On Maintaining Diversity in MOEA/D: Application to a Biobjective Combinatorial FJSP

Juan José Palacios Alonso
University of Oviedo
Department of Computer Science, 33204, Gijón (Spain)
palaciosjuan@uniovi.es

Bilel Derbel
Université Lille 1, CRIStAL (CNRS UMR 9189),
Inria Lille Nord Europe
59655 Villeneuve d'Ascq Cedex (France)
bilel.derbel@lifl.fr

ABSTRACT

MOEA/D is a generic decomposition-based multiobjective optimization framework which has been proved to be extremely effective in solving a broad range of optimization problems especially for continuous domains. In this paper, we consider applying MOEA/D to solve a bi-objective scheduling combinatorial problem in which task durations and due-dates are uncertain. Surprisingly, we find that the conventional MOEA/D implementation provides poor performance in our application setting. We show that this is because the replacement strategy underlying MOEA/D is suffering some shortcomes that lead to low population diversity, and thus to premature convergence. Consequently, we investigate existing variants of MOEA/D and we propose a novel and simple alternative replacement component at the aim of maintaining population diversity. Through extensive experiments, we then provide a comprehensive analysis on the relative performance and the behavior of the considered algorithms. Besides being able to outperform existing MOEA/D variants, as well as the standard NSGA-II algorithm, our investigations provide new insights into the search ability of MOEA/D and highlight new research opportunities for improving its design components.

Keywords

EMO, decomposition-based EA, fuzzy scheduling

1. INTRODUCTION

Multiobjective optimization problems (MOPs) are a natural and very frequent outcome in several application fields like logistics, planning, green-IT, clouds, etc. They are used to model a problem where one wants to optimize several, *a priori*, conflicting objectives. In practice, it is unlikely to find a solution which is able to optimize all target objectives simultaneously at the same time. Hence, solving a MOP turns out to finding a whole set of solutions; each one exhibiting a different compromise between objectives — which would allow a decision maker to choose among afterward.

More formally, a multi-objective optimization problem can be defined by a set of $M \geq 2$ objective functions $f = (f_1, f_2, \ldots, f_M)$, and a set X of feasible solutions in the *decision space*, where X is a discrete set in the combinatorial case. Let $Z = f(X) \subseteq \mathbb{R}^M$ be the set of feasible outcome vectors in the *objective space*. To each solution $x \in X$ is then assigned exactly one objective vector $z \in Z$, on the basis of the vector function $f : X \rightarrow Z$ with $z = f(x)$. Without loss of generality, in a maximization context, an objective vector $z \in Z$ is dominated by an objective $z' \in Z$, denoted by $z \prec z'$, iff $\forall m \in \{1, 2, \ldots, M\}$, $z_m \leq z'_m$ and $\exists m \in \{1, 2, \ldots, M\}$ such that $z_m < z'_m$. By extension, a solution $x \in X$ is dominated by a solution $x' \in X$, denoted by $x \prec x'$, iff $f(x) \prec f(x')$. A solution $x^\star \in X$ is termed *Pareto optimal*, if there does not exist any other solution $x \in X$ such that $x^\star \prec x$. The set of all Pareto optimal solutions is called the *Pareto set*. Its mapping in the objective space is called the *Pareto front*. One of the most challenging task in multi-objective optimization is to identify a minimal complete Pareto set, i.e., one Pareto optimal solution for each point from the Pareto front. This task is usually intractable for several optimization problems and one seeks instead a good *Pareto front approximation*.

In this context, we are interested in evolutionary multiobjective optimization (EMO) methods which have been shown to be well-applicable for a broad range of MOPs while being particularly accurate in finding high quality approximations [5, 4]. The EMO methods encompass several algorithms relying on different concepts and having seemingly different properties. Besides the well-established class of Pareto-dominance based algorithms, e.g. NSGA-II [6], we can report a recent and growing interest in the so-called *aggregation-based* algorithms, and especially the MOEA/D (multiobjective evolutionary algorithm based on decomposition) framework [19]. MOEA/D decomposes a given MOP into a number of single-objective optimization subproblems; Each single-objective optimization problem is defined by a scalarizing function using a different weight vector.

The original idea of MOEA/D is to define a neighborhood relation between subproblems and to solve each problem using standard evolutionary operators (crossover, mutation) as in the single-objective setting, but cooperatively based on the solutions computed at the neighboring subproblems. Each time a subproblem is considered, some parents are selected among neighbors in order to generate a new offspring which could replace the solutions of neighboring subproblems if an improvement is observed. As such, we claim that the selection and replacement components in MOEA/D are critical when evolving the population since it implies an

GECCO '15, July 11 - 15, 2015, Madrid, Spain

 ISBN 978-1-4503-3472-3/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754774

intensification-diversification trade-off — which is known to be a key ingredient in the success of any evolutionary algorithm. When applying MOEA/D for continuous MOP, this claim is rather well known and it was previously shown how to deal with this issue by designing MOEA/D variants that can improve on the conventional one [13]. For combinatorial problems, this issue is relatively less studied and one can only find few systematic investigations on the subject [16]. In this paper, we are specifically interested in the applicability of MOEA/D for a particular combinatorial optimization problem, namely, a bi-objective job shop scheduling problem with uncertainty. In this context, our contribution can be summarized in the three following points:

- We found that the previous claim is verified for the bi-objective combinatorial problem that we are tackling. In particular, we analyze the behavior of conventional MOEA/D and elicit its relative performance when compared to NSGA-II, considered as a baseline standard Pareto-dominance based algorithm. We illustrate how MOEA/D can in fact suffer from a loss in population diversity and premature convergence.
- We investigate the efficiency of existing mechanisms to deal with this issue, and we elicit their relative impact on performance. We illustrate how they can improve to some extent the performance of MOEA/D by avoiding solution duplication and hence increasing diversity.
- We propose a novel replacement strategy as an alternative to existing mechanisms. We show that besides being extremely simple, the so-obtained MOEA/D variant is successful in significantly improving the runtime performance. More importantly, the experimental study and the results obtained in this paper have the primary goal of providing a comprehensive analysis of MOEA/D in an attempt to unify our understanding of its algorithmic components; and thus to better grasp the future design challenges and research questions.

We also notice that although our results are with respect to a particular combinatorial optimization problem, which is challenging *per se*, the newly proposed mechanism is generic and fully compatible with MOEA/D viewed as a problem-independent EMO framework. Accordingly, we structure the paper following a step-by-step presentation allowing us to state our findings in an incremental manner. In the remainder, we first start by describing and formulating, in Section 2, the bi-objective fuzzy job shop scheduling problem we are interested in. In Section 3, we discuss the applicability of conventional MOEA/D, and we provide a brief analysis of its behavior compared to NSGA-II. In Section 4, we recall some existing variants of MOEA/D and we introduce the newly proposed strategy in order to deal with diversity issues. In Section 5, we provide our experimental results and analyze our findings. In Section 6, we conclude the paper and discuss some open research directions.

