Automatically Designing State-of-the-Art Multi- and Many-Objective Evolutionary Algorithms

Leonardo C. T. Bezerra leobezerra@imd.ufrn.br
Instituto Metrópole Digital (IMD), Universidade Federal do Rio Grande do Norte, Natal, RN, Brazil

Manuel López-Ibáñez manuel.lopez-ibanez@manchester.ac.uk
Alliance Manchester Business School, University of Manchester, UK

Thomas Stützle stuetzle@ulb.ac.be
IRIDIA, CoDE, Université Libre de Bruxelles, Belgium

https://doi.org/10.1162/evco_a_00263

Abstract

A recent comparison of well-established multiobjective evolutionary algorithms (MOEAs) has helped better identify the current state-of-the-art by considering (i) parameter tuning through automatic configuration, (ii) a wide range of different setups, and (iii) various performance metrics. Here, we automatically devise MOEAs with verified state-of-the-art performance for multi- and many-objective continuous optimization. Our work is based on two main considerations. The first is that high-performing algorithms can be obtained from a configurable algorithmic framework in an automated way. The second is that multiple performance metrics may be required to guide this automatic design process. In the first part of this work, we extend our previously proposed algorithmic framework, increasing the number of MOEAs, underlying evolutionary algorithms, and search paradigms that it comprises. These components can be combined following a general MOEA template, and an automatic configuration method is used to instantiate high-performing MOEA designs that optimize a given performance metric and present state-of-the-art performance. In the second part, we propose a multiobjective formulation for the automatic MOEA design, which proves critical for the context of many-objective optimization due to the disagreement of established performance metrics. Our proposed formulation leads to an automatically designed MOEA that presents state-of-the-art performance according to a set of metrics, rather than a single one.

Keywords

Multiobjective optimization, evolutionary algorithms, automatic algorithm design.

1 Introduction

Multiobjective optimization problems (MOPs) involve determining the best tradeoff solutions between various, typically conflicting objectives. In the most general case, MOPs are treated in the Pareto-sense and a best possible approximation of the Pareto-optimal front is desired. MOPs arise in many real-world situations and are more difficult to solve than their single-objective counterparts. Given the enormous challenge posed by MOPs, a large number of solution approaches for tackling them have been proposed (Paquete and Stützle, 2007; Ehrgott and Gandibleux, 2004; Deb, 2001; Coello Coello et al., 2007). As one of the main alternatives for tackling MOPs, multiobjective

Manuscript received: 7 December 2017; revised: 16 July 2019; accepted: 9 August 2019.

evolutionary algorithms (MOEAs) have become very popular (Deb, 2001; Coello Coello et al., 2007; Mezura-Montes et al., 2008). Relevant contributions from the MOEA community to solving MOPs include effective algorithmic components like dominance sorting (Goldberg, 1989; Deb, 2001), as well as the theoretical basis for the performance assessment of multiobjective algorithms (Zitzler et al., 2003).

Among the open challenges for the research on MOEAs are *many-objective optimization problems* (MaOPs), that is, MOPs with a large number of objectives (Fleming et al., 2005), which can be found in practical applications. Early MOEAs have shown performance limitations on MaOPs, which has motivated the proposal of a number of new algorithms over the last decade (Chand and Wagner, 2015; Li et al., 2015). In what follows, we will refer to MOEAs that have specifically been designed to tackle many-objective problems, as many-objective evolutionary algorithms (EAs). Although several many-objective EAs have been proposed, their effectiveness is not yet fully understood, as only few experimental comparisons exist (Bezerra et al., 2018; Tanabe et al., 2017). In general, such comparisons show that the improvements of the many-objective EAs considered over established MOEAs is less than expected once a rigorous experimental setup is adopted (e.g., parameters are properly configured).

Several issues help explain the results observed in these assessments. One is the difficulties posed by the increase in the number of objectives, such as increased dominance resistance, that is, the growth in the number of nondominated solutions on problems with specific characteristics (Schütze et al., 2011). Recent empirical analyses have reported disagreements between performance metrics (Jiang et al., 2014; Bezerra et al., 2018), which again vary as a function of problem characteristics. Another important issue is the rigor in the assessment of novel MOEAs. Specifically, the literature contains a large number of multi- and many-objective EAs, too large for researchers to properly assess their (dis)advantages in a practical way. In addition, MOEAs are commonly proposed as monolithic blocks, assuming that their components are equally effective and need to be jointly used. Hence, researchers rarely investigate how the proposed algorithmic components interact, or whether components from existing MOEAs could provide more benefits than components from the new MOEAs being proposed. This monolithic approach is reflected by various MOEA frameworks that offer limited composability of components (Bleuler et al., 2003; Igel et al., 2008; Biscani et al., 2010).

In previous works, we have contributed to help address many of these issues. In Bezerra et al. (2016), we proposed a general MOEA template from which a researcher can easily instantiate several existing MOEAs, but also generate a much larger number of novel MOEAs by combining the algorithmic components available from a component-wise algorithmic framework. Moreover, by applying an automatic algorithm design methodology (KhudaBukhsh et al., 2009; López-Ibáñez and Stützle, 2012; Dubois-Lacoste et al., 2011) to this AutoMOEA framework, we have shown that it is possible to generate several automatically designed MOEAs (AutoMOEAs) that consistently outperform established MOEAs on both continuous and combinatorial optimization problems. In another work (Bezerra et al., 2018), we have experimentally compared the performance of 14 MOEAs from the literature, after automatically tuning their parameter settings, on a variety of application scenarios considering four number of objectives (2, 3, 5, and 10 objectives), different termination criteria based on the maximum number of function evaluations (2 500, 10 000, and 40 000 FEs), and considering three different performance metrics (relative hypervolume, additive ϵ-indicator, and inverted generational distance, IGD). From that analysis, we have obtained various insights which should be taken into account by MOEA designers, for example, that using

different underlying EA operators can significantly boost the performance of MOEAs for continuous optimization. In addition, we have noticed that some recently proposed MOEAs do not present an empirical performance matching what their proposers had expected. That study not only gives us a solid basis for comparing the performance of automatically generated MOEAs to the performance of current state-of-the-art MOEAs, but it also indicates directions to extend our AutoMOEA framework.

In this work, we propose a strongly extended framework, hereon called AutoMOEA+. The main extensions to obtain AutoMOEA+ are the following. First, we integrate a further level of composability that allows us to separate between the multiobjective related aspects of the search from the underlying EA. Effectively, this composability refinement greatly expands the design space provided by our template, as any existing MOEA can be coupled with the most relevant EAs from the literature. More importantly, by doing so we contemplate the potential interactions between multiobjective components and underlying EAs. Second, we extend our template to comprise decomposition-based algorithms (Zhang and Li, 2007; Zhang et al., 2009; Hughes, 2003; Deb and Jain, 2014), in addition to the originally comprised dominance- (Deb et al., 2002; Zitzler et al., 2002) and indicator-based (Beume et al., 2007; Zitzler and Künzli, 2004; Bader and Zitzler, 2011) algorithms. Specifically, we implement components from relevant decomposition-based MOEAs such as MOPSO (Hughes, 2003) and NSGA-III (Deb and Jain, 2014), and we allow the free hybridization between all three design paradigms considered. In fact, this is the first work to consider such possibility, and it is one of the major contributions of our study. Third, we further exploit our unified definition of populations and archives to demonstrate how metrics that had been originally proposed as archive truncation techniques can be used as components of our general preference relations. We take as example the metric proposed for the adaptive grid archiver of PAES (Knowles and Corne, 2000), and recast it as a diversity component that can be used in combination with any of the other preference components available in our template.

The automatically devised MOEAs produced in this work (dubbed AutoMOEA+ algorithms) present better and/or more robust performance than the state-of-the-art results identified in Bezerra et al. (2018). Specifically, the AutoMOEA+ algorithms consistently outperform the 9 MOEAs (and their variants) used for that investigation, among which we highlight NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2002), IBEA (Zitzler and Künzli, 2004), SMS (Beume et al., 2007),[1] MOEA/D (Zhang and Li, 2007; Zhang et al., 2009), MO-CMA-ES (Igel et al., 2007; Voß et al., 2010), HypE (Bader and Zitzler, 2011), and NSGA-III (Deb and Jain, 2014). Interestingly, almost all novel components implemented in this work appear in the automatically generated designs, and often in ways that are very different from what human designers would tend to do. These results further evidence the need for flexible approaches that can be explored in a systematic, automated, and effective way, as we propose in this article.

On some many-objective scenarios, the challenge posed by the disagreements between performance metrics deceives the automatic methodology into selecting designs that are high performing according to some metrics, but not according to others. More precisely, the traditional approaches to automated algorithm design consider a single performance metric to be optimized, typically runtime or solution quality (Birattari, 2009; Hoos, 2012). The latter is typically assessed through a unary performance metric when dealing with the automatic design of a multiobjective algorithm (Dubois-Lacoste

[1]This algorithm is generally referred to as SMS-EMOA in the literature. Here, we dub it SMS for brevity.

et al., 2011; López-Ibáñez and Stützle, 2012; Bezerra et al., 2016). However, when the number of objectives is large, the disagreements between performance metrics become strong, and designing with a single metric in mind will inevitably lead to a design that is well performing according to some metrics, but poor-performing according to others (Jiang et al., 2014; Bezerra et al., 2018). To overcome this issue, we propose a multiobjective formulation of the design process, following a recent research trend on multiobjective configuration of algorithms (Bezerra et al., 2017a). Specifically, instead of evaluating solution quality through a single performance metric, we consider that the set of metrics used in the assessment should be jointly optimized during design, as in a multiobjective problem. Effectively, we propose a multiobjective design of multiobjective algorithms. Our experimental evaluation, which focuses on the most challenging experimental scenario, demonstrates the effectiveness of this approach. In particular, the newly devised algorithm shows robustness and effectiveness for all metrics considered.

The main contributions of this article can be summarized as follows:

1. An augmented framework for instantiating MOEAs, which comprises the most relevant underlying EAs and design paradigms (dominance-, indicator-, and decomposition-based).
2. An empirical demonstration that state-of-the-art MOEAs for continuous optimization can be automatically designed under different experimental scenarios, and that these designs combine elements from different MOEAs/design paradigms.
3. The proposal of a multiobjective formulation of the automatic MOEA design problem, from which one can automatically devise a state-of-the-art MOEA for MaOPs robust across a set of disagreeing performance metrics.

The remainder of this article is organized as follows. In Section 2, we review the original AutoMOEA framework and detail how we augment it in this work. In Section 3, we automatically design a set of MOEAs that display state-of-the-art performance on most experimental scenarios considered. Next, Section 4 details the multiobjective design formulation adopted for specific many-objective scenarios, and presents the experimental results confirming its effectiveness. We conclude in Section 5.

2 An Augmented MOEA Template

Automated algorithm design approaches observed in the literature can be broadly split into two main categories: (i) *bottom-up* approaches (e.g., hyperheuristics, Ross (2005)), where heuristics are crafted using little human insights and heavily relying on automatically discovered knowledge, and (ii) *top-down* approaches, where human knowledge provides a structural basis (e.g., a template or a grammar) and the automated design process attempts to design the best possible algorithm based on this structure. The scope of the former has been traditionally restricted to the design of heuristics. By contrast, top-down approaches have been increasingly proven effective whether for designing simple heuristics or complex algorithm portfolios (e.g., ensemble methods).[2]

In previous work (Bezerra et al., 2016), we studied the most popular MOEA designs found in the literature and proposed a template of the general design of a MOEA, shown

[2]Configurable algorithmic frameworks, such as ParadisEO-MOEO (Cahon et al., 2004) and Borg (Hadka and Reed, 2013), naturally enable top-down approaches. As we later further detail, the experimental assessment conducted in this work has partially relied on the former.

Algorithm 1 AutoMOEA+ template proposed in this work.

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

Algorithm 2 UnderlyingEA = GA

```
Input: pop
1: pop_new ←∅
2: pool ←Mating_GA(pop)
3: pop_new ←Variation_GA(pool)
4: return pop_new
```

in Algorithms 1 and 2. Each *abstract* component of this template represents a choice between different algorithmic components found in the literature. A component-wise algorithmic framework that implements this template and provides a set of options for each abstract component can instantiate diverse MOEAs. Our original AutoMOEA framework could instantiate at least six of the most relevant dominance- and indicator-based MOEAs from the literature (Fonseca and Fleming, 1993; Deb et al., 2002; Zitzler et al., 2002; Zitzler and Künzli, 2004; Beume et al., 2007; Bader and Zitzler, 2011), in addition to a large number of valid and novel MOEA designs.

In this work, we augment our previously proposed AutoMOEA framework in several directions, hereon called AutoMOEA+. First, we distinguish between multiobjective components and underlying EAs, and allow coupling the same set of MO components with different underlying EAs. This enables the AutoMOEA+ framework to instantiate many MOEAs from the literature that are based on differential evolution (Abbass et al., 2001; Abbass, 2002; Madavan, 2002; Robič and Filipič, 2005; Kukkonen and Lampinen, 2005; Tušar and Filipič, 2007; Tagawa et al., 2011). Second, we incorporate decomposition-based algorithmic components, and model these components in a manner that allows designers to combine, within a single algorithm, components originally proposed for dominance-, indicator-, and decomposition-based MOEAs. Finally, we also recast techniques originally proposed for archive truncation as options available for our preference components.

