Automatic Component-wise Design of Multi-objective Evolutionary Algorithms

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

***Abstract*—Multi-objective evolutionary algorithms are typically proposed, studied and applied as monolithic blocks with a few numerical parameters that need to be set. Few works have studied how the algorithmic components of these evolutionary algorithms can be classified and combined to produce new algorithmic designs. The motivation for studies of this latter type stem from the development of flexible software frameworks and the usage of automatic algorithm configuration methods to find novel algorithm designs. In this paper, we propose a multi-objective evolutionary algorithm template and a new conceptual view of its components that surpasses existing frameworks in both the number of algorithms that can be instantiated from the template and the flexibility to produce novel algorithmic designs. We empirically demonstrate the flexibility of our proposed framework by automatically designing multi-objective evolutionary algorithms for continuous and combinatorial optimization problems. The automatically designed algorithms are often able to outperform six traditional multi-objective evolutionary algorithms from the literature, even after tuning their numerical parameters.**

***Index Terms*—Multiobjective optimization, evolutionary algorithms, permutation flowshop problem, automatic algorithm configuration.**

I. Introduction

MULTI-OBJECTIVE evolutionary algorithms (MOEAs) are a central research topic in evolutionary computation as evidenced by the number of different proposals in the literature [1–8]. The traditional study of MOEAs considers these algorithms as monolithic units [9, 10] and only few articles have considered the contribution of individual algorithmic components of MOEAs to performance [11–13]. These latter empirical studies indicate that (i) performance improvements may be achieved by replacing the algorithmic components of well-established MOEAs by alternative options from other MOEAs and that (ii) the benefits of such changes become more important when applying MOEAs to scenarios different from those for which they were originally designed.

The component-wise view of MOEAs consists in identifying individual algorithmic components in different MOEAs that have the same function and, thus, could be replaced by alternative procedures taken either from different MOEAs or newly devised. Examples are the *fitness* and *diversity* components that appear in many MOEAs [11, 14]. This component-wise view has two main benefits. First, it allows algorithm designers to identify the various options available for each algorithmic component and whether a particular combination of components, i.e., an algorithm "design", has already been proposed. Second, it allows algorithm users to adapt the design of MOEAs to their particular application scenario.

One motivation for this component-wise view of MOEAs is the development of software frameworks that help practitioners apply and adapt MOEAs to their own application scenarios. Unfortunately, the design of most MOEA frameworks focuses on applying existing MOEAs to new scenarios, rather than on flexibly combining their components to produce new designs [15–17]. Even MOEA frameworks that allow the combination of algorithmic components from different MOEAs [14] are limited to MOEAs that are structurally similar, for example, based on the traditional *fitness* and *diversity* components. More recent MOEAs that differ from this template, such as HypE [5] and SMS-EMOA [6], cannot be instantiated from algorithmic components through such frameworks. We see two reasons behind this lack of flexibility. First, the analysis of MOEA components has relied on how the algorithms were described by their original authors, and only few works try to generalize functionally equivalent concepts of MOEAs into broader concepts [11, 14, 18, 19]. Second, a high degree of flexibility in a software framework may be deemed undesired, since some configurations may produce unreasonable algorithm designs or the number of possible configurations may be too large for human designers.

In this paper, we propose a new conceptual view of MOEA components that allows instantiating, from the same algorithmic template, a larger number of MOEAs from the literature than existing MOEA frameworks. For example, we are able to instantiate at least six well-known MOEAs from the literature: MOGA [1], NSGA-II [2], SPEA2 [3], IBEA [4], HypE [5], and SMS-EMOA [6]. More importantly, our framework allows to produce a large number of novel MOEA designs that are, in principle, reasonable from a human designer point of view. This is achieved by reformulating the traditional distinction between fitness and diversity components [11, 14] as preferences composed by set-partitioning, quality and diversity metrics [19]. In addition, different preferences may be used for mating and environmental selection. Our proposal also formalizes the distinction between internal and external populations and archives, which allows us to describe, using alternative options for the same components, algorithms as different as SPEA2 and SMS-EMOA. Our proposal is implemented in a

L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle are with the IRIDIA laboratory at CoDE, Université Libre de Bruxelles, 1050, Belgium.
email: {lteonaci,manuel.lopez-ibanez,stuetzle}@ulb.ac.be

This is the final draft version accepted by IEEE Transactions on Evolutionary Computation on 27 August, 2015. DOI: 10.1109/TEVC.2015.2474158

This research has received funding from the COMEX project (P7/36) within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office. L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a FRIA fellow, a post-doctoral researcher, and a senior research associate, respectively.

software framework from which novel MOEA designs can be instantiated by properly selecting the values of the various algorithmic components and numerical parameters.

Following previous work on automatic design [20], we apply an offline automatic configuration method, irace [21], to our proposed framework, considering the various algorithmic components as categorical parameters to be set. In this sense, the configuration space searched by irace is actually a design space, where different MOEA components can be combined to generate unique and possibly novel MOEA designs.

We automatically design several MOEAs, called here AutoMOEAs, for several application scenarios. Our scenarios were selected to provide insights on several questions about MOEA design. The first question is whether the benchmark that guides the design process has a strong influence in the performance of the resulting design. Thus, we consider continuous optimization problems, in particular, two benchmark sets, DTLZ [10] and WFG [22], with two, three, and five objectives, since MOEAs have been primarily designed for these problems. Our results indicate that the best MOEA design depends strongly on which benchmark is used for the automatic design.

A second question in MOEA design is the trade-off between computationally expensive components and the quality of the results. Thus, we consider two different stopping criteria for the MOEAs: maximum number of function evaluations (FEs) and maximum runtime. By using these two setups, we are able to represent problems with computationally demanding function evaluations as well as problems where the computational overhead of MOEA components is relevant. Although the former is the typical setup considered in the MOEA literature, the conclusions obtained may not apply to the latter setup. This is demonstrated by the fact that the AutoMOEAs produced for each setup show remarkable differences. In most cases, for both setups, the AutoMOEAs are able to match, and often significantly surpass, the results obtained by the MOEAs from the literature, even after tuning their numerical parameters.

Finally, we study the differences between the AutoMOEAs obtained for the continuous optimization benchmarks and those obtained for various multi-objective combinatorial optimization problems. In a preliminary version of this work [23], we considered four multi-objective variants of the *permutation flow shop problem* (PFSP), varying the number and nature of the objectives. Since MOEAs from the literature were not originally devised for such problems, it is not surprising that we were able to generate AutoMOEAs that outperformed them. Nonetheless, the best MOEA designs differ enough from what is considered the state-of-the-art in the MOEA literature that we briefly comment the results here. These results and the remarkable differences between the MOEAs designed for continuous optimization and those designed for combinatorial optimization provide further motivation for the automatic design of MOEAs.

Our overall goal is to empirically demonstrate that it is possible to find novel MOEA designs that outperform the MOEAs from the literature by means of automatically configuring the components of our proposed MOEA template. Another objective of this paper is to reformulate diverse MOEAs into a common conceptual view that generalizes functionally-equivalent algorithmic components and describes the available design choices. A final objective is to investigate which design choices (instead of which monolithic MOEAs) are more appropriate for the various scenarios described above.

Algorithm 1 AutoMOEA template proposed in this work.

```
1: pop ← Initialization ()
2: if type(pop_ext) != none
3:     pop_ext ← pop
4: repeat
5:     pool ← BuildMatingPool (pop)
6:     pop_new ← Variation (pool)
7:     pop_new ← Evaluation (pop_new)
8:     pop ← Replacement (pop, pop_new)
9:     if type(pop_ext) = bounded then
10:        pop_ext ← Replacement_Ext (pop_ext, pop_new)
11:    else if type(pop_ext) = unbounded then
12:        pop_ext ← pop_ext ∪ pop
13: until termination criteria met
14: if type(pop_ext) = none
15:     return pop
16: else
17:     return pop_ext
```

The remainder of this paper is structured as follows. Section II presents in detail our component-wise MOEA framework. We present empirical results and discussion for continuous and combinatorial problems in Sections III and IV, respectively, and conclude in Section V.

II. A Template for Designing MOEAs

The AutoMOEA template we propose for instantiating and designing MOEAs is shown in Algorithm 1. As we will explain below, from this template we can not only instantiate many well-known MOEAs, but also many new ones that have never been explored so far. The proposed template is based on the view that MOEAs can be seen as extensions of traditional single-objective EAs such as genetic algorithms [1–5], evolution strategies [6, 24], or differential evolution [25], extended by algorithm components that deal with the multi-objective aspects. In our template, we encapsulate the lower-level procedures in components such as Variation, which applies variation operators to the mating pool (pool). Additional components for tackling multi-objective problems in the Pareto sense are encapsulated in the BuildMatingPool and Replacement procedures (see lines 5 and 8 of the template, respectively). In addition, MOEAs often use their internal population (pop) as a bounded-size approximation to the Pareto front (i.e., as an archive) and many of them add the possibility of keeping an external (bounded or unbounded) archive ($\mathtt{pop}_{\mathtt{ext}}$).

Next, we describe the multi-objective components, how to instantiate some well-known MOEAs from our template, and how our approach differs from existing frameworks.

A. Preference relations in mating and replacement

The mating and environmental selection procedures performed by MOEAs depend on ranking solutions according to a preference relation. In general, given two solutions θ_1 and θ_2 and a metric Ψ to be minimized (without loss of generality), a relation $\prec_\Psi$ is defined as $\theta_1 \prec_\Psi \theta_2 \iff \Psi(\theta_1) < \Psi(\theta_2)$. In

Table I
MAIN ALGORITHMIC COMPONENTS OF AUTOMOEA

Component	Parameters
Preference	⟨ Set-partitioning, Quality, Diversity ⟩
BuildMatingPool	⟨ Preference$_{Mat}$, Selection ⟩
Replacement	⟨ Preference$_{Rep}$, Removal ⟩
Replacement$_{Ext}$	⟨ Preference$_{Ext}$, Removal$_{Ext}$ ⟩

Table II
ALGORITHMIC COMPONENT OPTIONS AVAILABLE FOR AUTOMOEA

Component	Domain
Set-partitioning	none (—) dominance count dominance rank dominance strength dominance depth dominance depth-rank
Quality	none (—) binary indicator (I_ϵ or I_H^-) exclusive hypervolume contribution (I_H^1) shared hypervolume contribution (I_H^h)
Diversity	none (—) niche sharing k-th nearest neighbor (kNN) crowding distance
Selection	deterministic tournament (DT) stochastic tournament (ST) random
Removal	sequential one-shot
type (pop)	{ fixed-size, bounded }
type (pop$_{\text{ext}}$)	{ none, bounded, unbounded }

our MOEA template, solutions are ranked according to *general preference relations* [19] defined as a sequence of three lower-level preference relations: a set-partitioning relation, a quality metric and a diversity metric. First, a *set-partitioning relation* ranks solutions in a Pareto-compliant way, but does not distinguish between nondominated solutions. These correspond to traditional fitness components such as dominance depth (NSGA-II) and dominance strength (SPEA2). Because of the nature of these metrics, multiple solutions are often equally ranked. Therefore, at a second step, we use refinement relations based on Pareto-compliant *quality indicators* to discriminate between equally ranked solutions. We apply these refinement relations to the equally ranked partitions, but we do not alter the cross-partition ranks. This means that if a solution θ_1 is ranked better than another solution θ_2 according to a set-partitioning relation, then a refinement relation would never contradict this. The third type of relation is based on *diversity metrics*. These metrics do not focus on Pareto dominance, but rather on allowing MOEAs to maintain a population that represents different trade-offs between the objectives. The structure for these general preference relations is encapsulated in component Preference (Table I). Additionally, any of the three components of Preference might be empty (*none*), which means that the next component takes effect. If all three components are empty, the ranking is random.

