A Study About Meta-Optimizing the NSGA-II Multi-Objective Evolutionary Algorithm

José F. Aldana-Martín[2], Antonio J. Nebro[1,2], Juan J. Durillo[3], and María del Mar Roldán García[1,2]

[1] Departamento de Lenguajes y Ciencias de la Computación. University of Málaga, 29071 Málaga, Spain
[2] ITIS Software, University of Málaga, 29071, Málaga, Spain
jfaldanam@uma.es, ajnebro@uma.es, mrgarcia@uma.es
[3] Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities, Germany
durillo@lrz.de

Abstract. The automatic design of multi-objective metaheuristics is an active research line aimed at, given a set of problems used as training set, to find the configuration of a multi-objective optimizer able of solving them efficiently. The expected outcome is that the auto-configured algorithm can be used of find accurate Pareto front approximations for other problems. In this paper, we conduct a study on the meta-optimization of the well-known NSGA-II algorithm, i.e., we intend to use NSGA-II as an automatic configuration tool to find configurations of NSGA-II. This search can be formulated as a multi-objective problem where the decision variables are the NSGA-II components and parameters and the the objectives are quality indicators that have to be minimized. To develop this study, we rely on the jMetal framework. The analysis we propose is aimed at answering the following research questions: RQ1 - how complex is to build the meta-optimization package?, and RQ2 - can accurate configurations be found? We conduct an experimentation to give an answer to these questions.

Keywords: Multi-objective optimization, auto-configuration of metaheuristics, NSGA-II

1 Introduction

The quality of the Pareto front approximations found by multi-objective evolutionary algorithms is affected on the values of their control parameters. This means that, given a set of problems to be optimized and a given algorithm, the user has to fine tune the algorithms parameters to get accurate results. The approach commonly adopted to carry out this task is to try to adjust the parameters manually by conducting pilot tests, which is a trial-and-error strategy. Furthermore, this process requires knowledge of the algorithm, which is not usually the case of the users expert in the problems. The consequence is that those users are likely to end up selecting a well-known algorithm, typically NSGA-II [1], with default settings.

In this context, an active research line is automatic algorithm configuration [2], consisting in taking a set of problems as training set to find a particular parameter configuration of the parameters to a produce version of the algorithm that, configured with them, can solve those problems efficiently. An extension of this idea is automatic algorithm design, where not only parameters but also algorithmic components can be combined to design a new algorithmic variant. An advantage of these approaches is that they can be supported by tools that help to find the configurations automatically, such as irace [3], paramILS [4], GSF [5] and SMAC3 [6]. Focusing on multi-objective evolutionary algorithms, irace has been applied in several works [7][8][9].

In this paper, we conduct a study about the use of NSGA-II to find configurations of NSGA-II, i.e., using NSGA-II as meta-optimizer. The basic idea is to consider the auto-design of NSGA-II as a multi-objective problem, where the decision variables represent parameters and components and the objectives can be combinations of quality indicators [10].

Our motivation stems, first, from our experiences in automatic design of multi-objective metaheuristics, which are based on combining the jMetal optimization framework [11, 12] with irace to find configurations of NSGA-II [13][14] and particle swarm optimizers [15]. Second, a recent survey [2] that remarked as future research prospects easy-to-use algorithm tuning and multi-objective approaches. Although our proposal does not include a toolbox (as suggested the mentioned survey [2]), we design a package based on jMetal, so we do not need to use external tools such as irace, thus simplifying the auto-design process in case the optimization problems are implemented with that framework.

We define two research questions that we intend to answer in our our study:

- RQ1: how complex is to build the meta-optimization package?. We are interested in a simple and easy-to-use software solution.
- RQ2: can accurate configurations be found? The search capabilities of the meta-optimizer must be validated by conducting representative experiments.

The rest of the paper is organized as follows. Section 2 provides a detailed description of the presented approach for the automated design of a meta-optimizer for NSGA-II. In Section 3, we present the results of three experiments conducted to validate our proposal. The findings and implications of these experiments are discussed in Section 4, while Section 5 provides conclusions about the effectiveness and usefulness of our study.

2 Meta-Optimization Approach

The process of auto-designing evolutionary algorithms requires three elements: the design space, an algorithmic template, and an auto-design tool. We describe these elements next, including how we cope with them.

2.1 Design space

The design space is composed of the algorithm parameters and components, their types, allowed values, and, optionally, constraints. In the case of NSGA-II, we consider a flexible definition of it, in which a multi-objective evolutionary algorithm adopting a replacement strategy based on dominance ranking and the crowding distance density estimator is considered a NSGA-II variant. We define the design space detailed in Table 1, which is similar to the ones used in former works [9][14] (please refer to these references for a detailed explanation of the parameters and components).