2. BACKGROUND ON THE FJSP

The *job shop scheduling problem* (*JSP*) consists in scheduling a set of jobs $J = \{J_1, \ldots, J_n\}$ on a set of physical resources or machines $M = \{M_1, \ldots, M_m\}$, subject to a set of constraints. There are *precedence constraints*, so each job J_i, $i = 1, \ldots, n$, consists of m tasks $\{\theta_{i1}, \ldots, \theta_{im}\}$ to be sequentially scheduled. Also, there are *capacity constraints*, whereby each task θ_{ij} requires the uninterrupted and exclusive use of one of the machines $\eta_{ij} \in M$ for its whole processing time p_{ij}. We can also consider due-date constraints, where each job has a maximum completion time and all its tasks must be scheduled to finish before this time. A feasible schedule is an allocation of starting times S_{ij} for each task θ_{ij} such that all constraints hold. The objective is to find a schedule which is *optimal* according to some criterion, e.g. the *makespan* or the delay w.r.t. the due-dates is minimal.

2.1 Uncertain durations

In real-world situations, it is often to find that some factors are unknown in advance and only some uncertain information is available. In the case of task processing times, an expert may be able to roughly estimate them based on his own experience. This knowledge is modeled using fuzzy intervals or fuzzy numbers, which have been extensively studied in the literature (cf. [8]). The simplest model of a fuzzy interval is a *triangular fuzzy number* or TFN, given by an interval $[a^1, a^3]$ of possible values and a most plausible value a^2 in it. That is, for a TFN A, denoted $A = (a^1, a^2, a^3)$, the membership function is determined as follows:

$$\mu_A(x) = \begin{cases} \frac{x-a^1}{a^2-a^1} & : a^1 < x \leq a^2 \\ \frac{x-a^3}{a^2-a^3} & : a^2 < x < a^3 \\ 0 & : x \leq a^1 \text{ or } a^3 \leq x \end{cases} \quad (1)$$

This model has been extensively used in the fuzzy scheduling literature [1]. However, when TFNs are to be used to extend the job shop to handle uncertainty, two issues must be addressed: the establishment of an order relation between TFNs and the means of extending the arithmetic operations of addition and maximum to work with TFNs (so to compute starting and completion times).

The fact that there is no natural total ordering in the set of TFNs makes concepts like "minimal makespan" ambiguous. The membership function of a fuzzy number can be interpreted as a possibility distribution on the real numbers; this allows to define its expected value, given for a TFN A by $E[A] = \frac{1}{4}(a^1 + 2a^2 + a^3)$. This induces a total ordering $\leq_E$ in the set of fuzzy intervals [9], where for any two fuzzy intervals A and B, $A \leq_E B$ iff $E[A] \leq E[B]$. The expected value coincides with the *neutral scalar substitute* of a fuzzy interval and can also be obtained as the centre of gravity of its *mean value* or using the *area compensation* method [7].

Regarding arithmetic operations, for the fuzzy job shop (*FJSP*), we need two operations on fuzzy numbers, the sum and the maximum. These are obtained by extending the corresponding operations on real numbers using the *Extension Principle*. In the case of the sum, for any pair of TFN A and B, the resulting expression is $A + B = (a^1 + b^1, a^2 + b^2, a^3 + b^3)$. However, computing the resulting expression of the maximum is cumbersome, if not intractable. For the sake of simplicity and tractability of numerical calculations, it is fairly common in the literature following [9], to approximate the result of this operation. So for any two TFNs A, B, if F denotes their maximum and $G = (\max\{a^1, b^1\}, \max\{a^2, b^2\}, \max\{a^3, b^3\})$ its approximated value, it holds that F and G have identical support and modal value. The most immediate consequence of this result in terms of scheduling is that, given a schedule, the makespan of all its possible realizations is contained within the support of the fuzzy makespan.

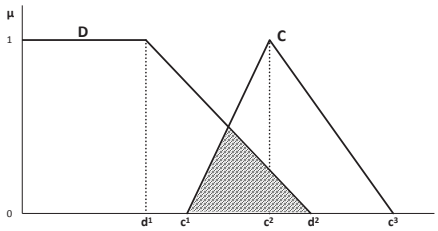


Figure 1: Intersection between a TFN and the due-date.

2.2 Flexible due dates

In the case in which due dates exist in real situations, they are often flexible: a customer is satisfied if his job finishes before a preferred due-date d^1 and after this time his satisfaction decreases until a limit due-date d^2, after which he is completely unsatisfied. A natural approach to modeling such satisfaction degrees is the use of a fuzzy set $D = (d^1, d^2)$ with linear decreasing membership function as follows:

$$\mu_D(x) = \begin{cases} 1 & : x \leq d^1 \\ \frac{x-d^2}{d^1-d^2} & : d^1 < x < d^2 \\ 0 & : d^2 \leq x \end{cases} \quad (2)$$

According to [7], this expresses a flexible threshold "less than" and the satisfaction degree of the customer for the job finishing at time x. However, in our case the completion time of the jobs is not a real number x, but a TFN. When the completion time of a job J_i is a TFN C_i, the degree to which D_i is satisfied is usually measured by the *agreement index* (AI) [12, 14] computed as follows (see Fig. 1):

$$AI(C, D) = \frac{Area(C \cap D)}{Area(C)} \quad (3)$$

Here the flexible due date D is completely satisfied when $AI = 1$ and unsatisfied when $AI = 0$.