2.1 Original Template

The main abstract components that characterize a MOEA depicted in Algorithms 1 and 2 are listed in Table 1, where both *atomic* and *composite* components are given. A composite

Table 1: Main algorithmic components of AutoMOEA+.

Component	Parameters
Preference	⟨ SetPart, Refinement, Diversity ⟩
Mating	⟨ Preference_{Mat}, Selection ⟩
Replacement	⟨ Preference_{Rep}, Removal_{Rep} ⟩
Replacement_{Ext}	⟨ Preference_{Ext}, Removal_{Ext} ⟩
UnderlyingEA*	⟨ Mating, Variation ⟩

(*) Novel component implemented in this work.

Table 2: Algorithmic component options available in the AutoMOEA+ framework. The symbol (*) denotes novel components implemented in this work.

Component	Domain
SetPart	none (—) dominance count dominance rank dominance strength dominance depth dominance depth-rank (DR)
Refinement	none (—) binary $I_{\epsilon+}$ binary I_H^- exclusive HV contribution (I_H^1) shared HV contribution (I_H^h) *weighted ranking (w-rank)**
Diversity	none (—) niche sharing (σ_{share}) nearest neighbors (NN) crowding distance *adaptive grid (AGA)** *reference lines**

Component	Domain
Selection	deterministic tourn. (DT) stochastic tourn. (ST) random
Removal	{ sequential, one-shot }
type ($\texttt{pop}$)	{ fixed-size, bounded }
type ($\texttt{pop}_{\texttt{ext}}$)	{ none, bounded, unbounded }
UnderlyingEA *	{ *GA*, *DE* }

Condition	Component	Domain
UnderlyingEA = *DE*	Online Replace*	*none*, *Pareto*, *WeakPareto*
Refin.= *w-rank* or Diversity = *ref. lines*	Λ_{dist}*	*uniform*, *dichotomic*, *two-layer*
Λ_{dist} = *two-layer*	Λ_{focus}*	*peripheral*, *central*, *balanced*

component such as Mating may comprise composite and/or atomic components. Algorithmic components (options for the abstract components) are listed in Table 2, where we make a distinction between components already available in the AutoMOEA framework and components implemented in this work for AutoMOEA+. Below we briefly summarize abstract components of the original AutoMOEA framework; for further details on the original components, we refer to Bezerra et al. (2016).

Preference is a *composite* component that models general preference relations (Zitzler et al., 2010), and comprises a sequence of three *atomic* components in the following order: (1) SetPart partitions solutions into dominance equivalent; (2) Refinement ranks solutions within each partition, e.g., using quality indicators; and (3) Diversity is a Pareto-noncompliant metric used to keep the population well-spread across the objective space. A Preference component can also contain less than three atomic components since SetPart, Refinement, and/or Diversity can be set to *none*.

Mating uses traditional Selection operators to select individuals to undergo variation. In the case of tournaments, solutions are compared based on a preference relation Preference_{Mat}.

The **Replacement** and $\textbf{Replacement}_{Ext}$ components define environmental selection and external archive truncation (if a bounded external archive is used), respectively. Both Replacement components ensure elitism, and comprise two other components, namely, a preference relation used to compare solutions (Preference_{Rep} and Preference_{Ext}, respectively), and a parameter that determines the frequency with which the preference relation is computed (the removal policies Removal_{Rep} and Removal_{Ext}, respectively): with *one-shot* removal, preferences are computed once and replacement takes place; *sequential* removal recomputes preferences every time a solution is discarded.

$\mathbf{pop}$ and $\mathbf{pop}_{\texttt{ext}}$ are sets of solutions that represent either populations or archives. $\texttt{pop}$ takes part in the evolutionary process and can be configured as a regular *fixed-size* population that may contain dominated solutions or as a *bounded-size* archive that only contains nondominated solutions. $\texttt{pop}_{\texttt{ext}}$ is an optional external archive that does not participate in the evolutionary process and may be either *bounded* or *unbounded*.

Initialization and **Variation** comprise problem-specific components, namely, the generation of an initial population and the variation operators that produce new solutions from existing ones.

2.2 Underlying EAs

Most MOEA proposals either specify the underlying EA as an integral part of a MOEA design or treat it as an irrelevant detail. However, our recent experimental assessment has shown that a proper choice of the underlying EA operators is critical to the effectiveness of a MOEA, given the strong interactions between MO-components and EA operators (Bezerra et al., 2018). Therefore, our augmented AutoMOEA+ framework allows the combination of MO-components with two different underlying EAs, namely *genetic algorithms* (Goldberg, 1989) and *differential evolution* (Price et al., 2005). We model the underlying EA as a composite component UnderlyingEA that comprises composite components Mating and Variation (Table 1), since the choice of EA not only affects variation operators but also the selection of the individuals that undergo variation. These two options are further explained next.

Genetic algorithms (GAs) was the only option in our original AutoMOEA framework, and is described in Algorithm 2. When this underlying EA is chosen (option *GA* in Table 2), a mating pool of solutions is built as described by component Mating_{GA}. Component Variation_{GA} comprises the sequential application of (domain-specific) crossover and mutation operators. In this article, we use the well-known SBX crossover and polynomial mutation operators.

Differential evolution (DE) is detailed in Algorithm 3, which generalizes the structure of most DE-based MOEAs. Component Mating_{DE} selects target and donor vectors, and component Variation_{DE} creates a trial vector through differential mutation and binomial crossover. A distinguishing feature of multi-objective DE algorithms is the OnlineReplacement component (Madavan, 2002; Robič and Filipič, 2005; Kukkonen and Lampinen, 2005). When this component is active, a newly created trial solution can immediately replace the target vector if a given acceptance criterion is satisfied. Some popular DE algorithms differ exactly in this acceptance criterion: DEMO (Robič

Algorithm 3 UnderlyingEA = DE

Input: $\mathtt{pop}$
1: $\mathtt{pop}_{\mathtt{new}} \leftarrow \varnothing$
2: **for** $i = 1$ **to** λ **do**
3: $\{target, donor\} \leftarrow \mathsf{Mating}_{DE}(\mathtt{pop})$
4: $trial \leftarrow \mathsf{Variation}_{DE}(target, donor)$
5: $S \leftarrow \mathsf{OnlineReplace}(\mathtt{pop}, target, trial)$
6: $\mathtt{pop}_{\mathtt{new}} \leftarrow \mathtt{pop}_{\mathtt{new}} \cup S$
7: **return** $\mathtt{pop}_{\mathtt{new}}$

and Filipič, 2005) uses Pareto dominance, whereas GDE3 (Kukkonen and Lampinen, 2005) uses weak Pareto dominance. We have added both options to our AutoMOEA+ framework (Table 2). If the target vector is replaced, then the size of $\mathtt{pop}_{\mathtt{new}}$ does not increase; else, if trial and target are nondominated (or if online replacement is not adopted at all), *trial* is added to $\mathtt{pop}_{\mathtt{new}}$. This is represented in Algorithm 3 by the set S produced by OnlineReplacement and later added to $\mathtt{pop}_{\mathtt{new}}$: if *trial* replaces *target*, S is an empty set and $\mathtt{pop}_{\mathtt{new}}$ remains unchanged; else, S is a singleton containing only *trial*, which is added to $\mathtt{pop}_{\mathtt{new}}$. Online replacement would be redundant together with steady state selection ($\lambda = 1$) since it becomes equivalent to Replacement.

Another novelty of our work is the possibility of using multiple DE schemes (Price et al., 2005). Specifically, so far only the *DE/rand/1* scheme has been adopted in the literature. Here, we also implement a preference-based selection scheme, which is an adaptation of the *DE/target-to-best/1* scheme (*TtoB*, for short). Concretely, designers may configure Mating_{DE} to select target and donor vectors using any $\mathsf{Preference}_{Mat}$ and Selection options, thus increasing the odds of producing a better trial vector. However, this scheme cannot be adopted in combination with online replacement, since $\mathsf{Preference}_{Mat}$ is computed before variation starts, and a trial vector replacing a target vector during variation would lead to an inconsistently evaluated population.[3]

2.3 Deconstructing Decomposition

Decomposition (Hughes, 2003; Zhang and Li, 2007; Zhang et al., 2009; Deb and Jain, 2014) is a search paradigm originally considered by the decision making community and adapted for MOEA research. The basic principle behind this paradigm is to decompose the original MOP into subproblems and optimize them in parallel. Each subproblem is a single-objective projection of the original MOP, which can be obtained using several different methods (Zhang and Li, 2007). An analysis of the decomposition-based MOEA literature reveals that most proposals can be classified as Refinement components. Specifically, most decomposition-based algorithms are able to simultaneously evaluate the convergence of a population (its closeness to the Pareto front) and its diversity (how well the front is covered), with the latter being ensured by the existence of multiple subproblems and the former by optimizing each subproblem. More importantly,

[3]It would also be possible to use a similar modeling to comprehend other underlying EAs, such as CMA-ES and even *particle swarm optimization* (PSO, Eberhart and Kennedy (1995)). However, this would require operators aware of population-related aspects, such as neighborhood topology in PSO. We leave such investigation for future work.

decomposition approaches are able to distinguish between dominance-equivalent solutions, the baseline definition for our Refinement components. One exception is NSGA-III (Deb and Jain, 2014), which uses decomposition only for diversity purposes and ensures convergence using the same SetPart component as NSGA-II (Deb et al., 2002). In this work, we implement two components from decomposition-based MOEAs:

Weighted ranking was originally proposed in MOPSO (Hughes, 2003). In our framework, it is provided as an option of component Refinement, and works as follows. Solutions are ranked according to their performance on each subproblem defined by a weight vector $\lambda \in \Lambda$, where Λ is a given set of weight vectors. The overall quality of a solution equals its aggregated performance considering the ranks from each subproblem. We use here the algebraic sum to aggregate the performance on the subproblems, but other aggregation functions are possible.

Reference lines correspond to the method used in NSGA-III to keep the population spread along the Pareto front. Thus, in our framework, it is an option of component Diversity. A reference line is the line intersecting the origin of the axes and a reference point defined by a weight vector $\lambda \in \Lambda$. When ranking solutions, each solution is first associated to its nearest reference line in terms of perpendicular distance in the objective space. Next, niche counts are computed for each reference line considering only solutions already selected for the next iteration by previous preference components. For example, when dominance depth is used as SetPart in NSGA-III, the selected solutions are those from the lowest depth fronts that fit in the next population. Finally, the procedure iteratively selects the reference line with lowest niche count and adds one of its associated solutions to the next population. Since we did not find a straightforward way to differentiate between a one-shot and a sequential Removal policy for this reference-lines procedure, we only combine it with the one-shot policy.

Two additional parameters are crucial in decomposition-based algorithms: the cardinality and the distribution of the generated weight set Λ. In our framework, the cardinality of Λ is upper-bounded by $\Lambda_r \cdot \mu$, where Λ_r is a numerical parameter and μ is the population size. If the number of weights generated by some method exceeds the upper bound, excessive weights are discarded at random. For the distribution of Λ, our framework provides methods for generating weights with a *uniform* distribution (Das and Dennis, 1997), the *dichotomic* method proposed by Aneja and Nair (1979) for bi-objective problems, and the *two-layer* method of NSGA-III (Deb and Jain, 2014) for many-objective problems. This latter method uses two numerical parameters H_1 and H_2 to determine how many weights will be generated using a uniform distribution in the outer and inner layers, respectively.[4] However, the effective goal of these parameters is to determine the *search focus* a designer wants to use, i.e., the proportion between weight vectors in the two layers. Rather than configuring these parameters independently, we provide a set of options Λ_{focus}, as follows:

Peripheral focus favors the outer layer by setting H_1 to the maximum value feasible so that there exists an H_2 value for which $|\Lambda| \leq \Lambda_r \cdot \mu$.

[4] A weight vector belongs to the outer layer if at least one of its components equals zero; otherwise it belongs to the inner layer.

Central focus favors the inner layer by setting H_2 to the maximum value feasible so that there exists an H_1 value for which $|\Lambda| \leq \Lambda_r \cdot \mu$.

Balanced tries to balance the importance of both layers. Concretely, H_1 and H_2 are set to a maximum feasible value h so that $|\Lambda| \leq \Lambda_r \cdot \mu$. If, however, it is still possible to increase either H_1 or H_2, that parameter is increased to prevent wasting weights.

2.4 Archive Truncation Techniques

Many different archive truncation techniques, or *archivers*, for short, have been proposed in the literature, as reviewed by López-Ibáñez et al. (2011), who consider that metrics proposed for environmental selection and metrics specifically proposed to keep an external archive bounded both serve the same purpose, and should altogether be considered archivers. In the original AutoMOEA framework we have followed this formulation and, if configured as bounded-size archives, the archiving of both pop and $\texttt{pop}_{\texttt{ext}}$ are commonly defined by Replacement components. In Bezerra et al. (2016), we have demonstrated the benefits of this formulation by recasting metrics originally proposed for environmental selection into archivers.