Parameter/Component	**Type**	**Domain**	
algorithmResult	c	{externalArchive, population}	
populationSizeWithArchive	i	[10, 200]	s.t. algorithmResult == externalArchive
externalArchive	c	{crowdingDistance, unbounded}	s.t. algorithmResult == externalArchive
offspringPopulationSize	i	[1, 400]	
selection	c	{tournament, random}	
selectionTournamentSize	i	[2, 10]	s.t. selection == tournament
createInitialSolutions	c	{random, latinHypercubeSampling, scatterSearch}	
crossover	c	{SBX, BLX_ALPHA, wholeArithmetic}	
crossoverProbability	r	[0.0, 1.0]	
crossoverRepairStrategy	c	{random, round, bounds}	
sbxDistributionIndex	r	[5.0, 400.0]	s.t. crossover == SBX
blxAlphaCrossoverAlphaValue	r	[0.0, 1.0]	s.t. crossover == BLX_ALPHA
mutation	c	{uniform, polynomial, linkedPolynomial, nonUniform}	
mutationProbabilityFactor	r	[0.0, 2.0]	
mutationRepairStrategy	c	{random, round, bounds}	
polynomialMutationDistributionIndex	r	[5.0, 400.0]	s.t. mutation ∈ {polynomial, linkedPolinomial}
uniformMutationPerturbation	r	[0.0, 1.0]	s.t. mutation == uniform
nonUniformMutationPerturbation	r	[0.0, 1.0]	s.t. mutation == nonUniform

Table 1: Design space of the configurable NSGA-II in jMetal. Types: (c)ategorical, (i)nteger, (r)eal.

An example of parameter is the offspring population size, which is an integer variable taking values in the range [1, 400] (a value of 1 would lead to a steady-state version of NSGA-II). Examples of components are the crossover and mutation operators. Each operators can in turn also have specific parameters, such as the distribution index for SBX crossover.

A design decision is whether NSGA-II uses an external archive (i.e., an auxiliary population) or not. If the population size is P, the idea is that any evaluated solution is inserted into the archive, which keeps only non-dominated solutions, and the result of the algorithm would be P solutions from the archive; in this case, the population size is not fixed and it can take a value between 10 and 200. The external archive can be bounded (the crowding distance is used as density estimator to remove solutions when the archive size is greater than P) or unbounded (in this case, all the evaluated solutions are inserted and, when the algorithm finishes, P evenly spread solutions are returned).

2.2 Algorithmic template

Since release 6.0, jMetal includes a *jmetal-auto* package containing an implementation of NSGA-II, called AutoNSGAII, which can take any valid combination of the parameters and components of Table 1, generating different NSGA-II versions.

The input of AutoNSGAII is a string containing all the parameter names and their values. This string is parsed internally and AutoNSGAII is configured with the parameter values and the components specified in the string. An example of a subset of this string is the following: "*–archiveResult externalArchive –offpringPopulation 40 –selection tournament ...*"

2.3 Meta-optimizer

In our previous works combining jMetal with irace [13][14], the finding of configurations is based on running irace, which generates combinations of valid configurations according to the design space. For each configuration, irace runs AutoNSGAII, which returns as a result the value of a quality indicator; this value is taken by irace as a measure of the quality of the configuration.

As we intend to replace irace by the NSGA-II algorithm implemented in jMetal, which would act as meta-optimizer, we have to formulate and implement the optimization problem that would to be solved by the meta-optimizer. This problem has the following parameters:

- List of problems used as training set.
- List of quality indicators, being each indicator an objective to be minimized.
- The population size of AutoNSGAII.
- The stopping condition of AutoNSGAII (in terms of number of evaluations).
- Number of independent runs of AutoNSGAII for each configuration to be evaluated.

To define the problem encoding, the approach we have adopted is simple: every parameter of Table 1 is represented as a real value in the range $[0.0, 1.0]$, so the solutions are composed of 18 decision variables. When a solution has to be evaluated, the variables are decoded to construct the parameter string that is used when calling AutoNSGAII. The decoding is done as follows:

- Real parameter: the value is scaled up from $[0.0, 1.0]$ to the range of the parameter (e.g.,. $[5.0, 400.0]$ in the case of the SBX distribution index).
- Integer parameter: same procedure as for real parameters, but the resulting value is truncated.
- Categorical parameter: the interval $[0, 0, 1, 0]$ is divided into sub-intervals according to the number of parameter values, and the index of the sub-interval is used to obtain the actual categorical value.

Once the parameter string is decoded, AutoNSGAII is called to solve all the problems of the training set as many times as the number of independent runs. For each obtained front, the quality indicators are computed and the resulting objectives values of evaluating a configuration is the median of the median of the quality indicators of all the problems of the training set.