The options available for composing preference relations in our template are given in Table II. This formulation of preference relations provides flexibility when designing MOEAs for different real-world optimization scenarios. For instance, set-partitioning relations may provide enough convergence given a problem with few objectives and for which the computation overhead of quality metrics may be deemed excessive. On the other extreme, given a problem with a large number of objectives, the number of incomparable candidate solutions may be too large such that set-partitioning relations do not provide enough convergence pressure. In other cases, the time required for computing the quality metrics may be negligible compared with the cost of evaluating candidate solutions. We also remark that the original proposal by Zitzler et al. [19] allows for more complex preference models (e.g. using multiple refinement relations based on quality indicators in a sequence), but our proposal here suffices to replicate most MOEAs from the literature and allows defining new preference relations in a flexible and consistent way. Furthermore, the general preference relations we adopt overcome the problems faced by existing frameworks when instantiating some recent MOEAs such as SMS-EMOA and HypE. In particular, these algorithms include components that simultaneously account for convergence and diversity, and hence do not fit the traditional separation between fitness and diversity metrics [11, 14].

The mating and environmental selection procedures (BuildMatingPool and Replacement) are defined in dependence of the general preference relations described above. BuildMatingPool comprises a preference relation Preference$_{Mat}$ and a selection method Selection as shown in Table I. The methods for selection we implement for this work are listed in Table II. In particular, the *tournament selection* method can be used either deterministically or stochastically. While deterministic tournaments always favor the best individual according to Preference$_{Mat}$, stochastic tournaments choose, with a probability γ, the solution preferable according to Preference$_{Mat}$. *Random selection* chooses individuals with uniform probability, and so no preference relation is used.

Component Replacement is composed of a preference relation Preference$_{Rep}$ and a removal policy Removal. Table II lists the removal policy options we implement. *Sequential* (or *iterative* [5]) *removal* [26] discards one solution at a time and recomputes the preference relation before the next solution is discarded. *One-shot removal* [5] computes preference relations once and discards the worst solutions altogether. Although the information provided by the sequential removal policy is more accurate, this policy is computationally more demanding, which may compromise its performance in time-constrained scenarios. Additionally, if the number of offspring per generation λ is set to 1 (steady-state selection), the different alternatives for component Removal become equivalent.

The ability of using different preference relations for mating and environmental selection is, in fact, another novel feature of our template over the templates implemented by existing MOEA frameworks. Although earlier MOEAs did not foresee the benefits of this design choice, more recent algorithms such

Table III
DIFFERENT TYPES OF ARCHIVES AVAILABLE FOR AUTOMOEA

Archive Type	μ_0	**FS**	**MC**	**DS**	**EP**	**Replacement**
type ($\mathtt{pop}$) = *fixed-size*	μ	+	μ	+	+	Replacement
type ($\mathtt{pop}$) = *bounded*	$\mu \cdot \mu_r$	−	μ	−	+	Replacement
type ($\mathtt{pop}_{\mathtt{ext}}$) = *bounded*	$\lvert\mathtt{pop}_0\rvert_{\prec}$	−	N_{ext}	−	−	Replacement*Ext*
type ($\mathtt{pop}_{\mathtt{ext}}$) = *unbounded*	$\lvert\mathtt{pop}_0\rvert_{\prec}$	−	∞	−	−	—

FS: *fixed-size*; MC: *maximum capacity*; DS: *dominated solutions*; EP: *part of the evolutionary process*; $\lvert\mathtt{pop}_0\rvert_{\prec}$: *size after removing dominated solutions*

as SMS-EMOA and HypE already make use of it to minimize the computational overhead of quality metrics such as the hypervolume. From a more general point of view, the flexibility provided by this design choice can be used to improve the effectiveness of the algorithm in several other ways, e.g., by combining exploitative and explorative strategies.

B. Population and archives

A population is a set of individuals, dominated and nondominated alike, that are subject to the evolutionary process. By contrast, an archive is an auxiliary set used for storing nondominated solutions found during a single run of the algorithm. In our template, we model an archive as a generalized population that may (i) only keep nondominated solutions, (ii) have unbounded capacity, and/or (iii) take part in the evolutionary process. We provide two archives for MOEAs to use: an *internal* archive $\mathtt{pop}$ that takes part in the evolutionary process and can be used as a regular population or as a bounded-size archive, and an *external* archive $\mathtt{pop}_{\mathtt{ext}}$ that does not participate in the evolutionary process.

All options implemented here for $\mathtt{pop}$ and $\mathtt{pop}_{\mathtt{ext}}$ are listed in Table II, and we present a summary of their characteristics in Table III. If $\mathtt{pop}$ is set to have a fixed size (*type* ($\mathtt{pop}$) = *fixed-size*), then $\mathtt{pop}$ behaves like a regular population of size μ and may contain dominated solutions. Otherwise, $\mathtt{pop}$ is used as a bounded internal archive (*type* ($\mathtt{pop}$) = *bounded*), accepting only nondominated solutions until its maximum capacity μ is reached. Once the maximum capacity is reached, a replacement is carried out by component Replacement mimicking an archive bounding method [27]. When used as a bounded internal archive, $\mathtt{pop}$ presents two other important characteristics. First, the initial number of solutions μ_0 in $\mathtt{pop}$ is controlled by a numerical parameter $\mu_r \in [0.1, 1]$, i.e., $\mu_0 = \mu_r \cdot \mu$. Second, the preference relation used by this bounded internal archive does not use set-partitioning relations, since all solutions are already nondominated. These characteristics make $\mathtt{pop}$ flexible enough to allow us instantiate archive-based algorithms such as PAES [24], as well as algorithms such as SPEA2, which are population-based but use an archive that interferes in the evolutionary process [11].

By contrast, the external archive $\mathtt{pop}_{\mathtt{ext}}$ can be used by MOEAs in three different ways. First, as traditionally used in the literature, the archive can be *bounded* to a maximum capacity N_{ext} and once the maximum capacity is reached, a replacement is carried out (see also line 10 of Algorithm 1). Component Replacement*Ext* is defined analogously to Replacement, but with its own Preference*Ext* and Removal*Ext* options (see Table I). Since all solutions kept by the archive are nondominated, Preference*Ext* does not use a set-partitioning relation. For application scenarios where the number of nondominated solutions is low, MOEAs can either use an archive without capacity constraints, i.e., *type* ($\mathtt{pop}_{\mathtt{ext}}$) = *unbounded*, or not use an external archive at all, i.e., *type* ($\mathtt{pop}_{\mathtt{ext}}$) = *none*.

The ability of using a different preference relation for maintaining the external archive opens a number of possibilities for MOEA designers. For example, the preference relations used for mating and replacement of the (internal) population could lack the limit-stable property [26] (that is, even given infinite time, the population will not converge to a stable set) in order to promote exploration, while the external preference relation could be both limit-stable and limit-optimal such that, eventually, the external archive will converge to an optimal (bounded) archive [26].

C. Differences from existing frameworks

A number of MOEA *software frameworks* can be found in the literature [14–17]. Together, they have made the application of MOEAs to new scenarios much easier by establishing a clear separation between problem-dependent and independent components. These software frameworks include implementations of the most popular MOEAs, however, their algorithmic components are often not directly inter-changeable. Thus, designing a novel MOEA by combining existing components in novel ways using most of these frameworks is not a straightforward task that can be done in an automatic manner, since they were not created with this goal in mind.

The framework that most closely resembles our proposed template is ParadisEO-MOEO [14], which provides a "unified model" for MOEAs that allows both the instantiation and the design of novel MOEAs using a template. However, the generality of the template used by ParadisEO-MOEO is limited when compared to the template we present here in at least four major aspects. First, ParadisEO-MOEO uses the traditional approach of preference relations built solely from fitness and diversity components, which is insufficient to represent complex preference relations as we do in this work. Second, these fitness and diversity components cannot be used with different behaviors for mating and replacement. This is highlighted by the fact that the default implementation of SPEA2 in ParadisEO-MOEO is not instantiated via their template, but requires an external archiver specifically designed for SPEA2 to work. Moreover, using the template provided by ParadisEO-MOEO, one cannot instantiate or design algorithms that use different preference relations for mating and replacement, such as HypE or SMS-EMOA. Third, our internal population definition is a unique contribution, since it allows us to instantiate both population-based and archive-based MOEAs, such as PAES. Finally, many of the components we use in this work are not available in ParadisEO-MOEO, such as the I^1_H and I^h_H quality indicators, or are only partially available, such as the sequential removal policy. Altogether, these aspects limit the number of MOEAs that can be represented using the template provided by ParadisEO-MOEO, and hence the number of possible designs one can instantiate through it.

Table IV
INSTANTIATION OF MOEAS FROM OUR PROPOSED TEMPLATE.

	BuildMatingPool				Replacement			
Algorithm	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal
MOGA [1]	DT	dom. rank	—	niche-sharing	—	—	—	generational
NSGA-II [2]	DT	dom. depth	—	crowding dist.	dom. depth	—	crowding dist.	one-shot
SPEA2 [3]	DT	dom. strength	—	kNN	dom. strength	—	kNN	sequential
IBEA [4]	DT	—	binary indicator	—	—	binary indicator	—	one-shot
HypE [5]	DT	—	I_H^h	—	dom. depth	I_H^h	—	sequential
SMS-EMOA [6]	random	—	—	—	dom. depth-rank	I_H^1	—	—

(All MOEAs above use *type* ($\texttt{pop}$) = *fixed-size* and *type* ($\texttt{pop}_{\texttt{ext}}$) = *none*; in addition, SMS-EMOA uses $\lambda = 1$)

The novelty of our proposal lies in the algorithmic template and the definition of its components, rather than in the software implementing them. In fact, the implementation of most of our algorithmic components is taken from the ParadisEO-MOEO [14], PISA [15], and PaGMO [16] frameworks, although we substantially modified them to work together within our algorithmic template. Nonetheless, it would be feasible to implement our proposed template within any of these frameworks, and we would like to encourage others to do so.

D. Standard MOEAs instantiated via the AutoMOEA template

By carefully selecting the values of each algorithmic component, we can instantiate many well-known MOEAs from the literature using the proposed template. Table IV shows how to instantiate the six MOEAs we consider in this work, which we have selected because of their relevance in the literature. We next describe each of these MOEAs.

