Université Libre de Bruxelles

*Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle*

Automatic Component-Wise Design of Multi-Objective Evolutionary Algorithms

L. C. T. BEZERRA, M. LÓPEZ-IBÁÑEZ, and T. STÜTZLE

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2014-012

August 2014

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle*
UNIVERSITÉ LIBRE DE BRUXELLES
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2014-012

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of IRIDIA. The authors take full responsibility for any copyright breaches that may result from publication of this paper in the IRIDIA – Technical Report Series. IRIDIA is not responsible for any use that might be made of data appearing in this publication.

Automatic Component-wise Design of Multi-objective Evolutionary Algorithms

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

Abstract—Multi-objective evolutionary algorithms (MOEAs) 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 MOEAs 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 MOEA template and a new conceptual view of MOEA components that surpasses existing frameworks in both the number of MOEAs that can be instantiated from the template and the flexibility to produce novel MOEA designs. We empirically demonstrate the flexibility of our proposed framework by automatically designing MOEAs for multi-objective continuous and combinatorial optimization problems. The automatically designed MOEAs are often able to outperform six traditional MOEAs from the literature, even after tuning their numerical parameters.

Index Terms—Multiobjective Optimization, Evolutionary Algorithms, Permutation Flowshop Problem, Automatic Algorithm Configuration.

I. INTRODUCTION

THE importance of the research on multi-objective evolutionary algorithms (MOEAs) is 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 [11–13]. More importantly, these empirical studies indicate that performance improvements may be achieved by replacing the algorithmic components of well-established MOEAs by alternative options from other MOEAs. These studies also indicate that 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 alternate procedures taken either from different MOEAs or being 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 configuration of components, that is, an algorithm “design”, has already been proposed in the literature. 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 is mostly focused 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, up to now 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 producing 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] for 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 diverse as SPEA2 and SMS-EMOA. Our proposal is implemented in a software framework from which novel MOEA designs can be instantiated by properly selecting the values of the various algorithmic components (and other numerical parameters).

In addition, we empirically demonstrate that it is possible to find novel MOEA designs that outperform the MOEAs from the literature by means of automatically configuring the

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

components of our proposed framework. Following previous work on automatic design [20], we apply an offline automatic configuration method, such as irace [21], to our proposed framework by considering the various algorithmic components as categorical parameters to be set by irace. 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 Auto-MOEAs, 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, since MOEAs have been primarily designed for this type of problems, and two benchmark sets, DTLZ [10] and WFG [22], with three and five objectives. Although MOEAs aim to be independent of the benchmark, they were evaluated using available benchmark sets and, thus, implicitly (or sometimes explicitly) also designed with the idea of solving well the particular available benchmark set. 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, very different setups, we are able to represent problems with computationally demanding function evaluations as well as problems where the computational overhead of the MOEA components is relevant. Although the former is the typical setup considered in the MOEA literature, the conclusions obtained might not be applicable 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 for completeness. 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.