We now look at the pros and cons of this approach. Starting by the cons, we are not considering parameter constraints, so all the elements of design space are included although some of them may be ignored (e.g., the uniform perturbation is useless if the selected mutation operator is polynomial), and the discretization of categorical parameters using sub-intervals can lead to different solutions being equivalent if all variables have the same values except one corresponding to a categorical parameter whose values are in the same sub-interval. As advantages, the encoding is very simple and any multi-objective algorithm in jMetal able of solving continuous problems can be used as meta-optimizer.

3 Experimentation

We aim to empirically validate our approach with a set of experiments grouped into two different scenarios. These experiments are described below, detailing their purpose, expected outcomes and results.

The meta-optimizer is configured with the additive epsilon (EP) and normalized hypervolume (NHV) quality indicators as the objective functions to be minimized. The first indicator measures the convergence of a Pareto front approximation while the second one takes into account both convergence and diversity [10]. We use NHV instead of plain hypervolume as for this latter, the bigger its value the better, while jMetal minimizes objective functions by default. NHV is defined as 1.0 minus the hypervolume of the front divided by the hypervolume of the reference front.

For the meta-optimizer, we have configured it with common NSGA-II parameter values. The population size is 50 and the variation operators are SBX crossover (with probability 0.9 and a distribution index value of 20.0) and polynomial mutation (with probability $1/n$, being n the number of decision variables of the problem, and a distribution index value of 20.0). We set the stopping to condition to 3000 function evaluations. The NSGA-II implementation in jMetal can be executed in parallel both using a synchronous or an asynchronous scheme [16].

Next, we define two scenarios and three experiments.

3.1 Scenario 1: Finding Configurations for Single Problems

The first scenario is aimed at determining whether our meta-optimization approach is able of finding well-performing configurations of NSGA-II for single problems. For that, we focus on experimenting with two problems:

- **Experiment 1 - problem ZDT4**: this problem [17] is a bi-objective multi-frontal problem, whose default configuration consists of 10 decision variables. The standard NSGA-II has difficulty in providing Pareto front approximations with a uniform spread of solutions. Previous studies [13] have shown that using a steady-approach can significantly improve the diversity of the fronts. Pilot tests also indicate that comparable improvements can be achieved when using an external bounded archive.
- **Experiment 2 - problem DTLZ3**: This problem belongs to the DTLZ benchmark [18]. It is formulated with a default configuration consisting of twelve decision variables and three objectives. DTLZ3 is also a multi-modal problem with a convex Pareto front. The study presented in [9] showed that both, NSGA-II and the AutoNSGAII, configured with irace were unable to find accurate approximated fronts in terms of convergence and diversity for this problem.According to other works [19], NSGA-II is able of finding accurate fronts for problem DTLZ2 when using an external unbounded archive and retrieving from it a subset of evenly distributed solutions. DLTZ2 is not multi-modal but shares many similarities with DTLZ3 (i.e., convex Pareto front, three objectives, and twelve decision variables). Our aim here is twofold: 1) to determine whether the meta-optimizer is able of finding a configuration to effectively solve DTLZ3; and, 2) to check if that configuration includes an unbounded archive.

3.2 Scenario 2: Finding Configurations for Sets of Problems

The above scenario must validate the potential of auto-configuration applied to optimize single problems. The found configurations may be, however, too specific to that particular problem, and perform poorly for other problems (overfitting). Our second scenario addresses this issue by auto-configuring the algorithm on a set of problems instead of just one. Additionally, the obtained configurations are validated using different sets of problems.

- **Experiment 3 - WFG benchmark**: This experiment aims to replicate the study presented in [9]. NSGA-II is configured for optimizing the nine problems of the WFG suite [20], which are *training set*. The found configurations are later used to solve both the WFG problems and the seven instances of the DTLZ family problems–the *validation set*. All the problems in this experiment are formulated as bi-objective ones.

Parameter	**NSGA-II**	**Exp. 1**	**Exp. 2**	**Exp. 3**
populationSize	100	100	100	100
createInitialSolutions	random	LHS	scatterSearch	random
algorithmResult	population	externalArchive	externalArchive	externalArchive
externalArchive	-	CD	unboundedArchive	CD
populationSizeWithArchive	-	106	58	61
offspringPopulationSize	100	60	130	68
crossover	SBX	SBX	SBX	BLX_ALPHA
crossoverProbability	0.9	0.991	0.942	0.858
crossoverRepairStrategy	random	round	random	bounds
sbxDistributionIndexValue	20.0	5.11	70.479	-
blxAlphaCrossoverAlphaValue	-	-	-	0.547
mutation	polynomial	polynomial	uniform	linkedPolynomial
mutationProbabilityFactor	1	0.76	0.699	0.161
mutationRepairStrategy	random	bounds	round	round
polynomialMutationDistributionIndex	20	32.23	-	-
linkedPolynomialMutationDistributionIndex	-	-	-	11.335
uniformMutationPerturbation	-	-	0.417	-
selection	tournament	tournament	random	tournament
selectionTournamentSize	2	9	-	4

Table 2: Best configuration found for the NSGA-II on each experiment. (LHS; latinHypercube-Sampling, CD; crowdingDistanceArchive)

3.3 Results

We report and analyze the results obtained on the three defined experiments. In all the cases, the number of independent runs per configuration is set to 3.