2.3 The bi-objective optimization problem

Having established how to model uncertain duration times and flexible due-dates, it is possible to find a schedule for a given FJSP instance and compute a fuzzy makespan, $C_{max} = \max C_i$, a TFN indicating the time when all jobs are finished. It is also possible to measure for every job $J_i, i = 1, \ldots, n$, to what degree its completion time C_i satisfies the flexible due-date D_i, as given by the agreement index $AI_i = AI(C_i, D_i)$. Our target is to minimize C_{max} and maximize the different AI_i, which shall be aggregated as to have an overall measure. In this paper we shall consider the optimization of the average AI (AI_{av}) together with the makespan. The problem can be formulated as follows:

$$\text{minimize } C_{max} = \max_{i=1\ldots n} C_i \quad (4)$$

$$\text{maximize } AI_{av} = 1/n \sum_{i=1}^{n} AI_i \quad (5)$$

subject to:

$$A_i = AI(C_i, D_i),\ 1 \leq i \leq n \quad (6)$$

$$C_i = \max_{j=1\ldots m} C_{ij},\ 1 \leq i \leq n \quad (7)$$

$$C_{ij} = S_{ij} + p_{ij},\ 1 \leq i \leq n, 1 \leq j \leq m \quad (8)$$

$$S_{ij} \geq C_{ij-1},\ 1 \leq i \leq n, 2 \leq j \leq m \quad (9)$$

$$S_{ij} \geq C_{lm} \vee C_{ij} \leq S_{lm}, \forall l, m,\ l \neq i \vee m \neq j : \eta_{ij} = \eta_{lm} \quad (10)$$

Algorithm 1: The conventional MOEA/D [19]

Input: $\{\lambda^1, \ldots, \lambda^N\}$: weight vectors w.r.t sub-problems;
$\mathcal{B}(i)$: the neighbors of sub-problem $i \in \{1, \ldots, N\}$;
$P = \{x^1, \ldots, x^N\}$: the initial population

```
while Not Stopping Condition do
    for i ∈ {1, ..., N} do
        // Mating selection
        (k, ℓ) ← rand(B(i));
        // Variation
        y ← Crossover_Mutation_Repair(x^k, x^ℓ);
        z* ← Update_Reference_Point(z*, y);
        // Environmental replacement
        shuffle(B_i);
        for j ∈ B_i do
            if g^te(y, λ^j) < g^te(x^j, λ^j) then x^j ← y;
```

In the rest of this paper, we use the fuzzy job shop instances proposed in [11]. These were obtained by fuzzyfying 12 well-known benchmark problems for deterministic job shop which are considered hard to solve: FT10 (size 10×10), FT20 (20×5), La21, La24, La25 (15×10), La27, La29 (20×10), La38, La40 (15×15), and ABZ7, ABZ8, ABZ9 (20×15). The task durations are generated from each original problem following [9], so they become symmetric TFNs where the modal value is the original duration. The due date d_i for each job J_i is defined following [2]: first, we define a generic due date $d_i = TF \sum_{j=1}^{m} p_{ij}^2$, where TF is a tightness factor; then, we use two different tightness factors to have the points that define the satisfaction function: d_i^1, with $TF = 1.3$, and d_i^2, with $TF = 1.7$.

3. MOEA/D APPLIED TO THE FJSP

In this section, we briefly review the conventional MOEA/D algorithm as originally described in [19] and we discuss its applicability for the bi-objective FJSP. Generally speaking, MOEA/D falls in the class of decomposition based EMO algorithms [10] which aim is to search for good solutions in multiple regions of the Pareto front. For this purpose, the original MOP is *decomposed* into a number of *scalarized* single-objective *sub-problems*. Different scalarizing functions have been proposed so-far [17]. In this paper, we use the weighted Tchebycheff (g^{te}) function which is to be minimized:

$$g^{te}(x, \lambda) = \max_{i \in \{1,2\}} \lambda_i \cdot \left| z_i^\star - f_i(x) \right|$$

where $\lambda = (\lambda_1, \lambda_2)$ is a positive weighting coefficient vector and $z^\star = (z_1^\star, z_2^\star)$ is a reference point.

Let $(\lambda^1, \ldots, \lambda^N)$ be a set of N uniformly distributed weighting coefficient vectors defining N sub-problems to be optimized. For each sub-problem $i \in \{1, \ldots, N\}$, the goal is to approximate the solution x with the best scalarizing function value $g^{te}(x, \lambda^i)$. For that purpose, MOEA/D maintains a population $P = (x^1, \ldots, x^N)$, each individual corresponding to a good-quality solution for one sub-problem. For each sub-problem $i \in \{1, \ldots, N\}$, a set of neighbors $\mathcal{B}(i)$ is defined by considering the T closest weight vectors (including i). To evolve the population, sub-problems are optimized iteratively as sketched in the high level template of Algorithm 1 where we omit some details for the sake of conciseness. At a given iteration corresponding to one sub-problem i, two solutions are selected at random from $\mathcal{B}(i)$, and an offspring solution y is created by means of variation oper-

Table 1: Comparing MOEA/D *vs.* NSGA-II. Average indicator value and the standard deviation (in parentheses, with a multiplicative factor of 10^{-1}) are reported.

instance	Hypervolume Indicator		Epsilon Indicator	
	MOEA/D	NSGA-II	MOEA/D	NSGA-II
ABZ7	$0{,}382_{(0,38)}$	$0{,}423_{(0,32)}$	$0{,}179_{(0,39)}$	$0{,}141_{(0,32)}$
ABZ8	$0{,}373_{(0,31)}$	$0{,}428_{(0,35)}$	$0{,}218_{(0,33)}$	$0{,}164_{(0,36)}$
ABZ9	$0{,}349_{(0,44)}$	$0{,}415_{(0,33)}$	$0{,}244_{(0,47)}$	$0{,}180_{(0,35)}$
FT10	$0{,}549_{(0,48)}$	$0{,}591_{(0,39)}$	$0{,}199_{(0,50)}$	$0{,}163_{(0,40)}$
FT20	$0{,}203_{(0,24)}$	$0{,}241_{(0,18)}$	$0{,}110_{(0,23)}$	$0{,}076_{(0,18)}$
La21	$0{,}427_{(0,33)}$	$0{,}449_{(0,37)}$	$0{,}158_{(0,34)}$	$0{,}142_{(0,40)}$
La24	$0{,}429_{(0,36)}$	$0{,}456_{(0,39)}$	$0{,}181_{(0,38)}$	$0{,}162_{(0,40)}$
La25	$0{,}440_{(0,30)}$	$0{,}445_{(0,32)}$	$0{,}163_{(0,32)}$	$0{,}165_{(0,33)}$
La27	$0{,}228_{(0,24)}$	$0{,}254_{(0,29)}$	$0{,}184_{(0,26)}$	$0{,}166_{(0,33)}$
La29	$0{,}235_{(0,31)}$	$0{,}254_{(0,27)}$	$0{,}201_{(0,36)}$	$0{,}190_{(0,30)}$
La38	$0{,}608_{(0,51)}$	$0{,}683_{(0,34)}$	$0{,}199_{(0,48)}$	$0{,}132_{(0,31)}$
La40	$0{,}618_{(0,45)}$	$0{,}691_{(0,36)}$	$0{,}204_{(0,43)}$	$0{,}145_{(0,35)}$

ators, e.g., mutation and crossover. Then, for every subproblem $j \in \mathcal{B}(i)$, if y improves over j's current solution x^j then y replaces it. The algorithm continues looping over sub-problems until a stopping condition is satisfied.

