Automatic Design of Evolutionary Algorithms for Multi-Objective Combinatorial Optimization

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

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

Abstract. Multi-objective evolutionary algorithms (MOEAs) have been the subject of a large research effort over the past two decades. Traditionally, these MOEAs have been seen as monolithic units, and their study was focused on comparing them as blackboxes. More recently, a component-wise view of MOEAs has emerged, with flexible frameworks combining algorithmic components from different MOEAs. The number of available algorithmic components is large, though, and an algorithm designer working on a specific application cannot analyze all possible combinations. In this paper, we investigate the automatic design of MOEAs, extending previous work on other multi-objective metaheuristics. We conduct our tests on four variants of the permutation flowshop problem that differ on the number and nature of the objectives they consider. Moreover, given the different characteristics of the variants, we also investigate the performance of an automatic MOEA designed for the multi-objective PFSP in general. Our results show that the automatically designed MOEAs are able to outperform six traditional MOEAs, confirming the importance and efficiency of this design methodology.

1 Introduction

Multi-objective evolutionary algorithms (MOEAs) are the most studied metaheuristics for solving multi-objective optimization problems and a large number of MOEAs have been proposed in the past two decades [1, 2, 8, 10, 21, 22]. A few surveys of the field have been conducted [19,24] and competitions have been held to identify the best MOEA for particular benchmarks [20]. These works study MOEAs as monolithic units and provide insights on which particular MOEAs are state of the art for specific problems, giving a baseline for future developments. More recently, a component-wise view of MOEAs has drawn the attention of the MOEA community [5,13]. "Deconstructed" MOEAs actually differ by a few main algorithmic components, which can be individually analyzed to assess their actual contribution to the overall efficiency of the algorithm. This component-wise view has been recently strengthened by the development of flexible algorithmic frameworks [11, 13], where novel MOEAs can be devised combining existing algorithmic components. However, the potential of such approach remains unclear as the number of possible combinations is extremely large to be fully explored.

In a parallel research trend, the automatic design of multi-objective optimizers has produced several state-of-the-art algorithms [3, 9, 16]. This methodology

T. Bartz-Beielstein et al. (Eds.): PPSN XIII 2014, LNCS 8672, pp. 508–517, 2014.

consists of applying automatic offline parameter configuration tools (*tuners*, for short) to flexible algorithmic frameworks, such as the ones recently proposed for MOEAs. By doing so, the parameter space searched by the tuner is actually a space of different algorithms, which allows the analysis and comparison of several novel algorithms at a time. The final configuration selected by the tuner is not necessarily the best possible algorithm as the configuration space is not searched exhaustively, but it is usually a high-performing one. In this work, we investigate the efficiency of the automatic algorithm design methodology for creating MOEAs for combinatorial optimization problems. We use four different variants of the permutation flowshop problem (PSFP) as test benchmarks. These variants combine, in different ways, the most relevant optimization criteria for the PFSP, namely *makespan* ($\mathsf{C_{max}}$), *total flow time* (TFT), and *total tardiness* (TT).

In a first step, we present the framework we use for this work, which combines components from ParadisEO-MOEO [13], PISA [7], and PaGMO [6]. Particularly, some of the MOEAs from the literature cannot be easily represented just by the combination of a fitness and a diversity component as proposed in ParadisEO-MOEO. Instead, following [23], we use a more general preference relation, defined as a combination of a set-partitioning criterion, a quality indicator and a diversity measure. Second, although some MOEAs claim to use an external archive, this archive is used during the evolutionary search process, either for mating or selection. In our implementation, we allow MOEAs to use two archives at a time: an internal one, which can replace the population of the algorithm; and an external one, which is only used for building the final approximation front. Finally, our implementation incorporates elements from new MOEAs, such as sequential versus one-shot removal of solutions [1], quality in terms of hypervolume contribution [1, 2], and adaptive-grid diversity [12].