Experiment 1 We set the stopping condition of AutoNSGAII to perform a total 15000 function evaluations. Fig. 1 shows computed fronts by the meta-optimizer at 1000, 2000, and 3000 evaluations. As shown, the final front is composed of only one solution, and the figure suggests that the meta-optimizer might not have converged in the performed evaluations. The found design in this experiment (see Table 2) includes a bounded external archive with crowding distance; as commented before, the use of this kind of archive is known to be beneficial for converging and for achieving a front of evenly spread solutions.

AutoNSGAII with the obtained configuration is compared with NSGA-II with default settings next. We set the stopping condition to 25000 function evaluations in both cases and compare the Pareto front approximations computed by both algorithms. We observe that the front computed with the found configuration (Fig. 2 right) has a noticeable better convergence and spread than the approximation computed by NSGA-II with standard the default setting (Fig. 2 left).

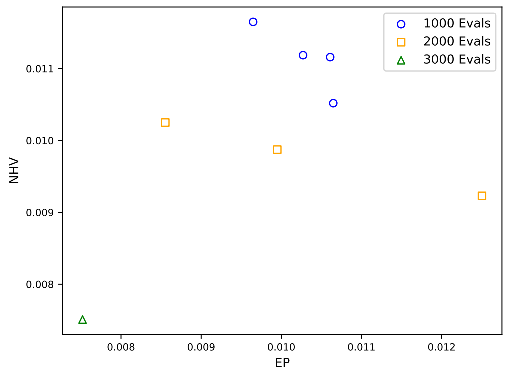

Fig. 1: Problem ZDT4. Evolution of the front generated by the meta-optimizer.

Experiment 2 In this experiment, the stopping criterion for AutoNSGAII has been raised to 20000 evaluations.

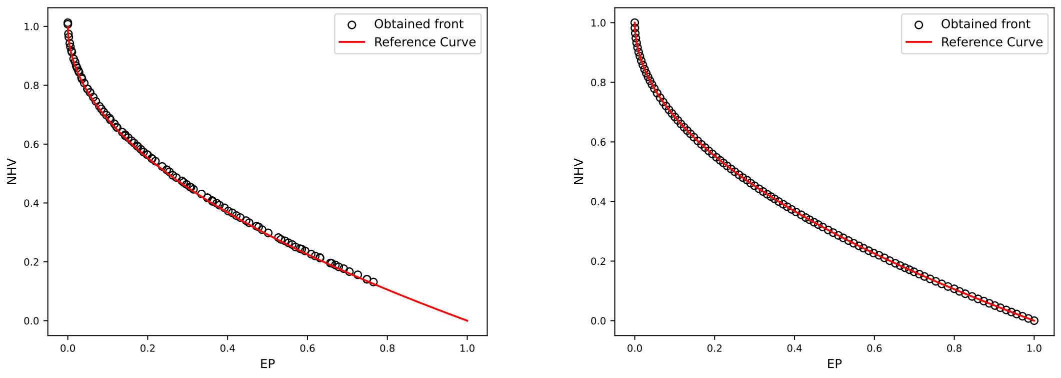

Fig. 2: Problem ZDT4. Pareto front approximation found by the standard NSGA-II (left), and Pareto front approximation found by the auto-designed NSGA-II (right).

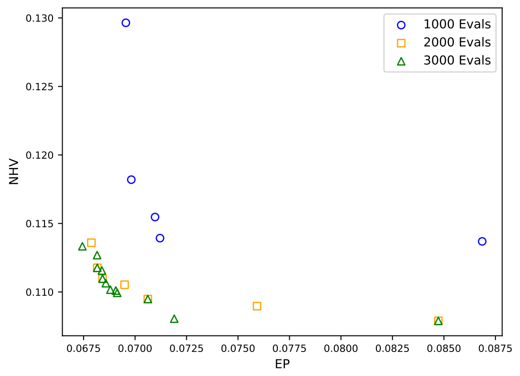

Fig. 3: Problem DTLZ3. Evolution of the front generated by the meta-optimizer.