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

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

```

continuous and combinatorial problems in Sections III and IV, respectively. Finally, Section V concludes the paper and discusses future work.

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 (pop_ext).

Next, we describe the multi-objective components as well as how to instantiate some well-known MOEAs from our template. We also explain 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 (without loss of generality) to be minimized, a relation $\prec_\Psi$ is defined as $\theta_1 \prec_\Psi \theta_2 \iff \Psi(\theta_1) < \Psi(\theta_2)$. In 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

Table I
MAIN ALGORITHMIC COMPONENTS OF AUTOMOEA

Component	Parameters
Preference	$\langle \text{Set-partitioning, Quality, Diversity} \rangle$
BuildMatingPool	$\langle \text{Preference}_{Mat}, \text{Selection} \rangle$
Replacement	$\langle \text{Preference}_{Rep}, \text{Removal} \rangle$
Replacement _{Ext}	$\langle \text{Preference}_{Ext}, \text{Removal}_{Ext} \rangle$

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 the set-partitioning relation, the refinement relations do not 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. Although 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), 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. 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, when *random selection* is used, individuals are selected with uniform probability, and no preference relation is used. 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} .

Table II
ALGORITHMIC COMPONENT OPTIONS AVAILABLE FOR AUTOMOEA

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

Table III
DIFFERENT TYPES OF ARCHIVES AVAILABLE FOR AUTOMOEA

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

FS: fixed-size; MC: maximum capacity; DS: dominated solutions; EP: part of the evolutionary process; $|\text{pop}_0|_{<}$: size after removing dominated solutions

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 (and, hence, may lead to better results), this policy is computationally more demanding, which may compromise its performance in time-constrained scenarios.

B. Population and archives

A population is a set of individuals, dominated and non-dominated 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 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 pop_{ext} that does not participate in the evolutionary process.

All options implemented here for pop and pop_{ext} are listed in Table II, and we present a summary of their characteristics in Table III. If pop is set to have a fixed size ($\text{type}(\text{pop}) = \text{fixed-size}$), then pop behaves like a regular population of size μ and may contain dominated solutions. Otherwise, pop is used as a bounded internal archive ($\text{type}(\text{pop}) = \text{bounded}$), 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 in this way an archive bounding method [27]. When used as a bounded internal archive, pop presents two other important characteristics. First, the initial number of solutions μ_0 in 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 of the internal archive pop make it flexible enough to allow us instantiate archive-based algorithms like 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 pop_{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, as shown in line 10 of Algorithm 1. Component $\text{Replacement}_{\text{Ext}}$ is defined analogously to Replacement, but with its own $\text{Preference}_{\text{Ext}}$ and $\text{Removal}_{\text{Ext}}$ options (see Table I). Since all solutions kept by the archive are nondominated, $\text{Preference}_{\text{Ext}}$ does not use a set-partitioning relation. For application scenarios where the number of non-dominated solutions is low, MOEAs can either use an archive without capacity constraints, i.e., $\text{type}(\text{pop}_{\text{ext}}) = \text{unbounded}$, or not use an external archive at all, i.e., $\text{type}(\text{pop}_{\text{ext}}) = \text{none}$.

C. Differences from existing frameworks

A number of MOEA 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. In fact, some components we use in the implementation of our template come from the ParadisEO-MOEO [14], PISA [15], and PaGMO [16] frameworks. However, designing a novel MOEA by combining existing components in novel ways using most of these frameworks is not a straightforward task, since they were not created with this goal in mind.

The framework that most closely resembles our approach is ParadisEO-MOEO [14], which 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_H^l and I_H^h quality indicators, and 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 novel MOEA designs one can instantiate through it.

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) *Multi-Objective Evolutionary Algorithm*: MOGA uses *dominance ranking* as set-partitioning relation and the *niche sharing* (also known as *fitness sharing*) diversity metric based on the objective values of the individuals. 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) *Non-dominated Sorting Genetic Algorithm II*: NSGA-II 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) *Strength Pareto Evolutionary Algorithm 2*: SPEA2 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 pop , and its internal population is equivalent to pop_{new} in our template.

4) *Indicator-Based Evolutionary Algorithm*: IBEA uses quality indicators to direct its search, as the ranking induced

Table IV
INSTANTIATION OF MOEAS FROM OUR PROPOSED TEMPLATE.

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

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

by this algorithm is 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) *Hypervolume Estimation Algorithm for Multi-objective Optimization*: In HypE, the search is 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) *S-Metric Selection Evolutionary Multi-objective Algorithm*: SMS-EMOA is a $(\mu+1)$ -ES algorithm that uses the *exclusive hypervolume contribution* (I_H^h) for replacement. This quality indicator evaluates individuals according to their unique contribution to the hypervolume of the population. 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.