From the previous description, we can see that MOEA/D is an extremely simple algorithm where only the variation step (line 4) needs to be expanded when applied to a particular problem. Specifically for the FJSP, there exist several well-established operators from the literature which might be applied. In this paper, solutions are encoded using permutations with repeated elements. This is a permutation of the set of tasks, each being represented by its job number. A given permutation is decoded into its associated schedule using an insertion strategy in the schedule-generating scheme [18]. We thereby use one of the most popular crossover operators for the JSP, the Generalized Order Crossover (GOX), and inversion as mutation strategy.

To assess the performance of the so-obtained algorithm, we consider a Pareto-dominance EMO method as a plausible alternative, namely, we use the well-known NSGA-II algorithm [6]. For completeness, we recall that in contrast to MOEA/D, the NSGA-II algorithm is based on a nondominated-sorting and crowding-distance estimation procedure for comparing qualities of different solutions, which serve for mating selection and environmental replacement. For fairness, we use the same mutation and crossover operators for both algorithms. We thereby conduct a first set of experiments using the previously described instances. Both MOEA/D and NSGA-II are run 30 independent times using the following setup: the population size is set to 100, the probability to apply crossover (resp. mutation) is set to 1 (resp. 0.1) and the stopping criterion is fixed to $5 \cdot 10^4$ function evaluations. Since the neighborhood size T is believed to have strong impact on performance, we tested a broad range of T-values in order to finely tune MOEA/D. Due to lack of space, we only show in this section the results obtained with the best value, namely, $T = 5$. Later, we will discuss further the impact of this parameter — for now we just focus on MOEA/D tuned with the best paramater setting.

We use the hypervolume[1] (to be maximized) and the epsilon indicators (to be minimized) [20] in order to appreciate the relative performance of the two designed approaches. Our findings are summarized in Table 1, where gray cells indicate that the algorithm in the corresponding column

[1]the reference point is set to the worst objective values ever observed in any run.

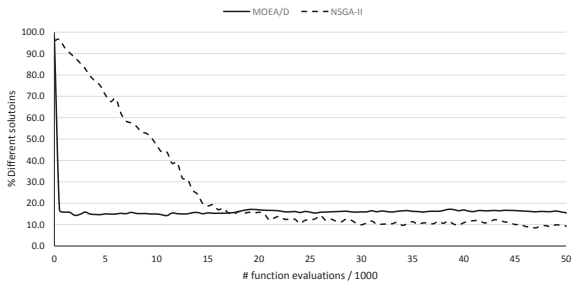


Figure 2: Evolution of the percentage of different objective vectors in the population.

is significantly better than the other one using a statistical t-test with confidence 0.05. We can see that NSGA-II is outperforming MOEA/D in all but one instance for both indicators. To understand this huge difference between the two approaches, we show in Fig. 2 the evolution of the average number of different solutions (in the objective space) in the population maintained by both approaches as a function of the number of function evaluations for instance La29 (similar observations were verified for the other instances). We observe that NSGA-II is able to maintain more different solutions than MOEA/D in the first stages of the run; or equivalently, that the number of copies in MOEA/D population is increasing very abruptly. We clearly attribute this to the fact that as soon as a good solution is found in MOEA/D, it will immediately replace all solutions from the neighbor subproblems. Although this strategy could speed-up convergence, it obviously prevents the genetic operators to produce improving offsprings in diversified regions in the objective space. Hence, we can argue that this replacement strategy is seemingly very aggressive, thus killing population diversity and leading to a premature convergence of MOEA/D.

Actually, the previous discussion holds for the original replacement component of MOEA/D which was first described in [19]. It is in fact important to grasp the algorithm behavior without necessarily being in a horse-racing condition, which might lead to diminish our understanding of the key ingredients towards a successful application. Having identified this weakness, we describe in the next section some existing alternatives together with our own proposal.

4. MAINTAINING DIVERSITY IN MOEA/D

Based on our observations, we are willing to provide improved variants of MOEA/D for the FJSP by addressing the raised diversity issue. In the following, we first review two variants (MOEA/D-n_r and MOEA/D-**ss**) which we believe to be the best existing alternatives from the literature. We then describe our own proposal MOEA/D-RO, a relatively simple variant that we will show to be extremely effective.

MOEA/D-n_r [13] **: Parametrizing the number of replacements.** Two modifications are proposed in [13] in order to better control the replacement and the selection components of conventional MOEA/D. The first one concerns the number of replacements that are allowed when an improving offspring is found. More specifically, once a new offspring has been generated for a given subproblem i in line 4, this offspring can only replace a maximum number of n_r other solutions in the T-neighborhood $\mathcal{B}(i)$ of subproblem i, where $n_r \leq T$ is a user-defined (typically small) parameter — the 'for loop' in line 7 is then modified accordingly. The

n_r parameter can be viewed as a mean to control how many solutions can be discarded from the population when a high quality offspring improving simultaneously several subproblems is found. The second modification described in [13] concerns the choice of parents in the mating selection step in line 3. Instead of systematically choosing parents from only the T-neighborhood at every iteration, an additional parameter $\delta \in [0, 1]$ is introduced in order to allow parent selection from the whole population with probability $1 - \delta$, i.e., at every iteration, the neighbors of a subproblem are defined by solely its T-neighborhood with probability δ, or using the whole population with a probability $1-\delta$ (typically δ is set to a high value). The δ parameter can be viewed as a mean to increase diversity among parents which might potentially results in increasing the diversity and the accuracy of generated offsprings. These two modifications are mostly studied for continuous MOPs and we are only aware of few related investigations in the combinatorial setting, e.g., [3].

MOEA/D-**xy** [16] **: A generational approach.** Specifically for the combinatorial case, a recent study was conducted in [16] where the authors advocated for the design of new replacement and selection strategies within the MOEA/D framework. The main idea developed therein is that since subproblems are processed in an iterative manner in conventional MOEA/D, as well as in its variants [13], once a solution gets replaced in a given iteration, it is definitively discarded and gets no chance to produce an offspring eventhough it could be potentially beneficial. Accordingly, the authors proposed to adopt a generational approach in which all individuals in the population are evolved simultaneously before the replacement can take place. They consequently described four variants denoted MOEA/D-**xy** with $\mathbf{x}, \mathbf{y} \in \{\mathbf{s}, \mathbf{c}\}$. The components of the so-denoted couple $(\mathbf{x}, \mathbf{y})$ correspond respectively to the selection and the replacement strategies; and the notation **s** (resp. **c**) refers to a so-called *selfish* (resp. *collective*) variant. A systematic study of these variants showed the superiority of MOEA/D-**ss** (when the crossover operator is not disabled) over conventional MOEA/D and its variants [19, 13]. Consequently, we only retain this variant as a competing algorithm for the problem we are tackling[2]. This variant introduces the following modifications. First, for each subproblem i, generate a new offspring y^i by fixing the first parent to be the current solution x^i and by selecting the second one from the remaining subproblems in the T-neighborhood of i. This is performed for *all* subproblems *before* any replacement is performed. Then, an offspring y^i can only replace the *current* solution x^i of the subproblem from where it was originated. Once the replacement is performed the next generational round is performed and so until termination. This variant is termed selfish because each subproblems privileges its own solution in the reproduction phase, and it also does not include its neighbors in the replacement phase.