In this work, we further demonstrate the benefits from this formulation by recasting metrics originally proposed to keep external archives bounded into Preference components that can be used for any preference-based selection. In addition, this formulation allows the free hybridization of archivers with other metrics. We take as example the *adaptive grid archiver* (AGA) from PAES (Knowles and Corne, 2000), which discretizes the objective space into grid cells that are dynamically computed as a function of the extreme solutions found during the run, and of a numerical parameter that specifies the number of cells per objective. Solutions are compared based on the crowdedness of the grid cell to which they belong, with less crowded regions being favored. In our framework, we model the adaptive grid approach as a Diversity component. As part of a Preference component, it can be used as the selection criterion for building the mating pool, as the replacement strategy of pop or as an archive truncation technique. For example, one could configure Mating to use a Preference relation that combines the decomposition-based weighted ranking as Refinement component with the adaptive grid approach as a Diversity component. Indeed, this is an improvement over the original applications of the AGA (Knowles and Corne, 2000), which had already been considered for mating selection, but in a more simplified Preference component.

The extensions detailed above over the original AutoMOEA framework greatly improve the flexibility of the resulting AutoMOEA+ framework, enabling us to automatically design state-of-the-art MOEAs for multi- and many-objective continuous optimization, as we discuss next.

3 Automatically Designing Effective MOEAs

In this section, we automatically design MOEAs by configuring our AutoMOEA+ framework using irace (López-Ibáñez et al., 2016), an automatic algorithm configuration tool. Our experimental analysis of the automatically designed MOEAs (hereon called AutoMOEA+ algorithms) has three main goals. First, we analyze to what extent the AutoMOEA+ algorithms match what human designers would choose as effective components, and whether the new components added to the framework appear in the AutoMOEA+ algorithms. Second, we assess whether the AutoMOEA+ algorithms can outperform the ones generated from our previous AutoMOEA framework (Bezerra et al.,

Table 3: Parameter space for tuning parameters of the AutoMOEA template.

Parameter	Domain
$\mu = \lvert\texttt{pop}\rvert$	$\{10, 20, \ldots, 100\}$
$\lambda = \lvert\texttt{pop}_{\texttt{new}}\rvert$	1 or $\lambda_r \cdot \mu$
λ_r	$[0.1, 2]$

Condition	Add. param.	Domain
type (pop) = *bounded*	μ_0	$\mu_r \cdot \mu$
	μ_r	$[0.1, 1]$
type ($\texttt{pop}_{\texttt{ext}}$) = *bounded*	N_{ext}	$\{100, 300, 500\}$
UnderlyingEA = *GA*	p_c, p_m	$[0, 1]$
	η_c, η_m	$\{1, \ldots, 50\}$
	MutScheme	{*bitwise*, *fixed*}
MutScheme = *bitwise*	p_v	$1/n_{\text{var}}$
MutScheme = *fixed*	p_v	$[0.01, 1]$

Condition	Add. param.	Domain
UnderlyingEA = *DE*	*CR*	$[0.01, 1]$
	F	$[0.1, 2]$
	Scheme	{*rand*, *TtoB*}
Selection = *ST*	γ	$[0.6, 0.9]$
Diversity = *sharing*	σ_{share}	$[0.1, 1]$
Diversity $_{Mat}$ = *NN*	k_{method}	{*default*, k}
	k	$\{1, \ldots, 9\}$
Diversity = *AGA*	l	$\{1, \ldots, 4\}$
Refine = *weighted rank.* or Diversity = *ref. lines*	Λ_{r}	$[0.5, 2]$

2016), and how they differ. Third, we assess whether the AutoMOEA+ algorithms are able to outperform the state-of-the-art MOEAs identified in Bezerra et al. (2018).[5]

3.1 Parameter Space of AutoMOEA+

The parameter space of the AutoMOEA+ framework contains, besides numerical MOEA parameters such as population size (μ) and number of offspring (λ) shown in Table 3, parameters for selecting options for abstract algorithmic components that define the MOEA design (Tables 1 and 2), and conditional parameters that need to be set if particular options are selected. Conditional parameters are listed in Table 2, already explained in the previous section, and in Table 3, which we detail next. A first group of parameters concerns archives. When pop is configured as a bounded-size archive instead of a fixed-size population, μ is interpreted as its maximum capacity and the initial number of solutions is given by $\mu_0 = \mu_r \cdot \mu$. When a bounded-size external archive $\texttt{pop}_{\texttt{ext}}$ is used, its capacity is given by N_{ext}. The next group of parameters concerns the underlying EA. When GA is selected, p_{m} and p_{c} give the probability of applying polynomial mutation and SBX crossover, respectively, for each individual or pair thereof. These operators have associated distribution indices η_{m} and η_{c} that must be configured. Our framework implements two different mutation schemes for real-parameter optimization (Deb and Deb, 2014): *bitwise* sets the mutation probability per variable p_{v} to $1/n_{\text{var}}$; *fixed* leaves p_v as a parameter to be configured. When the underlying EA is set to DE, only two parameters must be set, namely, the crossover probability (*CR*) and the scaling

[5]We have empirically verified that the performance of the implementations used in this work match what has been reported in the original papers. Specifically, we primarily adopt MOEA implementations from well-established frameworks such as ParadisEO-MOEO (Cahon et al., 2004), Shark (Igel et al., 2008), PaGMO (Biscani et al., 2010), and PISA (Bleuler et al., 2003). When unavailable from those sources, we adopt either official implementations (Zhang, 2007) or implement MOEAs ourselves reusing available code (DE-based and NSGA-III). In the case of NSGA-III, the core implementation is provided by Chiang (2014), from which we fixed existing issues.

Table 4: Experimental setup used for the design of the AutoMOEA+ algorithms. The differences in setup w.r.t. to the design of the original AutoMOEAs are underlined.

Factor	Details
Problems	DTLZ{2, 4–7} and WFG1–9
M	{2, 3, 5, <u>10</u>}
n_{var}	$\{20, 21, \ldots, 60\}$
$FE_{\max}$	{<u>2 500</u>, 10 000, <u>40 000</u>}
n_{testing}	{30, 40, 50}
n_{tuning}	$n_{\text{var}} \setminus n_{\text{testing}}$
$t_{\max}$	1h ($FE_{\max}$ = 2 500) 10min (otherwise)

Factor	Details
Configurator	irace
Configuration budget	20 000 MOEA runs
Configuration metric	$I_{\epsilon+}$ ($M = 10$), I_H^{rd} (otherwise)
Test metrics	I_H^{rd}, $I_{\epsilon+}$, <u>I_{IGD}</u>
Test repetitions	25
Upper bound ($\mathbf{u}$)	$\begin{cases} [10]^M, & \text{if } M \in \{2,3\} \\ [15]^5, & \text{if } M = 5 \\ [25]^{10}, & \text{if } M = 10 \end{cases}$
Reference point ($\mathbf{r}$)	$1.1 \cdot \mathbf{u}$

factor (F) of the DE operators. Whatever the underlying EA, when the Selection component of Mating is set to deterministic tournament (*DT*), then TournamSize controls the tournament size; if it is set to stochastic tournament (*ST*), then γ controls the probability of selecting the best contestant as the winner of a binary tournament. The last group of conditional parameters concerns Preference components. When the *sharing* diversity metric is selected, the radius of the niches (σ_{share}) must be configured. When the diversity metric is based on nearest neighbors (*NN*), mating considers the distance to the k-th nearest one, whereas replacement behaves as a nearest neighbor density estimation (see Bezerra et al., 2016 for details). When the AGA diversity metric is adopted, the number of grid cells is computed as a function of a discretization parameter l. Finally, if a decomposition-based preference component is selected, the upper bound size for the weight set Λ is a function of a numerical parameter Λ_r.

We remark that the parameter space used to configure the AutoMOEA+ algorithms ensures fairness in the comparisons to the original AutoMOEAs and also to the state-of-the-art MOEAs. Specifically, the domains we adopt in this work are reused from the original proposal of the AutoMOEA framework when the given component was already available in that work (Bezerra et al., 2016). In addition, MOEAs in the state-of-the-art assessment were given the same underlying EA choice and their associated parameters were configured using the same domains also adopted here (Bezerra et al., 2018).

3.2 Automatic Design Setup

The idea of automatic MOEA design by coupling a component-wise MOEA framework with an automatic configuration tool was originally proposed in Bezerra et al. (2016); thus, we refer to the original publication for the general details of the proposal. Here, we focus on the setup used in the present article, and later we briefly highlight relevant differences between the setups used for configuring AutoMOEA and AutoMOEA+. Several elements are needed to set up an automatic design scenario: the benchmark problems, the stopping criteria for the MOEAs, the parameters of the automatic configurator, and the unary performance metric that guides the configurator. Since we wish to compare with the state-of-the-art MOEAs from the literature previously identified in Bezerra et al. (2018), we use the same setup we adopted for tuning MOEAs in that work, summarized in Table 4 and detailed below. The values of M and $FE_{\max}$ and the

I_{IGD} metric that had not been considered for the design of the original AutoMOEAs are underlined.

Benchmark problems. We consider the box-constrained WFG (Huband et al., 2006) and most DTLZ (Deb et al., 2005) benchmark problems (Problems DTLZ1 and DTLZ3 are excluded due to ceiling effects (Bezerra et al., 2018)) with $M \in \{2, 3, 5, 10\}$ objectives and $n_{\text{var}} \in \{20, 21, \ldots, 60\}$ variables. We reserve sizes $n_{\text{testing}} \in \{30, 40, 50\}$ for comparing algorithms and only use sizes $n_{\text{var}} \setminus n_{\text{testing}}$ for the automatic design process to separate between training and testing sets.

Stopping criterion of MOEAs. Each MOEA run is stopped after using a maximum of $FE_{\max}$ function evaluations. Here, we evaluate several values of $FE_{\max} \in \{2\,500, 10\,000, 40\,000\}$. In addition, we set a maximum time limit per run to prevent very long runs making the automatic design process infeasible. The time limit is long ($t_{\max} =$ 1 hour) for $FE_{\max} = 2\,500$ assuming that such value represents an expensive evaluation scenario. Otherwise, the maximum time limit is $t_{\max} = 10$ minutes.

Configurator setup. We use irace (López-Ibáñez et al., 2016) as configurator. Given a set of training benchmark functions and a parameter space description, irace searches for good parameter configurations by evaluating configurations of the target algorithm (in our case, instantiations of the AutoMOEA+ framework) on the benchmark functions according to a given unary quality metric. Describing irace in detail is outside the scope of the article and more details can be found in López-Ibáñez et al. (2016). We run irace with its default settings and each run of irace has a budget of 20 000 MOEA runs.

Unary quality metrics. Since irace is a single-objective optimizer, it uses unary quality metrics to evaluate the performance of a MOEA run. In particular, we use the unary ϵ-metric ($I_{\epsilon+}$) for large number of objectives ($M = 10$) and the relative deviation from an approximation of the optimal hypervolume (I_H^{rd}) otherwise.[6] Before computing a metric, we discard objective vectors that exceed the bounds ($\mathbf{u}$) given in Table 4 to avoid strong outliers that would skew results. Finally, the nadir point in the computation of the I_H^{rd} is $\mathbf{r} = 1.1 \cdot \mathbf{u}$ to ensure extreme solutions contribute to the hypervolume.

With the above setup, we run irace once for each combination of M and $FE_{\max}$, resulting in 12 different AutoMOEA+ algorithms. For brevity, we refer to a particular scenario using the $\langle M, FE_{\max} \rangle$ notation, e.g., $\langle 2,\ 10k \rangle$ refers to a scenario where problems present two objectives and MOEAs are allowed to use 10 000 FEs. In order to assess their quality, we run each AutoMOEA+ algorithm on each of the benchmark functions with sizes n_{testing}. We perform 25 repetitions of each run and compute the mean value for each quality metric $I_{\epsilon+}$, I_H^{rd}, and inverted generational distance (I_{IGD}).[7] All experiments are run on a single core of Intel Xeon E5410 CPUs @ 2.33 GHz with 6 MB cache size under Cluster Rocks Linux version 6.2/CentOS 6.2.

[6] All metrics considered in this work and also in our state-of-the-art assessment (Bezerra et al., 2018) are unary formulations of binary metrics, using reference fronts for a common comparison. We adopt this formulation to avoid the total number of comparisons that truly binary metrics would require. For further details on the metrics and reference fronts used for their computation, see Bezerra et al. (2018).

[7] We are aware that IGD is not Pareto-compliant under some conditions. Yet, we use it to reproduce the setup used in a major share of the literature on many-objective EAs and also in our state-of-the-art assessment (Bezerra et al., 2018).