As said before, the benchmark problems that have been considered at the design time of an algorithm may implicitly or explicitly bias the algorithm design. Here we mimic 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.

Each benchmark set is used with three and five objectives. We separate between different number of objectives as the number of objectives 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 four scenarios obtained from the combinations of benchmark set and number of objectives: AutoMOEA_{D3} (DTLZ with 3 objectives), AutoMOEA_{D5} (DTLZ with 5 objectives), AutoMOEA_{W3} (WFG with 3 objectives), and AutoMOEA_{W5} (WFG with 5 objectives).

A. Tuning 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

Table V
PARAMETER SPACE FOR TUNING ALL MOEAs FOR CONTINUOUS OPTIMIZATION.

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

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

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, the approximation fronts produced are normalized to the range $[1, 2]$ to avoid bias from dissimilar domains. Finally, we compute the hypervolume using the reference point $r_i = 2.1$, $i = 1, \dots, M$, where M is the number of objectives considered.

Experiments are 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. To keep our experiments feasible in time, we limit the maximum runtime of a single run to 10 minutes. For all of the configurations that correspond to the standard MOEAs, this time limit is high enough for them to perform all 10 000 FEs. In fact, we have empirically verified that, on average, 85% of the candidates produced by *irace* use all of the FEs allowed within this time limit. The few configurations that do not use all of the 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

Figure 1. Boxplots of the relative hypervolume of selected MOEAs using *default* or *tuned* numerical parameter settings (WFG 3 objectives, 40 variables).

irace is equivalent to designing a MOEA over the weekend. Furthermore, we have also repeated our experiments with a reduced tuning budget of 10 000 runs to see how that would affect the results. Despite minor changes, results are consistent with the ones we present here.

B. Testing setup

To keep the analysis unbiased, we take two additional precautions. First, we use disjoint test problem sets for tuning and testing. We distinguish the tuning and the testing sets by the number of variables they consider (n_{var}). Concretely, we use $n_{\text{var}} \in \{20, \dots, 60\} \setminus n_{\text{testing}}$ for tuning, where $n_{\text{testing}} = \{30, 40, 50\}$ are the sizes we reserve for testing. For comparing the performance of different algorithms, we run each algorithm 25 times on the testing benchmarks. Then, we compute their *relative hypervolume* w.r.t to the hypervolume of the Pareto front (approximated using 10 000 Pareto-optimal solutions obtained as described in the original benchmark papers). Given an approximation front A and the Pareto front for a problem instance P , the relative hypervolume of A equals $I_H(A)/I_H(P)$. Hence, a relative hypervolume of 1.0 means the algorithm fully approximated the Pareto front.

The second precaution concerns 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. Figure 1 shows this performance improvement for some of the MOEAs on four problem instances of the 3-objective WFG benchmark set.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). Configurations that would change the MOEA design as defined by Table IV are not allowed when tuning numerical parameter settings. 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.

¹For brevity, few representative results are given here. The complete list of results is provided as supplementary material [30], together with the tuned configurations of all MOEAs.

Table VI
PARAMETERS SELECTED BY IRACE FOR THE AUTOMOEAS.

	BuildMatingPool				Replacement			Replacement _{Ext}		Numerical									
	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	Quality	Diversity	μ	μ_r	λ	λ_r	p_c	p_m	p_v	η_c	η_v
3-obj DTLZ	random	—	—	—	depth-rank	I_ϵ	—	—	I_H^1	sharing (0.3)	30	—	1	—	0.9	0.9	—	30	16
5-obj DTLZ	DT (4)	count	I_H^1	crowding	depth	I_ϵ	crowding	sequential	I_H^1	crowding	80	—	—	0.4	0.8	0.2	0.8	39	38
3-obj WFG	ST (0.8)	—	I_H^h	sharing (0.3)	—	I_H^h	kNN	sequential	I_H^1	sharing (0.9)	50	0.6	—	1.2	0.3	0.5	0.8	22	3
5-obj WFG	ST (0.6)	—	—	sharing (0.7)	—	I_H^1	—	sequential	I_ϵ	—	50	0.6	—	1.5	0.3	0.4	1.0	40	7

(All AutoMOEAs use $\text{type}(\text{pop}_{\text{ext}}) = \text{bounded}$ with sequential removal and $N_{\text{ext}} = 500$; only AutoMOEA_{D3} uses the bitwise mutation scheme; both AutoMOEA_{W3} and AutoMOEA_{W5} use bounded internal archives)

C. Results and discussion

The designs of the AutoMOEAs selected by irace for each of the scenarios we consider here are shown in Table VI. Our first observation is how different these designs are. In fact, the only component in common for all AutoMOEAs is the bounded external archive of size 500 with sequential removal. This was rather expected since the number of nondominated solutions for this type of problems is typically high, demanding an external archive, but prohibiting an unbounded one. We discuss next the remaining components of each AutoMOEA.

The design of AutoMOEA_{D3} can be described as an alternate version of SMS-EMOA. Both algorithms use random mating selection and the steady-state replacement using dominance depth-rank for set-partitioning. However, while population replacement in SMS-EMOA is based on the exclusive hypervolume contribution (I_H^1), AutoMOEA_{D3} uses instead the binary ϵ -indicator. Moreover, AutoMOEA_{D3} manages its external archive using the I_H^1 indicator, coupled with fitness sharing to promote diversity. This shows that the information provided by the hypervolume can still be improved, which is consistent with other results seen in the literature [19].