MOEA/D-RO **: Recursive opportunistic replacement**. This strategy is our own contribution and it can be viewed as a very intuitive answer to the question of how to maintain population diversity. More specifically, we explicitly check diversity of conventional MOEA/D locally at the neighborhood level, and we manage to design a non-oblivious replacement. First, we simply do not allow an offspring to replace a solution if there already exists a solution having the same objective values in the corresponding neighborhood. Moreover, every time this condition is satisfied and the replacement is activated with respect to an offspring, say y, and a neighboring subproblem solution, say x^j, we do the following. The offspring y becomes the new current solution for subproblem j, but the previous solution x^j is not discarded if it can improve the solution of other neighbors. Hence, we recursively check whether there is an opportunity that solution x^j replaces a solution j' in the neighborhood $\mathcal{B}(j)$ of subproblem j. If such a solution is found, a new replacement is activated and so on until no improvement is observed. Notice that with this strategy we do maintain diversity but we also attempt to improve convergence since we heuristically check whether a solution can serve for some subproblems before discarding it from the population.

In the remainder of this paper, we shall provide a comprehensive study allowing us to understand the impact of each of the previously described variants.

[2]Our tuning experiments have also confirmed its adequacy, we do not include the results due to lack of space.

Table 2: Performance of MOEA/D and MOEA/D-n_r depending on the neighborhood size T for instance La29.

	T	MOEA/D		MOEA/D-n_r = 1		MOEA/D-n_r = 2	
		δ=1.0	δ=0.9	δ=1.0	δ=0.9	δ=1.0	δ=0.9
Hypervol.	5	0.235	0.225	0.252	**0.258**	0.249	0.244
	25	0.235	0.239	**0.274**	0.264	0.252	0.255
	50	0.233	0.233	0.269	0.277	0.267	0.262
	75	0.238	0.240	0.264	**0.278**	0.257	0.268
	100	0.225	0.213	0.278	**0.288**	0.253	0.255
Epsilon	5	0.201	0.212	0.183	**0.176**	0.184	0.191
	25	0.204	0.199	**0.160**	0.172	0.184	0.181
	50	0.207	0.207	0.168	**0.157**	0.167	0.174
	75	0.200	0.198	0.172	**0.155**	0.178	0.167
	100	0.215	0.228	0.158	**0.145**	0.184	0.183

5. EXPERIMENTAL RESULTS

As previously, we use the same crossover and mutation operators and the same maximum budget of $5 \cdot 10^4$ function evaluations. In all reported tables, we use the same format by putting the best average indicator value in bold and having gray cells indicating that no significant differences can be reported w.r.t. the best with confidence 0.05. Unless stated explicitly the population size is set to 100.

5.1 Tuning MOEA/D

We start examining the relative performance of MOEA/D with respect to the MOEA/D-n_r variant to which it is more related and which was the first introduced to deal with diversity issues. In Table 2 we illustrate the influence of different parameters: T, nr and δ for instance La29, having similar results with the other instances (not reported here because of space limitations). In general, we appreciate that MOEA/D is not very sensible to the neighborhood size T. In fact, we can say that small neighborhood sizes help to delay the premature convergence just a little bit, but they don't overcome the issue. On the other hand, MOEA/D-n_r performs better with a large neighborhood size (for instance, 100 when $nr = 1$). This may sound shocking, but it is not so if we take into account that the nr parameter helps to control the diversity through a less aggressive replacement and the large neighborhood size allows to mate very different solutions. Finally, regarding the δ parameter, it turned out that running the algorithms without using it or using it with a quite standard value (0.9), have no statistically significant

Table 3: Comparing MOEA/D ($T = 5$) *vs.* MOEA/D-n_r ($T = 100$) for different nr values ($\delta = 1.0$).

instance	Hypervolume Indicator			Epsilon Indicator		
	MOEA/D	MOEA/D-n_r = 1	MOEA/D-n_r = 2	MOEA/D	MOEA/D-n_r = 1	MOEA/D-n_r = 2
ABZ7	$0{,}382_{(0,38)}$	$\mathbf{0{,}435}_{(0,35)}$	$0{,}406_{(0,33)}$	$0{,}179_{(0,39)}$	$\mathbf{0{,}128}_{(0,37)}$	$0{,}156_{(0,34)}$
ABZ8	$0{,}373_{(0,31)}$	$\mathbf{0{,}446}_{(0,33)}$	$0{,}425_{(0,33)}$	$0{,}218_{(0,33)}$	$\mathbf{0{,}144}_{(0,34)}$	$0{,}165_{(0,34)}$
ABZ9	$0{,}349_{(0,44)}$	$\mathbf{0{,}425}_{(0,41)}$	$0{,}404_{(0,45)}$	$0{,}244_{(0,47)}$	$\mathbf{0{,}167}_{(0,42)}$	$0{,}188_{(0,48)}$
FT10	$0{,}549_{(0,48)}$	$\mathbf{0{,}617}_{(0,45)}$	$0{,}596_{(0,44)}$	$0{,}199_{(0,50)}$	$\mathbf{0{,}135}_{(0,46)}$	$0{,}154_{(0,47)}$
FT20	$0{,}203_{(0,24)}$	$\mathbf{0{,}238}_{(0,20)}$	$0{,}227_{(0,22)}$	$0{,}110_{(0,23)}$	$\mathbf{0{,}079}_{(0,18)}$	$0{,}089_{(0,20)}$
La21	$0{,}427_{(0,33)}$	$\mathbf{0{,}470}_{(0,27)}$	$0{,}447_{(0,33)}$	$0{,}158_{(0,34)}$	$\mathbf{0{,}119}_{(0,29)}$	$0{,}142_{(0,34)}$
La24	$0{,}429_{(0,36)}$	$\mathbf{0{,}472}_{(0,47)}$	$0{,}460_{(0,34)}$	$0{,}181_{(0,38)}$	$\mathbf{0{,}141}_{(0,48)}$	$0{,}153_{(0,34)}$
La25	$0{,}440_{(0,30)}$	$\mathbf{0{,}472}_{(0,35)}$	$0{,}464_{(0,38)}$	$0{,}163_{(0,32)}$	$\mathbf{0{,}136}_{(0,37)}$	$0{,}142_{(0,38)}$
La27	$0{,}228_{(0,24)}$	$\mathbf{0{,}270}_{(0,29)}$	$0{,}259_{(0,26)}$	$0{,}184_{(0,26)}$	$\mathbf{0{,}142}_{(0,35)}$	$0{,}154_{(0,29)}$
La29	$0{,}235_{(0,31)}$	$\mathbf{0{,}278}_{(0,31)}$	$0{,}253_{(0,32)}$	$0{,}201_{(0,36)}$	$\mathbf{0{,}158}_{(0,34)}$	$0{,}184_{(0,35)}$
La38	$0{,}608_{(0,51)}$	$\mathbf{0{,}706}_{(0,36)}$	$0{,}687_{(0,33)}$	$0{,}199_{(0,48)}$	$\mathbf{0{,}112}_{(0,31)}$	$0{,}131_{(0,29)}$
La40	$0{,}618_{(0,45)}$	$\mathbf{0{,}684}_{(0,41)}$	$0{,}679_{(0,46)}$	$0{,}204_{(0,43)}$	$\mathbf{0{,}147}_{(0,37)}$	$0{,}150_{(0,44)}$