1) MOGA [1]: uses *dominance ranking* as set-partitioning relation and the *niche sharing* diversity metric based on the objective values of the individuals (also known as *fitness sharing*). Being one of the earliest MOEAs, MOGA does not use elitism, which can be implemented as a *generational* removal policy. Although we do not include this option for component Removal in our experiments with the framework, we use it in MOGA for fidelity to the original proposal.

2) NSGA-II [2]: uses *dominance depth* (also known as dominance sorting) for set-partitioning, as well as the *crowding distance* diversity metric. The same preference relation is used for mating and replacement. In addition, NSGA-II uses *one-shot removal*.

3) SPEA2 [3]: uses *dominance strength* as its set-partitioning relation. Concerning diversity, SPEA2 is the first algorithm to use different metrics for mating and replacement. For mating, SPEA2 considers the distance between each individual and its k-th nearest neighbor. Originally, the default is $k = \lfloor\sqrt{\mu}\rfloor$, where μ is the population size. In our implementation, we allow k to be set either according to this default definition or as a numerical parameter. For replacement, SPEA2 considers the distance from each individual to all of the remaining population, and uses *sequential removal*. Finally, the external archive of SPEA2 behaves like a population, as noted by other authors [11], hence it is equivalent to our fixed-size $\texttt{pop}$, and its internal population is equivalent to $\texttt{pop}_{\texttt{new}}$ in our template.

4) IBEA [4]: ranks solutions based on *binary quality indicators*. Concretely, IBEA computes the pairwise values of a given binary quality indicator for the whole population. Then, the preference of each individual is given by the aggregation of its binary values w.r.t. to the rest of the population. Since quality indicators sometimes account for diversity, the original proposal of IBEA does not present a diversity metric. The most used binary indicators for IBEA are the additive epsilon indicator ($I_{\epsilon+}$) and the hypervolume difference (I_H^-) [28], which are the options we implement here (see Table II). For replacement, IBEA also uses *sequential removal*.

5) HypE [5]: searches the solution space directed by the *shared hypervolume contribution* (I_H^h) of the individuals. This quality indicator measures the volume of the subspace an individual exclusively dominates, plus shares of the volumes that it jointly dominates with up to other h individuals of the population. Due to its computational overhead, HypE uses a Monte Carlo simulation to estimate these values for problems with more than three objectives. For mating selection, HypE uses the I_H^h indicator with $h = \mu$, and no set-partitioning relation or diversity metric. For replacement, HypE uses dominance depth as set-partitioning, the shared hypervolume contribution as refinement, sequential removal, and h equal to the number of solutions that have to be discarded from the first partition that does not fit in the new population.

6) SMS-EMOA [6]: is a (μ+1)-ES algorithm that uses the *exclusive hypervolume contribution* (I_H^1) for replacement (a particular case of the I_H^h where $h = 1$). Since it is computationally demanding, SMS-EMOA uses a set-partitioning relation called *dominance depth-rank* to reduce the number of times this indicator is employed. In dominance depth-rank, if dominance depth returns more than one partition, dominance rank is used to refine the first partition that does not fit the new population. The exclusive hypervolume contribution is only used otherwise.

To ensure the correctness of our implementation, we have empirically verified that its performance matches the original implementations of the MOEAs provided by the authors (or when not available, by third-party ones). In the following experiments, we compare these six standard MOEAs with novel MOEAs instantiated from our template. Our analysis covers several scenarios, ranging from continuous to combinatorial, presented in Sections III and IV, respectively.

III. Automatically Designing MOEAs for Continuous Optimization Problems

Our experimental investigation has two main goals. The first is to assess how automatically designed MOEAs (we call them AutoMOEAs in this paper) perform compared to several standard MOEAs that can be instantiated from our framework. Second, we want to investigate how much the structure of the AutoMOEAs varies depending on the benchmark and the number of objectives considered.

The benchmark problems that have been considered at the design time of an algorithm may implicitly or explicitly bias the algorithm design. Here we study this effect by considering two different benchmark sets, the DTLZ set [10] and the WFG set [22]. The DTLZ benchmark set was originally proposed as an improvement over the ZDT benchmark [9] to allow scalability in the number of objectives considered. Since our work focuses on unconstrained problems, we limit the DTLZ benchmark set to problems DTLZ1–DTLZ7. Although this benchmark has been widely used in the literature, some important characteristics of MOEAs are not assessed, particularly their performance on flat landscapes, deceptive, and nonseparable problems. Therefore, we also consider the WFG set, which was proposed later than many of the MOEAs we use in this work. Here, we use the nine exemplary functions proposed by the authors, WFG1–WFG9. Following [22], we set the ratio between position and distance variables to 1/6.

Each benchmark set is used with two, three and five objectives. We separate between different number of objectives as it is known before running an algorithm and, obviously, an algorithm configuration that performs well for a low number of objectives (e.g. 2 or 3) need not perform well for more objectives (e.g. 5). We then design AutoMOEAs for each of the six scenarios obtained from the combinations of benchmark set (DTLZ and WFG) and number of objectives (2, 3, and 5).

A. AutoMOEA design setup

The parameter space we use for the automatic design of the MOEAs is given in Tables I, II and V, where p_c and $p_m \in [0,1]$ respectively stand for the probability of applying crossover to a given pair of individuals, and the probability of applying mutation to a given individual. We use the SBX crossover operator and polynomial mutation, which have associated numerical parameters η_c and η_m (the distribution indices). Furthermore, different mutation schemes can be used by MOEAs for real-parameter optimization [29]. Here, we implement two options: (i) *bitwise*, which sets the mutation probability per variable such that on average one variable is mutated per individual chosen for mutation; and (ii) *fixed*, where the mutation probability per individual mutated is set by the user as a parameter $p_v \in [0.01, 1]$. We do not include more evolutionary operators and schemes to focus on the high-level multi-objective components that characterize MOEAs.

As the automatic offline parameter configuration tool we use irace [21], which has been adapted to handle multi-objective algorithms by using the hypervolume indicator as follows. First, the candidates generated by irace are given a maximum number of function evaluations (FE). Following [5], we set this number to 10 000 FEs. Then we assess the quality of the approximation fronts produced by each candidate by computing their *hypervolume relative percentage deviation* (I_H^{rpd}). Concretely, given a reference Pareto front P and an approximation front A, $I_H^{rpd}(A) = (I_H(P) - I_H(A)) \,/\, I_H(P)$. To compute the I_H metric, we discard all solutions with objective values worse than the upper bound $\mathbf{u} = [10]^M$, and use the reference point $\mathbf{r} = [11]^M$. Concerning the reference fronts we use in this work, we have initially generated 1 000 random Pareto optimal solutions for each problem size using the methodology described in the original papers where the benchmarks were proposed. However, we noticed that many of these initial sets presented poor hypervolume since solutions were not well spread. We improved these reference sets by adding nondominated solutions found by running traditional MOEAs for 100 000 FEs using their default parameters 10 times on each problem instance, for all problems where we identified this issue (WFG1–9 and DTLZ4).

Table V
Parameter space for tuning all MOEAs for continuous optimization.

Parameter	$\mu = \lvert\texttt{pop}\rvert$	$\lambda = \lvert\texttt{pop}_{\texttt{new}}\rvert$	p_c, p_m	η_c, η_m
Domain	$\{10, 20, \ldots, 100\}$	1 or $\lambda_r \cdot \mu$ $\lambda_r \in [0.1, 2]$	$[0, 1]$	$\{1, \ldots, 50\}$

Condition	**Additional parameter**	**Domain**
type (pop) = *bounded*	μ_r	$[0.1, 2]$
type ($\texttt{pop}_{\texttt{ext}}$) = *bounded*	N_{ext}	$\{100, 300, 500\}$
mutation scheme = *fixed*	p_v	$[0.01, 1]$
Selection = *DT*	*tournament size*	$\{2, 4, 8\}$
Selection = *ST*	γ	$[0.6, 0.9]$
Quality = *binary indicator*	*indicator*	I_ϵ, I_H^-
Diversity = *sharing*	σ_{share}	$[0.1, 1]$
Diversity = *kNN* (as part of Mating)	k_{method}	$\{default, k\}$ $k \in \{1, \ldots, 9\}$

Experiments are run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB cache size under Cluster Rocks Linux version 6.0/CentOS 6.3. To keep our experiments feasible in time, we limit the maximum runtime of a single run to 10 minutes. For all configurations that correspond to the standard MOEAs, this time limit is high enough to perform all 10 000 FEs. In fact, we have empirically verified that, on average, 85% of the candidates produced by irace use all FEs allowed within this time limit. The few configurations that do not use all available FEs are typically the ones that combine many computationally costly components at once, e.g., an external archive replacement based on the shared hypervolume contribution, nearest neighbor diversity, and sequential removal. For these configurations, we assess their performance based on the approximation fronts they return when reaching the time limit. Given the large search space for designing the AutoMOEAs, we give irace a tuning budget of 20 000 runs. In our computational setup, the wall-clock time used by irace is equivalent to designing a MOEA over the weekend.

Table VI
PARAMETERS SELECTED BY IRACE FOR THE AUTOMOEAS.

	BuildMatingPool				Replacement				Replacement$_{Ext}$			Numerical							
	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	Quality	Diversity	Removal	μ	μ_r	λ	λ_r	p_c	p_m	η_c	η_v
DTLZ 2-obj	DT (8)	—	—	crowding	depth-rank	I_ϵ	sharing	—	—	crowding	sequential	100	0.85	1	—	0.63	0.67	35	25
DTLZ 3-obj	DT (8)	depth-rank	I_ϵ	kNN	rank	I_H^1	sharing	—	I_H^1	—	sequential	80	0.77	1	—	0.58	0.63	2	15
DTLZ 5-obj	DT (8)	rank	I_H^1	crowding	depth	I_H^1	—	—	I_ϵ	crowding	one shot	40	—	1	—	0.35	0.62	42	5
WFG 2-obj	DT (8)	rank	—	crowding	depth-rank	I_H^1	—	—	I_H^h	crowding	one shot	20	—	1	—	0.11	0.33	31	11
WFG 3-obj	DT (4)	count	I_H^1	crowding	strength	I_H^1	sharing	sequential	I_H^1	kNN	sequential	10	—	—	0.86	0.11	0.49	39	13
WFG 5-obj	DT (8)	count	I_H^h	crowding	—	I_H^1	—	sequential	I_H^h	crowding	one shot	30	—	—	1.07	0.71	0.66	34	12

(All AutoMOEAs use the bitwise mutation scheme, and *type* ($\texttt{pop}_{\texttt{ext}}$) = *bounded* with $N_{\text{ext}} = 500$, except for Auto$_{\text{W2}}$, for which $N_{\text{ext}} = 300$. In addition, all but Auto$_{\text{D2}}$ and Auto$_{\text{D3}}$ use *type* ($\texttt{pop}$) = *fixed-size*, and all but Auto$_{\text{W3}}$ and Auto$_{\text{W5}}$ use steady-state replacement, i.e., λ = 1)