Fig. 3 shows the approximation fronts computed after 1000, 2000 and 3000 evaluations. In this case, the figure suggest that the meta-optimizer has almost converged after performing the 3000 function evaluations. The computed approximation front consists of twelve points. The configuration corresponding to the point with the lowest NHV value (on the right end) is included in Table 2. As expected, the configuration found by the meta-optimizer uses the unbounded external archive.

In Fig. 4, we compare the approximation front computed with NSGA-II and the one computed with the configuration found the meta-optimizer using AutoNSGAII. In both cases, we use 40000 function evaluations as stopping criterion. The graph shows remarkable differences between the front computed by NSGA-II (poor convergence and coverage of the Pareto front approximation) and AutoNSGAII.

Experiment 3 In alignment with existing work [9], we set the stopping criterion of AutoNSGAII to 25000 evaluations for this experiment. The evolution of the fronts over different number of evaluations is shown in Fig. 5. As in the previous experiment, the point with the minimum NHV value is taken and its corresponding configuration is used to compare with the results reported in [9]. The comparison in this case includes NSGA-II, and SMPSO [21] with their default settings and AutoNSGAII with the mentioned configuration.

The chosen configuration from the meta-optimizer is summarized in Table 2. Interestingly, this configuration is similar to the one computed in [9]: both share the use use BLX_ALPHA crossover and an external bounded archive).

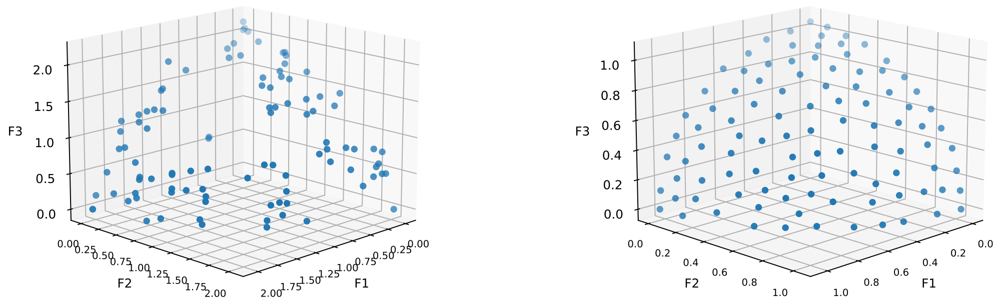

Fig. 4: Problem DTLZ3. Pareto front approximation found by the standard NSGA-II (left), and Pareto front approximation found by the auto-designed NSGA-II (right).

Table 3 showcase the validation results of the auto-designed NSGA-II for the WFG and DTLZ benchmarks. Tables (a) and (b) contains the Hypervolume indicator values and Tables (c) and (d) the Epsilon ones. As a general remark, the configurations found by our proposal yield similar indicator values (each cell includes the median of 25 independent runs) than those presented in previous work [9]. Additionally, this results are also supported by the Wilcoxon rank sum statistical test for significance. The results of the Wilcoxon test are included in Table 4. We can observe that statistical confidence has been found in most of the results.

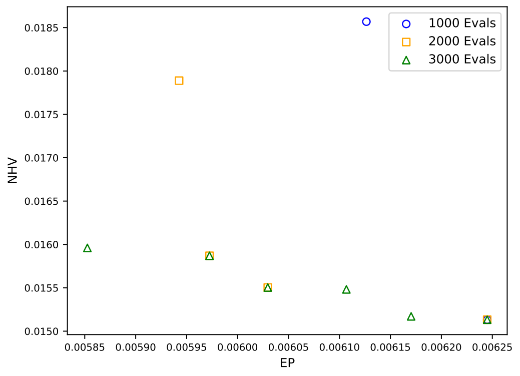

Fig. 5: WFG problem family. Evolution of the front generated by the meta-optimizer.

4 Discussion

Once we have conducted the two defined experiments, we revisit the two formulated research questions in the introduction, and we attempt to answer them based on the obtained results.

4.1 Research Questions

RQ1 - approach complexity: Our meta-optimization package relies only on jMetal code, so it does not require any external tool. The AutoNSGAII template within jMetal, designed and used in former studies, combined with irace simplifies the formulation of the auto-design of NSGA-II as a continuous optimization problem. We hope that researchers familiar with jMetal can benefit from the use of the meta-optimizer with little effort.

