Deconstructing Multi-objective Evolutionary Algorithms: An Iterative Analysis on the Permutation Flow-Shop Problem

Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle

IRIDIA, Université Libre de Bruxelles (ULB), Brussels, Belgium
{lteonaci,manuel.lopez-ibanez,stuetzle}@ulb.ac.be

Abstract. Many studies in the literature have applied multi-objective evolutionary algorithms (MOEAs) to multi-objective combinatorial optimization problems. Few of them analyze the actual contribution of the basic algorithmic components of MOEAs. These components include the underlying EA structure, the fitness and diversity operators, and their policy for maintaining the population. In this paper, we compare seven MOEAs from the literature on three bi-objective and one tri-objective variants of the permutation flowshop problem. The overall best and worst performing MOEAs are then used for a path-relinking-based iterative analysis, where each of the main components of these algorithms is analyzed to determine their contribution to the algorithms' performance. Results show some components not to be effective, while others only work well when simultaneously used.

1 Introduction

Evolutionary algorithms (EAs) are one of the most widely used metaheuristic algorithms and since a long time attract a large research community. Even when considering only applications to multi-objective optimization problems, many different multi-objective EAs (MOEAs) have been proposed [8,5,20,19,1,10,3]. In fact, MOEAs were among the first metaheuristics applied to multi-objective combinatorial optimization (MCOP) [16]. Moreover, several relevant developments in heuristic algorithms for multi-objective optimization have been advanced in the research efforts targeted to MOEAs. Such developments include archiving [10], dominance-compliant performance measures [21] and the performance assessment of multi-objective optimizers [22].

Despite the large number of MOEA proposals in the literature, little effort has been put into understanding the actual impact of specific algorithmic components. In general, the efficacy of MOEAs depends on a few main components. The first, common to single-objective optimization, is the underlying EA structure, which includes genetic algorithms (GA), evolutionary strategies (ES) and differential evolution (DE). The other two components, *fitness* and *diversity* operators, have been adapted from single-objective optimization to deal with search aspects particular to multi-objective problems. In MOEAs, the fitness operator typically considers the Pareto dominance relations between chromosomes in order to intensify the search. Conversely, the diversity operators focus on spreading

the solutions over the objective space in order to find a set of trade-off solutions representative of various possible preferences. Finally, the policy for the population management (that typically addresses the issue of which individuals to remove after new ones have been generated by the evolutionary operators recombination and mutation – one may call this also population reduction policy) determines which individuals are kept in the current population; this population reduction policy is using the measures on fitness and diversity measures to make these decisions.

Traditionally, a new MOEA is proposed as a monolithic block that integrates specific choices for the fitness, diversity, and population reduction. In this way, it is difficult to understand the actual impact each of these components has on performance. In fact, often algorithms that differ only by one such component have not been compared directly. In this paper, we compare seven different MOEA algorithms on the permutation flowshop problem (PFSP) to understand their performance. To make this analysis representative, we consider the three most relevant objective functions used in the PFSP literature, namely *makespan*, *total flow time*, and *total tardiness*. We then implement and compare all MOEAs for the three possible bi-objective variants as well as for the tri-objective variant.

While the performance of some algorithms is consistent across all PFSP variants, others present major differences. Concretely, the MOEA that ranks among the best for two of the variants ranks worst for the remaining. We then select two such variants and their best and worst ranked algorithms to iteratively move from the worst to the best configuration in the configuration space of MOEAs. This is done in a fashion akin to path relinking [9] by replacing component by component in the worst performing algorithm until obtaining the structure of the best performing one. Such a type of analysis has been proposed recently by Fawcett and Hoos [7] in the context of automatic algorithm configuration. The goal of our analysis is to identify the algorithm components that, for a specific problem, contribute most to algorithm performance. The results of the iterative analysis conducted show that, for runtime constrained scenarios such as the experimental setup traditionally adopted for the PFSP, the contribution of some components do not compensate for their computational overhead.

The paper is organized as follows. Section 2 presents the PFSP. Section 3 describes the algorithms we consider in this work, highlighting the differences between them. The experimental setup is presented in Section 4. The comparison of the MOEAs and the results of the analysis of MOEA components is presented in Sections 5 and 6, respectively. Finally, conclusions and possibilities for future work are discussed in Section 7.

2 The Permutation Flowshop Problem

The PFSP is one of the most widely studied scheduling problems in operations research. It arises in various industries such as chemical, steel, or ceramic tile production where jobs have to be executed by different machines in a given order. Since each execution takes a different amount of time, the order in which jobs are processed is of major importance for the efficiency of the process. An instance of the PFSP consists of a set of n jobs and m machines and a matrix

P of $n \times m$ processing times p_{ij}, where p_{ij} is the processing time of job i on machine j. For a permutation π that represents the order in which jobs will be executed, the completion times of all jobs on all machines are defined as

$$C_{\pi_0,j} = 0, \quad j = 1, ..., m, \qquad C_{\pi_i,0} = 0, \quad i = 1, ..., n, \tag{1}$$

$$C_{\pi,j} = \max\{C_{\pi_{i-1},j}, C_{\pi_i,j-1}\}, \qquad i = 1, ..., n,\ j = 1, ..., m \tag{2}$$

where π_i is the job at position i in the permutation. Typically, the PFSP has been studied using various objective functions, the most used being (i) *makespan* (C_{max}), i.e., the completion time of the last job on the last machine; (ii) *total flow time* (TFT), i.e., the sum of the completion times of each job on the last machine; and (iii) *total tardiness* (TT), the difference between the completion times of all jobs in the last machine and their due dates. In the latter case, a list of due dates is provided, where d_i is the due date of job i.

When more than one of these objectives is considered simultaneously, solutions are compared not based on a single objective value, but on an objective *vector*. Given a PFSP variant with objectives $f^i, i = 1, \ldots, k$, a solution s is said to be better (or to *dominate*) another solution s' if $\forall i$, $f^i(s) \leq f^i(s')$ and $\exists\, i$, $f^i(s) < f^i(s')$. If neither solution dominates the other, they are said to be *nondominated*. A typical goal of optimizers designed to solve a multi-objective problem is to find the set of nondominated solutions w.r.t. all feasible solutions, the Pareto set. Since this may prove to be computationally unfeasible, multi-objective metaheuristics have been used to find approximation sets, i.e., sets whose image in the objective space (called *approximation fronts*) best approximates the Pareto set image.