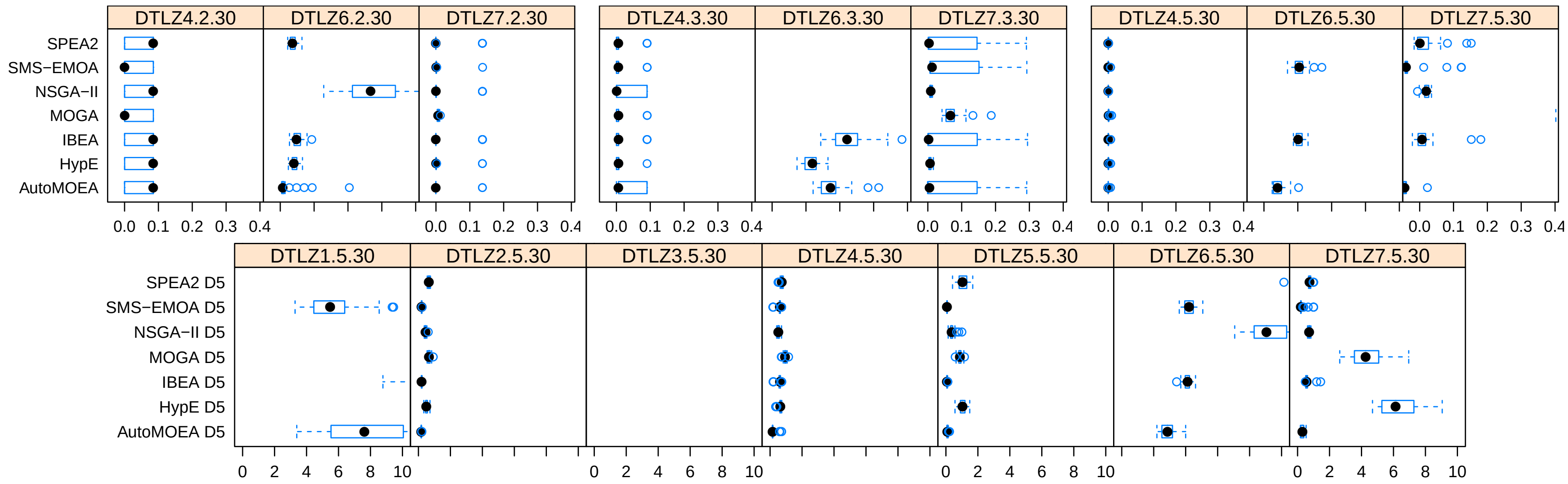

Figure 1. Performance boxplots for all algorithms on selected 30-variable DTLZ benchmark problems. Top: I_H^{rpd} for problems with 2, 3, and 5 objectives (from left to right, respectively). Bottom: I_ϵ for 5-objective problems.

B. Performance comparison setup

In the context of continuous optimization, benchmark sets try to be as heterogeneous as possible in order to capture as many potential features of unknown real-world problems as possible. As a consequence of how such benchmarks are designed, partitioning them into disjoint sets of functions would result in a training set for tuning that is not representative of the test set. Here, we go a step further of what is the standard in continuous optimization benchmarking, and the functions used in the tuning and test always differ in the number of variables (n_{var}). Concretely, we use $n_{\text{var}} \in \{20, \ldots, 60\} \setminus n_{\text{testing}}$ for tuning, where $n_{\text{testing}} = \{30, 40, 50\}$ are the sizes we reserve for testing. We also take the precaution of differentiating the effect of tuning the numerical parameters of MOEAs from the effect of designing novel MOEAs. Although the literature proposes default numerical parameters per benchmark [5, 6, 10, 22], we have found that major performance improvements can be achieved by tuning these numerical parameters[1]. Hence, in the remainder of the paper, all standard MOEAs have been tuned for the corresponding scenario, using the same numerical parameter space as for AutoMOEA (Table V)[2]. We prefer this approach of comparing standard MOEAs to the AutoMOEAs as there may be interactions between numerical and structural parameters that change the MOEA design and therefore transferring numerical parameter settings from one to another algorithm may bias the algorithm comparisons. For each of these tunings, we also give irace a tuning budget of 20 000 runs.

To compare different algorithms, we first run each algorithm 25 times on the testing benchmarks. In addition to the hypervolume RPD, we also compute the additive ϵ-indicator (I_ϵ) of the approximation sets w.r.t. the reference fronts. The comparison is done visually by means of boxplots, and analytically through rank sums. To assess statistical significance, we adopt Friedman's non-parametrical test and its associated post-hoc method at 99% confidence. For brevity, we omit the I_ϵ results when they agree with the I_H^{rpd} ones. The full set of results is provided as supplementary material [30].

C. Results and discussion

The designs of the AutoMOEAs selected by irace for each of the scenarios we consider are shown in Table VI. All AutoMOEAs use replacement preference relations comprising set-partitioning and indicator-based components (very often the I_H^1), as well as large external archives. Surprisingly, the only exception to this pattern is AutoMOEA$_{\text{W5}}$, which does not use any set-partitioning metric for replacement. Concerning the external archive, the number of nondominated solutions in these problems is large, demanding an external archive, but prohibiting an unbounded one. In particular, most

[1] For brevity, this analysis is provided as supplementary material [30], together with the tuned configurations of all MOEAs.

[2] Configurations that would change the MOEA design as defined by Table IV are not allowed when tuning numerical parameter settings.

Table VII
SUM OF RANKS DEPICTING THE PERFORMANCE OF MOEAS ON CONTINUOUS SETS ACCORDING TO THE I_H^{rpd}. ΔR IS THE CRITICAL RANK SUM DIFFERENCE FOR FRIEDMAN'S TEST WITH 99% CONFIDENCE.

DTLZ			WFG		
2-obj $\Delta R = 126$	3-obj $\Delta R = 127$	5-obj $\Delta R = 107$	2-obj $\Delta R = 169$	3-obj $\Delta R = 130$	5-obj $\Delta R = 97$
Auto$_{\mathrm{D2}}$ **(1339)**	**Auto$_{\mathrm{D3}}$** **(1500)**	**Auto$_{\mathrm{D5}}$** **(1002)**	**Auto$_{\mathrm{W2}}$** **(1692)**	**Auto$_{\mathrm{W3}}$** **(1375)**	**Auto$_{\mathrm{W5}}$** **(1170)**
SPEA2$_{\mathrm{D2}}$ (1562)	IBEA$_{\mathrm{D3}}$ (1719)	SMS$_{\mathrm{D5}}$ (1550)	SPEA2$_{\mathrm{W2}}$ (2097)	SMS$_{\mathrm{W3}}$ (1796)	SMS$_{\mathrm{W5}}$ (1567)
IBEA$_{\mathrm{D2}}$ (1940)	SMS$_{\mathrm{D3}}$ (1918)	IBEA$_{\mathrm{D5}}$ (1867)	NSGA-II$_{\mathrm{W2}}$ (2542)	IBEA$_{\mathrm{W3}}$ (1843)	IBEA$_{\mathrm{W5}}$ (1746)
NSGA-II$_{\mathrm{D2}}$ (2143)	HypE$_{\mathrm{D3}}$ (2019)	SPEA2$_{\mathrm{D5}}$ (2345)	SMS$_{\mathrm{W2}}$ (2621)	SPEA2$_{\mathrm{W3}}$ (2600)	SPEA2$_{\mathrm{W5}}$ (2747)
HypE$_{\mathrm{D2}}$ (2338)	SPEA2$_{\mathrm{D3}}$ (2164)	NSGA-II$_{\mathrm{D5}}$ (2346)	IBEA$_{\mathrm{W2}}$ (2777)	NSGA-II$_{\mathrm{W3}}$ (3315)	NSGA-II$_{\mathrm{W5}}$ (3029)
SMS$_{\mathrm{D2}}$ (2406)	NSGA-II$_{\mathrm{D3}}$ (2528)	HypE$_{\mathrm{D5}}$ (2674)	HypE$_{\mathrm{W2}}$ (2851)	HypE$_{\mathrm{W3}}$ (3431)	MOGA$_{\mathrm{W5}}$ (4268)
MOGA$_{\mathrm{D2}}$ (2970)	MOGA$_{\mathrm{D3}}$ (2851)	MOGA$_{\mathrm{D5}}$ (2915)	MOGA$_{\mathrm{W2}}$ (4320)	MOGA$_{\mathrm{W3}}$ (4540)	HypE$_{\mathrm{W5}}$ (4373)

AutoMOEAs use a Preference$_{Ext}$ that combines quality and diversity metrics, a combination that has been shown to work well in some cases [19]. One pattern we also observe in these external archives is that the exclusive hypervolume contribution (I_H^1) indicator is always coupled with sequential removal, while the remaining indicators are used with one shot replacement. This is likely explained by the increased computational overhead incurred by the computation of the hypervolume and our use of a maximum time limit.

Two other design choices have been frequently selected, namely steady-state replacement ($\lambda = 1$) and the BuildMatingPool component. Steady-state replacement has been shown to lead to effective results when runtime is not too limited [13]. As for BuildMatingPool, all AutoMOEAs use eight-ary deterministic tournament (except for AutoMOEA$_{\mathrm{W3}}$ which uses four-ary tournaments), reflecting the need for convergence pressure that the problems demand. In addition, all MOEAs use crowding distance as diversity metric, the most extreme case being AutoMOEA$_{\mathrm{D2}}$, which relies solely on this metric when selecting for mating.

Despite these patterns, it is hard to establish general guidelines for selecting components when we consider a specific benchmark or a specific number of objectives. However, as we will discuss in more detail below, the I_H^{rpd} rank sum analysis given in Table VII shows that each of these AutoMOEA variants perform very well on the scenarios for which they were designed. This confirms that different scenarios demand different components, and that the component-wise design proposed here provides enough flexibility to meet this need. Next, we discuss the performance of the algorithms for each of the benchmarks considered in detail.

1) DTLZ benchmark: Although this benchmark has been extensively used in the literature, most of the results can be considered novel, as the number of variables we use is larger than traditionally adopted. For DTLZ2 and DTLZ5, this increase in the number of variables is not enough to make these functions difficult, and all MOEAs find approximation sets with I_H^{rpd} very close to zero. Conversely, functions DTLZ1 and DTLZ3 become so difficult that no MOEA is able to find solutions within the bounds we set. The I_H^{rpd} of the remaining functions are shown in Figure 1 (top), and we examine them individually. We remark that we zoom these boxplots on the [0,0.4] range as this is the actual area of interest for this indicator. For brevity, we show only boxplots of the 30-variable functions, but we remark that the results for the other problem sizes are consistent with these ones.