In a second step, we generate five automatically designed MOEAs (AutoMOEAs): one for each of the PFSP variants considered here, and an extra one for solving all variants ($\text{AutoMOEA}_{\text{PFSP}}$). Since related works have shown these variants present different search space characteristics [5, 9], it is important to understand whether (i) the automatic design methodology can produce a single optimizer that performs better than the others for all variants, or; (ii) the best strategy is to have an AutoMOEA tailored for each variant. We then conduct an experimental analysis on a PFSP variant basis. For each PFSP variant, we (i) compare the structure of the variant-specific AutoMOEA to $\text{AutoMOEA}_{\text{PFSP}}$; and (ii) compare their performance to six traditional MOEAs. Results show that the AutoMOEAs are often able to outperform the traditional MOEAs. Although the more general $\text{AutoMOEA}_{\text{PFSP}}$ reaches better results than the traditional MOEAs, the variant-specific AutoMOEAs perform better in most scenarios, reinforcing the need for the automatic algorithm design methodology.

2 A Framework for Instantiating MOEAs

A summary of the components that differentiate most current MOEA algorithms is given in Table 1. The most important components are (i) the mating selection (Mating), and (ii) the environmental selection or truncation (Replacement).

Table 1. Algorithmic components of the MOEA framework used

Component	Domain	Description
μ	$\mathbb{N}^+$	Population size
μ_0	$[0.1, 1] \subset \mathbb{R}$	Num. of initial solutions is $\mu \cdot \mu_0$
λ	$\mathbb{N}^+$	Number of offspring
pop	{ fixed-size, bounded }	Population type
$\texttt{pop}_{\text{ext}}$	{ none, bounded, unbounded }	External archive type
N_{ext}	$\mathbb{N}^+$	$\texttt{pop}_{\text{ext}}$ size, if $\texttt{pop}_{\text{ext}}$ type = *bounded*
Selection	random deterministic tournament stochastic tournament	Mating selection operator
Removal	sequential one shot	Population replacement policy
Set-partitioning	none dominance rank dominance strength dominance depth dominance depth-rank	
Quality indicator	none binary indicator (ϵ or hypervolume) (IBEA) hypervolume contribution (SMS-EMOA) h-hypervolume contribution (HypE)	
Diversity	none niche sharing k-th nearest neighbor crowding distance adaptive grid (PAES)	

Traditionally, mating and replacement have been defined as compositions of a *fitness* component, designed to favor convergence, and a *diversity* component, meant to keep the population spread across the objective space. Following Zitzler et al. [23], we consider general preference relations comprising three lower-level components: (i) a *set-partitioning relation*, which partitions a set of solutions in a manner consistent with Pareto dominance but cannot discriminate between nondominated solutions; (ii) a *quality indicator*, which assigns a quality value to solutions in a manner that does not contradict Pareto dominance (often used as a refinement of the partitions); and (iii) a diversity metric, which does not need to be consistent with Pareto dominance. All the options implemented for these low-level components are shown in Table 1.

We define the Mating component as being composed of a preference relation and a selection method. The selection method may be any of the traditional methods in EAs: random, tournament, etc. Conversely, component Replacement is composed of a preference relation and a removal policy. The latter may be either *sequential* [15] (also called *iterative* [1]), i.e., one individual/solution is removed at a time and preference relations are re-computed before the next is discarded; or *one-shot*, i.e., preference relations are computed once and the worst solutions are removed altogether [1].

Table 2. Instantiation of MOEAs using our framework. Other components are parameters except $\texttt{pop} = \textit{fixed-size}$ and $\texttt{pop}_{\texttt{ext}} = \textit{none}$. SMS-EMOA uses $\lambda = 1$ by design.