differences for any of the three configurations described in the table. We attribute this to the fact that the parameter δ can only help to diversify the search in the selection step, which is completely annealed if either the population becomes very little diverse due to the aggressive replacement of MOEA/D or if the neighborhood size T is set to a large value in the case of MOEA/D-n_r with a small nr parameter.

In Table 3 we show the performance of the three algorithms for all the instances. Based on the previous analysis, we only show results with a small T-value for MOEA/D (namely, $T = 5$), and a large value for MOEA/D-n_r (namely, $T = 100$). Also based on the lack of differences between using δ or not, we remove this parameter from all configurations. We can appreciate in the table that MOEA/D-n_r clearly outperforms MOEA/D in both indicators and for all instances. This behavior is due to the less aggressive replacement strategy which allows the algorithm to keep more diversity. Regarding the best value for the nr parameter, we obtain the best results for all instances when $nr = 1$, even though in some of them the differences with respect to the use of $nr = 2$ are not statistically significant. Due to this, we shall fix $nr = 1$ and $T = 100$.

5.2 Comparative analysis

Overall performance. In the following we analyze the behavior of the remaining strategies, i.e., MOEA/D-**ss** and MOEA/D-RO. In Table 4, where we also include NSGA-II as a baseline reference approach, we can see that for all instances except for FT20, the newly proposed MOEA/D-RO strategy provides the best results for both indicators. However, the differences between this algorithm and the already existing MOEA/D-**ss** are not statistically significant, but there is a huge difference between them when examining more carefully their runtime behavior as discussed in the following.

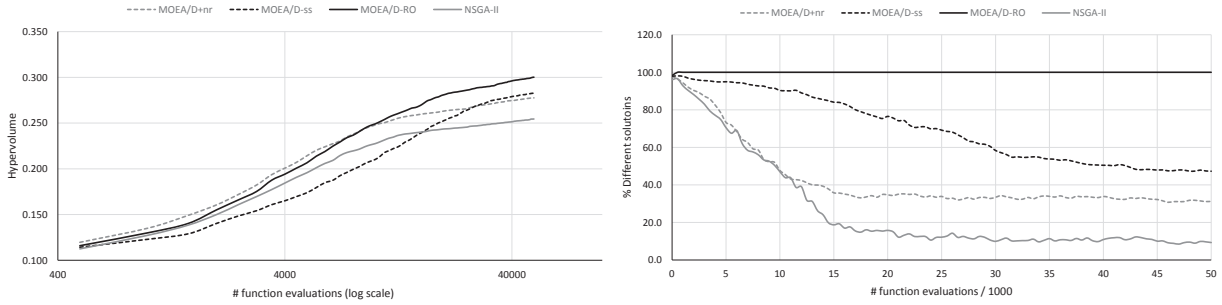
Runtime behavior. In Fig. 3, we provide a sample extracted from our data to illustrate this claim. In the first sub-figure, the Y-axis represents the average hypervolume obtained by each algorithm, while the X-axis measures the number of calls to the evaluation function (in the logarithmic scale). We remark that the behavior observed for instance La29 is seemingly the same for all the other instances, except for instances La21 and FT20 for which the behavior is slightly different (not shown in the figure due to space limitations).

Overall, even though MOEA/D-**ss** and MOEA/D-RO obtain similar hypervolume values at the end of the $5 \cdot 10^4$ function evaluations, we see that the evolution of both in time is different. For MOEA/D-RO, MOEA/D-n_r, and even NSGA-II, they obtain good results even with a short amount of function evaluations, while MOEA/D-**ss** is much slower converging to good results. Actually, in the first stages MOEA/D-**ss** performs worse than MOEA/D-n_r, but it is able to outperform it (and also MOEA/D-RO for instance FT20) after a given number of evaluations. The only exception to this is for La21, where MOEA/D-n_r remains better than MOEA/D-**ss**. In the right part of Fig. 3, we see the number of different objective vectors that are maintained within the population during the evolution of the algorithms. The MOEA/D-RO has a 100% of different solutions, as expected by taking T equal to the population size, but what is more interesting is that MOEA/D-**ss** offers more diversity than MOEA/D-n_r. In our judgment, this is what makes this strategy better in the long term, as in the first iterations of the algorithm it is more focused on keeping the diversity that will allow it to reach better solutions at the end.

EAF analysis. In order to have a more clear information about the relative performance of the different competing strategies, we compare the sets of solutions obtained by each pair of the algorithms by means of the empirical attainment functions (EAFs) [15]. The EAF provides the probability, estimated from several runs, that an arbitrary objective vector is dominated by, or equivalent to, a solution obtained by a single run of the algorithm. The difference between the EAFs for two different algorithms enables to identify the regions of the objective space where one algorithm performs better than another. The magnitude of the difference in favor of one algorithm is plotted within a gray-colored graduation as illustrated in Fig. 4. First of all, we can appreciate that the solutions obtained with MOEA/D-RO dominate those obtained by the MOEA/D-**ss** and MOEA/D-n_r in almost every region of the front. Furthermore, none of the algorithms are able to obtain solutions dominating those obtained by MOEA/D-RO. Interestingly, we can observe a specific feature for the NSGA-II algorithm that we were not able to elicit from the previous indicator based results. In fact, we can appreciate that it is quite good finding solutions in the middle part of the front. We hence argue that the decomposition nature of the MOEA/D algorithms makes solutions spread much better, thus forcing the population to stay spread even at the extreme parts and thus investing the running resources at looking for good solutions along all the front, while NSGA-II is focusing its "efforts" in finding good solutions in the middle part, especially if a loss in diversity occurs at the early stage of the optimization process.