The structure of AutoMOEA_{D5} differs in many components from that of AutoMOEA_{D3}. First, its population size (μ) is much larger and the number of offspring per generation ($\lambda = \lambda_r \cdot \mu$) is larger than one, although it remains low when compared to the value $\lambda_r = 1.0$ traditionally adopted by MOEAs. Second, instead of the random mating selection used by AutoMOEA_{D3}, AutoMOEA_{D5} uses a four-ary deterministic tournament based on dominance count, the exclusive hypervolume contribution, and the crowding distance diversity metric. In fact, this preference relation reinforces what we had previously discussed for AutoMOEA_{D3}: that the hypervolume contribution can still benefit from additional convergence pressure and diversity preserving metrics. In particular, the choice of dominance count for set-partitioning is rather interesting, as it is a metric that privileges solutions in the center of the objective space.

Differently from the AutoMOEAs designed for the DTLZ benchmark, both AutoMOEA_{W3} and AutoMOEA_{W5} use pop as a bounded internal archive, which indicates that the WFG benchmark requires a higher convergence pressure. The structure of AutoMOEA_{W3} relies heavily on the information provided by the hypervolume: the exclusive hypervolume contribution (I_H^1) is used for the external archive replacement and the shared hypervolume contribution (I_H^h) is used both for mating and for the internal archive replacement. Another interesting feature

Figure 2. Relative hypervolume for all algorithms on the DTLZ benchmark with 40 variables and either three objectives (top), where differences are found only for DTLZ4, DTLZ6, and DTLZ7; or five objectives (bottom), where differences are also found for DTLZ5.

of the preference relations used by AutoMOEA_{W3} is that they all present diversity components. For mating and the external archive replacement, fitness sharing is used, but with very different σ_{share} values. For the internal archive replacement, the diversity metric based on the distance to the nearest neighbors is adopted.

Finally, the structure of AutoMOEA_{W5} is far simpler than that of AutoMOEA_{W3}. While all the preference relations used by AutoMOEA_{W3} combined a quality indicator with a diversity metric, AutoMOEA_{W5} uses a single component per preference relation. For mating, only fitness sharing is used, which resembles the niche-formation strategy used by MOGA. For the replacement of both archives, only quality indicators are employed. While the internal archive focuses on maximizing the exclusive hypervolume contribution, the external archive focuses on minimizing the binary ϵ -indicator. Next, we discuss the performance of the algorithms for each of the scenarios considered.

1) *3-objective DTLZ (D3)*: Concerning the performance of the algorithms, the 3-objective DTLZ benchmark can be split into two groups. For the first group, formed by DTLZ1–DTLZ3 and DTLZ5, all algorithms are able to approximate the Pareto front accurately, and hence we omit these problems from the boxplots we present. For the second group, formed by the remaining problems, SMS-EMOA and IBEA perform much better than the other standard MOEAs, as shown in Figure 2 (top). The AutoMOEAs are able to achieve the max-

Table VII

SUM OF RANKS DEPICTING THE OVERALL PERFORMANCE OF MOEAS ON CONTINUOUS PROBLEMS. ΔR IS THE CRITICAL RANK SUM DIFFERENCE FOR FRIEDMAN'S TEST WITH 95% SIGNIFICANCE. ALGORITHMS IN BOLDFACE PRESENT RANK SUMS NOT SIGNIFICANTLY HIGHER THAN THE LOWEST RANKED.

3-obj DTLZ $\Delta R = 165$	SMS-EMOA_{D3} (1779)	Auto_{D3} (1880)	Auto _{D5} (2147)	IBEA _{D3} (2397)	SPEA2 _{D3} (2823)	Auto _{W5} (3053)	NSGA-II _{D3} (3321)	HypE _{D3} (3602)	Auto _{W3} (3834)	MOGA _{D3} (4036)
5-obj DTLZ $\Delta R = 137$	Auto_{D5} (1039)	SMS-EMOA _{D5} (1763)	Auto _{W5} (2157)	IBEA _{D5} (2278)	Auto _{D3} (2380)	NSGA-II _{D5} (3533)	HypE _{D5} (3619)	Auto _{W3} (3683)	SPEA2 _{D5} (3918)	MOGA _{D5} (4503)
3-obj WFG $\Delta R = 144$	Auto_{W3} (2066)	Auto _{W5} (2487)	SPEA2 _{W3} (2546)	NSGA-II _{W3} (2676)	SMS-EMOA _{W3} (2676)	IBEA _{W3} (3084)	MOGA _{W3} (4742)	HypE _{W3} (5020)	Auto _{D5} (5356)	Auto _{D3} (6472)
5-obj WFG $\Delta R = 127$	Auto_{W5} (1372)	SMS-EMOA _{W5} (1929)	IBEA _{W5} (2795)	Auto _{W3} (2991)	SPEA2 _{W5} (3025)	NSGA-II _{W5} (3175)	Auto _{D5} (4195)	MOGA _{W5} (5556)	HypE _{W5} (5924)	Auto _{D3} (6163)

Figure 3. Relative hypervolume for all algorithms on the WFG benchmark with 40 variables (top: 3-objectives; bottom: 5-objectives).

imum hypervolume for all problems except for AutoMOEA_{D3} on DTLZ6, AutoMOEA_{W3} for DTLZ7, and AutoMOEA_{D5} for DTLZ4 and DTLZ7. The similarities between AutoMOEA_{D3} and SMS-EMOA reflect directly on their overall performance. As seen in the rank sum analysis given in Table VII, both algorithms get equivalent rank sums considering Friedman's test with 95% significance level. These results confirm what was expected: that the standard MOEAs perform particularly well for the benchmark they were tested on at design time. Nonetheless, the automatically designed MOEAs are able to match their performance.

2) *5-objective DTLZ (D5)*: All algorithms were again able to accurately approximate the Pareto fronts for functions DTLZ1–DTLZ3, and hence we omit these functions from the plots. Figure 2 (bottom) shows the performance of the algorithms for the remaining problems. Those MOEAs that did not perform consistently well across the whole three objectives DTLZ benchmark (e.g., MOGA and HypE) perform even worse when five objectives are considered. Other algorithms (SMS-EMOA, IBEA and the AutoMOEAs except for AutoMOEA_{W3}) perform similarly well for both three and five objective problems, which shows that these algorithms present scalable designs. Moreover, AutoMOEA_{W5} and AutoMOEA_{D5} are nearly indistinguishable in the boxplots, which indicates that the number of objectives is a more important feature when designing a MOEA than the actual benchmark one intends to solve, in the case where the benchmark is rather easy as

the DTLZ one. Nonetheless, the rank sum analysis given in Table VII confirms that AutoMOEA_{D5} returns the best results overall, with much lower rank sums than the second best-ranked algorithm, SMS-EMOA.