3.3 Trends from the Generated AutoMOEA+ Algorithms

The designs of the automatically designed AutoMOEA+ algorithms are given in Table 5. Although it is not possible to tell whether a particular component significantly contributes to the performance of a MOEA design without a component-by-component analysis (Fawcett and Hoos, 2016) and possibly various repetitions of the configuration process for each scenario, some trends appear in the designs that have been obtained for the 12 different scenarios.

We first focus on the general trends among the AutoMOEA+ designs for scenarios with a number of objectives $M \in \{2, 3, 5\}$ (top three blocks of Table 5, with a grey background), as these designs share a number of similarities. First, the underlying EA choice differs from what human designers have typically adopted in the literature, as DE was always selected instead of GA. Even concerning the multiobjective DE literature we see a contrast to the traditional designs, since irace always chooses the preference-based scheme (i.e., DE/target-to-best/1, which we propose here) and, as a consequence, never the online replacement. However, it is difficult to find an overall pattern for Mating, though deterministic tournaments are used more often than stochastic ones. Second, in contrast to Preference_{Mat}, patterns are clear for component Replacement, where hypervolume-based Refinement is almost always used, and scenarios are nearly evenly split between using steady-state selection or sequential replacement. Finally, external archives are more frequently used for lower M values, and the same pattern is observed for Refinement components in Replacement_{Ext}.[8]

We next discuss the similarities and differences in the structure of the AutoMOEA+ algorithms focusing on the experimental factors that constitute scenarios:

- M: First, while all $M < 10$ designs use UnderlyingEA $= DE$, two of three $M = 10$ designs adopt UnderlyingEA $= GA$. A second affected component is Refinement. The I_H^h component is clearly frequent in bi-objective scenarios, whereas the I_H^1 indicator is chosen more frequently when $M \in \{3, 5\}$. Likely, this is explained by the computational overhead of the I_H^h indicator since, when $M = 3$, no Refinement component is used for the external archives except for the scenario with a larger cutoff time. When $M = 10$, $I_{\epsilon+}$ becomes the standard refinement option for Preference_{Rep}, but further investigation would be required to determine if this is a consequence of changing the performance metric for irace when $M = 10$. Finally, the occurrences of decomposition-based components increase as M grows: component weighted rank is the most selected Refinement option for mating selection when $M = 5$, and all algorithms for $M = 10$ use at least one decomposition-based component.
- $FE_{\max}$: The most evident insight we observe is that external archives tend to become prohibitive when this budget is increased but the maximum runtime is kept constrained. This is initially observed for scenarios with $M = 3$, where only Diversity components are used when $FE_{\max} \in \{10\,000, 40\,000\}$, and made worse on scenarios with $M = 5$, where external archives are not used at all when $FE_{\max} \in \{10\,000, 40\,000\}$. The extreme situation is observed for $\text{AutoMOEA+}_{\langle 10, 40k \rangle}$, where no refinement metrics nor external archives are used. An exception to this pattern

[8] One may assume that using an external archive is always better than not using it when function evaluations are used as stopping criterion, but the time-constrained setups we adopt help explain the cases in which no external archive is used.

Table 5: Configurations of the AutoMOEA+ framework selected by irace for each $\langle M, FE_{\max}\rangle$ design scenario. Each row describes a different AutoMOEA+ design. The complete set of parameters is provided as supplementary material (Bezerra et al., 2017b). Notice that $\lambda = 1$ indicates steady-state selection.

	Mating				Replacement				Replacement$_{Ext}$				pop				Mating$_{DE}$
$\langle M, FE_{\max}k\rangle$	Selection	SetPart	Refine	Diversity	SetPart	Refine	Diversity	Removal	N_{ext}	Refine	Diversity	Removal	*type*	μ	μ_r	λ	*Scheme*
$\langle 2, 2.5k\rangle$	DT(8)	strength	$I_{\epsilon+}$	—	—	I_H^h	—	—	300	I_H^h	NN	1-shot	*fixed-size*	60	—	1	*TtoB*
$\langle 2, 10k\rangle$	DT(4)	depth	I_H^1	NN	DR	I_H^h	—	—	—	—	—	—	*bounded*	100	0.44	1	*TtoB*
$\langle 2, 40k\rangle$	ST(0.79)	—	—	sharing	—	$I_{\epsilon+}$	sharing	seq.	500	I_H^h	crowd.	1-shot	*bounded*	60	0.84	λ_r	*TtoB*
$\langle 3, 2.5k\rangle$	DT(8)	DR	I_H^1	crowding	rank	I_H^1	NN	—	500	I_H^h	ref. lines[a]	1-shot	*fixed-size*	60	—	1	*TtoB*
$\langle 3, 10k\rangle$	ST(0.77)	—	I_H^1	AGA (1)	—	I_H^1	AGA(2)	seq.	500	—	AGA(3)	seq.	*fixed-size*	40	—	λ_r	*TtoB*
$\langle 3, 40k\rangle$	DT(2)	—	I_H^1	—	—	I_H^1	—	seq.	500	—	sharing	seq.	*bounded*	90	0.76	λ_r	*TtoB*
$\langle 5, 2.5k\rangle$	DT(4)	depth	w-rank[b]	crowd.	rank	I_H^1	crowd.	—	300	—	AGA(2)	1-shot	*fixed-size*	30	—	1	*TtoB*
$\langle 5, 10k\rangle$	DT(4)	DR	w-rank[c]	AGA(1)	depth	I_H^1	sharing	—	—	—	—	—	*fixed-size*	70	—	1	*TtoB*
$\langle 5, 40k\rangle$	ST(0.82)	—	I_H^1	NN(8)	—	I_H^1	crowd.	seq.	—	—	—	—	*fixed-size*	60	—	λ_r	*TtoB*
$\langle 10, 2.5k\rangle$	random	—	—	—	—	$I_{\epsilon+}$	NN	—	500	w-rank	ref. lines	1-shot	*fixed-size*	50	—	1	*rand*
$\langle 10, 10k\rangle$	random	—	—	—	rank	$I_{\epsilon+}$	sharing	—	500	I_H^h	ref. lines	1-shot	*fixed-size*	90	—	λ_r	—
$\langle 10, 40k\rangle$	ST(0.84)	count	—	sharing	DR	—	ref. lines	—	—	—	—	—	*fixed-size*	20	—	1	—

[a] with $\Lambda_r = 1.15$, $\Lambda_{\text{dist}} =$ uniform; [b] with $\Lambda_r = 1.54$, $\Lambda_{\text{dist}} =$ two-layer, $\Lambda_{\text{focus}} =$ peripheral; [c] with $\Lambda_r = 0.98$, $\Lambda_{\text{dist}} =$ uniform

Table 6: Structure of the AutoMOEAs automatically designed in Bezerra et al. (2016). Each row represents an AutoMOEA version, designed for a specific benchmark (D: DTLZ; W: WFG), $M \in \{2, 3, 5\}$, and $FE_{\max} = 10\,000$.

	Mating				Replacement				Replacement$_{Ext}$			*Num.*
	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	Quality	Diversity	Removal	μ
D2	DT (8)	—	—	crowd.	DR	$I_{\epsilon+}$	sharing	—	—	crowd.	seq.	100
D3	DT (8)	DR	$I_{\epsilon+}$	NN	rank	I_H^1	sharing	—	I_H^1	—	seq.	80
D5	DT (8)	rank	I_H^1	crowd.	depth	I_H^1	—	—	$I_{\epsilon+}$	crowd.	1-shot	40
W2	DT (8)	rank	—	crowd.	DR	I_H^1	—	—	I_H^h	crowd.	1-shot	20
W3	DT (4)	count	I_H^1	crowd.	strength	I_H^1	sharing	seq.	I_H^1	NN	seq.	10
W5	DT (8)	count	I_H^h	crowd.	—	I_H^1	—	seq.	I_H^h	crowd.	1-shot	30

(All AutoMOEAs use $N_{\text{ext}} = 500$, except Auto$_{\text{W2}}$ which uses $N_{\text{ext}} = 300$. In addition, all but Auto$_{\text{D2}}$ and Auto$_{\text{D3}}$ use *type* (pop) = *fixed-size*, and all but Auto$_{\text{W3}}$ and Auto$_{\text{W5}}$ use steady-state replacement ($\lambda = 1$).)

is AutoMOEA+$_{\langle 10,\, 10k\rangle}$, which uses an external archive with hypervolume estimation.[9] Yet, this AutoMOEA+ is the only algorithm for $M = 10$ scenarios that does not use steady-state selection, so the adoption of a more costly external archive may be a design compromise between expensive components.

Finally, we observe that the design of AutoMOEA+$_{\langle 10,\, 40k\rangle}$ differs the most from all other AutoMOEA+ designs. Specifically, this algorithm resembles NSGA-III in its randomized mating selection, absence of refinement metrics, and use of GA. However, given that no external archive is used, the population size is rather small for a many-objective scenario. As we will discuss later, this design conducts a search that is too restricted in the objective space, and yet it performs well according to the $I_{\epsilon+}$ indicator.

3.4 Comparison between Designs from AutoMOEA+ and AutoMOEA

To assess the improvements provided by the extensions proposed in this work, we first compare the AutoMOEA+ algorithms to the AutoMOEAs designed in Bezerra et al. (2016). In particular, the AutoMOEAs were created for scenarios with $FE_{\max} = 10\,000$ and $M \in \{2, 3, 5\}$, and we therefore only compare the algorithms on these scenarios; that is, in Table 5, only the rows $\langle 2,\, 10k\rangle$, $\langle 3,\, 10k\rangle$, and $\langle 5,\, 10k\rangle$ are considered in the following discussion. The AutoMOEAs have been tuned separately for the DTLZ and WFG benchmarks, while AutoMOEA+ is tuned across the two benchmark sets. Hence, the AutoMOEAs benefit potentially stronger from tuning than the AutoMOEA+ algorithms (or the state-of-the-art MOEAs, which use the same setup as the AutoMOEA+ algorithms). Thus, the results in favor of the AutoMOEA+ algorithms are even more remarkable.

We start with a structural comparison of the AutoMOEA and AutoMOEA+ designs. To aid this analysis we show in Table 6 the structure of the AutoMOEAs from Bezerra et al. (2016). The main trends are the following. As previously discussed, DE is always used in the AutoMOEA+ algorithms. This design choice highlights the importance of providing different underlying EAs for a component-wise design, as for the AutoMOEAs only the GA operators have been available. Selection approaches are similar between AutoMOEA and AutoMOEA+ designs, as tournaments are always used.

[9] When $M > 3$, component I_H^h uses a Monte Carlo estimation instead of an exact computation.

However, the tournaments from the original AutoMOEAs are deterministic and enforce greater convergence pressure due to the choice of four-ary (once) and eight-ary (five times) tournaments. This design difference is likely explained by the different underlying EAs used. While environmental selection is similar in all designs, they differ in the usage of external archives, which AutoMOEAs select more often.

We proceed to a performance comparison, and to this end a rank sum analysis is given in Table 7 for all $FE_{\max} = 10\,000$ scenarios. The analysis in this section focuses on the first three scenarios, for which the AutoMOEAs were originally designed. Moreover, for a particular scenario, the entry labeled as Auto represents an aggregation of results from the AutoMOEAs designed for each benchmark on that scenario. For instance, the rank sum entry labeled as Auto on scenario $\langle 2,\ 10k \rangle$ considers runs from $\text{AutoMOEA}_{\text{D2}}$ on the DTLZ benchmark and from $\text{AutoMOEA}_{\text{W2}}$ on the WFG benchmark. Thus, the AutoMOEAs have the advantage of being two separate MOEAs tuned for each specific benchmark, while the AutoMOEA+ algorithms are a single design that must generalize over both benchmark sets.

Nevertheless, as seen in Table 7, the rank sums achieved by the AutoMOEA+ algorithms are statistically significantly better than the sums achieved by the original AutoMOEAs when $M \in \{2, 3\}$, whichever the metric considered. When $M = 5$, all automatically designed MOEAs reach statistically equivalent results, but the original AutoMOEAs achieve the lowest rank sums for the $I_{\epsilon+}$ and I_{IGD} metrics. Two factors can help explain this result. First, the disagreements between performance metrics are known to increase with the increase in M (Jiang et al., 2014). Second, the AutoMOEAs were custom-designed for each of the benchmarks. With the increase in M, it is natural that the difficulty posed by the benchmarks also increases (Bezerra et al., 2018). Given that each benchmark comprises problems with different characteristics, it seems natural that the need for specialized algorithmic components become stronger.

3.5 State-of-the-Art Comparison

To assess whether the performance of the AutoMOEA+ algorithms can match that of the state-of-the-art algorithms identified in Bezerra et al. (2018), we start with the aggregative analysis depicted by the rank sums given in Table 7, focusing on the comparison of AutoMOEA+ algorithms to the best-performing MOEAs from the literature.