Table 4: Overall comparisons of competing algorithms ($T = 100$, $nr = 1$, $\delta = 1$ when it holds).

instance	Hypervolume Indicator				Epsilon Indicator			
	MOEA/D-n_r	MOEA/D-**ss**	MOEA/D-RO	NSGA-II	MOEA/D-n_r	MOEA/D-**ss**	MOEA/D-RO	NSGA-II
ABZ7	$0{,}435_{(0,35)}$	$0{,}446_{(0,27)}$	$\mathbf{0{,}456}_{(0,30)}$	$0.423_{(0,32)}$	$0{,}128_{(0,37)}$	$0{,}112_{(0,28)}$	$\mathbf{0{,}106}_{(0,34)}$	$0{,}141_{(0,32)}$
ABZ8	$0{,}446_{(0,33)}$	$0{,}459_{(0,32)}$	$\mathbf{0{,}477}_{(0,33)}$	$0{,}428_{(0,35)}$	$0{,}144_{(0,34)}$	$0{,}129_{(0,34)}$	$\mathbf{0{,}112}_{(0,34)}$	$0{,}164_{(0,36)}$
ABZ9	$0{,}425_{(0,41)}$	$0{,}435_{(0,35)}$	$\mathbf{0{,}445}_{(0,38)}$	$0{,}415_{(0,33)}$	$0{,}167_{(0,42)}$	$0{,}156_{(0,38)}$	$\mathbf{0{,}148}_{(0,48)}$	$0{,}180_{(0,35)}$
FT10	$0{,}617_{(0,45)}$	$0{,}634_{(0,41)}$	$\mathbf{0{,}643}_{(0,42)}$	$0{,}591_{(0,39)}$	$0{,}135_{(0,46)}$	$0{,}117_{(0,43)}$	$\mathbf{0{,}109}_{(0,47)}$	$0{,}163_{(0,40)}$
FT20	$0{,}238_{(0,20)}$	$\mathbf{0{,}249}_{(0,20)}$	$0{,}243_{(0,23)}$	$0{,}241_{(0,18)}$	$0{,}079_{(0,18)}$	$\mathbf{0{,}068}_{(0,18)}$	$0{,}073_{(0,20)}$	$0{,}077_{(0,18)}$
La21	$0{,}470_{(0,27)}$	$0{,}456_{(0,29)}$	$\mathbf{0{,}478}_{(0,33)}$	$0{,}449_{(0,37)}$	$0{,}119_{(0,29)}$	$0{,}132_{(0,32)}$	$\mathbf{0{,}112}_{(0,34)}$	$0{,}142_{(0,40)}$
La24	$0{,}472_{(0,47)}$	$0{,}491_{(0,30)}$	$\mathbf{0{,}507}_{(0,36)}$	$0{,}456_{(0,39)}$	$0{,}141_{(0,48)}$	$0{,}122_{(0,30)}$	$\mathbf{0{,}108}_{(0,34)}$	$0{,}162_{(0,40)}$
La25	$0{,}472_{(0,35)}$	$0{,}485_{(0,30)}$	$\mathbf{0{,}494}_{(0,32)}$	$0{,}445_{(0,32)}$	$0{,}136_{(0,37)}$	$0{,}120_{(0,30)}$	$\mathbf{0{,}115}_{(0,38)}$	$0{,}165_{(0,33)}$
La27	$0{,}270_{(0,29)}$	$0{,}274_{(0,22)}$	$\mathbf{0{,}279}_{(0,28)}$	$0{,}254_{(0,29)}$	$0{,}142_{(0,35)}$	$\mathbf{0{,}134}_{(0,24)}$	$0{,}134_{(0,29)}$	$0{,}166_{(0,33)}$
La29	$0{,}278_{(0,31)}$	$0{,}283_{(0,32)}$	$\mathbf{0{,}300}_{(0,31)}$	$0{,}254_{(0,27)}$	$0{,}158_{(0,34)}$	$0{,}148_{(0,34)}$	$\mathbf{0{,}134}_{(0,35)}$	$0{,}190_{(0,30)}$
La38	$0{,}706_{(0,36)}$	$0{,}709_{(0,36)}$	$\mathbf{0{,}721}_{(0,30)}$	$0{,}683_{(0,34)}$	$0{,}112_{(0,31)}$	$0{,}111_{(0,34)}$	$\mathbf{0{,}101}_{(0,29)}$	$0{,}132_{(0,31)}$
La40	$0{,}684_{(0,41)}$	$0{,}697_{(0,35)}$	$\mathbf{0{,}726}_{(0,31)}$	$0{,}691_{(0,36)}$	$0{,}147_{(0,37)}$	$0{,}135_{(0,34)}$	$\mathbf{0{,}109}_{(0,44)}$	$0{,}145_{(0,35)}$



(a) Hypervolume evolution for La29 (b) Diversity evolution for La29

Figure 3: Evolution of the hypervolume (% of different solutions in the population) depending on the number of evaluations