3) *3-objective WFG (W3)*: The performance comparison for the 3-objective WFG problems is shown in Figure 3 (top). The first observation we make is the poor results of all MOEAs on some problems. For example, the relative hypervolume achieved by all MOEAs on WFG2 and WFG3 is very close to zero, and therefore we exclude these problems from the plots. Moreover, on most WFG functions, no MOEA considered in this work is able to achieve a relative hypervolume larger than 0.6. Despite these observations, we see that AutoMOEA_{W3} improves over the results of the standard MOEAs for functions WFG4, WFG7, and WFG9. All algorithms obtain very similar results on the remaining problems, except on WFG1, where AutoMOEA_{W3} is outperformed by SMS-EMOA and SPEA2. Overall, AutoMOEA_{W3} performs better than the standard MOEAs considered in this work for the 3-objective WFG set, as shown by Table VII. Moreover, AutoMOEA_{W3} design appears more robust, since the performance of the standard MOEAs varies considerably per problem, as shown in the boxplots and confirmed by the high rank sums they achieve.

For the remaining AutoMOEAs, we see here that the benchmark used for tuning is more important than the number of objectives. While AutoMOEA_{W5} performs as well as the best standard MOEAs, the AutoMOEAs designed using DTLZ

Figure 4. Relative hypervolume boxplots for MOEAs tuned for a maximum runtime on 40-variable WFG problems (top: 3-objectives; bottom: 5-objectives).

perform poorly, particularly AutoMOEA_{D3} . Since AutoMOEA_{D3} and SMS-EMOA obtain very different results in spite of their similar designs, the difference may be due to their numerical parameters. As explained earlier, the parameters of the standard MOEAs are tuned for each scenario, whereas the parameters of AutoMOEA_{D3} are tuned for the DTLZ (3-objectives) scenario.

4) *5-objective WFG (W5)*: Figure 3 (bottom) shows the boxplots for the 5-objective WFG set, on which AutoMOEA_{W5} performs particularly well. For WFG1, WFG4, WFG7, and WFG9, AutoMOEA_{W5} is clearly able to produce better approximation fronts than the other MOEAs. Moreover, this AutoMOEA presents a very robust behavior, being better than the other algorithms in most of the problems considered. This robust performance is confirmed by its much lower rank sum given in Table VII, which indicates that the simplicity of its design is actually an advantage of this AutoMOEA . In fact, while the performance of the standard MOEAs depends on the problem, AutoMOEA_{W5} only performs worse than the others for WFG8. Apparently, the characteristics of this problem make it impossible for irace to select a configuration that performs simultaneously well in all WFG problems. The other AutoMOEA s, however, perform on this benchmark set as poorly as the MOEAs that perform worse overall (NSGA-II, MOGA and HypE). In the case of AutoMOEA_{W3} , its worse results here may be explained by the use of the shared hypervolume indicator, which is computed exactly for problems with up to three objectives, but estimated for five objectives.

D. Experiments with a different stopping criterion

As shown by the results discussed above, standard MOEAs tend to perform better on the scenarios on which they have been tested during design time. 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 AutoMOEA s 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 AutoMOEA s designed and tested on the WFG benchmark for three and five objectives. We obtained two very different AutoMOEA s (AutoMOEA_{W3} and AutoMOEA_{W5} , respectively) with only two components in common: the bounded internal archive and a very high number of offspring per generation. AutoMOEA_{W3} uses the shared hypervolume contribution coupled with the kNN diversity metric for mating, and the exclusive hypervolume contribution coupled with crowding distance for the replacement of its internal archive. It also uses an external bounded archive of size 500 based only on crowding distance. By contrast, AutoMOEA_{W5} uses no external archive, random mating selection and the binary ϵ -indicator coupled with crowding distance for its internal archive replacement. The different structures of these algorithms reflect the characteristics of the scenarios that they were designed for. AutoMOEA_{W3} uses components that would be too computationally costly for five objectives, but can lead to better results for three-objective problems. Conversely, AutoMOEA_{W5} uses much less costly components. For instance, the population replacement based on the ϵ -indicator is justified both by its reduced complexity and by the additional convergence pressure it provides, by contrast to

Table VIII
SUM OF RANKS DEPICTING THE OVERALL PERFORMANCE OF MOEAS
TUNED FOR A MAXIMUM RUNTIME SCENARIO (WFG, 40 VARIABLES,
3 AND 5 OBJECTIVES). ALGORITHMS IN BOLDFACE PRESENT RANK SUMS
NOT SIGNIFICANTLY HIGHER THAN THE LOWEST RANKED.

3-obj	AutoW5 (1452)	IBEA (1507)	AutoW3 (1595)	NSGA-II (1962)	SMS (2036)	SPEA2 (2755)	HypE (3546)	MOGA (4047)
5-obj	AutoW5 (1453)	IBEA (1466)	SMS (1855)	NSGA-II (1877)	SPEA2 (1877)	AutoW3 (2564)	MOGA (3743)	HypE (3947)

the exclusive hypervolume contribution used by AutoMOEA_{W3}.

The relative hypervolume boxplots for the WFG problems with 40 variables are shown in Figure 4, while the rank sum analysis is given in Table VIII. As expected, IBEA is the algorithm that benefits the most from the different setup, since now it can evaluate many more solutions. On the other hand, NSGA-II, which is also a fast algorithm, gives unequal results depending on the problem, particularly for five objectives. SMS-EMOA, which performed particularly well when runtime was not an issue, is now always worse than IBEA. By contrast, AutoMOEA_{W5} presents the best overall performance for both benchmark sets, matching the rank sums of IBEA. AutoMOEA_{W3} presents competitive results for the 3-objective problems, with a rank sum equivalent to those of IBEA and AutoMOEA_{W5}. For five objectives, however, AutoMOEA_{W3} is unable to scale and does not perform so well.