In this paper, we implement the MOEAs to solve the three possible bi-objective variants that combine C_{max}, TFT, and TT, namely C_{max}-TFT, C_{max}-TT, TFT-TT. Moreover, we also consider the tri-objective variant C_{max}-TFT-TT. To ensure the algorithms efficacy, we use the same algorithmic components typically found in the PFSP literature. Solutions are represented through direct encoding, that is, each individual in the MOEA is a permutation of the jobs. Initial solutions are constructed using the well-known NEH heuristic [15] that is extended from the usual makespan to the other objectives following [6]. The crossover operator applied is the two-point crossover operator. Finally, two mutation operators are considered: *insert*, which selects a job and reinserts it in a position of the permutation that is chosen uniformly at random, and *exchange*, which swaps two jobs, chosen uniformly at random, of the permutation.

3 Multi-objective evolutionary algorithms

Since the first proposal of a MOEA [16], several different algorithmic structures and components have been devised. To better understand the commonalities and peculiarities of the most relevant approaches, we review various proposals here. All MOEAs described above and used in this work are summarized in Table 1.

Many extensions of EAs to multi-objective optimization rely mostly on the extension of the concepts of fitness and diversity. In a MOEA, the fitness of a solution is generally calculated by means of dominance compliant metrics, meaning

Table 1. Main algorithmic components of the MOEAs considered in this work. For an explanation of the table entries we refer to the text. Structure here refers to the choice whether the MOEA is based on either a traditional genetic algorithm (GA) or an evolution strategy (ES).

algorithm	**fitness**	**diversity**	**reduction**	**structure**
MOGA [8]	*dominance rank*	*niche sharing*	*one shot*	*GA*
NSGA-II [5]	*dominance depth*	*crowding distance*	*one shot*	*GA*
SPEA2 [20]	*dominance count/rank*	*k-NN*	*iterative*	*GA*
IBEA [19]	*binary indicator*	*none*	*iterative*	*GA*
HypE [1]	*hypervolume contribution*	*none*	*iterative*	*GA*
PAES [10]	*none*	*grid crowding*	*one shot*	$(1+1)$-*ES*
SMS-EMOA [3]	*three-way fitness*	*none*	*steady-state*	$(\mu+1)$-*ES*

the algorithm will favor solutions according to Pareto dominance. Several fitness metrics can be found in the literature, such as dominance rank [8], dominance count [20], and dominance depth [5,1]. Besides fitness measures, the population is also evaluated according to diversity metrics. In single-objective optimization, diversity metrics are used to prevent the algorithm from stagnating by spreading individuals across the decision space. This concept becomes even more important for multi-objective optimization since multiple solutions need to be found. In this context, diversity is generally measured in terms of the objective space, the main concern being to have a well-distributed approximation to the Pareto front. The most commonly used metrics include crowding distance [5], niche sharing [17], and k-nearest-neighbor [20]. Finally, algorithms also differ as to the frequency with which these values are calculated. *One shot* algorithms compute fitness and diversity values once before population reduction and then discard the worst individuals. By contrast, *iterative* algorithms re-calculate fitness and diversity values every time a solution is discarded from the population. Although this second alternative is known to be computationally more expensive, initial results have shown this strategy to produce better results when runtime is not an issue [1].

The particular choice of fitness, diversity metrics and how often these are computed are distinguishing features of different multi-objective EAs. The most relevant algorithms propose their own fitness and diversity strategies. MOGA [8], for instance, uses dominance ranking and niche sharing. The population reduction adopts the one shot policy. NSGA-II [5] uses dominance depth and crowding distance, and also uses one shot population reduction. SPEA2 [20] uses a combination of dominance count and dominance rank for the fitness computation and a k-NN metric for diversity, but discards individuals using an iterative reduction policy. IBEA [19] uses binary quality indicators to compare solutions. The two most commonly adopted are: (i) the ϵ-indicator, that computes the ϵ value that would have to be added (or multiplied) to one solution for it to be dominated by another, and; (ii) the hypervolume difference, which computes the volume of the subspace one individual dominates that the other do not. These binary values are computed for each pair of solutions in the population. The fitness of

an individual is then equal to the aggregation of its indicator w.r.t. the rest of the population.

More recently, the hypervolume indicator has been used to evaluate fitness and diversity simultaneously during an MOEA run. In this case, it computes the volume of the objective space dominated by a given approximation set, bounded by a reference point. The hypervolume used as a fitness metric captures both concepts of closeness to the Pareto front and well-spread approximation, thus replacing the explicit diversity measure in other algorithms [19,1]. In this work, we consider HypE, a traditional genetic algorithm (GA). HypE [1] uses the hypervolume contribution, that is, the volume of the subspace dominated exclusively by a given solution. HypE evaluates the fitness of the individuals at two moments: (i) before mating, when all individuals are assessed, and; (ii) during population reduction, when a speed-up is employed: the hypervolume contribution is used as a tie-breaker for the dominance depth approach.

Several MOEAs are based on a structure that is rather typical for evolution strategies (ES). PAES [10] is a (1+1)-ES that actually resembles a local search procedure. At each iteration, an incumbent solution is mutated and compared to the population of the algorithm, which maintains only nondominated solutions. The actual efficiency of the algorithm lies in the intelligent adaptive procedure used to keep this population well-spread and to direct the search towards regions that are little explored. If the population size has not yet reached the maximum allowed size, the new solution is accepted as long as it is not dominated by an existing solution. Otherwise, the new solution is only accepted in the population if it either dominates an existing solution or if it is located in a region of the objective space where the algorithm still has not found many solutions.