DTLZ4 is a function that presents bias, and MOEAs are sometimes unable to find well-spread approximation fronts. This explains the variance we observe on the 2-objective boxplots. Still, algorithms like SMS-EMOA and MOGA are able to perform well on most runs. Function DTLZ6 presents a different kind of bias, making it difficult for several MOEAs to converge to the actual fronts, specially as the number of objectives grows. This time the only MOEAs that maintain good performance in all scenarios are IBEA and the AutoMOEAs. Finally, DTLZ7 is a disconnected function that MOEAs are able to solve with two objectives, but that becomes much harder with five objectives. SMS-EMOA and the AutoMOEAs are the only algorithms that present high performance in all scenarios, but once more the AutoMOEAs find approximation fronts with lower I_H^{rpd} more often than SMS-EMOA. Overall, the rank sums achieved by the AutoMOEAs are much lower than those of the remaining algorithms as shown in Table VII, which considers all problems and all test sizes.

The results given by the I_ϵ indicator are mostly consistent with the ones provided by the I_H^{rpd}, except for the 5-objective problems shown in Figure 1 (bottom). As discussed for the I_H^{rpd}, the performance of all MOEAs in DTLZ1 and DTLZ3 is so poor that solutions lie outside the boundaries we pre-established. Only SMS-EMOA and AutoMOEA$_{\mathrm{D5}}$ are able to reach results on DTLZ1 inside these boundaries. Concerning DTLZ2, DTLZ4, and DTLZ5, even though all MOEAs find approximation sets with I_H^{rpd} close to zero, the I_ϵ tells us that only AutoMOEA$_{\mathrm{W5}}$, SMS-EMOA, and IBEA are able to converge to the actual fronts in functions DTLZ2 and DTLZ5, and only AutoMOEA$_{\mathrm{W5}}$ in DTLZ4. Another function where the performance of the MOEAs is worse according to the I_ϵ than according to the I_H^{rpd} is DTLZ6, which is explained by the difficulty of converging we have previously discussed. Finally, the general performance of all MOEAs according to the I_ϵ for DTLZ7 are actually better than according to the I_H^{rpd} ones, which can be explained by the disconnectedness of this problem. Overall, the I_ϵ rank sum analysis is consistent with the I_H^{rpd} analysis given in Table VII.

2) WFG benchmark: The performance of all MOEAs in the WFG problems is shown in Fig. 2. For brevity, we omit functions WFG4–WFG7 and WFG9 as we have noticed that the performance of all MOEAs is very similar on the WFG concave functions (WFG4–WFG9). We also remark that the boxplots of the I_H^{rpd} and the I_ϵ indicators are very similar, and for this reason we omit the latter. Concerning the functions depicted in Fig. 2, one can clearly see a separation between WFG1 and WFG2 from the remaining functions. These two convex problems pose difficulties for MOEAs to converge regardless of the number of objectives. As for the other group of problems, MOEAs are able to perform well both on 2

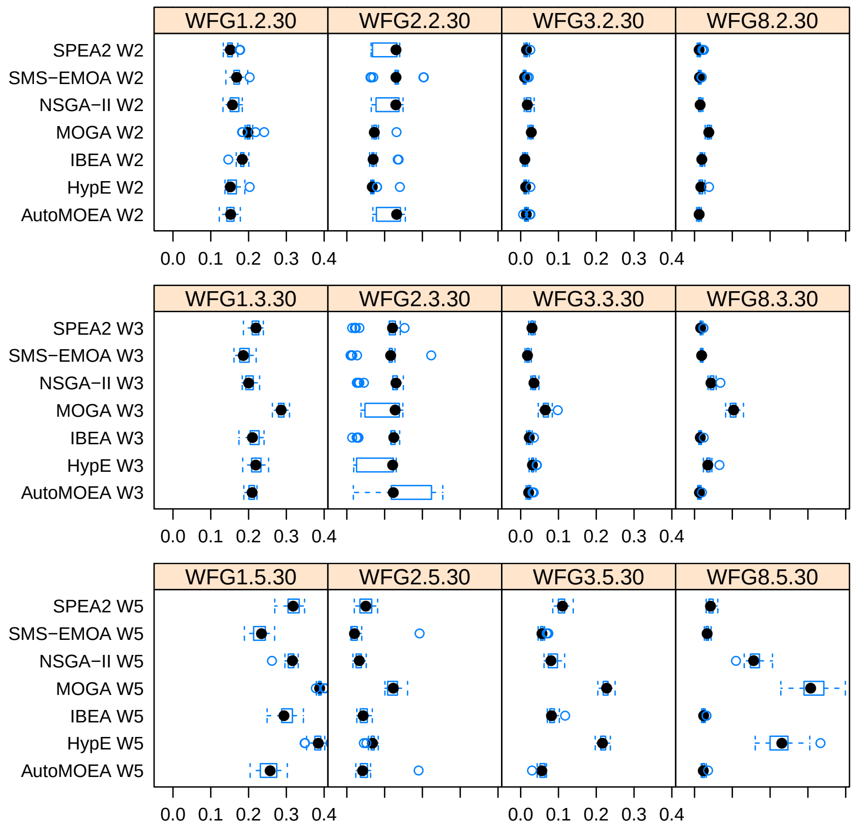

Figure 2. Hypervolume RPD on selected WFG benchmark problems with 30 variables. From top to bottom, 2, 3, and 5 objectives.

and 3 objectives, with the exception of MOGA. Looking into the 5-objective problems in more detail, we notice that MOGA, NSGA-II, and HypE are unable to converge to the actual fronts, and so is SPEA2 for WFG3, WFG5, WFG7, and WFG9. IBEA, SMS-EMOA, and AutoMOEA$_{\text{W5}}$ show the best performance on all problems except WFG3, where no MOEA is able to match the performance of AutoMOEA$_{\text{W5}}$.

The rank sum analysis of the I_H^{rpd} results given in Table VII confirms that the AutoMOEAs designed for the WFG benchmark display the best performance among all MOEAs considered, and so does the rank sum analysis of the I_ϵ indicator (see [30]). As for the remaining MOEAs, the rank sums of the algorithms are not consistent across different metrics, which indicates that some MOEAs favor convergence while others favor keeping a good trade-off between solutions (or are simply unable to find/preserve extreme solutions).

D. Experiments with a different stopping criterion

As shown by the results discussed above, standard MOEAs tend to perform better on the scenarios for which they have been properly tuned. Besides the benchmark set and the number of objectives considered, another major factor that affects the performance of algorithms is the stopping criterion used to terminate their runs. In continuous optimization, a maximum number of function evaluations (FE) is typically used because some applications present computationally costly FEs. As a result, algorithm designers tend to devise algorithms that are able to reach high-quality solutions with as few FEs as possible. Moreover, the time spent by the algorithms computing metrics or discarding solutions is not considered an issue in these scenarios and, hence, very fast and very slow algorithms are often considered equal. For instance, SMS-EMOA requires almost 10 minutes for executing 10 000 FEs in our computer environment, while IBEA terminates in seconds. However, in many practical situations the computational cost of the FEs may not be high enough to justify large computation times. In such scenarios, fast algorithms such as IBEA or NSGA-II could likely outperform slow ones such as SMS-EMOA by seeing many more solutions within a maximum runtime. By contrast, our design approach should be able to deal with such changes naturally. In this section, we investigate the structure of the AutoMOEAs generated and their performance relative to other MOEAs when all algorithms are given a maximum time limit of *one minute* CPU time. For brevity, we focus only on the AutoMOEAs designed and tested on the WFG benchmark for two, three and five objectives:

1) WFG, 2-objective (W2): AutoMOEA$_{\text{W2}}^{\text{1min}}$ uses a intermediate population size, a high number of offspring, and a large external archive based on crowding. Mating relies solely on crowding distance, whereas replacement is done based on dominance depth and the I_ϵ indicator. Both the external archive and the population removal are sequential. The structure of this algorithm is quite interesting because it combines computationally expensive components (large external archive with sequential removal) with computationally cheap ones (crowding distance, dominance depth, and the I_ϵ), adapting to the maximum runtime it is given. The I_H^{rpd} performance of AutoMOEA$_{\text{W2}}^{\text{1min}}$ is shown in Fig. 3 (top). We notice that the problem with most significant changes is WFG1, where the I_H^{rpd} of all algorithms is greatly improved. Although all MOEAs perform similarly, the rank sum analysis given in Table VIII indicates that AutoMOEA$_{\text{W2}}^{\text{1min}}$ performs better on a larger number of problems, both according to the I_H^{rpd} and the I_ϵ. Concerning the remaining MOEAs, IBEA and NSGA-II rank equivalently according to the I_H^{rpd}, but IBEA performs better than NSGA-II more often according to the I_ϵ.

2) WFG, 3-objective (W3): AutoMOEA$_{\text{W3}}^{\text{1min}}$ uses a large bounded internal archive, default number of offspring ($\lambda_r = 1.0$), and no external archive. The Preference$_{Mat}$ component is based on nearest neighbor density, while Preference$_{Rep}$ uses the exclusive hypervolume contribution (I_H^1) and sequential removal. This combination of components is coherent with this scenario, as the bounded internal archive and the I_H^1 provide convergence pressure at a low computational cost, and the nearest neighbor diversity has shown good results for SPEA2. Performance-wise, we see from Fig. 3 (second top-most plot) that again many algorithms present good performance according to the I_H^{rpd}. Overall, the rank sum analysis given in Table VIII indicates IBEA displays better I_H^{rpd} performance more often than AutoMOEA$_{\text{W3}}^{\text{1min}}$, but the opposite happens for I_ϵ. This is actually surprising, since AutoMOEA$_{\text{W3}}^{\text{1min}}$ has been tuned for the I_H^{rpd} and uses the I_H^1 indicator as its replacement mechanism, when IBEA uses the I_ϵ instead.