E. 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. When given a maximum number of FEs to use, SMS-EMOA is the MOEA that produced the best results among the standard MOEAs. However, when a maximum runtime is used as stopping criterion, the performance of SMS-EMOA is surpassed by IBEA for most of the problems considered. In both scenarios, however, the AutoMOEAs designed here performed consistently well.

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 have shown that the component-wise approach we propose here led to much better performance, and therefore we discuss the insights they produced.

The multi-objective PFSP is a widely studied, practically relevant problem [31, 32] that is structurally rather different from continuous multi-objective 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* ($C_{\max}$), *total flow time* (TFT), and *total tardiness* (TT). In particular, we considered three bi-objective variants, $C_{\max}$ -TFT, $C_{\max}$ -TT, and TFT-TT, and the three-objective variant $C_{\max}$ -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.

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. For tuning, irace was given a budget of 20 000 algorithm runs for designing the general AutoMOEA_{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].

B. Experimental results and discussion

Next, we shortly discuss the main experimental results and insights we obtained from the AutoMOEA tuning for the MO-PFSPs.

1) *AutoMOEA designs for the PFSP*: The tuned designs of the AutoMOEAs are shown in Table IX. These design 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 higher than 1.4. This is explained by the time-constrained setup used for combinatorial optimization since the number of offspring per generation, given by $\lambda = \lambda_r \cdot \mu$, influences the trade-off between the number of solutions seen versus the time spent computing metrics [12].

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

	BuildMatingPool				Replacement				<i>Numerical</i>					
	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
PFSP	random	—	—	—	—	I_H^h	—	one-shot	60	0.73	1.53	0.17	0.76	0.40

Table X
SUM OF RANKS DEPICTING THE OVERALL PERFORMANCE OF MOEAs ON COMBINATORIAL PROBLEMS. ΔR IS THE CRITICAL RANK SUM DIFFERENCE FOR FRIEDMAN'S TEST WITH 95% SIGNIFICANCE. ALGORITHMS IN BOLDFACE PRESENT RANK SUMS NOT SIGNIFICANTLY HIGHER THAN THE LOWEST RANKED.

Cmax-TFT $\Delta R = 53$	AutoMOEA PFSP (249)	AutoMOEA Cmax-TFT (301)	IBEA (398)	NSGA-II (472)	SPEA2 (479)	HypE (585)	MOGA (687)	SMS-EMOA (788)
Cmax-TT $\Delta R = 45$	AutoMOEA Cmax-TT (209)	AutoMOEA PFSP (253)	NSGA-II (357)	SPEA2 (464)	IBEA (547)	HypE (574)	SMS-EMOA (770)	MOGA (786)
TFT-TT $\Delta R = 64$	MOGA (304)	IBEA (371)	AutoMOEA TFT-TT (475)	HypE (499)	NSGA-II (499)	AutoMOEA PFSP (553)	SPEA2 (615)	SMS-EMOA (644)
Cmax-TFT-TT $\Delta R = 42$	AutoMOEA Cmax-TFT-TT (161)	AutoMOEA PFSP (251)	IBEA (417)	SPEA2 (525)	HypE (528)	NSGA-II (541)	SMS-EMOA (735)	MOGA (802)

More precisely, if an algorithm produces too few solutions per generation, it will spend most of its time computing metrics for selection rather than evaluating new solutions. Conversely, the number of offspring cannot be set to a very high value, because this will reduce the number of generations and hinder 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_{PFSP} uses the same BuildMatingPool components from AutoMOEA_{Cmax-TT}. Also, the only design that uses a quality indicator in component Preference_{Mat} is AutoMOEA_{Cmax-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_{TFT-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_{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, the design choices that are more frequently observed in both sets of AutoMOEAs is the use of a bounded internal archive with a high number of offspring per generation. In fact, the only AutoMOEAs that do not follow this pattern are the ones designed for the DTLZ benchmark, but this is likely explained by the easiness with which effective MOEA designs

solve this benchmark. In addition, all AutoMOEAs use quality metrics in component Preference_{Rep}. The different with respect to the continuous scenarios is that the AutoMOEAs for the PFSP almost never use the exclusive hypervolume, but they often use the shared hypervolume. As explained earlier, the shared hypervolume, when computed exactly in problems with two or three objectives, typically leads to better results.

Apart from these common components, most of the remaining design choices differ substantially when switching from continuous to combinatorial optimization. While component Preference_{Mat} often used quality indicators, the only AutoMOEA designed for the PFSP that follows this pattern is AutoMOEA_{Cmax-TFT}. Moreover, while fitness sharing and sequential removal were often selected for continuous scenarios, the crowding distance and one-shot removal components appear to be more effective for the PFSP scenarios. This highlights the importance of a flexible and representative framework, since the application scenarios one intends to tackle may contradict tendencies observed for the traditional benchmark sets used in the literature.

3) *Overall performance comparison:* We analyzed the performance of all algorithms and variants using the rank sum analysis given in Table X. 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 these general observations 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_{PFSP} performs particularly well for the 20-machine instances (the size used for the tuning), but for the remaining instances the

Figure 5. Mean hypervolume on 10 instances of TFT-TT. Left: 50 jobs, 20 machines. Right: 200 jobs, 20 machines. Each vertical line depicts one instance.

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 $C_{\max}$ -TFT, where $\text{AutoMOEA}_{\text{PFSP}}$ obtains a lower rank sum than $\text{AutoMOEA}_{C_{\max}\text{-TFT}}$, and $C_{\max}$ -TFT-TT, where the performance of $\text{AutoMOEA}_{C_{\max}\text{-TFT-TT}}$ is statistically significantly better than that of $\text{AutoMOEA}_{\text{PFSP}}$.