	NSGAII	SMPSO	AutoNSGAII
WFG1	$4.35e-01_{1.8e-01}$	$1.17e-01_{8.0e-03}$	$6.34e-01_{1.8e-05}$
WFG2	$5.61e-01_{2.3e-03}$	$5.61e-01_{1.6e-03}$	$5.64e-01_{9.2e-05}$
WFG3	$4.92e-01_{8.3e-04}$	$4.92e-01_{6.1e-04}$	$4.95e-01_{6.1e-05}$
WFG4	$2.17e-01_{3.7e-04}$	$2.03e-01_{2.4e-03}$	$2.18e-01_{1.5e-03}$
WFG5	$1.95e-01_{3.5e-04}$	$1.96e-01_{7.8e-05}$	$1.96e-01_{9.8e-05}$
WFG6	$2.01e-01_{1.3e-02}$	$2.09e-01_{5.0e-04}$	$2.02e-01_{1.4e-02}$
WFG7	$2.09e-01_{5.5e-04}$	$2.09e-01_{2.7e-04}$	$2.11e-01_{3.5e-05}$
WFG8	$1.47e-01_{2.7e-03}$	$1.47e-01_{3.4e-03}$	$1.40e-01_{3.1e-03}$
WFG9	$2.37e-01_{1.8e-03}$	$2.35e-01_{5.8e-04}$	$2.39e-01_{2.0e-03}$
DTLZ1	$4.88e-01_{7.9e-03}$	$4.94e-01_{2.7e-04}$	$0.00e+00_{4.9e-01}$
DTLZ2	$2.09e-01_{4.7e-04}$	$2.10e-01_{1.6e-04}$	$2.11e-01_{3.6e-05}$
DTLZ3	$0.00e+00_{1.8e-02}$	$2.10e-01_{1.2e-01}$	$0.00e+00_{0.0e+00}$
DTLZ4	$2.09e-01_{2.1e-01}$	$2.10e-01_{8.7e-05}$	$2.11e-01_{7.2e-05}$
DTLZ5	$2.11e-01_{3.4e-04}$	$2.12e-01_{2.2e-04}$	$2.12e-01_{4.5e-05}$
DTLZ6	$1.82e-01_{3.6e-02}$	$2.12e-01_{8.1e-05}$	$2.12e-01_{4.2e-05}$
DTLZ7	$3.34e-01_{3.0e-04}$	$3.35e-01_{1.2e-04}$	$3.35e-01_{9.2e-05}$

(a) Current study results using as objective the Hypervolume.

	NSGAII	SMPSO	AutoNSGAII
WFG1	$4.49e-01_{7.6e-02}$	$1.16e-01_{7.7e-03}$	$6.34e-01_{2.6e-05}$
WFG2	$5.64e-01_{9.5e-04}$	$5.62e-01_{1.2e-03}$	$5.65e-01_{5.1e-05}$
WFG3	$4.41e-01_{3.8e-04}$	$4.41e-01_{2.2e-04}$	$4.42e-01_{1.1e-05}$
WFG4	$2.17e-01_{7.6e-04}$	$2.03e-01_{2.4e-03}$	$2.17e-01_{3.0e-03}$
WFG5	$1.95e-01_{2.9e-04}$	$1.96e-01_{7.5e-05}$	$1.96e-01_{1.0e-04}$
WFG6	$2.03e-01_{8.9e-03}$	$2.09e-01_{4.3e-04}$	$2.08e-01_{1.3e-02}$
WFG7	$2.09e-01_{3.5e-04}$	$2.09e-01_{3.2e-04}$	$2.11e-01_{3.1e-05}$
WFG8	$1.48e-01_{2.3e-02}$	$1.48e-01_{1.0e-03}$	$1.39e-01_{2.3e-03}$
WFG9	$2.37e-01_{2.9e-03}$	$2.35e-01_{8.2e-04}$	$2.39e-01_{1.9e-03}$
DTLZ1	$4.66e-01_{1.6e-01}$	$4.94e-01_{1.9e-04}$	$0.00e+00_{0.0e+00}$
DTLZ2	$2.09e-01_{2.7e-04}$	$2.10e-01_{1.5e-04}$	$2.11e-01_{4.1e-05}$
DTLZ3	$0.00e+00_{0.0e+00}$	$2.10e-01_{6.3e-02}$	$0.00e+00_{0.0e+00}$
DTLZ4	$2.10e-01_{7.1e-04}$	$2.10e-01_{1.5e-04}$	$2.11e-01_{4.2e-05}$
DTLZ5	$2.11e-01_{3.5e-04}$	$2.12e-01_{1.3e-04}$	$2.12e-01_{4.1e-05}$
DTLZ6	$1.89e-05_{1.4e-03}$	$2.12e-01_{6.9e-05}$	$2.12e-01_{5.6e-05}$
DTLZ7	$3.29e-01_{2.8e-04}$	$3.30e-01_{9.8e-05}$	$3.30e-01_{7.3e-05}$

(b) Results obtained from [9] with irace using as objective the Hypervolume.