	Mating				Replacement			
Algorithm	SetPart	Quality	Diversity	Selection	SetPart	Quality	Diversity	Removal
MOGA [10]	rank	—	sharing	det. tour.	rank	—	sharing	one-shot
NSGA-II [8]	depth	—	crowding	det. tour.	depth	—	crowding	one-shot
SPEA2 [22]	strength	—	k-NN	det. tour.	strength	—	k-NN	sequential
IBEA [21]	—	bin. ind.	—	det. tour.	—	bin. ind.	—	sequential
SMS-EMOA [2]	—	—	—	random	depth rank	hyperv. contrib.	—	—
HypE [1]	—	h-hyperv. contrib.	—	det. tour.	depth	h-hyperv. contrib	—	sequential

Table 3. Parameter space for tuning the numerical parameters of all MOEAs

Parameter	μ	λ	p_C	p_{mut}	p_X
Domain	$\{10, 20, \ldots, 100\}$	1 or $\lambda_r \cdot \mu$ $\lambda_r \in [0.1, 2]$	$[0, 1]$	$[0, 1]$	$[0, 1]$

Algorithm	IBEA	MOGA	SPEA2
Parameter	*indicator*	σ_{share}	k
Domain	$\{I_\epsilon, I_{\overline{H}}\}$	$[0.1, 1]$	$\{1, \ldots, 9\}$

Besides the components explained above, populations and archives also need to be highlighted. Traditionally, a population is a set of individuals (dominated and nondominated alike) that are subject to the evolutionary process. As discussed elsewhere [19], an archive can be seen as a generalized population that may (i) keep only nondominated solutions and/or (ii) have unlimited capacity. In this work, we implement an internal archive $\texttt{pop}$ (in place of the population), as well as an external archive $\texttt{pop}_{\texttt{ext}}$, which does not interfere with the evolutionary process. Concretely, if $\texttt{pop}$ is set to *fixed-size*, it behaves like a regular population. If it is set to *bounded*, it behaves like an internal archive, accepting only nondominated solutions until its maximum capacity is reached. At this point, a replacement is carried out. By using this flexible implementation, we are able to instantiate algorithms such as SPEA2, which were originally described with an external archive that interferes in the evolutionary process. Concerning the external archive as implemented in this work, a MOEA may choose (i) not to use it ($\texttt{pop}_{\texttt{ext}} = \textit{none}$), (ii) to use it without capacity constraints ($\texttt{pop}_{\texttt{ext}} = \textit{unbounded}$), or (iii) to use it with a maximum capacity N_{ext} ($\texttt{pop}_{\texttt{ext}} = \textit{bounded}$). In the latter case, an additional replacement component $\mathsf{Replacement}_{Ext}$ needs to be defined, similar to the replacement component used by the internal archive. Moreover, in the case of $\texttt{pop} = \textit{bounded}$, we add a numerical parameter μ_0 that determines the number of initial solutions as $\mu \cdot \mu_0$.

As shown in Table 2, by selecting the corresponding components for each algorithm, we can instantiate at least six well-known MOEAs from the literature. Particularly, SMS-EMOA and HypE use slightly different definitions of hypervolume contribution, and the definition used by HypE is further parametrized by a parameter h (k in the original paper [1]). SPEA2 uses a pre-defined formula for computing its k parameter value. Here, we implement both the formula and an option for manual input of k, i.e., k becomes a numerical parameter.

Table 4. The MOEAs are sorted according to their sum of ranks (in parenthesis)

Cmax	AutoMOEA	AutoMOEA	IBEA	NSGA-II	SPEA2	HypE	MOGA	SMS
TFT	$_{\text{PFSP}}$ (249)	$_{\text{Cmax-TFT}}$ (302)	(398)	(472)	(479)	(585)	(687)	(788)
Cmax	AutoMOEA	AutoMOEA	NSGA-II	SPEA2	IBEA	HypE	SMS	MOGA
TT	$_{\text{Cmax-TT}}$ (209)	$_{\text{PFSP}}$ (253)	(357)	(464)	(547)	(574)	(770)	(786)
TFT	MOGA	IBEA	AutoMOEA	HypE	NSGA-II	AutoMOEA	SPEA2	SMS
TT	(304)	(371)	$_{\text{TFT-TT}}$ (475)	(499)	(499)	$_{\text{PFSP}}$ (553)	(615)	(644)
Cmax	AutoMOEA	AutoMOEA	IBEA	SPEA2	HypE	NSGA-II	SMS	MOGA
TFT-TT	(161)	$_{\text{PFSP}}$ (251)	(417)	(525)	(528)	(541)	(735)	(802)