Concerning the TFT-TT, it is a problem with a high correlation between the two objectives as noted earlier [32]. As a result, as the instance size grows, the number of nondominated solutions becomes particularly small and MOEAs sometimes return non-dominated 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 Figure 5. For smaller instance sizes like the ones given in Figure 5 (left), all algorithms perform similarly, with $\text{AutoMOEA}_{\text{TFT-TT}}$ performing best in several instances. For the larger instance sizes given in Figure 5 (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, $\text{AutoMOEA}_{\text{TFT-TT}}$, 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 is high enough to prove the efficacy of the automatic MOEA design also for combinatorial optimization. This also highlights the importance of having a flexible and representative MOEA framework. The good performance of the variant-specific AutoMOEAs further reinforces this, showing that designing MOEAs for specific problem variants can lead to promising improvements.

V. CONCLUSION

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

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 used here with iterative design-space analysis tools such as ablation analysis [33]. 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 consider MOEAs that differ considerably from the structure proposed here, such as MOEA/D [25], MO-CMA-ES [34], and GDE3 [35]. Moreover, recent work in the MOEA literature has considered problems with a large number of objectives (10–20) [36]. The specific search components used by MOEAs designed for these scenarios should also be considered, such as space partitioning [37], decomposition [25], and using reference points [38].

REFERENCES

- [1] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *ICGA*, S. Forrest, Ed. 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*, K. C. Giannakoglou, D. T. Tsahalis, J. Periaux, K. D. Papaliliou, and T. Fogarty, Eds. CIMNE, Barcelona, Spain, 2002, pp. 95–100.
- [4] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *PPSN*, ser. LNCS, X. Yao *et al.*, Eds. 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 multi-objective optimization," Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, Tech. Rep. 112, 2001.
- [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, P. M. Pardalos, M. G. C. Resende, C. Vogiatzis, and J. L. Walteros, Eds. 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, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds. 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, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds. Springer, 2003, 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, Jun. 2008. [Online]. Available: <http://www.jmlr.org/papers/volume9/igel08a/igel08a.pdf>
- [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, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011. [Online]. Available: <http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf>
- [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, J. Smith *et al.*, Eds. Springer, 2014, vol. 8672.
- [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, R. H. C. Takahashi *et al.*, Eds. 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, X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, Eds. 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 multi-objective 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] C. Fawcett and H. H. Hoos, "Analysing differences between algorithm configurations through ablation," in *Proceedings of MIC 2013, the 10th Metaheuristics International Conference*, 2013, pp. 123–132.
- [34] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evol. Comput.*, vol. 15, no. 1, pp. 1–28, 2007.
- [35] 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.
- [36] C. von Lüken, 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.
- [37] H. E. Aguirre and K. Tanaka, "Many-objective optimization by space partitioning and adaptive ϵ -ranking on MNK-landscapes," in *EMO*, ser. LNCS, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds. Springer, 2009, vol. 5467, pp. 407–422.
- [38] 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.
- [39] M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds., *Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009*, ser. LNCS. Springer, 2009, vol. 5467.

Automatic Component-wise Design of Multi-objective Evolutionary Algorithms

Supplementary Material

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

Abstract—Multi-objective evolutionary algorithms (MOEAs) 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 MOEAs 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 MOEA template and a new conceptual view of MOEA components that surpasses existing frameworks in both the number of MOEAs that can be instantiated from the template and the flexibility to produce novel MOEA designs. We empirically demonstrate the flexibility of our proposed framework by automatically designing MOEAs for multi-objective continuous and combinatorial optimization problems. The automatically designed MOEAs are often able to outperform six traditional MOEAs from the literature, even after tuning their numerical parameters.

This supplementary material provides all the data required for the analysis of the main paper that had been omitted for brevity.

In Sections I and II, we provide the numerical parameters used by the traditional MOEAs after being tuned by irace. In particular, we place the parameters selected after a 10 000-experiment tuning (Section I) and a 20 000-experiment tuning (Section II) side-by-side so readers can contrast the difference in these settings.

Sections III and IV show the performance comparison for the algorithms tuned using the 10 000-experiment and 20 000-experiment tuning budgets, respectively. For each tuning budget, we show the performance comparison boxplots for all scenarios so the reader can see how algorithms performed across scenarios.

In Section V, only the 20 000-experiment tuning budget is used, and so we present the numerical parameter settings and the performance comparison boxplots altogether. As explained in the paper, only the WFG benchmark set is used in this comparison.