Results shown in Table 7 lead to different conclusions depending on the given performance metric. In general, the AutoMOEA+ algorithms are either the top-ranking algorithms or at least statistically equivalent to the best-performing manually designed MOEA, which may vary according to the scenario and/or metric. Differences in favor of the AutoMOEA+ algorithms are always statistically significant according to I_{IGD}, only for $M \in \{2, 3\}$ scenarios according to $I_{\epsilon+}$, and never according to I_H^{rd}. Indeed, it never happens that an AutoMOEA+ algorithm is statistically significantly better than the best-performing manually designed MOEA according to the metric used for tuning (I_H^{rd} when $M < 10$; $I_{\epsilon+}$, otherwise). Overall, these results lead to two important conclusions. First, they effectively mean that a single MOEA, automatically designed for a given scenario, is able to perform at least as well as, and in many cases better than, the best manually designed MOEAs, even after their parameters have been properly tuned. Second, we see that the disagreements between metrics like I_H^{rd} and I_{IGD} can be overcome to some extent by the automatic design process given that, even if differences in performance according to I_H^{rd} are not statistically significant, differences for I_{IGD} are. Altogether, the results confirm that the AutoMOEA+ algorithms present either the most effective or the most robust performance for the scenarios considered so far.

Table 7: Rank sum difference (in parentheses) between the given MOEA and the lowest ranked for different scenarios. MOEAs highlighted in boldface present rank sums statistically significantly lower than the others according to Friedman's test. Notice that results for scenario $\langle 10,\ 10k \rangle$ do not include an AutoMOEA, since this scenario was not considered in the original proposal of the AutoMOEA framework (Bezerra et al., 2016). Moreover, state-of-the-art MOEAs are run as configured in Bezerra et al. (2018); for details on the selected underlying EA for a given MOEA on a given scenario, we refer to that work.

					Scenario $\langle 2,\ 10k \rangle$					
I_H^{rd}	**Auto+**	**IBEA** (37)	SMS (56)	Auto (92)	SPEA2 (99)	NSGA-II (113)	CMA (193)	HypE (194)	MOEA/D (196)	NSGA-III (237)
$I_{\epsilon+}$	**Auto+**	SMS (59)	IBEA (64)	SPEA2 (99)	NSGA-II (129)	Auto (141)	HypE (186)	CMA (210)	MOEA/D (220)	NSGA-III (248)
I_{IGD}	**Auto+**	SMS (48)	SPEA2 (75)	IBEA (104)	HypE (139)	NSGA-II (157)	Auto (160)	CMA (268)	NSGA-III (268)	MOEA/D (277)
					Scenario $\langle 3,\ 10k \rangle$					
I_H^{rd}	**Auto+**	**SMS** (34)	IBEA (75)	Auto (79)	SPEA2 (143)	HypE (158)	MOEA/D (159)	CMA (204)	NSGA-II (248)	NSGA-III (253)
$I_{\epsilon+}$	**Auto+**	SMS (70)	IBEA (96)	Auto (166)	SPEA2 (177)	CMA (208)	HypE (223)	MOEA/D (252)	NSGA-III (267)	NSGA-II (293)
I_{IGD}	**Auto+**	IBEA (88)	SMS (118)	Auto (129)	SPEA2 (139)	MOEA/D (187)	HypE (213)	NSGA-II (252)	CMA (278)	NSGA-III (280)
					Scenario $\langle 5,\ 10k \rangle$					
I_H^{rd}	**Auto+**	**SMS** (3)	**Auto** (26)	MOEA/D (99)	IBEA (109)	CMA (165)	SPEA2 (172)	NSGA-II (201)	NSGA-III (218)	HypE (267)
$I_{\epsilon+}$	**Auto**	**Auto+** (10)	**SMS** (31)	IBEA (104)	MOEA/D (138)	CMA (146)	NSGA-II (209)	NSGA-III (231)	SPEA2 (241)	HypE (283)
I_{IGD}	**Auto**	**Auto+** (17)	SMS (65)	IBEA (146)	NSGA-II (163)	MOEA/D (167)	CMA (177)	SPEA2 (225)	HypE (280)	NSGA-III (300)
					Scenario $\langle 10,\ 10k \rangle$					
I_H^{rd}	**IBEA**	**SMS** (13)	**Auto+** (39)	CMA (63)	SPEA2 (123)	NSGA-III (124)	NSGA-II (164)	MOEA/D (230)	HypE (234)	
$I_{\epsilon+}$	**MOEA/D**	**Auto+** (39)	IBEA (67)	SMS (148)	CMA (150)	NSGA-III (158)	NSGA-II (176)	SPEA2 (206)	HypE (244)	
I_{IGD}	**Auto+**	NSGA-III (65)	IBEA (67)	SPEA2 (96)	CMA (140)	NSGA-II (140)	SMS (142)	HypE (173)	MOEA/D (230)	

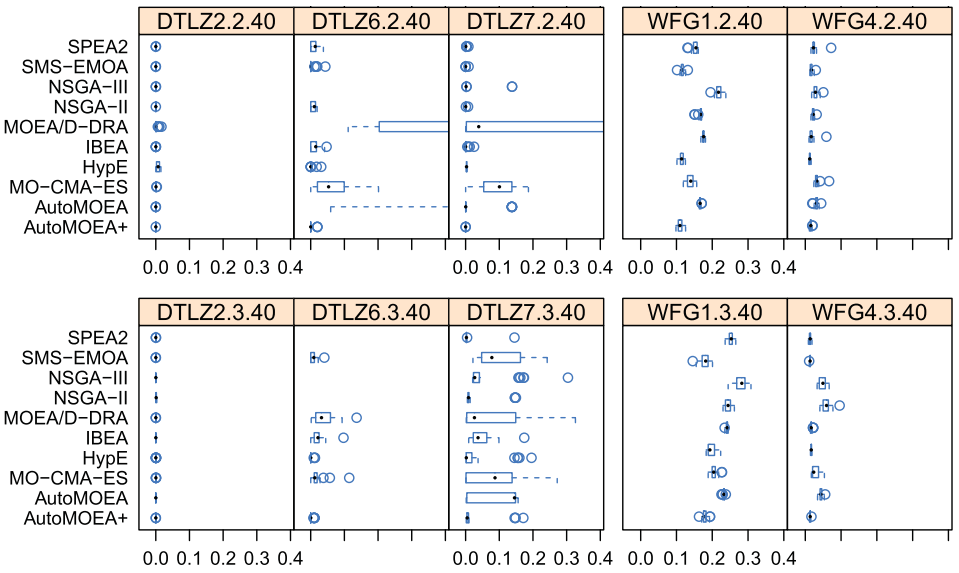

Figure 1: Performances of MOEAs given 10 000 FEs on selected problems with two (top) and three (bottom) objectives and 40 variables according to the I_H^{rd}.

We proceed to a more fine-grained analysis of the results to better visualize the effects of problem characteristics. Figures 1–3 show boxplots depicting the performance of these algorithms on selected problems from both benchmark sets and increasing M. Specifically, problems DTLZ2 and WFG4 represent concave problems, although the difficulty posed by the WFG concave problems tends to be higher than that of the DTLZ ones. Concerning the remaining DTLZ problems, DTLZ6 represents problems with a strong presence of local Pareto fronts, whereas DTLZ7 represents problems with disconnected Pareto fronts. Regarding non-concave WFG problems, we take WFG1 as illustrative. In general, the boxplots evidence the effects of problem characteristics and performance metric disagreements, which become ever stronger with the increase in M. Next, we discuss results from each scenario.

$\langle 2,\ 10k \rangle$: Figure 1 (top) depicts I_H^{rd} results, with which the remaining metrics agree. Effects from problem characteristics are seen in the differences between the relative performance of the algorithms when optimizing DTLZ or WFG problems. In more detail, for the DTLZ benchmark, the best-performing MOEAs present equivalent performance. Conversely, on the WFG set the AutoMOEA+ algorithm clearly achieves better and more consistent results for the non-concave problems represented by WFG1, and competitive performance on the concave ones, illustrated by WFG4. In addition, for no problem the original AutoMOEAs are able to outperform AutoMOEA+$_{\langle 2,\ 10k \rangle}$, whereas the opposite often happens. We also remark the rather different performances of both automatically designed algorithms in DTLZ6. In previous works, we have demonstrated that GA-based MOEAs struggle to solve this problem for a moderate number of variables, whereas DE-based ones can solve it far more easily (Bezerra et al., 2016, 2018); these considerations illustrate the importance of allowing configuration of the underlying EA and/or the variation operators.

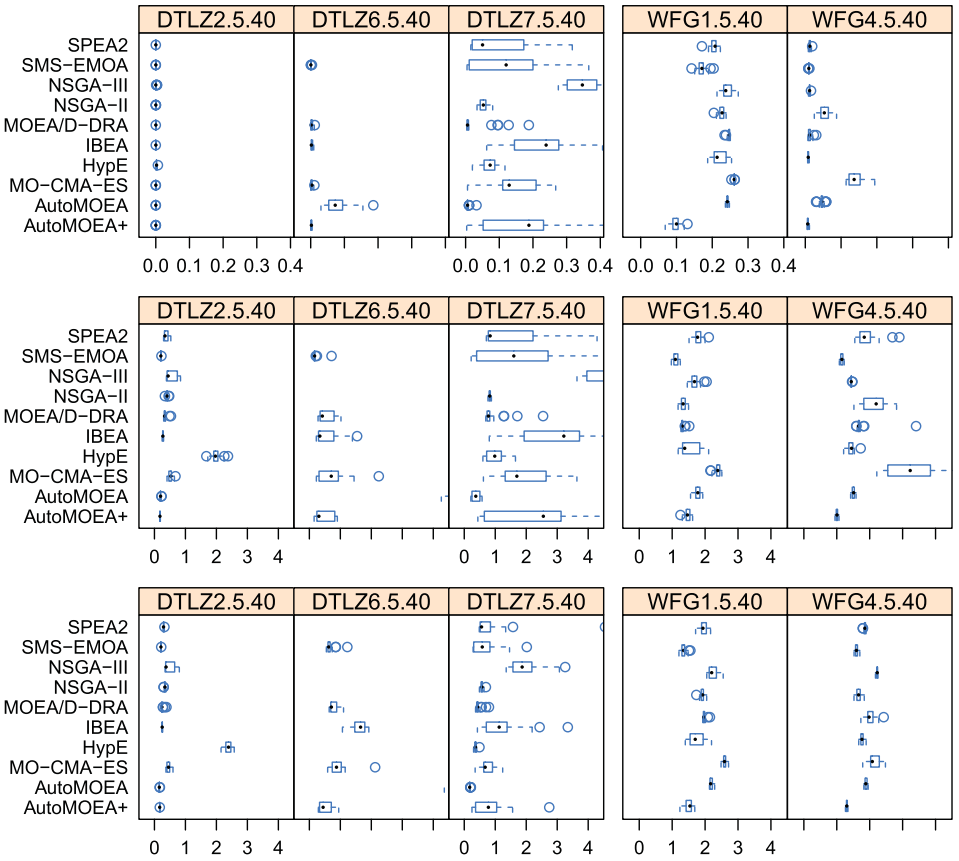

Figure 2: Performances of MOEAs on the scenario with $M = 5$ and $FE_{\max} = 10\,000$ for selected problems with $n_{\text{var}} = 40$. From top to bottom, I_H^{rd}, $I_{\epsilon+}$, and I_{IGD}.

$\langle 3, 10k \rangle$: A similar pattern is observed on this scenario, given on Figure 1 (bottom), where results from I_H^{rd} are depicted. The performances on problems DTLZ2 and WFG1 repeat the pattern discussed for the previous scenario. Yet, results for remaining problems become more spread, providing an indication that few algorithms such as AutoMOEA+$_{\langle 3, 10k \rangle}$ perform better and more consistently than the remaining MOEAs. Once again, the AutoMOEA+ algorithm outperforms the AutoMOEAs for all problems.