3 Experimental Setup

We consider the six standard MOEAs in Table 2 and we compare them with the novel, automatically designed MOEAs instantiated using this framework for the multi-objective PFSP variants. Our goal is to identify good combinations of such components, specially if they differ from (and are better than) existing combinations used by standard MOEAs.

We use irace [14] as tuner, adapting it to multi-objective optimization by using the hypervolume quality measure [16]. For computing the hypervolume, we normalize the objective values to the range $[1, 2]$ using the maximum and minimum ever found per problem, and use $(2.1, 2.1)$ and $(2.1, 2.1, 2.1)$, respectively, as reference points for the bi- and tri-objective variants. We set a budget of 5 000 runs for each tuning in order to design each variant-specific AutoMOEA. Since $\text{AutoMOEA}_{\text{PFSP}}$ needs to see four times more instances than the variant-specific AutoMOEAs, we give it a tuning budget of 20 000 runs. Moreover, since the traditional MOEAs do not present default values for an application to the PFSP, we tune their numerical parameters using the same budget given to the variant-specific AutoMOEAs (5 000 runs per MOEA per variant per tuning). The parameter space used for the numerical parameters is the same for all MOEAs (Table 3). Following [5], all MOEAs use random initial solutions, unbounded external archives, two-point crossover and two problem-specific mutation operators, namely *insert* and *exchange*. In Table 3, p_{mut} stands for the probability of applying mutation to a given individual, while p_X is the probability of using the exchange operator if the first test is successful ($1 - p_X$ for the insertion operator).

To ensure that our results are not affected by overtuning, we use two independent benchmark sets for testing and tuning. The tuning benchmark set contains the instances with 20 machines from the tuning benchmark set proposed by [9]. The testing benchmark set is an adaptation of the benchmark set proposed by Taillard [18], following what is traditionally done in the multi-objective PFSP literature [9, 17]. Once the algorithms are tuned, we run them 10 times on each test instance, compute the average hypervolume, and compare the results using rank sum analysis and parallel coordinate plots. Experiments are run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB of cache size under Cluster Rocks Linux version 6.0/CentOS 6.3. For brevity, few representative results are given here. The complete list of results is provided as supplementary material [4], as well as the parameter settings used by irace.

Table 5. Parameters selected by irace for AutoMOEA$_{\text{Cmax-TFT}}$, AutoMOEA$_{\text{Cmax-TT}}$, AutoMOEA$_{\text{TFT-TT}}$, AutoMOEA$_{\text{Cmax-TFT-TT}}$, and AutoMOEA$_{\text{PFSP}}$, respectively. All AutoMOEAs use pop = *bounded*.

Cmax-TFT	Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	p_c	p_{mut}	p_X
Value	det. tour. (2)	rank	hyp. contr.	crowding	depth	bin. ind. (I_ϵ)	crowding	one-shot	80	0.3	1.5	0.38	0.82	0.71

Cmax-TT	Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	p_c	p_{mut}	p_X
Value	random	—	—	—	—	h-hyp. contr.	crowding	one-shot	30	0.94	1.63	0.34	0.95	0.81

TFT-TT	Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	p_c	p_{mut}	p_X
Value	stoch. tour. (0.9)	—	—	crowding	count	bin. ind. (I_ϵ)	sharing (0.87)	sequential	70	0.94	1.47	0.77	0.99	0.63

Cmax-TFT-TT	Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	p_c	p_{mut}	p_X
Value	det. tour. (2)	rank	—	crowding	—	h-hyp. contr.	crowding	one-shot	40	0.26	1.68	0.36	0.85	0.74