3) WFG, 5-objective (W5): As the previous scenarios have indicated, IBEA is quite effective when facing a runtime-constrained scenario. The structure of AutoMOEA$_{\text{W5}}^{\text{1min}}$ confirms this, as this algorithm presents the same exact components from IBEA, but can be considered a refinement of that algorithm as AutoMOEA$_{\text{W5}}^{\text{1min}}$ uses crowding diversity both for mating and environmental selection. The similarity between these algorithms reflects on the boxplots shown in Fig. 3 (two bottom-most plots), and is confirmed by the rank sums given in Table VIII. We see that the crowding distance metric is unable

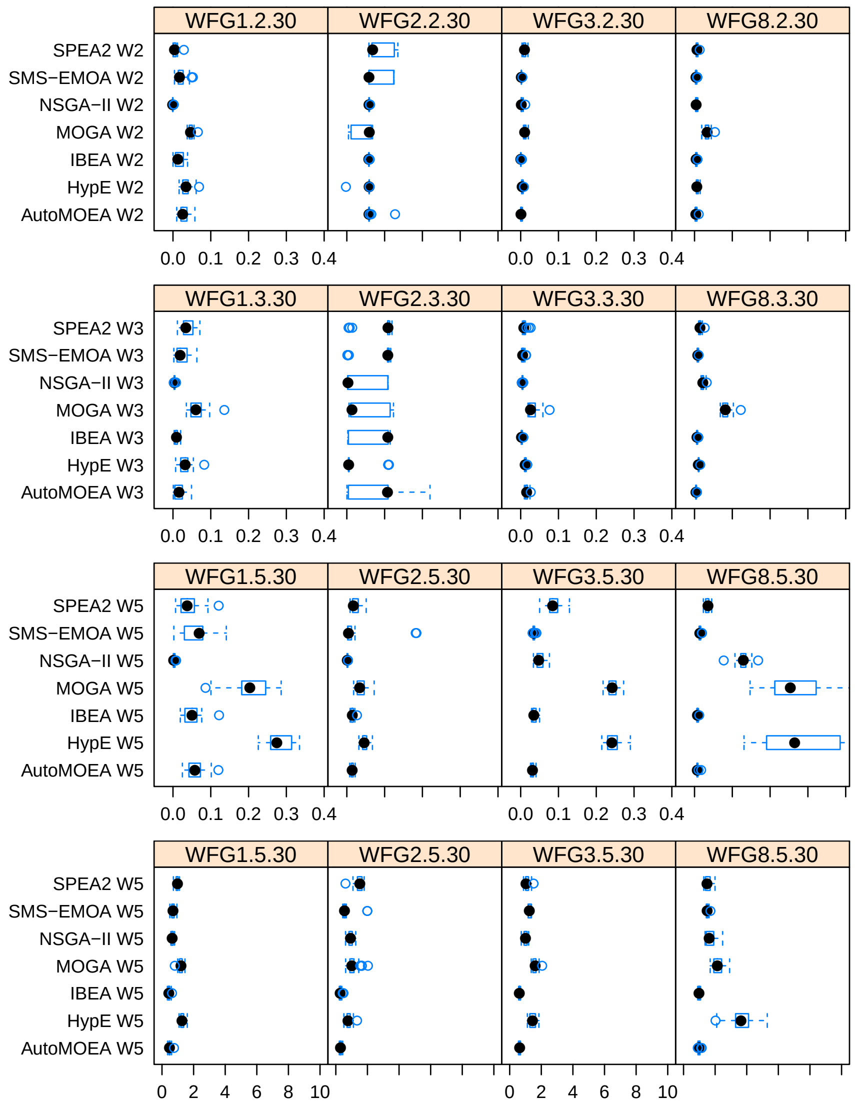

Figure 3. Performance of all MOEAs tuned for a maximum runtime on selected 30-variable WFG problems. From top to bottom, I_H^{rpd} results for 2, 3, and 5 objectives.The bottom-most plot depict the I_ϵ results on 5 objectives.

to improve the I_ϵ performance of a MOEA, but the I_H^{rpd} performance of AutoMOEA$^{1min}_{W5}$ is greatly improved. Concerning the performance of the remaining algorithms on WFG1, we see that the I_ϵ points to a better performance of all MOEAs than the I_H^{rpd}, except for NSGA-II and SPEA2. Additional experiments have shown that this is explained by the way MOEAs approach this function's front: they first converge to a small region of the objective space and, only later, they start spreading accross the Pareto front. We also observe the same discrepancy between the different metrics when we analyze MOGA on WFG2. This discrepancy holds for all MOEAs on WFG3, except two: SMS-EMOA, which now performs better on the I_H^{rpd}, and SPEA2, which now performs better on the I_ϵ. On the concave functions, represented by WFG8, the hypervolume-based SMS-EMOA and HypE present better performance on the I_H^{rpd} than on the I_ϵ, and so does SPEA2.

Overall, the results shown in this section have confirmed that the overhead incurred by MOEA components can greatly impair their efficiency when facing a problem that is not computationally expensive, but requires a constrained runtime.

E. Cross-benchmark setup

One may suspect that the better performance of the AutoMOEAs for the specific benchmark sets towards which they are tuned comes at the price of poorer performance on other benchmark sets. To examine whether this happens in our case, we applied the various MOEA algorithms tuned for one benchmark set to the respective other one, that is, the algorithms tuned on the WFG training set of functions to the DTLZ benchmark set and vice versa. We did this analysis only for the setup where MOEAs are given a maximum number of FEs to use, and we focus on the rank sum analysis of the I_H^{rpd}. The results of this analysis are given in Table IX. In most cases, the relative order among the algorithms remains very similar to the one encountered in Table VII. In five out of six cases the AutoMOEA algorithms remain the best performing ones, AutoMOEA$_{W2}$ being the only exception. The results for the I_ϵ are consistent with these ones for all scenarios, despite minor differences. The full analysis of this cross-benchmark setup is provided as supplementary material [30].

Table VIII
SUM OF RANKS DEPICTING THE PERFORMANCE OF MOEAS TUNED FOR A MAXIMUM RUNTIME. ALGORITHMS IN BOLDFACE PRESENT RANK SUMS NOT SIGNIFICANTLY HIGHER THAN THE LOWEST RANKED.

W2	I_H^{rpd}	**Auto$_{W2}$ (1700)**	NSGA-II (1909)	IBEA (1912)	SMS (2678)	SPEA2 (3082)	HypE (3360)	MOGA (4259)
	I_ϵ	**Auto$_{W2}$ (1402)**	IBEA (1868)	NSGA-II (2158)	HypE (2995)	SMS (3050)	SPEA2 (3190)	MOGA (4237)
W3	I_H^{rpd}	**IBEA (1363)**	Auto$_{W3}$ (1651)	SMS (2328)	HypE (2566)	NSGA-II (2986)	SPEA2 (3445)	MOGA (4561)
	I_ϵ	**Auto$_{W3}$ (1184)**	IBEA (1380)	SMS (2240)	SPEA2 (2959)	NSGA-II (3109)	HypE (3658)	MOGA (4369)
W5	I_H^{rpd}	**Auto$_{W5}$ (1192)**	IBEA (1446)	SMS (2072)	NSGA-II (2676)	SPEA2 (2857)	MOGA (4274)	HypE (4383)
	I_ϵ	**Auto$_{W5}$ (1052)**	**IBEA (1084)**	SMS (2717)	NSGA-II (2721)	SPEA2 (2932)	MOGA (4075)	HypE (4319)

(ΔR values from top to bottom: 157, 155, 136, 122, 102, 97)

Table IX
SUM OF RANKS DEPICTING THE PERFORMANCE OF MOEAS FOR THE CROSS-BENCHMARK SETUP. ΔR IS THE CRITICAL RANK SUM DIFFERENCE FOR FRIEDMAN'S TEST WITH 99% CONFIDENCE.

DTLZ			WFG		
2-obj $\Delta R = 118$	3-obj $\Delta R = 126$	5-obj $\Delta R = 118$	2-obj $\Delta R = 165$	3-obj $\Delta R = 119$	5-obj $\Delta R = 118$
Auto$_{W2}$ (1142)	**Auto$_{W3}$ (1292)**	**Auto$_{W5}$ (1420)**	**SPEA2$_{D2}$ (1376)**	**Auto$_{D3}$ (1491)**	**Auto$_{D5}$ (1420)**
SPEA2$_{W2}$ (1692)	IBEA$_{W3}$ (1692)	**SMS$_{W5}$ (1485)**	NSGA-II$_{D2}$ (2334)	IBEA$_{D3}$ (1663)	**SMS$_{D5}$** (1485)
IBEA$_{W2}$ (1858)	SMS$_{W3}$ (1937)	IBEA$_{W5}$ (1774)	IBEA$_{D2}$ (2409)	SMS$_{D3}$ (1739)	IBEA$_{D5}$ (1774)
NSGA-II$_{W2}$ (1929)	SPEA2$_{W3}$ (2067)	NSGA-II$_{W5}$ (2279)	HypE$_{D2}$ (2666)	SPEA2$_{D3}$ (2395)	NSGA-II$_{D5}$ (2279)
SMS$_{W2}$ (2443)	NSGA-II$_{W3}$ (2451)	SPEA2$_{W5}$ (2291)	SMS$_{D2}$ (2904)	NSGA-II$_{D3}$ (3360)	SPEA2$_{D5}$ (2291)
MOGA$_{W2}$ (2791)	MOGA$_{W3}$ (2547)	HypE$_{W5}$ (2625)	Auto$_{D2}$ (2966)	HypE$_{D3}$ (3702)	HypE$_{D5}$ (2625)
HypE$_{W2}$ (2844)	HypE$_{W3}$ (2712)	MOGA$_{W5}$ (2824)	MOGA$_{D2}$ (4245)	MOGA$_{D3}$ (4550)	MOGA$_{D5}$ (2824)

F. Concluding remarks

The experiments conducted in this section have confirmed the importance of the automatic design methodology for developing MOEAs for continuous optimization, highlighting both its effectiveness and flexibility. Under all application scenarios and setups considered here, the AutoMOEAs were able to present a robust behavior and often outperform all standard MOEAs. At the same time, the performance of these standard

MOEAs varied considerably. Although IBEA performed well on both setups we adopted, the AutoMOEAs designed here were able to consistently outperform it in the majority of cases.

IV. Automatically Designing MOEAs for Combinatorial Optimization Problems

In a preliminary evaluation of our proposed framework [23], we applied the automatic MOEA design for tackling four multi-objective permutation flow shop problems (MO-PFSP), a well-known class of multi-objective combinatorial problems. Although many MOEAs have not been designed with combinatorial optimization problems in mind, many of the MOEAs we considered in Section III have been adapted to such problems using problem-specific variation operators [31]. Our results with the component-wise approach we propose here led to much better performance than standard MOEAs, and therefore we discuss the insights they produced.

The multi-objective PFSP is a widely studied, practically relevant problem [31, 32] that is structurally different from continuous problems. Thus, we expected that the AutoMOEA designs for the MO-PFSP use different components when compared to those we have previously discussed in Section III. We considered four variants of the MO-PFSP that combine the three most used objective functions from the PFSP literature, which are *makespan* ($\mathsf{C_{max}}$), *total flow time* (TFT), and *total tardiness* (TT). In particular, we considered three bi-objective variants, $\mathsf{C_{max}}$-TFT, $\mathsf{C_{max}}$-TT, and TFT-TT, and the three-objective variant $\mathsf{C_{max}}$-TFT-TT. All problems are NP-hard as already the underlying single-objective PFSPs are.

We used irace to devise five AutoMOEAs: a variant specific for each of the four variants, and a general one that tackles all four MO-PFSP variants (PFSP).

A. Experimental setup

The experimental setup for the MO-PFSP experiments follows runtime-constrained scenarios as presented in Section III but with some differences to be mentioned. First, following [32], we allow algorithms to run for a maximum of $t = 0.1 \cdot n \cdot m$ seconds, where n and m are the number of jobs and machines, respectively. Second, the MO-PFSP literature has already shown that the number of nondominated solutions for these problems is low, and hence we run all algorithms with an unbounded external archive. Third, we use the regular I^H indicator instead of the previous metrics we used for continuous optimization. For tuning, irace was given a budget of 20 000 algorithm runs for designing the general AutoMOEA$_{\mathsf{PFSP}}$, and 5 000 for designing each of the variant-specific AutoMOEAs and for tuning the standard MOEAs. The parameter space used for tuning the standard MOEAs and all AutoMOEAs is the same as presented in the previous section, except for the problem-dependent ones. Following [12], all MOEAs use two-point crossover, and the *insert* and *exchange* mutation operators. For testing, each algorithm was run 10 times on each test instance, and the results presented here are the mean hypervolume over these 10 runs. All test instances are different from the instances used in the tuning. In addition, the testing set considers instances with 5, 10, and 20 machines, while the tuning set uses only instances with 20 machines. For full details we refer to the original paper [23] and to the supplementary material [30]. Next, we discuss the main experimental results and insights from this analysis.