$\langle 5, 10k \rangle$: Figure 2 shows results from I_H^{rd} (top), $I_{\epsilon+}$ (middle), and I_{IGD} (bottom), where one observes two different situations on DTLZ problems. For the problem characteristics represented by DTLZ2 and DTLZ6, we notice that some state-of-the-art MOEAs are able to present performance equivalent to AutoMOEA+$_{\langle 5, 10k \rangle}$, but the latter stands out according to I_{IGD}. By contrast, on DTLZ7 AutoMOEA+$_{\langle 5, 10k \rangle}$ performs much worse than the best-performing MOEAs. This drawback is understandable when one observes that the automatic design methodology considers benchmarks as a whole, and specific functions may constitute exceptions to a broader picture. Regarding WFG problems, we see that performance metrics strongly disagree. For instance, the performance of AutoMOEA+$_{\langle 5, 10k \rangle}$ on WFG1 is remarkable according to I_H^{rd}, the metric used for tuning,

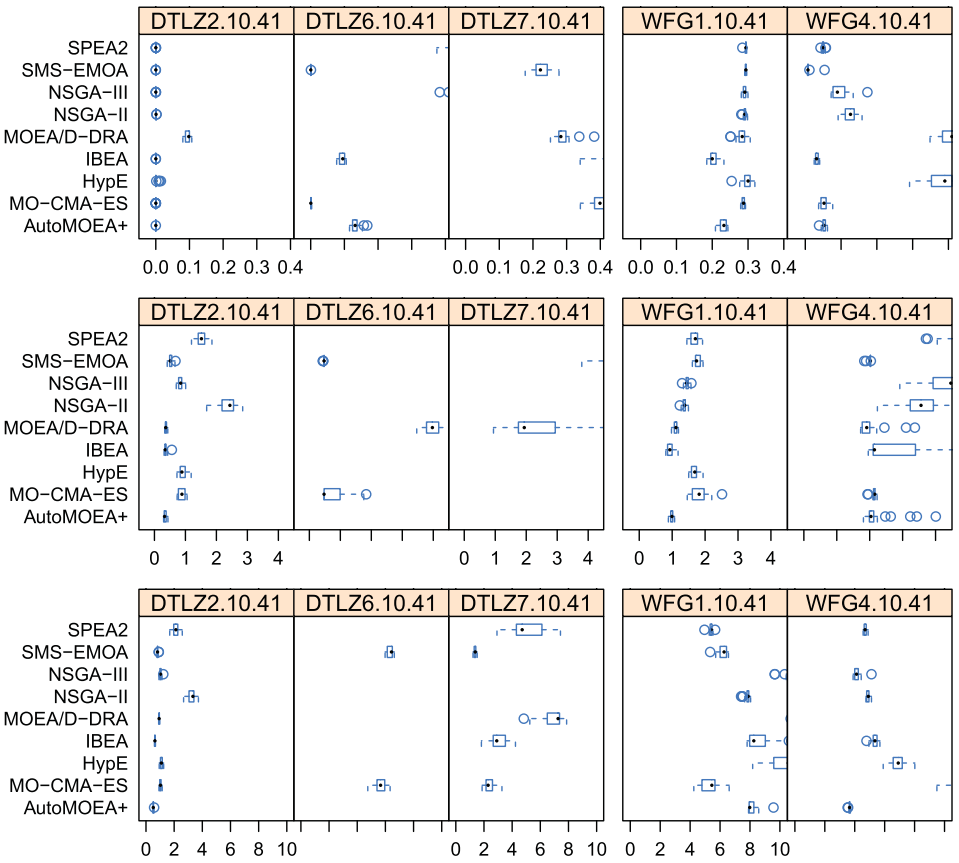

Figure 3: Performances of MOEAs on the $\langle 10,\ 10k \rangle$ scenario for selected problems with 40 variables. From top to bottom, I_H^{rd}, $I_{\epsilon+}$, and I_{IGD}.

but it is surpassed by SMS according to I_{IGD} and also by other MOEAs according to $I_{\epsilon+}$. As for the concave WFG problems, the AutoMOEA+ algorithm is always the best-performing, but the gap w.r.t. other MOEAs is very different depending on the metric considered. Finally, we see that AutoMOEA+$_{\langle 5,\ 10k \rangle}$ improves over the original AutoMOEA for all problems except DTLZ7 according to all metrics.

$\langle 10,\ 10k \rangle$: As previously reported in Bezerra et al. (2018), the performance from MOEAs is quite different from the remaining scenarios, in that metric ranges for given problems become much more spread. For this reason, our scaling is unable to fit results from many MOEAs; yet, if boxplots were to render all results visible, the differences between the best-performing algorithms would become difficult to distinguish, concealing the most important part of our analysis. In general, no single MOEA is considered consistent across all problems and metrics. AutoMOEA+$_{\langle 10,\ 10k \rangle}$, for instance, performs poorly on DTLZ6 and DTLZ7. The extreme situations are observed for I_{IGD} results: a poor relative performance on non-concave problems, contrasting to a very good relative performance on concave problems. Altogether, in comparison to other MOEAs, we see a generally competitive performance from AutoMOEA+$_{\langle 10,\ 10k \rangle}$.

Table 8: Summary of the statistical test results from each metric (respectively, I_H^{rd}, $I_{\epsilon+}$, I_{IGD}) per scenario (rows: $FE_{\max}$; columns: M). The given AutoMOEA+ algorithm may have been considered better than (+), equivalent to (=), or worse than (-) the best-performing MOEA(s) for the given scenario. MOEAs that match or outperform an AutoMOEA+ according to all metrics are indicated in parentheses.

	2	**3**	**5**	**10**
2 500	=== (SMS)	==+	==+	==+
10 000	=++	=++	=== (AutoMOEA)	===
40 000	+++	==+	==-	- - - (IBEA)

3.6 Stopping Criteria Effects

The analysis above has focused only on scenarios with $FE_{\max} = 10\,000$, for which we have identified two factors that appear critical for the trends observed in the results: (i) the heterogeneity of the problem characteristics that comprise the benchmark sets, and (ii) the disagreements between performance metrics, which become stronger with increasing number of objectives. We next further investigate this latter factor, with a focus on the role played by the stopping criterion in this interaction.

A summary of the rank sum analyses conducted on all scenarios is given in Table 8. Each entry summarizes the result of the statistical tests applied to the rank sums produced by each metric, respectively I_H^{rd}, $I_{\epsilon+}$, and I_{IGD}, and denotes that the given AutoMOEA+ algorithm was considered better than (+), equivalent to (=), or worse than (-) the best state-of-the-art MOEA. In addition, if a MOEA is considered better than or equivalent to the AutoMOEA+ algorithm for all metrics, it is indicated in parentheses. Results are provided in full as supplementary material, for brevity (Bezerra et al., 2017b).

A few important patterns are noticeable from Table 8. First, the AutoMOEA+ algorithms are generally able to match or improve over the performance of the state-of-the-art MOEAs for each scenario. Specifically, improvements are often observed on I_{IGD} analyses, and eventually on $I_{\epsilon+}$ ones, whereas equivalence is far more often observed on I_H^{rd} analyses. Second, in more than half of the scenarios the AutoMOEA+ algorithms improve over the state-of-the-art according to at least one metric, and rarely they do not match the performance of the state-of-the-art MOEAs for all metrics at the same time. Indeed, the only scenarios where this latter situation occurs are $\langle 5,\ 40k \rangle$ and $\langle 10,\ 40k \rangle$, both scenarios with a larger $FE_{\max}$ value. Third, it is far more likely that an AutoMOEA+ algorithm improve over the state-of-the-art according to a given metric when M and/or $FE_{\max}$ are low—the only exception to this pattern is scenario $\langle 2,\ 2.5k \rangle$. Finally, only two manually designed MOEAs are able to either match or outperform a given AutoMOEA+ algorithm according to all metrics at the same time: (i) SMS, which matches the performance of AutoMOEA+$_{\langle 2,\ 2.5k \rangle}$, and (ii) IBEA, which outperforms AutoMOEA+$_{\langle 10,\ 40k \rangle}$. We next further discuss results grouped by $FE_{\max}$.

2 500 FEs: When only a limited number of FEs is given to MOEAs, the differences between the best-performing algorithms are reduced. In fact, in many scenarios it is

Table 9: Rank sum difference (in parentheses) between the given MOEA and the lowest ranked for different scenarios. MOEAs in boldface present rank sums statistically significantly lower than the others according to Friedman's test.

			Scenario $\langle 5, 40k \rangle$			
I_H^{rd}	**Auto+** (0)	**SMS** (1)	IBEA (50)	MOEA/D (95)	SPEA2 (122)	CMA (125)
$I_{\epsilon+}$	**SMS** (0)	**IBEA** (5)	**Auto+** (21)	CMA (89)	MOEA/D (94)	SPEA2 (156)
I_{IGD}	**IBEA** (0)	**MOEA/D** (19)	**SMS** (25)	Auto+ (53)	SPEA2 (84)	CMA (105)
			Scenario $\langle 10, 40k \rangle$			
I_H^{rd}	**IBEA** (0)	SMS (63)	SPEA2 (66)	CMA (92)	Auto+ (97)	NSGA-III (110)
$I_{\epsilon+}$	**MOEA/D** (0)	**IBEA** (7)	Auto+ (55)	NSGA-III (97)	SMS (114)	NSGA-II (130)
I_{IGD}	**IBEA** (0)	**NSGA-III** (37)	SPEA2 (48)	NSGA-II (118)	HypE (131)	CMA (167)

not possible to identify statistically significant differences between the performances of the AutoMOEA+ algorithms, SMS, and IBEA according to Friedman's test. Yet, only once we observe that a single manually designed MOEA is able to match the performance of an AutoMOEA+ algorithm for all metrics at the same time, namely SMS when $M = 2$. Indeed, for all remaining M values we observe that the I_{IGD} performance of the AutoMOEA+ algorithms is considered statistically significantly better than that of the remaining MOEAs. Altogether, these results indicate that it is possible to design MOEAs that obtain good results according to all metrics at the same time even for scenarios where few FEs are available.

40 000 FEs: When a larger FE budget is considered, the best-performing MOEAs are able to approximate well the Pareto front for most bi-objective problems. Nonetheless, the rank sum analysis demonstrates that the performance of AutoMOEA+$_{\langle 2, 40k \rangle}$ is statistically significantly better than that of the state-of-the-art MOEAs for all metrics. For $M = 3$, the results according to I_{IGD} are statistically significantly only in favor of AutoMOEA+$_{\langle 3, 40k \rangle}$. Differences in conclusions in dependence of the metrics used become stronger for even larger numbers of objectives, as shown in Tables 8 and 9. When $M = 5$, we see a different set of best-performing MOEAs depending on the metric considered. AutoMOEA+$_{\langle 5, 40k \rangle}$ is either the best ranking (for I_H^{rd}) or statistically not significantly different from the top-ranking MOEA (for $I_{\epsilon+}$). This is different for I_{IGD}, where it ranks fourth. Finally, for $M = 10$ the AutoMOEA+ algorithm is unable to outperform or even match the overall performance of the best-performing MOEAs for any of the metrics. In fact, the performance of AutoMOEA+$_{\langle 10, 40k \rangle}$ resembles the performance previously reported for MOEA/D on this scenario (Bezerra et al., 2018), being considered good only according to $I_{\epsilon+}$, even if not competitive with the best-performing MOEAs. A surprising result from this scenario is that IBEA displays very good performance according to all metrics. Considering, however, that the tuning of IBEA and the design of the AutoMOEA+ are given the same budget and the tuning of AutoMOEA+ does not use any initial configurations, the much larger configuration space of AutoMOEA+ may lead to the fact that a design such as IBEA is not (yet) found by irace. Yet, this is the only scenario in which having a smaller configuration space translates into a clear advantage for the manually designed MOEAs.

From a more general perspective, the automatic MOEA design is strongly affected by the disagreements between metrics, and this challenge grows not only as a function

Table 10: Parameters selected by irace for AutoMOEA+$_{\langle 5,40k,I_{\epsilon+}\rangle}$.

Mating				Replacement			Replacement$_{Ext}$			pop		
Selection	SetPart	Refine	Diversity	SetPart	Refine	Removal	Refine	Diversity	Removal	*type*	μ	λ
DT(2)	depth	I_H^1	NN	strength	I_H^1	seq.	$I_{\epsilon+}$	ref. lines	1-shot	*fixed*	80	λ_r

(UnderEA = *DE* (TtoB); $N_{\text{ext}} = 500$; *Ref. lines*: Λ_{dist} = *two-layer*, $\Lambda_{\text{focus}} = \textit{balanced}$, $\Lambda_{\text{r}} = 0.74$.)

Table 11: Rank sum difference (in parentheses) between the given MOEA and the lowest ranked ($FE_{\max} = 40\,000$). MOEAs in boldface present rank sums statistically significantly lower than the others according to Friedman's test. For brevity, only the six best-performing MOEAs from each scenario are shown, and AutoMOEA+$_{\langle 5,40k,I_{\epsilon+}\rangle}$ is abbreviated as Auto-ϵ.

	Scenario $\langle 5,\ 40k\rangle$					
I_H^{rd}	**Auto+** (0)	**SMS** (1)	**Auto-ϵ** (31)	IBEA (58)	MOEA/D (103)	SPEA2 (138)
$I_{\epsilon+}$	**Auto-ϵ** (0)	**SMS** (39)	IBEA (44)	Auto+ (61)	CMA (129)	MOEA/D (134)
I_{IGD}	**Auto-ϵ** (0)	IBEA (89)	MOEA/D (106)	SMS (113)	Auto+ (142)	SPEA2 (173)

of M, but also of $FE_{\max}$. More precisely, when MOEAs are given more resources, it is natural that their search converges to the region of interest of the performance metric used to guide tuning or the implicit preferences from the algorithm designer. In other words, due to the known disagreements of the metrics, further optimizing the metric that guides the automatic tuning by increasing $FE_{\max}$ leads to worse results on the other metrics. From the rank sum analyses, we see that the automatic design is sensitive to this issue.