Another multi-objective ES proposal is SMS-EMOA [3], a steady-state ES. Similar to HypE, SMS-EMOA uses dominance depth followed by hypervolume contribution for tie-breaking. In fact, the combined fitness metric used by this algorithm can be described as a three-way fitness metric. First, individuals are sorted according to dominance depth. If all individuals in the population are given the same fitness value, the hypervolume contribution is used to break ties. Otherwise, all fronts that fit the new population are preserved, and tie-breaking (by means of the dominance rank metric) is applied for the first front that does not fit fully into the population. As SMS-EMOA is a steady-state algorithm, the offspring always replaces the worst individual of the parent population, and the mating selection is always done at random.

4 Experimental setup

We use the benchmark set provided by Taillard [18] following previous work on multi-objective PFSP [6,14]. This benchmark set contains instances with all combinations of $n \in \{20, 50, 100, 200\}$ jobs and $m \in \{5, 10, 20\}$ machines, except for $n = 200$ and $m = 5$ (200x5). We consider 10 instances of each size, 110 instances in total. The maximum runtime per instance equals $t = 0,1 \cdot n \cdot m$ seconds. All experiments were run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB of cache size under Cluster Rocks Linux version 6.0/CentOS 6.3.

Table 2. Parameter space for tuning the MOEA numerical parameters.

Parameter	*pop*	*offspring*	*pCross*	*pMut*	*pExchange*	*archive*
Domain	$\{10, 20, 30, 50, 80, 100\}$	1 or $[0.1, 2]$	$[0, 1]$	$[0, 1]$	$[0, 1]$	*none* or $\{30, 70, 100, 300, \{500, 700, 1000\}$

Table 3. Parameter space for tuning additional MOEA-specific parameters.

Algorithm	IBEA	MOGA	PAES	SPEA2
Parameter	*indicator*	σ_{share}	l	k
Domain	{epsilon, hypervolume}	$[0.1, 1]$	$\{1, 2\}$	$\{1, \ldots, 9\}$

The MOEAs used in this algorithm were instantiated using the C++ ParadisEO framework [11]. We implemented several algorithmic components required for our study that were not available in ParadisEO. We also extended ParadisEO's PFSP library to handle all PFSP variants considered in this paper. The original MOEAs were not designed for the PFSP, and, hence, their parameter settings are likely not well suited for this problem. Therefore, we tuned the parameter settings of all MOEAs using irace [2,12] with a tuning budget of 1000 experiments. As training instances during tuning, we used a different benchmark set [6] from the one used in the final analysis. irace was originally designed for single-objective algorithms, but it has been extended to handle the multi-objective case by using the hypervolume quality measure. For computing the hypervolume, we normalize objective values in the range $[1, 2]$ and use $(2.1, 2.1)$ as the reference point. The parameter space considered for all algorithms is the same, depicted in Table 2. The additional parameters required by specific algorithms are listed in Table 3.

To compare the tuned MOEAs, we consider the average hypervolume over 10 runs of each algorithm per instance. We then plot parallel coordinate plots of these results for each problem variant. Concretely, for each variant we produce 11 plots (one per instance size), each depicting the behavior of all MOEAs in ten different instances. Due to space limitations, few representative results are shown here. The complete set of results are made available as a supplementary page [4]. Finally, to select the source and target algorithms for the ablation analysis, we compute rank sums considering the average hypervolume of the 10 runs per instance size.

5 Comparison of MOEAs

The four variants of the PFSP considered in this paper differ significantly from each other in the shape of the non-dominated reference fronts and the number of non-dominated points [6]. As a consequence, the performance of the MOEAs may vary greatly from one variant to another. In this section, we discuss the most representative results considering one variant at a time. For each PFSP variant, we present the tuned MOEAs, discuss their performance and conclude with a rank sum analysis.

Table 4. Rank sum analysis: The MOEAs are sorted according to their sum of ranks (in parenthesis) for each MO-PFSP variant. Lower rank-sums indicate better performance.

Cmax-TFT	HypE (224.5)	SMS-EMOA (260)	SPEA2 (262.5)	IBEA (476)	NSGA-II (521)	MOGA (640)	PAES (696)
Cmax-TT	NSGA-II (190)	IBEA (253)	SMS-EMOA (320)	SPEA2 (356)	MOGA (590)	PAES (657)	HypE (714)
TFT-TT	HypE (269)	SMS-EMOA (275)	NSGA-II (282)	SPEA2 (333)	PAES (560)	IBEA (642)	MOGA (719)
Cmax-TFT-TT	SMS-EMOA (123)	SPEA2 (267)	IBEA (417)	PAES (443)	NSGA-II (505)	MOGA (584)	HypE (741)

5.1 Cmax-TFT

Table 5 shows the best parameter configuration found by irace for each MOEA for Cmax-TFT. These configurations show some commonalities. First, the population sizes are usually small, and sometimes no external archive is used to store non-dominated solutions. This is justified by the small number of non-dominated solutions that exist in the Pareto front of this problem. The number of offspring is always around 100% as traditionally used by GAs. The mutation rate $pMut$ is always very high and the insertion operator is used much more frequently than the exchange one. Finally, SPEA2 considers the first nearest neighbor for its diversity operator and IBEA uses the epsilon indicator.

Figure 1 gives parallel coordinate plots for the average hypervolume (given on y-axis) measured on each of the 10 instances of size $n = 200$ and $m = 20$. As shown in Fig. 1, HypE, SMS-EMOA and SPEA2 perform best. By contrast, the remaining algorithms perform poorly, with PAES and MOGA being the ones that produce the worst average hypervolumes. The fact that HypE and SMS-EMOA perform similarly well is interesting, because both algorithms use similar fitness and diversity components, but very different underlying EA structures. On the other hand, the performance difference between SPEA2 and NSGA-II or MOGA, for instance, is most likely related to the MOEA components they use, since their underlying structure is the same.

Next, we computed for each algorithm its rank sum. In particular, we ranked the average hypervolume of each algorithm on each instance from one (best) to seven (worst) and then summed the ranks of each algorithm across all 110 instances. The rank sums are given in Table 4. HypE presents the lowest rank sum, but SMS-EMOA and SPEA2 also show low values. PAES performs worst when measured across all instance sizes even though it performs better than MOGA and NSGA-II on the largest instances (Fig. 1).