PFSP	Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	p_c	p_{mut}	p_X
Value	random	—	—	—	—	h-hyp. contr.	—	one-shot	60	0.73	1.53	0.17	0.76	0.40

4 Results and Discussion

As shown in Table 5, the structures of AutoMOEA$_{\text{Cmax-TFT}}$ and AutoMOEA$_{\text{PFSP}}$ differ substantially. AutoMOEA$_{\text{Cmax-TFT}}$ combines components from almost all specific known MOEAs that built the basis for the algorithmic components included in the framework. Its mating preference relation joins components from MOGA (*dominance rank*), SMS-EMOA (*hypervolume contribution*), and NSGA-II (*crowding distance*). The replacement preference relation is a combination of the components from NSGA-II and IBEA. Interestingly, this very heterogeneous MOEA is able to outperform all traditional MOEAs on many of the tested instances. In fact, it is only outperformed by AutoMOEA$_{\text{PFSP}}$. The structure of AutoMOEA$_{\text{PFSP}}$ is radically simpler. The mating selection is performed randomly, so no preference relation is required. For replacement, only the shared hypervolume contribution proposed for HypE is used. However, the *one-shot* removal policy is adopted, most likely to cope with the computational overhead for the three-objective variant. Despite its relatively simple structure, AutoMOEA$_{\text{PFSP}}$ outperforms, in the Cmax-TFT problem variant, all MOEAs considered, including AutoMOEA$_{\text{Cmax-TFT}}$.

The similarity of the structures of AutoMOEA$_{\text{Cmax-TT}}$ and AutoMOEA$_{\text{PFSP}}$ is remarkable. In fact, the only significant differences between these two algorithms lie in the addition of the crowding distance diversity operator in AutoMOEA$_{\text{Cmax-TT}}$ and the population size, which in AutoMOEA$_{\text{Cmax-TT}}$ is half the size used by AutoMOEA$_{\text{PFSP}}$. These small differences are reflected on the similar rank sums they achieve. This time, however, the variant-specific AutoMOEA$_{\text{Cmax-TT}}$ outperforms the general AutoMOEA$_{\text{PFSP}}$ for several instances. Both algorithms obtain better hypervolume values for almost all test instances in comparison to the traditional MOEAs.

For the TFT-TT variant, results reflect the peculiarity of the problem's structure. For this variant, the number of nondominated solutions decreases as the problem size increases. This is confirmed by the fact that many algorithms applied to this variant tend to find a single solution for the larger instances. Having this mixed nature, the design of AutoMOEA$_{\text{TFT-TT}}$ is unable to outperform some algorithms across the whole benchmark. In fact, MOGA is the only algorithm to perform well in the instances with few nondominated solutions. We suspect this unexpectedly good performance of MOGA can be explained by its capacity to establish and maintain niches, an effective strategy for single-objective EAs. Given the peculiarities of this variant, the high rank sum achieved by AutoMOEA$_{\text{PFSP}}$ confirms that using general-purpose designed algorithms may eventually present sub-optimal performance, and that using a variant-specific design one can overcome to some extent this type of drawbacks.

This mixed characteristics of this variant are shown in the parallel coordinate plots depicted in Fig. 1. For the "multi-objective" instances like the ones with 50 jobs and 20 machines (top), AutoMOEA often performs best. However, for the "single-objective" ones, like the ones with 200 jobs and 20 machines (bottom), all algorithms perform clearly worse than MOGA. To verify whether we could automatically generate an AutoMOEA as good as MOGA for this problem,