	NSGAII	SMPSO	AutoNSGAII
WFG1	$2.94e-01_{2.7e-01}$	$4.56e-01_{1.3e-02}$	$6.19e-03_{5.6e-04}$
WFG2	$1.81e-01_{1.7e-01}$	$6.70e-03_{2.8e-03}$	$4.03e-03_{5.0e-04}$
WFG3	$1.33e-02_{2.8e-03}$	$7.39e-03_{8.4e-04}$	$5.40e-03_{2.2e-04}$
WFG4	$1.21e-02_{3.6e-03}$	$2.19e-02_{2.6e-03}$	$6.33e-03_{1.1e-03}$
WFG5	$3.31e-02_{2.8e-03}$	$2.78e-02_{3.9e-04}$	$2.76e-02_{1.8e-04}$
WFG6	$1.49e-02_{1.0e-02}$	$6.19e-03_{5.9e-04}$	$1.00e-02_{9.0e-03}$
WFG7	$1.20e-02_{4.4e-03}$	$6.34e-03_{9.1e-04}$	$5.18e-03_{2.6e-04}$
WFG8	$2.44e-01_{1.0e-01}$	$1.75e-01_{2.1e-02}$	$2.45e-01_{1.0e-03}$
WFG9	$1.47e-02_{2.7e-03}$	$1.12e-02_{1.2e-03}$	$7.19e-03_{1.6e-03}$
DTLZ1	$1.60e-02_{5.2e-03}$	$6.30e-03_{7.0e-04}$	$5.16e-01_{1.0e+00}$
DTLZ2	$1.23e-02_{2.8e-03}$	$5.59e-03_{3.7e-04}$	$5.23e-03_{2.2e-04}$
DTLZ3	$1.14e+00_{1.4e+00}$	$6.20e-03_{7.0e-01}$	$1.16e+01_{8.4e+00}$
DTLZ4	$1.18e-02_{9.9e-01}$	$5.68e-03_{3.7e-04}$	$5.38e-03_{3.0e-04}$
DTLZ5	$1.14e-02_{2.5e-03}$	$5.25e-03_{2.5e-04}$	$5.05e-03_{2.1e-04}$
DTLZ6	$2.77e-02_{2.4e-02}$	$5.22e-03_{6.6e-04}$	$5.09e-03_{1.9e-04}$
DTLZ7	$1.02e-02_{3.5e-03}$	$4.68e-03_{4.6e-04}$	$4.50e-03_{3.5e-04}$

(c) Current study results using as objective the Epsilon.

	NSGAII	SMPSO	AutoNSGAII
WFG1	$4.52e-01_{2.4e-01}$	$4.55e-01_{9.8e-03}$	$5.99e-03_{7.3e-04}$
WFG2	$5.41e-03_{2.4e-03}$	$6.04e-03_{1.1e-03}$	$3.88e-03_{5.5e-04}$
WFG3	$3.34e-01_{5.3e-04}$	$3.34e-01_{1.9e-04}$	$3.33e-01_{2.7e-07}$
WFG4	$1.29e-02_{2.2e-03}$	$2.16e-02_{3.4e-03}$	$6.72e-03_{1.2e-03}$
WFG5	$3.31e-02_{3.7e-03}$	$2.77e-02_{1.9e-04}$	$2.76e-02_{1.6e-04}$
WFG6	$1.50e-02_{5.9e-03}$	$6.37e-03_{4.9e-04}$	$6.13e-03_{7.2e-03}$
WFG7	$1.28e-02_{4.0e-03}$	$6.52e-03_{4.9e-04}$	$5.20e-03_{1.9e-04}$
WFG8	$1.68e-01_{1.0e-01}$	$1.69e-01_{1.7e-02}$	$2.45e-01_{1.3e-03}$
WFG9	$1.42e-02_{2.0e-03}$	$1.10e-02_{1.9e-03}$	$7.39e-03_{1.2e-03}$
DTLZ1	$3.53e-02_{1.5e-01}$	$6.30e-03_{5.5e-04}$	$3.08e+01_{1.7e+01}$
DTLZ2	$1.12e-02_{4.2e-03}$	$5.53e-03_{3.2e-04}$	$5.28e-03_{2.5e-04}$
DTLZ3	$1.00e+01_{6.4e+00}$	$5.97e-03_{3.5e-01}$	$1.05e+02_{3.6e+01}$
DTLZ4	$1.18e-02_{5.4e-03}$	$5.61e-03_{3.1e-04}$	$5.32e-03_{1.9e-04}$
DTLZ5	$1.01e-02_{2.2e-03}$	$5.12e-03_{3.5e-04}$	$5.03e-03_{2.7e-04}$
DTLZ6	$3.72e-01_{5.3e-02}$	$5.15e-03_{4.5e-04}$	$5.06e-03_{2.9e-04}$
DTLZ7	$7.78e-03_{2.6e-03}$	$4.30e-03_{2.3e-04}$	$4.06e-03_{2.8e-04}$

(d) Results obtained from [9] with irace using as objective the Epsilon.