5.2 Cmax-TT

The parameters selected by irace for the Cmax-TT variant (Table 6) do not differ much from those obtained above for the Cmax-TFT variant. We identify two important differences. First, the number of offspring is now set either to a much larger or a smaller value than the population size, instead of being roughly equal. Second, many algorithms use external archives with a large maximum size. An analysis of the number of non-dominated solutions returned by the algorithms

Table 5. Parameter settings chosen by irace for all MOEAs on Cmax-TFT; the column *Other* refers to algorithm specific parameters while the others are common to all MOEAs.

MOEA	*pop*	*offspring*	*pCross*	*pMut*	*pExchange*	*archive*	*Other*
HypE	50	81%	52%	95%	54%	-	-
IBEA	20	109%	75%	82%	19%	-	*Epsilon*
MOGA	30	186%	21%	89%	31%	-	$\sigma = 0.31$
NSGA-II	30	103%	87%	70%	33%	700	-
PAES	10	-	-	-	19%	70	$l = 2$
SMS-EMOA	10	-	83%	88%	52%	-	-
SPEA2	10	92%	58%	65%	44%	70	$k = 1$

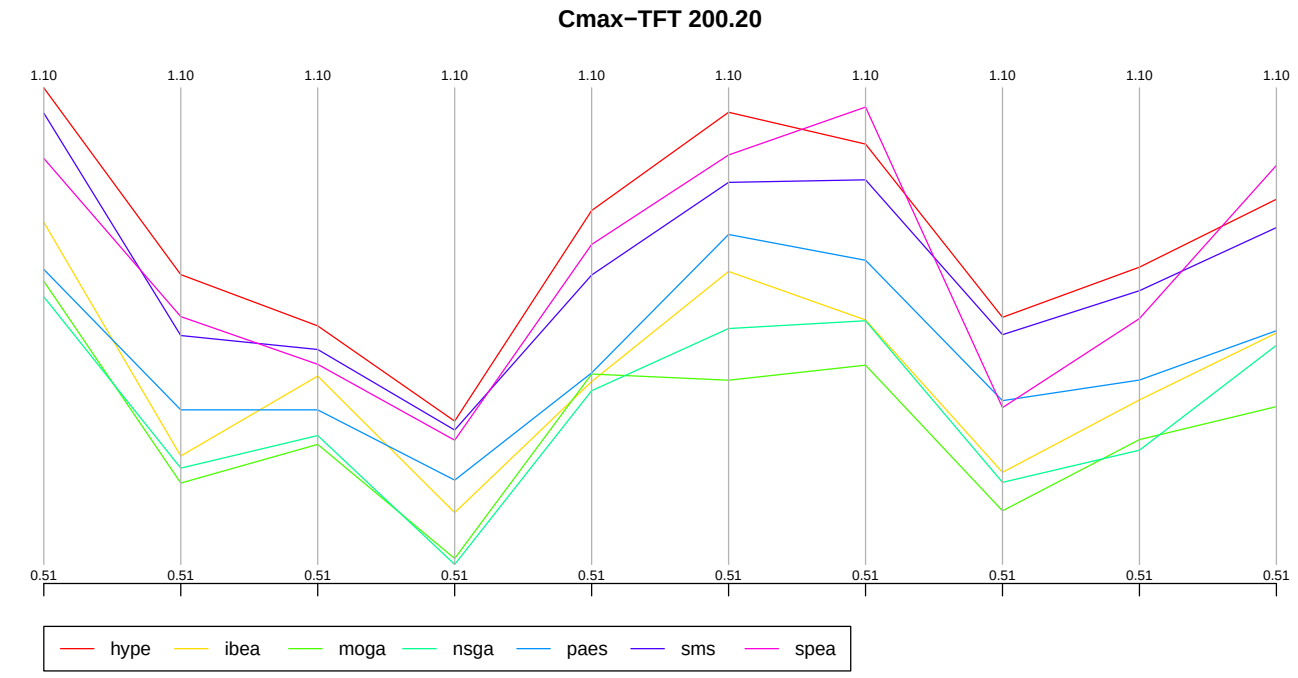

Fig. 1. Average hypervolume over 10 runs on 10 instances of size 200x20 of Cmax-TFT.

revealed, however, that this number was actually much lower than the maximum limits.

The main differences observed in Fig. 5.3 concern HypE, NSGA-II, and IBEA. Hype was the best MOEA for the Cmax-TFT variant. However, for Cmax-TT it is the worst among all algorithms compared. The same loss of quality is not observed for SMS-EMOA, which suggests that the underlying EA structure this time became of major importance. Second, NSGA-II and IBEA did not perform well for Cmax-TFT, but they now produce the best results among all MOEAs. This is confirmed by the rank sum analysis in Table 4, where NSGA-II obtains the lowest rank sum.

5.3 TFT-TT

Compared to the parameters used for the two previous PFSP variants, the configurations tuned for TFT-TT present two features worth highlighting. First, the frequency of usage of the exchange operator has generally increased, with

Table 6. Parameter settings chosen by irace for all MOEAs on Cmax-TT; the column *Other* refers to algorithm specific parameters while the others are common to all MOEAs.