B. Experimental results and discussion

1) AutoMOEA designs for the PFSP: The tuned designs of the AutoMOEAs are shown in Table X, and present two commonalities. First, all AutoMOEAs use a *bounded internal archive*, which reflects the need for convergence pressure for solving combinatorial problems such as the MO-PFSP. Second, all algorithms use a high value for the offspring factor (λ_r), which is always larger than 1.4. This is explained by the time-constrained setup used for combinatorial optimization since the number of offspring per generation influences the trade-off between the number of solutions seen versus the time spent computing metrics [12]. More precisely, if an algorithm produces too few solutions per generation, it will spend most of its time computing selection metrics rather than evaluating new solutions. Conversely, the number of offspring cannot be set to a very high value or this would reduce the number of generations, hindering the evolutionary process.

Besides these two components that are used by all AutoMOEAs, we also highlight other design choices that were often selected by irace. For mating, the crowding diversity operator was used by three of the five AutoMOEAs. Interestingly, the more general AutoMOEA$_{\mathsf{PFSP}}$ uses the same BuildMatingPool components from AutoMOEA$_{\mathsf{Cmax\text{-}TT}}$. Also, the only design that uses a quality indicator in component Preference$_{Mat}$ is AutoMOEA$_{\mathsf{Cmax\text{-}TFT}}$. For replacement, three components are again used by most of the designs: the binary ϵ-indicator or the shared hypervolume contribution as quality indicators, crowding distance as diversity metric, and one-shot removal. In fact, the only design that contradicts this pattern is AutoMOEA$_{\mathsf{TFT\text{-}TT}}$, which uses fitness sharing for diversity and sequential removal. Concerning numerical parameters, an interesting (and apparently contradictory) observation is the fact that all variant-specific AutoMOEA designs favor the exchange mutation operator ($p_X > 0.5$), but AutoMOEA$_{\mathsf{PFSP}}$ favors the insertion mutation operator. This is likely explained by the different tuning budgets used, since larger budgets are particularly beneficial for fine-tuning numerical parameters.

2) Design comparison with previous AutoMOEAs: When compared to the AutoMOEAs devised for continuous optimization, we noticed that most design choices differ considerably with the switch in application domain. For instance, while all continuous AutoMOEAs use deterministic tournament for mating, three out of five AutoMOEAs for the MO-PFSP use some form of randomized selection. In addition, nearly none of the AutoMOEAs designed for the MO-PFSP use quality metrics in Preference$_{Mat}$, while almost all continuous AutoMOEAs did. We do remark, though, the number of AutoMOEAs that use crowding diversity, both for continuous and combinatorial domains. Concerning component Replacement, we first remark that all PFSP use a bounded internal archive and none are steady-state, the opposite of what often happened with the continuous AutoMOEAs. As for Preference$_{Rep}$, both

Table X
PARAMETERS SELECTED BY IRACE FOR THE AUTOMOEAS. ALL DESIGNS USE AN INTERNAL ARCHIVE INSTEAD OF A REGULAR POPULATION.

	BuildMatingPool				Replacement				*Numerical*					
	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_r	λ_r	p_c	p_{mut}	p_X
Cmax-TFT	DT (2)	—	I_H^1	crowding	—	I_ϵ	crowding	one-shot	80	0.3	1.5	0.38	0.82	0.71
Cmax-TT	random	—	—	—	—	I_H^h	crowding	one-shot	30	0.94	1.63	0.34	0.95	0.81
TFT-TT	ST (0.9)	—	—	crowding	—	I_ϵ	sharing (0.87)	sequential	70	0.94	1.47	0.77	0.99	0.63
Cmax-TFT-TT	DT (2)	—	—	crowding	—	I_H^h	crowding	one-shot	40	0.26	1.68	0.36	0.85	0.74
All variants (PFSP)	random	—	—	—	—	I_H^h	—	one-shot	60	0.73	1.53	0.17	0.76	0.40

Table XI
SUM OF RANKS DEPICTING THE OVERALL PERFORMANCE OF MOEAS ON COMBINATORIAL PROBLEMS. ALL ALGORITHMS HAVE BEEN TUNED FOR THE SCENARIOS CONSIDERED. ΔR IS THE CRITICAL RANK SUM DIFFERENCE FOR FRIEDMAN'S TEST WITH 99% CONFIDENCE. ALGORITHMS IN BOLDFACE PRESENT RANK SUMS NOT SIGNIFICANTLY HIGHER THAN THE LOWEST RANKED.

Cmax-TFT $\Delta R = 68$	**AutoMOEA** **PFSP (249)**	**AutoMOEA** **Cmax-TFT (301)**	IBEA (398)	NSGA-II (472)	SPEA2 (479)	HypE (585)	MOGA (687)	SMS-EMOA (788)
Cmax-TT $\Delta R = 55$	**AutoMOEA** **Cmax-TT (209)**	**AutoMOEA** **PFSP (253)**	NSGA-II (357)	SPEA2 (464)	IBEA (547)	HypE (574)	SMS-EMOA (770)	MOGA (786)
TFT-TT $\Delta R = 85$	**MOGA** **(304)**	**IBEA** **(371)**	AutoMOEA TFT-TT (475)	HypE (499)	NSGA-II (499)	AutoMOEA PFSP (553)	SPEA2 (615)	SMS-EMOA (644)
Cmax-TFT-TT $\Delta R = 55$	**AutoMOEA** **Cmax-TFT-TT (161)**	AutoMOEA PFSP (251)	IBEA (417)	SPEA2 (525)	HypE (528)	NSGA-II (541)	SMS-EMOA (735)	MOGA (802)

continuous and combinatorial AutoMOEAs consider dominance, quality metrics, and (very often) the crowding distance diversity metric. Finally, if we compare the external archive removal policy used by continuous AutoMOEAs with the internal archive ones used by the PFSP AutoMOEAs, we notice a much clearer pattern here, pointing to the effectiveness of the one shot policy for the PFSP.

3) Overall performance comparison: We analyzed the performance of all algorithms and variants using the rank sum analysis given in Table XI. In general, the AutoMOEAs perform much better than the standard MOEAs. Among these, IBEA is the algorithm that most often outperforms the others, while SMS-EMOA presents a particularly poor performance. The only scenario that contradicts this pattern is TFT-TT.

Regarding the comparison between the variant-specific and the general AutoMOEAs, for most of the variants considered these two algorithms can be qualified as equally good. The difference in their rank sums is due to the fact that AutoMOEA$_{\text{PFSP}}$ performs particularly well for the 20-machine instances (the size used for the tuning), but for the remaining instances the variant-specific AutoMOEAs consistently outperform it. This indicates that the representativeness of the tuning set with respect to the test set is limited due to the different instance sizes present on each. The only variants where results differ slightly from this pattern are C$_{\text{max}}$-TFT, where AutoMOEA$_{\text{PFSP}}$ obtains a lower rank sum than AutoMOEA$_{\text{Cmax-TFT}}$, and C$_{\text{max}}$-TFT-TT, where the performance of AutoMOEA$_{\text{Cmax-TFT-TT}}$ is statistically significantly better than that of AutoMOEA$_{\text{PFSP}}$.

Concerning the TFT-TT, it is a problem with a high correlation between the two objectives [32]. As a result, for large instances the number of nondominated solutions becomes particularly small and MOEAs sometimes return nondominated sets with very few solutions. This heterogeneity of the testing set limits the representativeness of the tuning set affecting irace, as it cannot select a configuration that performs well across the whole set. To illustrate, we show the performance of all algorithms on two sets of 20-machine instances in Fig. 4. For smaller instance sizes like the ones given in Fig. 4 (left), all algorithms perform similarly, with AutoMOEA$_{\text{TFT-TT}}$ performing best in several instances. For the larger instance sizes given in Fig. 4 (right), however, all algorithms perform poorly, but MOGA is the one clearly less affected. When we limit the rank sum analysis to the testing instances with 20 machines, no significant difference is found for MOGA, AutoMOEA$_{\text{TFT-TT}}$, IBEA, and NSGA-II, the algorithms that present better performance.

C. Concluding remarks

As seen in this section, the designs of the AutoMOEAs devised for the PFSP differed in many aspects from those devised for continuous optimization problems. Nonetheless, the performance of the AutoMOEAs proves the efficacy of the automatic MOEA design also for combinatorial optimization. This further highlights the importance of having a flexible and representative MOEA framework. The good performance of the variant-specific AutoMOEAs reinforce this, showing that designing MOEAs for specific problem variants can lead to promising improvements. A comparison to a current state-of-the-art algorithm for the three bi-objective PFSPs is given in the supplementary material for interested readers [30]. However, reaching state-of-the-art results would require AutoMOEAs incorporate fine-tuned local search algorithms [31, 32], which is beyond the scope of this paper.

V. CONCLUSION

In this paper, we have proposed a novel conceptual view of MOEAs to improve the way such algorithms are designed

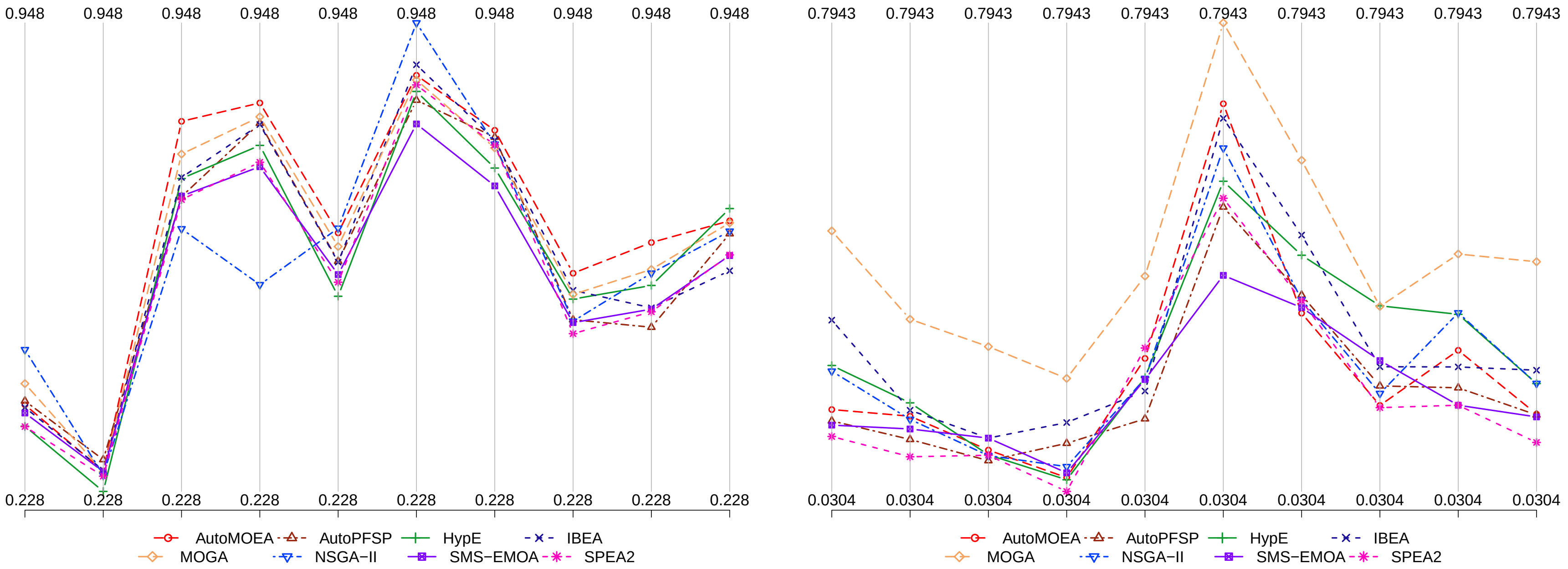

Figure 4. Mean hypervolume on 10 instances of TFT-TT. Left: 50 jobs, 20 machines. Right: 200 jobs, 20 machines. Each vertical line depicts one instance.