CONTENTS

I	Parameter settings – first termination criterion (10 000 FEs), 10 000-run tuning budget	2
II	Parameter settings – first termination criterion (10 000 FEs), 20 000-run tuning budget	2
III	Boxplot comparison – first termination criterion (10 000 FEs), 10 000-run tuning budget	3
IV	Boxplot comparison – first termination criterion (10 000 FEs), 20 000-run tuning budget	5
V	Parameter settings and performance comparison – second termination criterion (1-min runtime)	7

I. PARAMETER SETTINGS – FIRST TERMINATION CRITERION (10 000 FES), 10 000-RUN TUNING BUDGET

Table I
NUMERICAL PARAMETERS SELECTED BY IRACE FOR MOEAS. IBEA USES I_e AND SPEA2 USES $t_{size} = 8$ FOR ALL SCENARIOS.

	Numerical								
HYPE	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	10	0.1	2	0.4	0.6	T	—	19	12
5-obj DTLZ	100	0.1	4	0.8	0.1	F	0.5	48	19
3-obj WFG	60	0.1	2	0.3	0.8	F	0.9	27	5
5-obj WFG	90	0.1	8	0.9	0.5	F	0.5	6	9

IBEA	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	100	0.28	4	0.96	0.36	F	0.22	11	35
5-obj DTLZ	100	0.63	4	0.83	0.25	F	0.58	37	21
3-obj WFG	100	1.19	4	0.15	0.4	F	0.84	31	3
5-obj WFG	100	1.43	4	0.38	0.55	F	0.69	41	3

MOGA	σ_{share}	μ	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	0.77	100	4	0.53	0.32	T	—	46	5
5-obj DTLZ	0.15	100	4	0.64	0.2	T	—	47	2
3-obj WFG	0.98	100	2	0.19	0.29	F	0.93	44	14
5-obj WFG	0.98	100	2	0.46	0.89	F	0.68	45	11

NSGA-II	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	100	0.18	8	0.3	0.32	T	—	38	4
5-obj DTLZ	70	0.84	8	0.2	0.12	T	—	49	14
3-obj WFG	100	0.2	8	0.61	0.69	F	0.66	27	9
5-obj WFG	100	1.03	8	0.28	0.27	F	0.61	14	4

SMS-EMOA	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	90	—	—	0.79	0.18	F	0.94	48	49
5-obj DTLZ	90	—	—	0.28	0.11	F	0.92	50	33
3-obj WFG	60	—	—	0.18	0.39	F	0.95	33	9
5-obj WFG	70	—	—	0.18	0.19	F	0.72	42	5

SPEA2	μ	λ_r	k_{method}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	90	0.13	$k = 1$	0.29	0.24	T	—	43	9
5-obj DTLZ	70	1.04	$k = 8$	0.01	0.35	T	—	39	15
3-obj WFG	90	1.89	<i>default</i>	0.03	0.07	F	0.56	34	5
5-obj WFG	100	1.83	<i>default</i>	0.05	0.19	F	0.73	34	5

II. PARAMETER SETTINGS – FIRST TERMINATION CRITERION (10 000 FES), 20 000-RUN TUNING BUDGET

Table II
NUMERICAL PARAMETERS SELECTED BY IRACE FOR MOEAS. IBEA USES I_e FOR ALL SCENARIOS. SPEA2 USES $t_{size} = 8$ FOR ALL SCENARIOS, EXCEPT FOR SPEA2_{W3}, WHERE $t_{size} = 4$.

Numerical									
HYPE	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	100	0.1	8	0.9	0.6	T	—	2	20
5-obj DTLZ	100	0.1	4	1.0	0.3	T	—	45	21
3-obj WFG	70	0.1	4	0.4	0.6	F	0.9	21	4
5-obj WFG	50	0.1	8	0.3	0.2	F	0.8	33	1

IBEA	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	100	0.13	8	0.88	0.96	T	—	40	13
5-obj DTLZ	100	0.52	4	0.99	0.31	F	0.49	38	36
3-obj WFG	100	1.69	8	0.15	0.39	F	0.88	9	7
5-obj WFG	100	0.56	4	0.40	0.46	F	0.69	43	2

MOGA	σ_{share}	μ	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	0.69	100	4	0.81	0.40	T	—	50	8
5-obj DTLZ	0.25	100	4	0.77	0.02	F	0.38	45	2
3-obj WFG	0.89	100	2	0.19	0.35	F	0.63	48	1
5-obj WFG	0.99	100	4	0.85	0.6	F	0.81	44	12

NSGA-II	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	80	0.12	8	0.39	0.35	T	—	41	6
5-obj DTLZ	60	0.57	8	0.23	0.16	T	—	48	40
3-obj WFG	100	0.36	8	0.39	0.45	F	0.64	33	4
5-obj WFG	100	0.28	8	0.32	0.26	F	0.82	50	5

SMS-EMOA	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	70	—	—	0.70	0.27	F	0.9	41	45
5-obj DTLZ	100	—	—	0.97	0.33	F	0.97	41	38
3-obj WFG	60	—	—	0.1	0.18	F	0.91	38	2
5-obj WFG	60	—	—	0.15	0.19	F	0.89	21	2

SPEA2	μ	λ_r	k_{method}	p_c	p_m	$bitwise$	p_v	η_c	η_v
3-obj DTLZ	100	0.16	$k = 5$	0.26	0.22	T	—	47	6
5-obj DTLZ	70	0.12	default	0.0	0.52	T	—	—	9
3-obj WFG	100	0.15	default	0.11	0.18	F	0.95	34	1
5-obj WFG	100	0.15	default	0.11	0.19	F	0.64	35	5

III. BOXPLOT COMPARISON – FIRST TERMINATION CRITERION (10 000 FES), 10 000-RUN TUNING BUDGET

Figure 1. Boxplots of the relative hypervolume for 3-objective DTLZ problems with $\{30, 40, 50\}$ variables).

Figure 2. Boxplots of the relative hypervolume for 5-objective DTLZ problems with $\{30, 40, 50\}$ variables).

Figure 3. Boxplots of the relative hypervolume for 3-objective WFG problems with $\{30, 40, 50\}$ variables).

Figure 4. Boxplots of the relative hypervolume for 5-objective WFG problems with $\{30, 40, 50\}$ variables).

IV. BOXPLOT COMPARISON – FIRST TERMINATION CRITERION (10 000 FES), 20 000-RUN TUNING BUDGET

Figure 5. Boxplots of the relative hypervolume for 3-objective DTLZ problems with $\{30, 40, 50\}$ variables).

Figure 6. Boxplots of the relative hypervolume for 5-objective DTLZ problems with $\{30, 40, 50\}$ variables).

Figure 7. Boxplots of the relative hypervolume for 3-objective WFG problems with $\{30, 40, 50\}$ variables).

Figure 8. Boxplots of the relative hypervolume for 5-objective WFG problems with $\{30, 40, 50\}$ variables).

V. PARAMETER SETTINGS AND PERFORMANCE COMPARISON – SECOND TERMINATION CRITERION (1-MIN RUNTIME)

Table III
NUMERICAL PARAMETERS SELECTED BY IRACE FOR MOEAS FOR MAXIMUM RUNTIME SCENARIOS. IBEA USES I_ϵ AND SPEA2 USES $t_{size} = 8$ FOR ALL SCENARIOS.

HYPER	Numerical									
	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v	
3-obj WFG	40	0.1	2	0.5	0.6	F	0.7	8	4	
5-obj WFG	100	2.0	8	0.9	0.9	F	1.0	26	5	

IBEA	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v	
	100	1.83	4	0.5	0.84	F	0.42	42	4	
5-obj WFG	100	1.27	2	0.33	1.0	F	0.25	50	6	

MOGA	σ_{share}	μ	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v	
	0.96	100	2	0.28	0.22	F	0.18	1	38	
5-obj WFG	0.93	100	2	0.13	0.34	F	0.05	5	13	

NSGA-II	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v	
	100	1.01	8	0.59	0.59	T	—	29	4	
5-obj WFG	100	1.57	8	0.45	0.58	F	0.04	47	2	

SMS-EMOA	μ	λ_r	t_{size}	p_c	p_m	$bitwise$	p_v	η_c	η_v	
	70	—	—	0.4	0.97	F	0.29	27	2	
5-obj WFG	50	—	—	0.43	0.87	F	0.78	41	17	

SPEA2	μ	λ_r	k_{method}	p_c	p_m	$bitwise$	p_v	η_c	η_v	
	90	0.3	$k = 6$	0.33	0.98	F	0.06	20	1	
5-obj WFG	90	0.38	default	0.11	0.97	F	0.12	4	6

Figure 9. Boxplots of the relative hypervolume for 3-objective WFG problems with $\{30, 40, 50\}$ variables).

Figure 10. Boxplots of the relative hypervolume for 3-objective WFG problems with $\{30, 40, 50\}$ variables).