MOEA	*pop*	*offspring*	*pCross*	*pMut*	*pExchange*	*archive*	*Other*
HypE	50	122%	73%	95%	35%	500	-
IBEA	30	80%	41%	85%	38%	-	*Epsilon*
MOGA	30	138%	51%	99%	19%	70	$\sigma = 0.19$
NSGA-II	50	75%	28%	77%	66%	-	-
PAES	30	-	-	-	48%	300	$l = 2$
SMS-EMOA	10	-	89%	86%	8%	100	-
SPEA2	10	142%	81%	70%	30%	300	$k = 5$

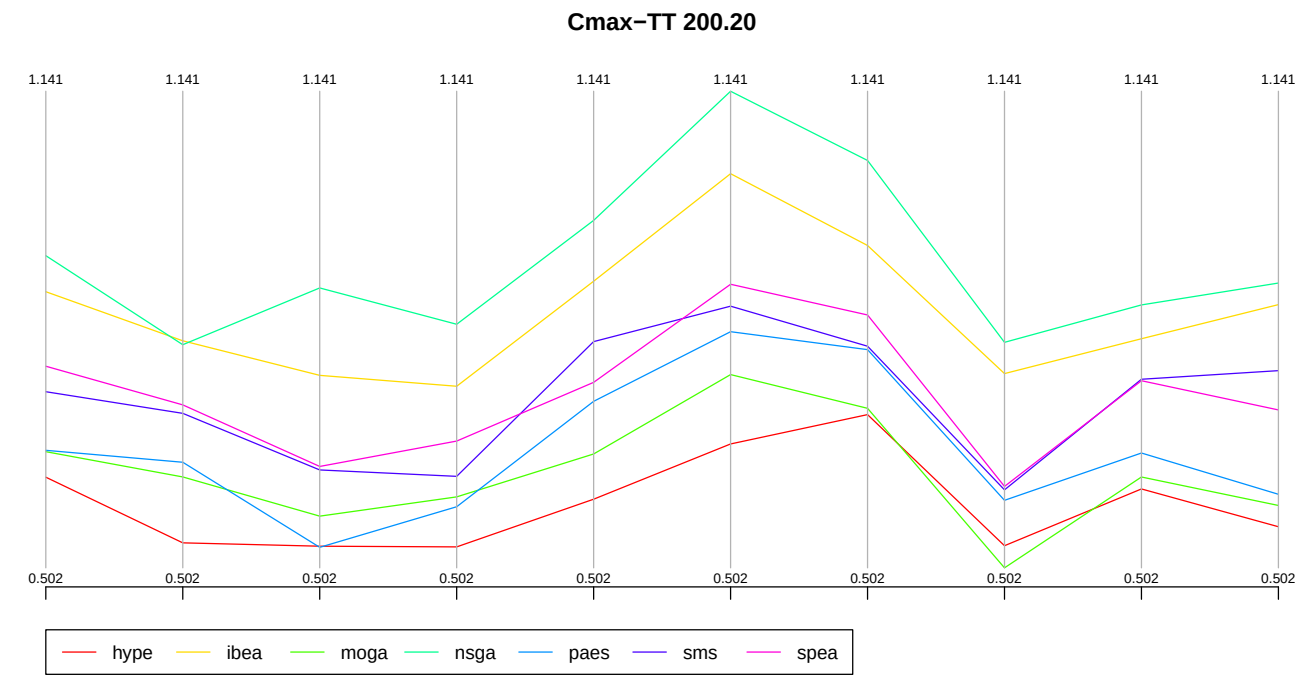

Fig. 2. Average hypervolume over 10 runs on 10 instances of size 200x20 of Cmax-TT.

the exception of NSGA-II. In fact, even the mutation probability is very low in NSGA-II if compared to the other MOEAs. The second is the increase in the number of MOEAs that use external archives with large size. Concerning solution quality, Fig. 5.3 shows that the performance of the algorithms is very similar to their performance in the Cmax-TFT variant. The main differences are the very good performance of NSGA-II contrasting with the poor performance of IBEA and MOGA (see Table 4).

5.4 Cmax-TFT-TT

The last variant considered in this section, Cmax-TFT-TT, is the only one considering three objectives. It is expected that the number of non-dominated solutions be much larger than in the bi-objective variants. This is reflected in the use of larger external archives by the best configurations found by irace (see Table 8). It is also expected that the hypervolume-based algorithms show a significant performance decrease, since the computing the hypervolume contributions becomes

Table 7. Parameter settings chosen by irace for all MOEAs on TFT-TT; the column *Other* refers to algorithm specific parameters while the others are common to all MOEAs.

MOEA	*pop*	*offspring*	*pCross*	*pMut*	*pExchange*	*archive*	*Other*
HypE	50	78%	78%	87%	62%	-	-
IBEA	50	131%	87%	81%	27%	70	*Epsilon*
MOGA	50	147%	37%	87%	48%	300	$\sigma = 0.40$
NSGA-II	30	90%	57%	50%	16%	100	-
PAES	10	-	-	-	44%	-	$l = 2$
SMS-EMOA	10	-	80%	97%	42%	300	-
SPEA2	10	105%	29%	76%	68%	1000	$k = 5$

