generations have been performed.

In the following two subsections we describe the MOEA/DEP parameters' calibration and the results of the comparison with MEDA/D-MK.

5.1 Parameters calibration

MOEA/DEP has different components and parameters that need to be appropriately tuned.

One of the initial individual is generated by means of the popular Liu-Reeves procedure $LR(n/m)$ [10] used in variety of PFSP algorithms (see for example [13, 20]), while the other ones are randomly initialized. The differential evolution parameters F and CR are self-adapted during the evolution by means of the popular self-adaptive scheme jDE [5]. After some preliminary experiments we set $N = 100$, $T = 20$, $U = 2$, while the other parameters have been experimentally tuned by means of a full factorial analysis. The chosen ranges are:

- $g \in \{g^{ws}, g^{tc}\}$;
- $q \in \{0.25, 0.5, 0.75\}$;
- $pm \in \{0.7, 0.85, 1\}$;
- $gset \in \{ASW, EXC, INS\}$;
- $CrossOver \in \{TPII, OBX, TPII^{CR}, OBX^{CR}\}$.

Hence, a total of 216 MOEA/DEP settings have been executed on the first instance of every $n \times m$ problem configuration by performing 10 executions per setting. The total number of executions amounts to 23 760.

For every run, the hypervolume of the obtained Pareto set is computed. Then, on every instance $inst$, the hypervolumes of the same algorithm setting alg have been aggregated using the average relative percentage deviation measure

$$ARPD_{inst}^{alg} = \frac{1}{10} \sum_{i=1}^{10} \frac{\left(HV_{inst}^{best} - HV_{inst}^{alg,i}\right)}{HV_{inst}^{best}} \times 100, \tag{16}$$

where $HV_{inst}^{alg,i}$ is the hypervolume obtained by the setting alg on its i-th run on the instance $inst$, while HV_{inst}^{best} is the best hypervolume obtained by any setting in any run on the instance $inst$.

The setting with the best Quade average rank [8] computed on the ARPD values is chosen. Briefly, in any instance a ranking of the algorithms/settings is obtained basing on its ARPD value. The instances are weighted basing on the variability of its ARPDs. Finally, the ranks of an algorithm on the different instances are aggregated using a weighted average. For further details, see [8].

The winning MOEA/DEP configuration is

$$g = g^{ws}, q = 0.75, pm = 0.7, gset = INS, CrossOver = TPII. \quad (17)$$

5.2 Comparison with MEDA/D-MK

MEDA/D-MK has been recently proposed in [20] where it has been shown to be the state-of-the-art algorithm for the MoPFSP with the objectives MS and TFT.

In order to compare MOEA/DEP with MEDA/D-MK, the tuned MOEA/DEP setting (see Section 5.1) has been executed, 20 times per instance, on the 110 benchmark instances, while the Pareto fronts of MEDA/D-MK have been obtained from the website of their authors[1]. Moreover, as done in [20], given an instance, the 20 Pareto fronts obtained by MOEA/DEP have been aggregated into a single Pareto front by merging them and removing the dominated solutions.

The comparison between the Pareto fronts has been performed using both the hypervolume and the C-metric. The experimental results, averaged for any $n \times m$ problem configuration, are provided in Table 1, where A and B denote, respectively, MOEA/DEP and MEDA/D-MK.

Table 1. Comparison between A=MOEA/DEP and B=MEDA/D-MK

Problem	HV_A	HV_B	$C(A,B)$	$C(B,A)$
20×5	**0.884**	0.612	**0.988**	0
20×10	**0.901**	0.640	**0.997**	0
20×20	**0.889**	0.648	**0.993**	0
50×5	**0.981**	0.730	**0.995**	0
50×10	**0.904**	0.623	**0.937**	0
50×20	**0.906**	0.635	**0.938**	0.003
100×5	**0.984**	0.597	**0.989**	0
100×10	**0.944**	0.521	**0.949**	0
100×20	**0.885**	0.622	**0.884**	0
200×10	**0.959**	0.497	**0.997**	0
200×20	**0.862**	0.547	**0.882**	0

Table 1 clearly shows that MOEA/DEP systematically outperforms MEDA/D-MK. The difference in terms of hypervolumes is particularly large on the instances with $n = 200$, though it is never below 0.2. Moreover, remarkable results

[1] https://github.com/murilozangari

are obtained also considering C-metric. In general, around 90% of the solutions in the Pareto sets of MEDA/D-MK are dominated by our results.

Finally, note that also the Pareto fronts obtained in a single run of MOEA/DEP are generally better than the reference ones, though with a slightly smaller difference with respect to the data provided in Table 1.