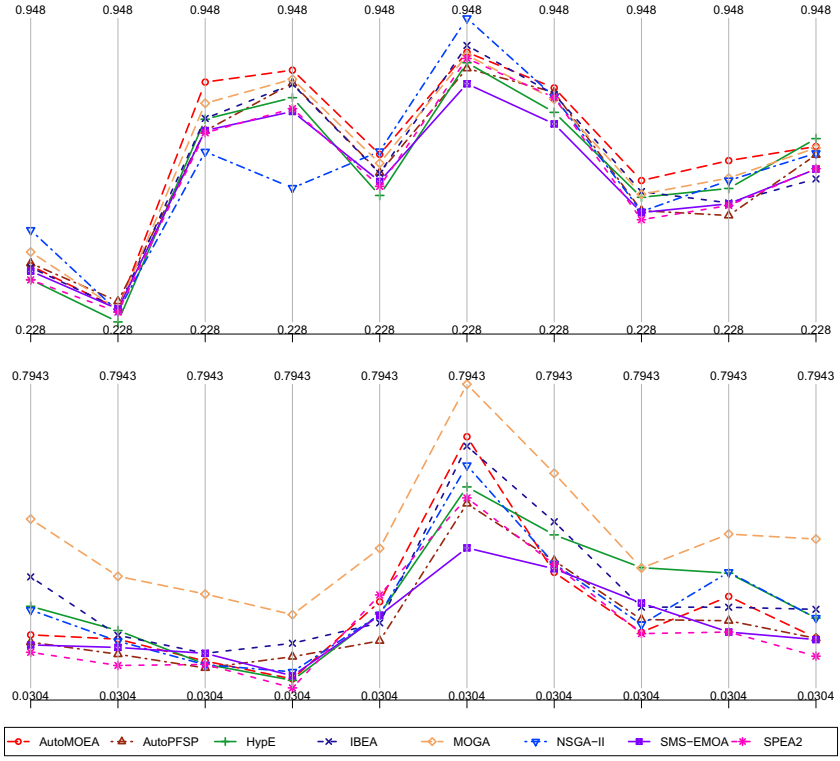

Fig. 1. MOEA's average hypervolume on 10 instances of Cmax-TFT. Top: 50 jobs and 20 machines. Bottom: 200 jobs and 20 machines. Each vertical line depicts one instance.

we ran irace again under a different setup. First, we added the generational removal policy from MOGA, which we had excluded from the parameter space in an early development stage because of the evidence favoring elitism throughout the MO literature. Second, we added the configuration of MOGA as an initial candidate for irace. However, this candidate was discarded in the first iteration, and the final configuration was very similar to the one presented here. This could also be explained by possible differences in the training and test sets.

To conclude, the structure of $\text{AutoMOEA}_{\text{Cmax-TFT-TT}}$ combines elements from $\text{AutoMOEA}_{\text{Cmax-TFT}}$ (mating) and $\text{AutoMOEA}_{\text{Cmax-TT}}$ (replacement), which was expected since these components proved to be effective for these objectives. Interestingly, only the replacement hypervolume-based component is maintained. This is likely explained by the computational overhead that the hypervolume-based components introduce, particularly for three-objective scenarios. Concerning the performance of this AutoMOEA, the rank sum depicted in Table 4 shows that this AutoMOEA achieves hypervolumes that are better over almost all test instances considered. $\text{AutoMOEA}_{\text{PFSP}}$ ranks second, confirming that it is a flexible but effective algorithm.

5 Conclusions

In this paper, we have investigated the automatic design of multi-objective evolutionary algorithms (MOEAs). We used an extended framework that combines components from ParadisEO-MOEO, PISA, and PaGMO to deal with four different variants of the permutation flowshop problem (PFSP). We have used irace, an offline algorithm configuration tool, to automatically select the components that were more effective for each PFSP variant, thus producing novel MOEAs. These automatically designed MOEAs (AutoMOEAs) have outperformed the traditional MOEAs on all variants considered, confirming that novel combinations of existing MOEAs can lead to much improvement on the state-of-the-art, and reinforcing the need for research works on automatic algorithm design.