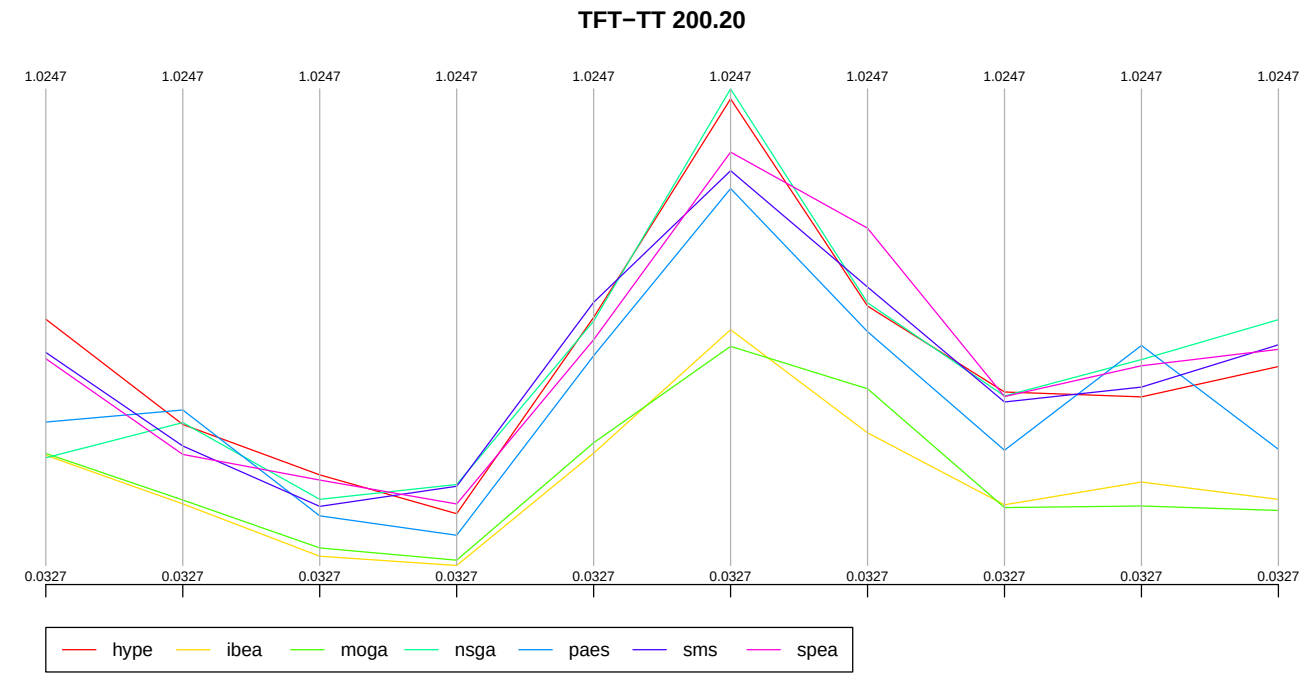

Fig. 3. Average hypervolume over 10 runs on 10 instances of size 200x20 of TFT-TT.

more expensive. Interestingly, SMS-EMOA is able to scale well and performs the best, as shown in Fig. 5.3 and Table 4, probably due to the use of a small population size. SPEA2 ranked second, as shown in Table 4, achieving good results throughout all instances.

Among the remaining algorithms, the biggest change is the performance of PAES. While for the previous PFSP variants it always performed rather poorly, for Cmax-TFT-TT its performance greatly improved. This is most likely due to two main characteristics of this algorithm. First, PAES always explores the "neighborhood" of an individual until it moves on to the next little explored region. In practice, many individuals kept in the population never get to be used as incumbent solution, and thus a lot of the exploration power of algorithm is lost. For bi-objective scenarios this is problematic, since the algorithm converges quickly. When more objectives are considered, though, intensification becomes more important, and hence the improvement in performance. Second, PAES generates only one offspring per iteration and uses only mutation operators. Being

Table 8. Parameter settings chosen by irace for all MOEAs on Cmax-TFT-TT; the column *Other* refers to algorithm specific parameters while the others are common to all MOEAs.

MOEA	*pop*	*offspring*	*pCross*	*pMut*	*pExchange*	*archive*	*Other*
HypE	100	61%	86%	100%	46%	-	-
IBEA	30	116%	48%	95%	19%	-	*Epsilon*
MOGA	30	199%	49%	95%	68%	1000	$\sigma = 0.40$
NSGA-II	80	110%	45%	73%	37%	-	-
PAES	10	-	-	-	13%	300	$l = 2$
SMS-EMOA	10	-	54%	92%	60%	1000	-
SPEA2	10	175%	68%	67%	31%	700	$k = 5$

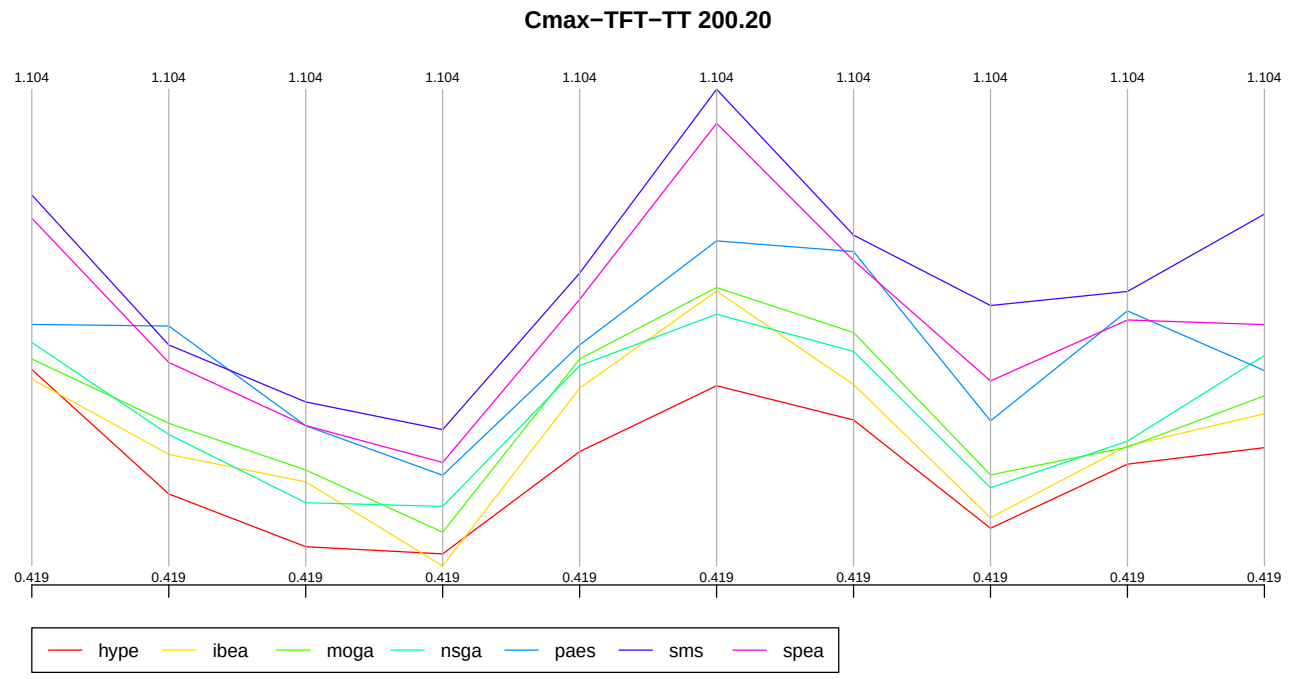

Fig. 4. Average hypervolume over 10 runs on instance sizes 200x20: Cmax-TFT-TT.

faster, it can perform more iterations in a runtime constrained experimentation setup such as the one we consider in this paper.