6 Conclusion and Future Work

In this paper, basing on the recently proposed algebraic framework for combinatorial optimization, we have introduced MOEA/DEP: a decomposition-based algebraic evolutionary algorithm for multiobjective permutation-based problems.

The selection and replacement mechanisms of MOEA/DEP are based on the MOEA/D framework for multiobjective optimization. Moreover, in order to mitigate the diversity loss during the evolution, MOEA/DEP introduces some additional components and variants such as: a pre-mutation procedure, a (possibly) extended neighborhood for individuals selection, and a limited update of the neighboring individuals.

Experiments have been held on the multiobjective permutation flowshop scheduling problem (MoPFSP) considering the two popular criteria: makespan and total flowtime. A commonly used benchmark suite has been considered. The MOEA/DEP parameters have been properly calibrated, and the experimental results have been compared with respect to recent state-of-the-art results for MoPFSP [20]. The experimental analysis has been conducted using two popular performance metrics for multiobjective optimization. The experimental results clearly show that MOEA/DEP systematically outperforms the previous state-of-the-art algorithm.

Starting from the significant results obtained in this work, future lines of research will include: (i) the application of MOEA/DEP to other MoPFSP objectives and to other permutation-based multiobjective problems, (ii) the use of the three generating sets altogether by means of a self-adaptive scheme.

References

1. Baioletti, M., Milani, A., Santucci, V.: Linear ordering optimization with a combinatorial differential evolution. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics. pp. 2135–2140 (2015)
2. Baioletti, M., Milani, A., Santucci, V.: Algebraic particle swarm optimization for the permutations search space. In: Proc. of IEEE Congress on Evolutionary Computation CEC 2017. pp. 1587–1594 (June 2017)
3. Baioletti, M., Milani, A., Santucci, V.: An extension of algebraic differential evolution for the linear ordering problem with cumulative costs. In: Proc. of - 14th International Conference on Parallel Problem Solving from Nature - PPSN XIV. pp. 123–133 (2016), http://dx.doi.org/10.1007/978-3-319-45823-6_12
4. Beume, N., Fonseca, C.M., López-Ibáñez, M., Paquete, L., Vahrenhold, J.: On the complexity of computing the hypervolume indicator. IEEE Transactions on Evolutionary Computation 13(5), 1075–1082 (Oct 2009)

5. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10(6), 646–657 (Dec 2006)
6. Chang, P.C., Chen, S.H., Zhang, Q., Lin, J.L.: MOEA/D for flowshop scheduling problems. In: Proc. of IEEE Congress on Evolutionary Computation CEC 2008. pp. 1433–1438 (June 2008)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (Apr 2002)
8. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1(1), 3 – 18 (2011)
9. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Computers & Operations Research 38(8), 1219 – 1236 (2011)
10. Liu, J., Reeves, C.R.: Constructive and composite heuristic solutions to the $P//\Sigma C_i$ scheduling problem. European Journal of Operational Research 132(2), 439 – 452 (2001)
11. Minella, G., Ruiz, R., Ciavotta, M.: A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS Journal on Computing 20(3), 451–471 (2008)
12. Minella, G., Ruiz, R., Ciavotta, M.: Restarted iterated pareto greedy algorithm for multi-objective flowshop scheduling problems. Computers & Operations Research 38(11), 1521 – 1533 (2011)
13. Santucci, V., Baioletti, M., Milani, A.: Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. IEEE Transactions on Evolutionary Computation 20(5), 682–694 (2016), http://dx.doi.org/10.1109/TEVC.2015.2507785
14. Santucci, V., Baioletti, M., Milani, A.: Solving permutation flowshop scheduling problems with a discrete differential evolution algorithm. AI Communications 29(2), 269–286 (2016)
15. Schutze, O., Esquivel, X., Lara, A., Coello, C.A.C.: Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation 16(4), 504–522 (Aug 2012)
16. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997)
17. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64(2), 278 – 285 (1993)
18. Trivedi, A., Srinivasan, D., Sanyal, K., Ghosh, A.: A survey of multiobjective evolutionary algorithms based on decomposition. IEEE Transactions on Evolutionary Computation 21(3), 440–462 (June 2017)
19. Varadharajan, T., Rajendran, C.: 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)
20. Zangari, M., Mendiburu, A., Santana, R., Pozo, A.: Multiobjective decomposition-based mallows models estimation of distribution algorithm. a case of study for permutation flowshop scheduling problem. Information Sciences 397-398(Supplement C), 137 – 154 (2017)

21. Zhang, Q., Li, H.: Moea/d: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11(6), 712–731 (Dec 2007)
22. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (Nov 1999)
23. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)