or applied to new scenarios. By considering MOEAs as combinations of lower-level components, such as preference relations and archives, our approach allows tailoring algorithms according to the characteristics of the target application. We have empirically demonstrated the efficacy of this component-wise view by automatically designing novel, efficient MOEAs for several application scenarios comprising continuous and combinatorial optimization problems. Concretely, we have proposed a flexible framework that extends both the number of algorithms that can be instantiated from a single template and the number of novel MOEA designs that can be produced from it. To navigate this large design space, we have used an offline parameter configuration tool, irace, following similar research work on other multi-objective metaheuristics [20].

An important focus of our work is the generalization aspect of the automatic design process. More precisely, a critical attribute of automatic configuration is the separation between training and testing scenarios, as highlighted in Birattari's PhD thesis [33], increasingly many other publications in this area [34–37], and also in the policy on heuristic search of the Journal of Heuristics [38]. In the context of continuous optimization, we have implemented this separation by using different dimensions of the functions within a benchmark set for training and testing, and additionally by conducting cross-benchmark experiments. For the MO-PFSPs studied, we have training instances that are different from the testing instances.

The applications of the conceptual view we propose here are numerous. First, by designing novel MOEAs to specific problem classes, one could identify particularly effective components for a given class. For instance, continuous benchmark sets often include disconnected problems. Using the methodology proposed here, one could create a benchmark set of disconnected problems and analyze AutoMOEAs specifically devised for this benchmark. Moreover, by comparing designs devised for several characteristic-driven benchmark sets like this one, patterns could likely be found, helping in future MOEA design when the characteristics of a given application are known in advance.

A second and promising application is to couple the automatic design methodology with iterative design-space analysis tools such as ablation analysis [39]. This method generates intermediate configurations between pairs of algorithm designs, and hence can provide important insights about the contribution of individual components. One example of such approach would be to ablate between the AutoMOEAs devised for three and five objectives, and see how the intermediate configurations perform in these scenarios. Since related work has already shown that the contributions of some components can be much larger than that of others [12], one could possibly reduce the number of components used by some of these AutoMOEAs and understand better why they work so well.

Finally, the research presented here can be extended into a number of interesting directions. Future work should extend the proposed template to integrate MOEAs that differ considerably from the structure proposed here, such as MOEA/D [25], MO-CMA-ES [40], and GDE3 [41]. Future extensions should also allow hybridization with local search, which is crucial to achieve state-of-the-art results in many combinatorial problems, such as the MO-PFSP [31, 32]. Moreover, recent work in the MOEA literature has considered problems with a large number of objectives (10–20) [42]. The specific search components used by MOEAs designed for these scenarios should also be considered, such as space partitioning [43], decomposition [25] and reference points [44].

References

[1] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *ICGA*. Morgan Kaufmann Publishers, 1993, pp. 416–423.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[3] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *EUROGEN*, CIMNE, Barcelona, Spain, 2002, pp. 95–100.

[4] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *PPSN*, ser. LNCS, Springer, 2004, vol. 3242, pp. 832–842.

[5] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.

[6] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, 2007.

[7] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, UK: Wiley, 2001.

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

[9] E. Zitzler, L. Thiele, and K. Deb, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.

[10] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization*, ser. Advanced Information and Knowledge Processing, Springer, London, UK, Jan. 2005, pp. 105–145.

[11] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evol. Comput.*, vol. 8, no. 2, pp. 125–147, 2000.

[12] L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle, "Deconstructing multi-objective evolutionary algorithms: An iterative analysis on the permutation flowshop," in *LION*, ser. LNCS, Springer, 2014, vol. 8426, pp. 57–172.

[13] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba, "On the effect of the steady-state selection scheme in multi-objective genetic algorithms," in *EMO*, ser. LNCS, Springer, 2009, vol. 5467, pp. 183–197.

[14] A. Liefooghe, L. Jourdan, and E.-G. Talbi, "A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO," *Eur. J. Oper. Res.*, vol. 209, no. 2, pp. 104–112, 2011.

[15] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA – a platform and programming language independent interface for search algorithms," in *EMO*, ser. LNCS, Springer, 2003, vol. 2632, pp. 494–508.

[16] F. Biscani, D. Izzo, and C. H. Yam, "A global optimisation toolbox for massively parallel engineering optimisation," in *ICATT*, 2010.

[17] C. Igel, V. Heidrich-Meisner, and T. Glasmachers, "Shark," *J. Mach. Learn. Res.*, vol. 9, pp. 993–996, 2008.

[18] M. Laumanns, E. Zitzler, and L. Thiele, "A unified model for multi-objective evolutionary algorithms with elitism," in *IEEE CEC*. Piscataway, NJ: IEEE Press, Jul. 2000, pp. 46–53.

[19] E. Zitzler, L. Thiele, and J. Bader, "On set-based multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 58–79, 2010.

[20] M. López-Ibáñez and T. Stützle, "The automatic design of multi-objective ant colony optimization algorithms," *IEEE Trans. Evol. Comput.*, vol. 16, no. 6, pp. 861–875, 2012.

[21] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," IRIDIA, ULB, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011.

[22] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, 2006.

[23] L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle, "Automatic design of evolutionary algorithms for multi-objective combinatorial optimization," in *PPSN 2014*, ser. LNCS, Springer, 2014, vol. 8672, pp. 508–517.

[24] J. D. Knowles and D. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, 2000.

[25] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007.

[26] M. López-Ibáñez, J. D. Knowles, and M. Laumanns, "On sequential online archiving of objective vectors," in *EMO*, ser. LNCS, Springer, 2011, vol. 6576, pp. 46–60.

[27] J. D. Knowles and D. Corne, "Bounded Pareto archiving: Theory and practice," in *Metaheuristics for Multiobjective Optimisation*, ser. LNEMS, Springer, Berlin, Germany, 2004, vol. 535, pp. 39–64.

[28] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, 2003.

[29] K. Deb and D. Deb, "Analysing mutation schemes for real-parameter genetic algorithms," *Intern. J. Artif. Intell. Soft. Comput.*, vol. 4, no. 1, pp. 1–28, 2014.

[30] L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle, "Automatic component-wise design of multi-objective evolutionary algorithms," http://iridia.ulb.ac.be/supp/IridiaSupp2014-010/, 2015.

[31] G. Minella, R. Ruiz, and M. Ciavotta, "A review and evaluation of multiobjective algorithms for the flowshop scheduling problem," *INFORMS Journal on Computing*, vol. 20, no. 3, pp. 451–471, 2008.

[32] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems," *Comput. Oper. Res.*, vol. 38, no. 8, pp. 1219–1236, 2011.

[33] M. Birattari, "The problem of tuning metaheuristics as seen from a machine learning perspective," Ph.D. dissertation, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[34] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *GECCO*, Morgan Kaufmann Publishers, San Francisco, CA, 2002, pp. 11–18.

[35] H. H. Hoos, "Programming by optimization," *Commun. ACM*, vol. 55, no. 2, pp. 70–80, Feb. 2012.

[36] S. Nguyen, M. Zhang, M. Johnston, and K. Chen Tan, "Genetic programming for evolving due-date assignment models in job shop environments," *Evol. Comput.*, vol. 22, no. 1, pp. 105–138, 2014.

[37] ——, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 193–208, 2014.

[38] "Journal of heuristic policies on heuristic search research," http://www.springer.com/journal/10732.

[39] C. Fawcett and H. H. Hoos, "Analysing differences between algorithm configurations through ablation," in *MIC*, 2013, pp. 123–132.

[40] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evol. Comput.*, vol. 15, no. 1, pp. 1–28, 2007.

[41] S. Kukkonen and J. Lampinen, "GDE3: the third evolution step of generalized differential evolution," in *IEEE CEC*. Piscataway, NJ: IEEE Press, Sep. 2005, pp. 443–450.

[42] C. von Lücken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," *Computational Optimization and Applications*, vol. 58, no. 3, pp. 707–756, 2014.

[43] H. E. Aguirre and K. Tanaka, "Many-objective optimization by space partitioning and adaptive ϵ-ranking on MNK-landscapes," in *EMO*, ser. LNCS, Springer, 2009, vol. 5467, pp. 407–422.

[44] K. Deb and S. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, 2014.

Leonardo C. T. Bezerra received the M.S. degree in systems and computing from the Universidade federal do Rio Grande do Norte, Brazil, in 2011. He is currently a FRIA doctoral fellow of the Belgian F.R.S.-FNRS at the IRIDIA laboratory, Université libre de Bruxelles (ULB), Brussels, Belgium. His research interests concern artificial intelligence techniques such as metaheuristics and machine learning, and their application to operations research, in particular the automatic design of metaheuristics for multi-objective optimization.

Manuel López-Ibáñez received the M.S. degree in computer science from the University of Granada, Granada, Spain, in 2004, and the Ph.D. degree from Edinburgh Napier University, Edinburgh, U.K., in 2009. He is currently a Postdoctoral Researcher of the F.R.S.-FNRS at IRIDIA, ULB, Brussels, Belgium. His current research interests are the engineering, experimental analysis and automatic configuration of stochastic optimization algorithms for single and multi-objective optimization problems.

Thomas Stützle received the Ph.D. degree in computer science from Technische Universität Darmstadt, Darmstadt, Germany, in 1998. He is a Senior Research Associate of the F.R.S.-FNRS working at IRIDIA, ULB, Belgium. He has published extensively in the area of metaheuristics (more than 200 peer-reviewed articles in journals, conference proceedings, or edited books). His research interests range from stochastic local search (SLS) algorithms, large scale experimental studies, automated design of algorithms, to SLS algorithms engineering.