6 Iterative analysis

The experiments in the previous section showed important differences in MOEA performance in dependence of particular problems. In this section, we conduct an iterative analysis to understand which algorithm components cause the main differences between the best and worst performing MOEA variants. This analysis can be seen as a path relinking in the configuration space and it has been applied in the context of automatic algorithm configuration before by Fawcett and Hoos [7]. The main motivation for this analysis is to get insight into the contribution of specific components on algorithm performance [7]. We do so by generating intermediate configurations between the two algorithms, referred to as source and target. At each step, we modify all individual algorithm com-

ponents in which the two algorithms differ, and follow the path that has the maximum impact on performance. In this way, the analysis of the intermediate configurations allows us to understand the actual contribution of the individual components to the performance of the algorithm.

For this analysis, we have chosen the TFT-TT and the Cmax-TFT-TT variant. While in TFT-TT HypE is the best performing algorithm, in Cmax-TFT-TT it is the worst performing. The other two algorithms for our analysis are MOGA (worst on TFT-TT) and SMS-EMOA (best on Cmax-TFT-TT).

6.1 TFT-TT

The components of the source and target algorithms for TFT-TT, MOGA (worst on TFT-TT) and HypE (best on TFT-TT), are listed in Table 1. All intermediate configurations tested are shown in Fig. 6.2. The y-axis represents the rank sums. The x-axis contains the steps of the procedure. In step 0, only the source algorithm is depicted, in this case MOGA. In step 1, we modify the three components that differ between MOGA and HypE, namely fitness, diversity, and replacement (see Table 1), thus generating three new algorithms.

As shown in Fig. 6.2, the component that leads to the strongest decrease of the rank sum is the diversity component. The interesting aspect of this step is that HypE does not use diversity metrics. This means that the change that most improves MOGA is to remove its diversity component. This may be explained by the extra number of iterations the intermediate configuration is able to perform. In fact, for some instances this configuration performs nine times more iterations than the original MOGA.

Step 2 modifies the fitness and replacement components, starting from the best algorithm in step 1. However, none of these algorithmic components by itself leads to an improvement. The performance of HypE is only matched once both components are changed to entirely match HypE. Two main conclusions can be drawn from this result: (i) using either a one shot or an iterative policy with the fitness component used by MOGA (dominance ranking) does not make a difference, and; (ii) the hypervolume contribution needs to be coupled with the iterative policy.

6.2 Cmax-TFT-TT

The source and target algorithms for Cmax-TFT-TT are respectively HypE (worst) and SMS-EMOA (best). Three of their components differ, namely fitness, replacement and EA structure (see Table 1). All intermediate configurations tested are shown in Fig. 6.2. In step 1, the component that leads to strongest improvement from HypE is the EA structure. When moving from Hype to SMS-EMOA, the EA structure is characterized by two features: (i) only one offspring is produced by iteration, and; (ii) individuals are randomly selected for mating. Further analysis showed that this random selection is key to the performance of the algorithm. The explanation is that the computationally most costly component of HypE is computing the fitness of the individuals for mating selection. By removing this component, the number of iterations grows significantly, which

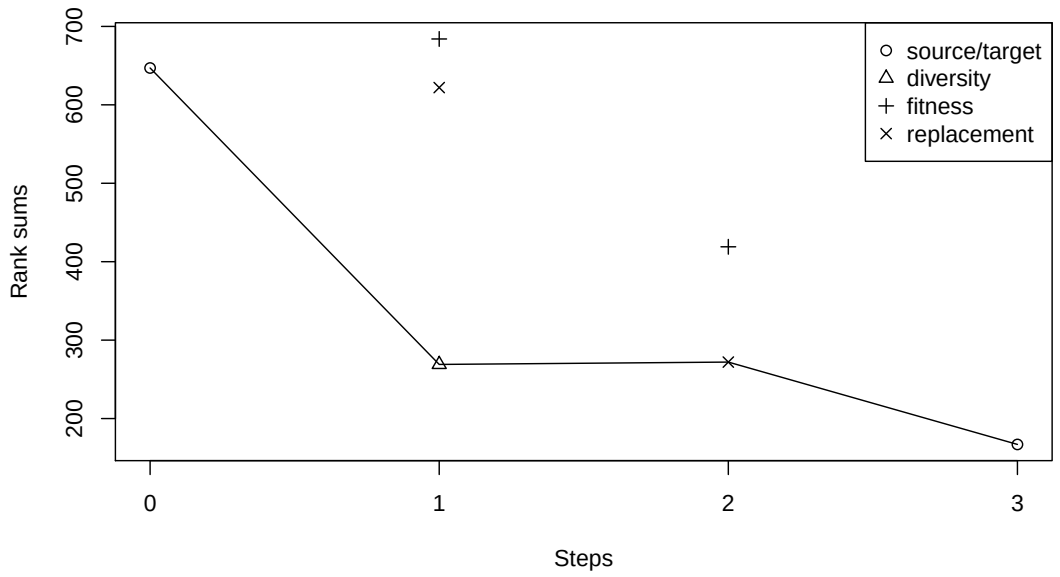

Fig. 5. Intermediate configurations tested for TFT-TT. The line connects the changes that caused the largest performance improvement.