3.7 Tuning Metric Effects on Single-Objective MOEA Design

The disagreements between performance metrics have evidenced the importance of the tuning metric for the effectiveness of the automatically designed algorithms. We next investigate the effects of designing an AutoMOEA+ algorithm for scenario $\langle 5,\ 40k\rangle$, optimizing $I_{\epsilon+}$ instead of I_H^{rd}. The structure of this AutoMOEA+ algorithm (hereon called AutoMOEA+$_{\langle 5,40k,I_{\epsilon+}\rangle}$) is given in Table 10. The most significant structural change w.r.t. AutoMOEA+$_{\langle 5,\ 40k\rangle}$ depicted in Table 5 concerns the adoption of an external archive with a Preference$_{Ext}$, which comprises an indicator-based Refinement ($I_{\epsilon+}$) and a decomposition-based Diversity (reference lines). This preference relation is complementary to Preference$_{Mat}$ and Preference$_{Rep}$, which both use as Refinement component the I_H^1 indicator. This change in the design seems to reflect the change in tuning metric, as AutoMOEA+$_{\langle 5,\ 40k\rangle}$ was heavily I_H^1-based, whereas the search of AutoMOEA+$_{\langle 5,40k,I_{\epsilon+}\rangle}$ is more balanced w.r.t metrics; however, more repetitions of the design process would be required to corroborate this hypothesis.

The rank sum analysis given in Table 11 shows the results from a comparison that includes all MOEAs considered so far. (For brevity, AutoMOEA+$_{\langle 5,40k,I_{\epsilon+}\rangle}$ is abbreviated as Auto-ϵ in the table, and only the top-performing MOEAs are given.) The performance of AutoMOEA+$_{\langle 5,40k,I_{\epsilon+}\rangle}$ reflects the balance between metrics discussed above. Specifically, its performance is considered statistically significantly better than the state-of-the-art MOEAs according to I_{IGD}, and equivalent to the best MOEA according to the remaining

metrics. In fact, the results according to I_H^{rd} are remarkable, given that the performance of AutoMOEA+$_{\langle 5,40k,I_{\epsilon+}\rangle}$ is considered equivalent to that of AutoMOEA+$_{\langle 5,\,40k\rangle}$, which was configured for I_H^{rd}. More importantly, results from AutoMOEA+$_{\langle 5,40k,I_{\epsilon+}\rangle}$ evidence that the challenge posed by metric disagreements can be alleviated even for larger $FE_{\max}$ values, at least for a moderate number of objectives.

4 A Multiobjective Formulation to Automatic MOEA Design

The disagreement between performance metrics is most evident in the many-objective scenarios, where the best-ranked manually designed MOEA strongly depends on the metric used to measure quality. One possible way to overcome this disagreement is to optimize all metrics simultaneously during the automatic design process. In this section, we propose such a multiobjective formulation, following related work on multiobjective configuration of algorithms (Dréo, 2009; Bezerra et al., 2017a). First, we briefly define the concept of multiobjective configuration and detail our proposal. We then present an experimental investigation to evaluate this formulation on scenario $\langle 10,\ 40k\rangle$, the most challenging scenario we have identified so far.

4.1 Multiobjective MOEA Design

The fields of automatic algorithm configuration and multiobjective optimization intersect in two main ways (Bezerra et al., 2017a). The first one concerns the automatic configuration of multiobjective algorithms (López-Ibáñez and Stützle, 2012; Dubois-Lacoste et al., 2011; Bezerra et al., 2016), that is, the target algorithm tackles multiobjective problems and, hence, returns a set of mutually nondominated solutions. This is the context of the work presented so far in this article. The second one concerns multiobjective configuration of algorithms (Dréo, 2009; Blot et al., 2016), that is, the configuration of algorithms according to several metrics simultaneously.

In this section, we consider the *multiobjective design of MOEAs*, where the configurator searches for a MOEA design that optimizes multiple performance metrics simultaneously. In particular, we propose to aggregate the various performance metrics that we have used before to evaluate the performance of the MOEAs; that is, we consider an aggregation of the metrics $\mathcal{C} = \{I_H^{rd}, I_{\epsilon+}, I_{IGD}\}$. For this aggregation we use the hypervolume (I_H) metric (Zitzler et al., 2002) to make the multiobjective nature of the configuration problem transparent to irace. Concretely, candidate evaluation is done in two stages. First, each metric in $\mathcal{C}$ is computed, following the same setup described in the previous section. Second, the I_H of the subspace dominated by the objective vector representing the performance of the candidate in the *metric space* is computed. To ensure each metric is equally assessed by the I_H metric, we use a two-stage normalization approach, as follows. First, we discard points outside the upper bounds defined for each metric,[10] to avoid strong outliers. Next, we normalize each metric value to the [1,2] interval. The I_H metric is computed using point 2.2 as reference.

We use an aggregation for two main reasons. First, to simplify the choice of the final AutoMOEA+ to be compared to other MOEAs from several, possibly mutually nondominated AutoMOEA+ designs (nondominated w.r.t. the metrics in $\mathcal{C} = \{I_H^{rd}, I_{\epsilon+}, I_{IGD}\}$). Second, to be able to directly use the irace configurator that expects configurations to be evaluated w.r.t. a single metric. Given the large configuration space, we believe that this approach also helps to better direct the search of irace to very high-performing AutoMOEA+ designs.

[10] For $I_{\epsilon+}$ and I_{IGD}, the bound is set to 100. For the I_H^{rd}, the bound is set to 1.0.

Table 12: Parameters selected by irace for AutoMOEA+$_{\langle 10,40k,MO\rangle}$.

Mating				Replacement			Replacement$_{Ext}$			pop		
Selection	SetPart	Refine	Diversity	SetPart	Refine	Removal	Refine	Diversity	Removal	*type*	μ	λ
DT(4)	count	$I_{\epsilon+}$	crowd.	rank	$I_{\epsilon+}$	1-shot	$I_{\epsilon+}$	NN	seq.	*bound.*	90	λ_r

(UnderlyingEA = *DE* (TtoB) and $N_{\text{ext}} = 500$.)

Table 13: Rank sum difference (in parentheses) between the given MOEA and the lowest ranked ($FE_{\max} = 40\,000$). MOEAs in boldface present rank sums statistically significantly lower than the others according to Friedman's test. For brevity, only the six best-performing MOEAs from each scenario are shown.

	Scenario $\langle 10,\ 40k\rangle$					
I_H^{rd}	**Auto-MO** (0)	**IBEA** (48)	SMS (104)	SPEA2 (114)	CMA (143)	Auto+ (143)
$I_{\epsilon+}$	**Auto-MO** (0)	**MOEA/D** (40)	IBEA (55)	Auto+ (98)	NSGA-III (149)	SMS (163)
I_{IGD}	**Auto-MO** (0)	IBEA (67)	NSGA-III (103)	SPEA2 (115)	NSGA-II (185)	HypE (201)

4.2 Empirical Assessment

The experimental setup we adopt to evaluate our approach is similar to the setup adopted in the previous section. The only differences are (i) the number of scenarios, as only scenario $\langle 10,\ 40k\rangle$ is considered, and (ii), the way candidate configurations are evaluated, as we adopt the multiobjective formulation described above.

The structure of the AutoMOEA+ algorithm designed using the proposed multiobjective formulation (hereon called AutoMOEA+$_{\langle 10,40k,MO\rangle}$) is given in Table 12. Compared to the structure of AutoMOEA+$_{\langle 10,\ 40k\rangle}$ given in Table 5, designed to optimize the $I_{\epsilon+}$, we notice significant structural differences. In fact, the only component in common between both algorithms is dominance count for Preference$_{Mat}$. We also remark how interesting it is that a configuration designed to optimize multiple metrics selects the $I_{\epsilon+}$ as Refinement component for every Preference relation adopted, when this component had not been used at all on the algorithm meant to optimize the $I_{\epsilon+}$ metric. In addition, it is surprising that a large size external archive using NN as Diversity and sequential removal gets selected in a runtime-constrained setup. Yet, the computationally most expensive component of AutoMOEA+$_{\langle 10,\ 40k\rangle}$, steady-state selection, is not present in AutoMOEA+$_{\langle 10,40k,MO\rangle}$. Finally, DE is selected in place of GA as underlying EA. Altogether, one can understand these structural changes as a trade-off between computationally demanding components, with irace favoring the combination of refinement metrics and an external archive over steady-state replacement.

The rank sum analysis in Table 13 shows that this change in structure leads to a remarkable performance. Not only AutoMOEA+$_{\langle 10,40k,MO\rangle}$ ranks first according to all metrics, it is also considered statistically significantly better than all MOEAs according to I_{IGD}. This achievement is made yet more important in light of what had been observed in the state-of-the-art assessment conducted in Bezerra et al. (2018), and also in Section 3. Specifically, some algorithms are able to excel according to given metrics at the cost of others, such as NSGA-III for I_{IGD} (interestingly, the metric used in the original article to evaluate NSGA-III performance). Still, AutoMOEA+$_{\langle 10,40k,MO\rangle}$ is an algorithm that is able to excel according to all metrics, even outperforming NSGA-III for I_{IGD}. In fact, the only

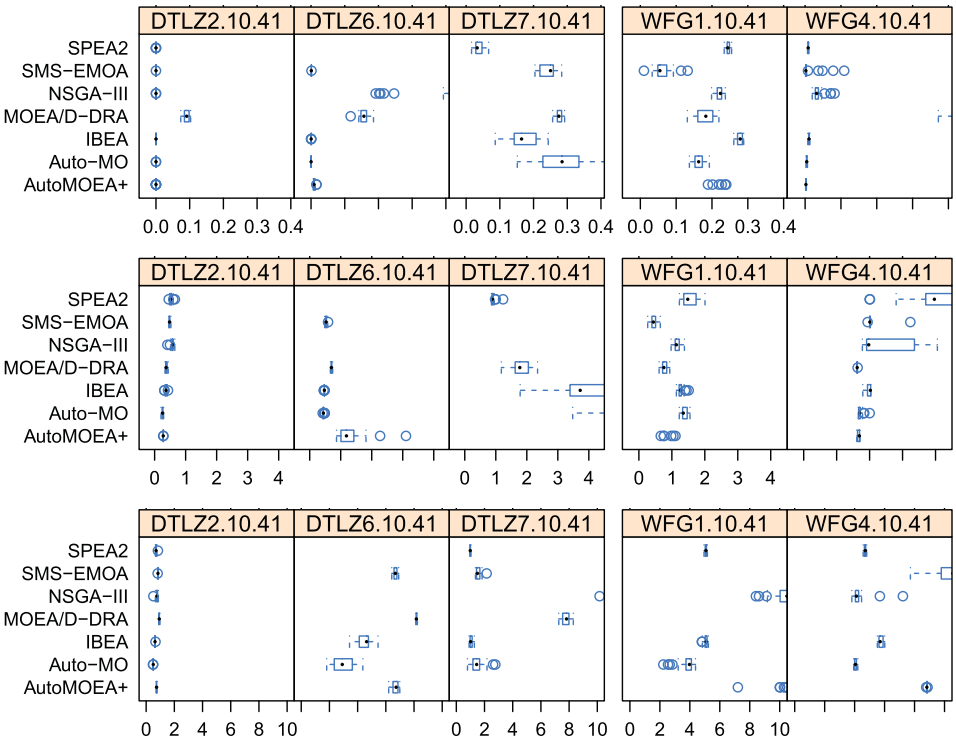

Figure 4: Performances of MOEAs on the $\langle 10,\ 40k \rangle$ scenario for selected problems with 40 variables. From top to bottom, I_H^{rd}, $I_{\epsilon+}$, and I_{IGD}.

manually designed MOEA that is able to achieve a balanced, yet effective performance on this scenario is IBEA. Yet, AutoMOEA+$_{\langle 10,40k,MO \rangle}$ is considered statistically significantly better than IBEA for all metrics but I_H^{rd}. Finally, it is interesting to observe the large difference in rank sums between AutoMOEA+$_{\langle 10,40k,MO \rangle}$ and AutoMOEA+$_{\langle 10,\ 40k \rangle}$: the single-objective design guided by $I_{\epsilon+}$ produces an algorithm that is unable to excel for the very metric for which it was created to optimize. By contrast, the multiobjective formulation leads to an algorithm that is balanced, yet effective for all metrics.

We conclude with a problem-wise assessment, using the boxplots depicted in Figure 4. For clarity, only MOEAs that rank up to fourth according to any of the metrics in the rank sum analyses are included in the comparison. The only problem for which all metrics agree is DTLZ2; for all others, at least one metric strongly disagrees with the remaining ones. This pattern also applies to AutoMOEA+$_{\langle 10,40k,MO \rangle}$, which is never outperformed by other MOEAs according to all metrics at the same time. The largest gaps between AutoMOEA+$_{\langle 10,40k,MO \rangle}$ and the remaining MOEAs are seen for I_{IGD}, whereas the smallest are observed for I_H^{rd}. Indeed, the contrasting results between I_H^{rd} and I_{IGD} further corroborate the need for algorithm engineering approaches that simultaneously consider multiple metrics, especially given that I_{IGD} has been so widely employed in the design and assessment of many-objective MOEAs. Finally, even when comparing according to $I_{\epsilon+}$, the AutoMOEA+ designed to optimize $I_{\epsilon+}$ outperforms only AutoMOEA+$_{\langle 10,40k,MO \rangle}$ on a few functions, confirming that a design which balances the importance of different metrics can lead to a better overall performance.