The analysis conducted here has reached its two main goals. The first was to assess whether the state of the art of MOEAs could be improved (and by how much) by using an automatic algorithm design methodology. The results shown here confirm that improvements can be achieved in the context of combinatorial optimization, particularly for the PFSP. The second was to investigate whether a single AutoMOEA could potentially outperform the variant-taylored AutoMOEAs, given that the variants present different search space characteristics. In terms of its structure, AutoMOEA$_{\mathsf{PFSP}}$ is a simple yet flexible and efficient algorithm. Concerning performance, the more general version outperforms the variant-specific AutoMOEA for the Cmax-TFT variant, but it is unable to match the performance of the other variant-specific AutoMOEAs. However, AutoMOEA$_{\mathsf{PFSP}}$ often outperforms the other MOEAs considered in this work.

Finally, the results presented here are preliminary. More experimental analysis, ranging from combinatorial to continuous, would be required to assess the full potential of the automatic design of MOEAs. Nonetheless, these initial results suggest that our approach is not only feasible but that it may give significant insights about the role various algorithmic components of MOEAs play and lead to completely new MOEA designs that have never been tested or considered previously. Moreover, deeper analysis on individual components and reasons for the high performance of the automatically designed configurations are an important open research question.

References

1. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. Evolutionary Computation 19(1), 45–76 (2011)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. 181(3), 1653–1669 (2007)
3. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Automatic generation of multi-objective ACO algorithms for the biobjective knapsack problem. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Engelbrecht, A.P., Groß, R., Stützle, T. (eds.) ANTS 2012. LNCS, vol. 7461, pp. 37–48. Springer, Heidelberg (2012)
4. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Automatic design of evolutionary algorithms for multi-objective combinatorial optimization: Supplementary material (2014), http://iridia.ulb.ac.be/supp/IridiaSupp2014-007/

5. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Deconstructing multi-objective evolutionary algorithms: An iterative analysis on the permutation flowshop. In: Pardalos, P., et al. (eds.) LION 8. LNCS. Springer (2014)
6. Biscani, F., Izzo, D., Yam, C.H.: A global optimisation toolbox for massively parallel engineering optimisation (2010), http://arxiv.org/abs/1004.3824
7. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA – a platform and programming language independent interface for search algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 181–197 (2002)
9. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Comput. Oper. Res. 38(8), 1219–1236 (2011)
10. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Forrest, S. (ed.) ICGA, pp. 416–423. Morgan Kaufmann Publishers (1993)
11. Igel, C., Heidrich-Meisner, V., Glasmachers, T.: Shark. Journal of Machine Learning Research 9, 993–996 (2008)
12. Knowles, J.D., Corne, D.: Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computation 8(2), 149–172 (2000)
13. Liefooghe, A., Jourdan, L., Talbi, E.G.: A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. Eur. J. Oper. Res. 209(2), 104–112 (2011)
14. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011)
15. López-Ibáñez, M., Knowles, J., Laumanns, M.: On sequential online archiving of objective vectors. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 46–60. Springer, Heidelberg (2011)
16. López-Ibáñez, M., Stützle, T.: The automatic design of multi-objective ant colony optimization algorithms. IEEE Trans. Evol. Comput. 16(6), 861–875 (2012)
17. Minella, G., Ruiz, R., Ciavotta, M.: A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS Journal on Computing 20(3), 451–471 (2008)
18. Taillard, É.D.: Benchmarks for basic scheduling problems. Eur. J. Oper. Res. 64(2), 278–285 (1993)
19. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. Evolutionary Computation 8(2), 125–147 (2000)
20. Zhang, Q., Suganthan, P.N.: Special session on performance assessment of multi-objective optimization algorithms/CEC 2009 MOEA competition (2009)
21. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN VIII. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
22. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., et al. (eds.) EUROGEN 2001, pp. 95–100. CIMNE, Barcelona (2002)
23. Zitzler, E., Thiele, L., Bader, J.: On set-based multiobjective optimization. IEEE Trans. Evol. Comput. 14(1), 58–79 (2010)
24. Zitzler, E., Thiele, L., Deb, K.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)