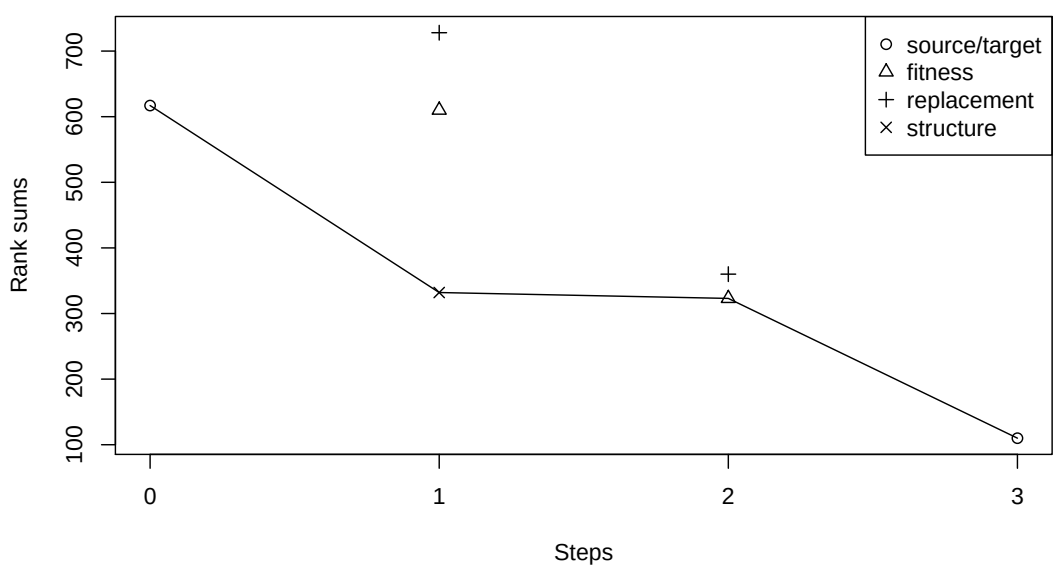

Fig. 6. Intermediate configurations tested for Cmax-TFT-TT. The line connects the changes that caused the largest performance improvement.

explains the performance gain. The second step of the iterative analysis considers changing the fitness and replacement components. Again, no significant modifications are identified, which means both components need to be modified together.

7 Conclusions and Future work

Traditionally, MOEAs have been seen as monolithic blocks, which is reflected by the fact that many of these algorithms have been proposed and analyzed as such. However, for a more detailed analysis it is often preferable to decompose the algorithms into building blocks or, say, main algorithm components. In this paper, we have followed this direction and deconstructed MOEAs into four main

components. These components are the underlying EA algorithm, the fitness and diversity operators, and the population management policy. We believe that analyzing these components and their contributions to performance is key to understanding which MOEA works best on each problem and to develop better MOEAs in the future.

In this work, we deconstructed seven relevant MOEAs, namely HypE, IBEA, MOGA, NSGA-II, PAES, SMS-EMOA, and SPEA2. We compared them on three bi-objective and one tri-objective variants of the permutation flowshop problem (PFSP). The results are strongly variant and also problem dependent, highlighting particular strengths and weaknesses of each MOEA. Overall, SMS-EMOA and SPEA2 are always able to find good approximation sets. Maybe surprisingly, some algorithms such as HypE are sometimes among the best for some problem variant while they perform poorly for other problem variants. Furthermore, we conduct an iterative analysis interpolating between the best and worst performing algorithms for two PFSP variants. We show that, sometimes, not using diversity operators or adopting random mating selection leads to a significant improvements in performance, in part due to the more computation time efficiency obtained by the modified algorithms.

The conclusions drawn from this work confirm the great performance variability metaheuristics generally present for combinatorial optimization problems. As said, the best performing MOEA (and with this the algorithm components that should be chosen to obtain a high-performing MOEA algorithm) strongly depend on the specific problem or problem variant being tackled. This fact also motivates the need for a flexible, component-wise implementation of MOEAs, as well as the specialization of MOEAs to specific problems through automatic algorithm configuration. Initial results on flexible, configurable frameworks for other multi-objective search techniques have shown that this is a very promising path for research [13].

Acknowledgments. The research leading to the results presented in this paper has received funding from the Meta-X project from the Scientific Research Directorate of the French Community of Belgium and the COMEX project within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office. Leonardo C. T. Bezerra, Manuel López-Ibáñez and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a FRIA doctoral fellow, a postdoctoral researcher and a senior research associate, respectively.

References

1. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. Evolutionary Computation 19(1), 45–76 (2011)
2. Balaprakash, P., Birattari, M., Stützle, T.: Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. In: Bartz-Beielstein, T., et al. (eds.) Hybrid Metaheuristics, LNCS, vol. 4771, pp. 108–122. Springer, Heidelberg, Germany (2007)
3. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. 181(3), 1653–1669 (2007)

4. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Deconstructing multi-objective evolutionary algorithms: An iterative analysis on the permutation flowshop: Supplementary material. `http://iridia.ulb.ac.be/supp/IridiaSupp2013-010/` (2013)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 181–197 (2002)
6. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Comput. Oper. Res. 38(8), 1219–1236 (2011)
7. Fawcett, C., Hoos, H.H.: Analysing differences between algorithm configurations through ablation. In: Proceedings of MIC 2013, the 10th Metaheuristics International Conference. pp. 123–132 (2013)
8. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Forrest, S. (ed.) ICGA. pp. 416–423. Morgan Kaufmann Publishers (1993)
9. Glover, F.: A template for scatter search and path relinking. In: Hao, J.K., et al. (eds.) Artificial Evolution, LNCS, vol. 1363, pp. 1–51. Springer, Heidelberg, Germany (1998)
10. Knowles, J.D., Corne, D.: Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computation 8(2), 149–172 (2000)
11. Liefooghe, A., Jourdan, L., Talbi, E.G.: A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. Eur. J. Oper. Res. 209(2), 104–112 (2011)
12. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011)
13. López-Ibáñez, M., Stützle, T.: The automatic design of multi-objective ant colony optimization algorithms. IEEE Trans. Evol. Comput. 16(6), 861–875 (2012)
14. 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)
15. Nawaz, M., Enscore, Jr, E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA 11(1), 91–95 (1983)
16. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette, J.J. (ed.) ICGA-85. pp. 93–100. Lawrence Erlbaum Associates (1985)
17. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2(3), 221–248 (1994)
18. Taillard, É.D.: Benchmarks for basic scheduling problems. Eur. J. Oper. Res. 64(2), 278–285 (1993)
19. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) Parallel Problem Solving from Nature, PPSN VIII, LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg, Germany (2004)
20. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control. pp. 95–100. CIMNE, Barcelona, Spain (2002)
21. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto evolutionary algorithm. IEEE Trans. Evol. Comput. 3(4), 257–271 (1999)
22. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. 7(2), 117–132 (2003)