5 Conclusion

In this work, we have automatically designed state-of-the-art multi- and many-objective evolutionary algorithms (MOEAs) for box-constrained continuous optimization problems. Specifically, we have considered a range of experimental factors such as benchmark problems, number of variables and objectives, stopping criteria, and performance metrics. The AutoMOEA+ algorithms designed in this work have demonstrated a remarkably robust performance to all of these factors, especially the latter three. In particular, these results were only made possible through a series of investigations upon which this article builds, namely our proposal of the automatic MOEA design methodology (Bezerra et al., 2016), our review of multiobjective algorithm configuration (Bezerra et al., 2017a), and our assessment of the state-of-the-art in MOEAs for box-constrained continuous optimization (Bezerra et al., 2018).

The convergence of the insights obtained from those studies were translated into two major proposals in this work. The first is the AutoMOEA+ framework, which comprises the most relevant MOEA design paradigms (dominance-, indicator-, and decomposition-based), underlying evolutionary algorithms (genetic algorithms and differential evolution), and archive truncation techniques. From this framework, we have automatically designed state-of-the-art MOEAs for all experimental scenarios considered in multiobjective optimization, and for nearly all scenarios considered in many-objective optimization. Many of the design choices present in the AutoMOEA+ algorithms differ considerably from what human designers have so far considered; some designs couple components from entirely different design paradigms to produce high-performing MOEA designs. Indeed, all novel components implemented in this article have been used in one or more automatically designed MOEAs, except for the online replacement component. Performance improvements from the AutoMOEA+ algorithms over the AutoMOEAs produced from the original framework corroborate the benefits of the extensions we propose in this work.

The second proposal focused on many-objective scenarios, and consists in a multiobjective formulation of the automatic MOEA design. Using this formulation, one can automatically design MOEAs which simultaneously optimize a set of relevant, yet disagreeing metrics (I_H^{rd}, $I_{\epsilon+}$, and I_{IGD}). Perhaps surprisingly, we have shown that an algorithm designed to optimize a set of metrics can even outperform algorithms created with a single metric in mind according to that metric. Overall, the performance of the resulting AutoMOEA+$_{\langle 10,40k,MO\rangle}$ algorithm is remarkable for several reasons: (1) it ranks first according to all metrics in rank sum analyses; (2) it is considered statistically significantly better than the state-of-the-art MOEAs according to I_{IGD}; and (3) it is considered statistically significantly better than IBEA for all but the I_H^{rd} metric, the best-performing MOEA for this scenario according to all metrics.

The implications of this work are many and its applications are numerous. First, although we have produced a number of novel state-of-the-art algorithms for the main application domain of MOEAs, our actual contribution is the empirical demonstration that this task is likely feasible for any application domain. The only imperative requirement is the a priori identification of effective domain-specific components, to which the automatic design approach is flexible enough to adapt. Second, our multiobjective formulation of MOEA design is an elegant solution to the disagreement between Pareto-compliant performance metrics, and yet its major contribution is the empirical demonstration that a MOEA should not be engineered with a single metric in mind, regardless of the scenario considered. In addition, our multiobjective formulation of

MOEA design has only been tested so far for solution quality assessment. Yet, it seems imperative to account also for runtime in search for algorithms with better anytime behavior.

A final implication of our work concerns the limitations of MOEAs that our approach did not propose to solve, but to put in evidence. Specifically, our previous investigation on manually designed state-of-the-art MOEAs had indicated that a few factors pose challenges that MOEAs are yet to overcome: (i) having too little function evaluations available; (ii) accounting for very heterogeneous problem characteristics; and (iii) scaling to deal with a significant number of variables and/or objectives. In all of these scenarios, the automatically designed MOEAs match or surpass the performance of the manually designed state-of-the-art MOEAs. Still, it becomes evident that the MOEA research community needs to keep pushing in these directions if effective algorithms are to be designed, either manually or automatically.

Acknowledgments

The research presented in this article has received funding from the COMEX project (P7/36) within the IAP programme of BelSPO. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is the research director.

References

Abbass, H. A. (2002). The self-adaptive Pareto differential evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pp. 831–836.

Abbass, H. A., Sarker, R., and Newton, C. (2001). PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems. In *Proceedings of the 2001 Congress on Evolutionary Computation*, pp. 971–978.

Aneja, Y. P., and Nair, K. P. K. (1979). Bicriteria transportation problem. *Management Science*, 25(1):73–78.

Bader, J., and Zitzler, E. (2011). HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76.

Beume, N., Naujoks, B., and Emmerich, M. T. M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

Bezerra, L. C. T., López-Ibáñez, M., and Stützle, T. (2016). Automatic component-wise design of multi-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(3):403–417.

Bezerra, L. C. T., López-Ibáñez, M., and Stützle, T. (2017a). Automatic configuration of multi-objective optimizers and multi-objective configuration. Technical Report TR/IRIDIA/2017-011, IRIDIA, Université Libre de Bruxelles, Belgium.

Bezerra, L. C. T., López-Ibáñez, M., and Stützle, T. (2017b). Automatically designing state-of-the-art multi- and many-objective evolutionary algorithms: Supplementary material. Retrieved from http://iridia.ulb.ac.be/supp/IridiaSupp2016-004/

Bezerra, L. C. T., López-Ibáñez, M., and Stützle, T. (2018). A large-scale experimental evaluation of high-performing multi- and many-objective evolutionary algorithms. *Evolutionary Computation*, 26(4):621–656.

Birattari, M. (2009). *Tuning metaheuristics: A machine learning perspective*, Vol. 197 of *Studies in computational intelligence*. Berlin, Heidelberg: Springer.

Biscani, F., Izzo, D., and Yam, C. H. (2010). A global optimisation toolbox for massively parallel engineering optimisation. In *4th International Conference on Astrodynamics Tools and Techniques*.

Bleuler, S., Laumanns, M., Thiele, L., and Zitzler, E. (2003). PISA—A platform and programming language independent interface for search algorithms. In *Evolutionary Multi-criterion Optimization, EMO 2003*, pp. 494–508. Lecture Notes in Computer Science, Vol. 2632.

Blot, A., Hoos, H. H., Jourdan, L., Kessaci-Marmion, M.-E., and Trautmann, H. (2016). MO-ParamILS: A multi-objective automatic algorithm configuration framework. In *10th International Conference on Learning and Intelligent Optimization*, pp. 32–47. Lecture Notes in Computer Science, Vol. 10079.

Cahon, S., Melab, N., and Talbi, E.-G. (2004). ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380.

Chand, S., and Wagner, M. (2015). Evolutionary many-objective optimization: A quick-start guide. *Surveys in Operations Research and Management Science*, 20(2):35–42.

Chiang, T.-C. (2014). nsga3cpp: A C++ implementation of NSGA-III. Retrieved from http://web.ntnu.edu.tw/~tcchiang/publications/nsga3cpp/nsga3cpp.htm

Coello Coello, C. A., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. New York: Springer.

Das, I., and Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley.

Deb, K., and Deb, D. (2014). Analysing mutation schemes for real-parameter genetic algorithms. *International Journal of Artificial Intelligence and Soft Computing*, 4(1):1–28.

Deb, K., and Jain, S. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In A. Abraham, L. Jain, and R. Goldberg (Eds.), *Evolutionary multiobjective optimization*, *Advanced information and knowledge processing*, pp. 105–145. London: Springer.

Dréo, J. (2009). Using performance fronts for parameter setting of stochastic metaheuristics. In R. Rothlauf (Ed.), *GECCO (Companion)*, pp. 2197–2200.

Dubois-Lacoste, J., López-Ibáñez, M., and Stützle, T. (2011). Automatic configuration of state-of-the-art multi-objective optimizers using the TP+PLS framework. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 2019–2026.

Eberhart, R., and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43.

Ehrgott, M., and Gandibleux, X. (2004). Approximative solution methods for combinatorial multicriteria optimization. *TOP*, 12(1):1–88.

Fawcett, C., and Hoos, H. H. (2016). Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22(4):431–458.

Fleming, P. J., Purshouse, R. C., and Lygoe, R. J. (2005). Many-objective optimization: An engineering design perspective. In *Evolutionary Multi-criterion Optimization*, pp. 14–32. Lecture Notes in Computer Science, Vol. 3410.

Fonseca, C. M., and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest (Ed.), *ICGA*, pp. 416–423. Burlington, MA: Morgan Kaufmann.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Boston: Addison-Wesley.

Hadka, D., and Reed, P. (2013). Borg: An auto-adaptive many-objective evolutionary computing framework. *Evolutionary Computation*, 21(2):231–259.

Hoos, H. H. (2012). Automated algorithm configuration and parameter tuning. In Y. Hamadi, E. Monfroy, and F. Saubion (Eds.), *Autonomous search*, pp. 37–71. Berlin: Springer.

Huband, S., Hingston, P., Barone, L., and While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506.

Hughes, E. J. (2003). Multiple single objective Pareto sampling. In *Proceedings of the 2003 Congress on Evolutionary Computation*, pp. 2678–2684, Vol. 4.

Igel, C., Hansen, N., and Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28.

Igel, C., Heidrich-Meisner, V., and Glasmachers, T. (2008). Shark. *Journal of Machine Learning Research*, 9:993–996.

Jiang, S., Ong, Y. S., Zhang, J., and Feng, L. (2014). Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(12):2391–2404.

KhudaBukhsh, A. R., Xu, L., Hoos, H. H., and Leyton-Brown, K. (2009). SATenstein: Automatically building local search SAT solvers from components. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pp. 517–524.

Knowles, J. D., and Corne, D. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172.

Kukkonen, S., and Lampinen, J. (2005). GDE3: The third evolution step of generalized differential evolution. In *Proceedings of the 2005 Congress on Evolutionary Computation*, pp. 443–450.

Li, B., Li, J., Tang, K., and Yao, X. (2015). Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys*, 48(1):1–35.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

López-Ibáñez, M., Knowles, J. D., and Laumanns, M. (2011). On sequential online archiving of objective vectors. In *Evolutionary Multi-criterion Optimization*, pp. 46–60. Lecture Notes in Computer Science, Vol. 6576.

López-Ibáñez, M., and Stützle, T. (2012). The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861–875.

Madavan, N. K. (2002). Multiobjective optimization using a Pareto differential evolution approach. In *Proceedings of the 2002 World Congress on Computational Intelligence*, pp. 1145–1150.

Mezura-Montes, E., Reyes-Sierra, M., and Coello Coello, C. A. (2008). Multi-objective optimization using differential evolution: A survey of the state-of-the-art. In U. K. Chakraborty (Ed.), *Advances in differential evolution*, pp. 173–196. Heidelberg: Springer.

Paquete, L., and Stützle, T. (2007). Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In T. F. Gonzalez (Ed.), *Handbook of approximation algorithms and metaheuristics*, pp. 29–1—29–15. Boca Raton, FL: Chapman & Hall/CRC.

Price, K., Storn, R. M., and Lampinen, J. A. (2005). *Differential evolution: A practical approach to global optimization*. New York: Springer.

Robič, T., and Filipič, B. (2005). DEMO: Differential evolution for multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, pp. 520–533. Lecture Notes in Computer Science, Vol. 3410.

Ross, P. (2005). Hyper-heuristics. In E. K. Burke and G. Kendall (Eds.), *Search methologies*, pp. 529–556. Boston: Springer.

Schütze, O., Lara, A., and Coello Coello, C. A. (2011). On the influence of the number of objectives on the hardness of a multiobjective optimization problem. *IEEE Transactions on Evolutionary Computation*, 15(4):444–455.

Tagawa, K., Shimizu, H., and Nakamura, H. (2011). Indicator-based differential evolution using exclusive hypervolume approximation and parallelization for multi-core processors. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 657–664.

Tanabe, R., Ishibuchi, H., and Oyama, A. (2017). Benchmarking multi- and many-objective evolutionary algorithms under two optimization scenarios. *IEEE Access*, 5:19597–19619.

Tušar, T., and Filipič, B. (2007). Differential evolution versus genetic algorithms in multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, pp. 257–271. Lecture Notes in Computer Science, Vol. 4409.

Voß, T., Hansen, N., and Igel, C. (2010). Improved step size adaptation for the MO-CMA-ES. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 487–494.

Zhang, Q. (2007). MOEA/D homepage. Retrieved from https://dces.essex.ac.uk/staff/zhang/webofmoead.htm

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

Zhang, Q., Liu, W., and Li, H. (2009). The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In *Proceedings of the 2009 Congress on Evolutionary Computation*, pp. 203–208.

Zitzler, E., and Künzli, S. (2004). Indicator-based selection in multiobjective search. In *Proceedings of PPSN-VIII, Eighth International Conference on Parallel Problem Solving from Nature*, pp. 832–842. Lecture Notes in Computer Science, Vol. 3242.

Zitzler, E., Laumanns, M., and Thiele, L. (2002). SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Periaux, K. D. Papaliliou, and T. Fogarty (Eds.), *Evolutionary Methods for Design, Optimisation and Control*, pp. 95–100.

Zitzler, E., Thiele, L., and Bader, J. (2010). On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14(1):58–79.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Grunert da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.