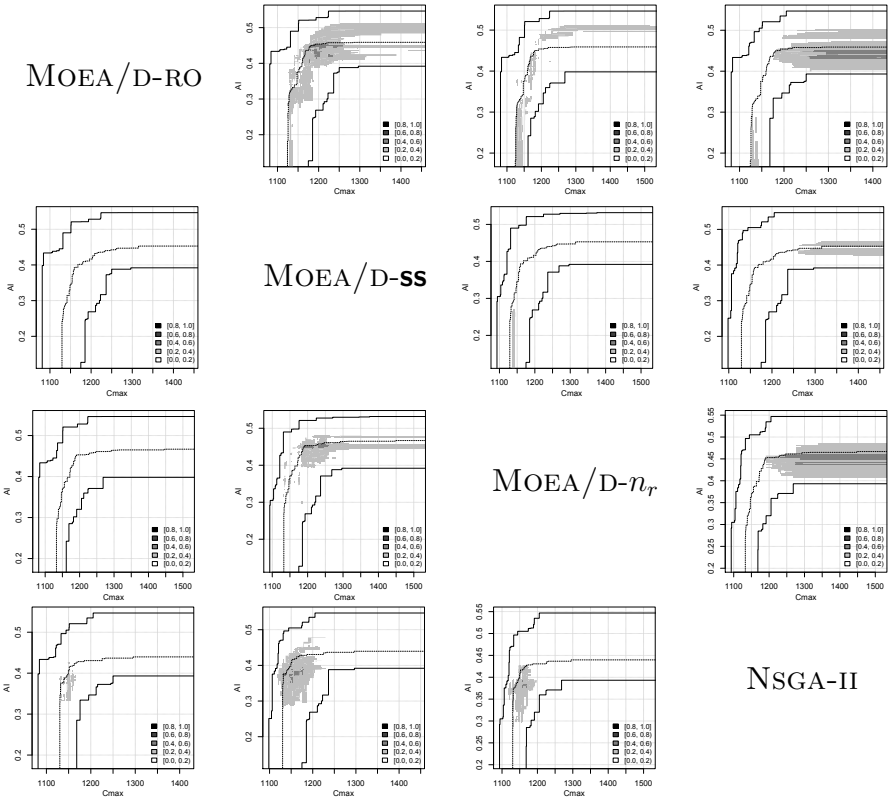


Figure 4: Comparison of MOEA/D-RO, MOEA/D-**ss**, MOEA/D-n_r, and NSGA-II using the EAF differences. Results are for instance La21. The plot in row i and column j shows the areas where algorithm i improves over algorithm j.

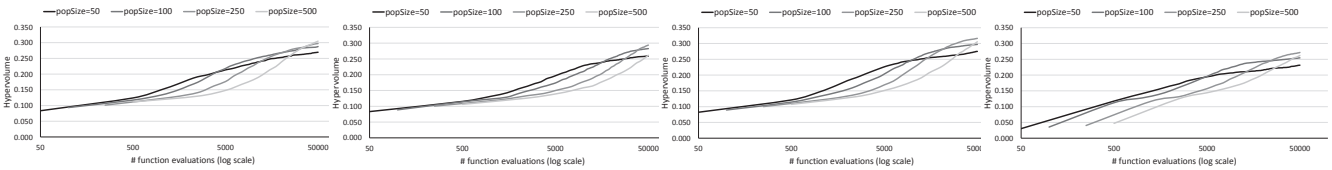


Figure 5: Runtime evolution of the hypervolume using different population sizes for instance La29 and different algorihtms. From Left to right: MOEA/D-RO, MOEA/D-**ss**, MOEA/D-n_r, NSGA-II.

Impact of population size. When a diversity issue is found in some algorithm, it is a common practice to increase the population size as to have a larger pool of solutions to evolve. To further study the influence of this parameter on the considered MOEA/D variants, in Fig. 5 we plot their runtime behavior using four different population sizes 50, 100, 250 and 500; but keeping as stopping criterion the same number of function evaluations. Even though the figure shows the results for instance La29, the behavior is the same for the other instances. We observe that increasing the population size is a good option if one has a large budget, but in case of having a short budget of time it is much better to have small population sizes and let the algorithm converge for more iterations. Furthermore, this behavior is common for all the tested algorithms, which indicates that all of them scale consistently in a similar way. Finally we can confirm that the MOEA/D-**ss** is the slowest algorithm converging, even when using larger the population size.

6. CONCLUSION

In this paper, we addressed some diversity issues of MOEA/D variants applied to the FJSP. The MOEA/D-n_r and MOEA/D-**ss** algorithms proposed in the literature are good alternatives, as they improve the performance of MOEA/D, but they can be improved substantially as illustrated by the proposed MOEA/D-RO variant independently of the available budget. We also confirmed that all MOEA/D variants perform very competitive compared to the Pareto-dominance based algorithm NSGA-II. However, it was interesting to observe that NSGA-II is able to find better solutions than MOEA/D-n_r and MOEA/D-**ss** in the medium part of the front, while the MOEA/D variants are all better at finding good solutions at the extremes of the front due to the nature of the MOEA/D framework.

It is our hope that the new findings elicited by the throughout analysis presented in this paper would enhance our understanding of the main ingredients for designing highly efficient decomposition-based algorithm. In particular, it would be interesting to conduct similar investigations for other combinatorial problems and to study more deeply the behavior of MOEA/D when additionally plugging local search based operators.

7. REFERENCES

[1] S. Abdullah and M. Abdolrazzagh-Nezhad. Fuzzy job-shop scheduling problems: A review. *Information Sciences*, 278:380–407, 2014.

[2] M. Andresen, H. Bräsel, M. Mörig, J. Tusch, F. Werner, and P. Willenius. Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. *Mathematical and Computer Modelling*, 48:1279–1293, 2008.

[3] P. C. Chang, S. H. Chen, Q. Zhang, and J. L. Lin. MOEA/D for flowshop scheduling problems. In *CEC*, pages 1433–1438, 2008.

[4] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2007.

[5] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[6] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE TEC*, 6(2):182–197, 2002.

[7] D. Dubois, H. Fargier, and P. Fortemps. Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147:231–252, 2003.

[8] D. Dubois and H. Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York (USA), 1986.

[9] P. Fortemps. Jobshop scheduling with imprecise durations: a fuzzy approach. *IEEE Transactions of Fuzzy Systems*, 7:557–569, 1997.

[10] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming. Generalized Decomposition. In *EMO*, pages 428–442, 2013.

[11] I. González Rodríguez, C. R. Vela, A. Hernández-Arauzo, and J. Puente. Improved local search for job shop scheduling with uncertain durations. In 19^{th} *Inter. Conf. on Automated Planning and Scheduling*, pages 154–161, 2009.

[12] A. Kaufmann and M. Gupta. *Introduction to Fuzzy Arithmetic*. Van Nostrand Reinhold, New York, 1991.

[13] H. Li and Q. Zhang. Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE TEC*, 13(2):284–302, 2009.

[14] L. D. Long and A. Ohsato. Fuzzy critical chain method for project scheduling under resource constraints and uncertainty. *International Journal of Project Management*, 26(6):688–698, 2008.

[15] M. López-Ibáñez, L. Paquete, and T. Stützle. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, chapter 9, pages 209–222. Springer.

[16] G. Marquet, B. Derbel, A. Liefooghe, and E.-G. Talbi. Shake them all! Rethinking selection and replacement in MOEA/D. In *PPSN XIII*, pages 641–651. 2014.

[17] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer, Boston, MA, USA, 1999.

[18] J. J. Palacios, C. R. Vela, I. González-Rodríguez, and J. Puente. Schedule generation schemes for job shop problems with fuzziness. In *ECAI*, pages 687–692, 2014.

[19] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE TEC*, 11(6):712–731, 2007.

[20] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE TEC*, 7(2):117–132, 2003.