Table 3: The cells include the median and interquartile range of 25 independent runs. The dark-grey and light-grey background cells indicate, respectively, the best and second best indicator values.

	SMPSO	AutoNSGAII
NSGAII	▲ – ▽ ▲ ▽ ▽ ▽ – ▲ ▽ ▽ ▽ ▽ ▽ ▽ ▽	▽ ▽ ▽ – ▽ – ▽ ▲ ▽ ▲ ▽ ▲ ▽ ▽ ▽ ▽
SMPSO		▽ ▽ ▽ ▽ – ▲ ▽ ▲ ▽ ▲ ▽ ▲ ▽ ▽ ▽ –

(a) Current study results using as objective the Hypervolume.

	SMPSO	AutoNSGAII
NSGAII	▲ ▲ ▽ ▲ ▽ ▽ – – ▲ ▽ ▽ ▽ ▽ ▽ ▽ ▽	▽ ▽ ▽ – ▽ – ▽ ▲ ▽ ▲ ▽ – ▽ ▽ ▽ ▽
SMPSO		▽ ▽ ▽ ▽ – – ▽ ▲ ▽ ▲ ▽ ▲ ▽ ▽ ▽ ▽

(b) Results obtained from [9] with irace using as objective the Hypervolume.

	SMPSO	AutoNSGAII
NSGAII	▲ ▽ ▽ ▲ ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽	▽ ▽ ▽ ▽ ▽ ▽ ▽ ▲ ▽ – ▽ ▲ ▽ ▽ ▽ ▽
SMPSO		▽ ▽ ▽ ▽ ▽ ▲ ▽ ▲ ▽ ▲ ▽ ▲ ▽ ▽ ▽ –

(c) Current study results using as objective the Epsilon.

	SMPSO	AutoNSGAII
NSGAII	– – ▽ ▲ ▽ ▽ ▽ – ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽	▽ ▽ ▽ ▽ ▽ ▽ ▽ ▲ ▽ ▲ ▽ ▲ ▽ ▽ ▽ ▽
SMPSO		▽ ▽ ▽ ▽ ▽ – ▽ ▲ ▽ ▲ ▽ ▲ ▽ ▽ – ▽

(d) Results obtained from [9] with irace using as objective the Epsilon.

Table 4: Wilcoxon rank sum test results. The symbols in each cell correspond to problems WFG1-9 and DTLZ1-7. The symbols indicate: "–" no stadistical significance, "▲" the algorithm in the row has a better indicator value than the algorithm in the row with confidence and "▽" the algorithm in the row has a worse indicator value than the algorithm in the row with confidence

RQ2 - finding of accurate designs: We have adopted a simple encoding for the NSGA-II configurations consisting in codifying each parameters as a floating point value in the range $[0.0, 1.0]$, and these values are further decoded into a string that used as the input of the AutoNSGAII template. This configuration has been proved effectively by our empirical experiments. The first two experiments showed that the meta-optimizer has been able to generate the expected key components required by NSGA-II to converge to the Pareto front of the selected problems. For these experiments we provided visual evidence. Additionally, the generalization capabilities of the auto-tuner were challenged by requiring it to find an accurate configuration for a training set composed of nine problems, and validating the found configurations on a set of additional seven problems. The experiment we have accomplished shows almost identical results to the previously published work were irace was used as auto-configuration tool.

4.2 Further Remarks

Our empirical evaluation also revealed a few issues that are worth discussing:

- The formulation of searching designs for NSGA-II as a continuous problem opens the opportunity of using most of the metaheuristics provided by jMetal as meta-optimizers. This enable the easy development comparative studies based on configuring AutoNSGAII with different training sets.
- Although we have used two quality indicators, EP and NHV, as objectives for guiding the search, the inclusion of additional ones (e.g., spread, inverted generational distance, etc.) could reveal new insights regarding the configurations for solving different problems.
- We used NSGA-II with standard settings as the meta-optimizer. The obtained results in this paper could be used in order to analyze whether its performance could be improved if using different parameter settings.
- We have performed only a run of the meta-optimizer in the experiments. A deeper study should be carried out by performing a number of independent runs and making statistical analysis of the results.

5 Conclusions and Future Work

In this paper we have presented a study in which the NSGA-II algorithm is used as a meta-optimizer, i.e., as a tool that, given a set of problems as training set, is aimed at finding configurations that include NSGA-II parameters and components. By using a simple encoding scheme and the features existing in jMetal that were developed in former studies, our proposal is 100% developed in jMetal, so no external tools are required.

We have defined an experimentation to validate our proposal considering two scenarios and three experiments to cover both automatic search of NSGA-II designs for single and multi-problem training sets. The outcomes of these experiments reveal that the meta-optimizer is able of finding configurations of NSGA-II that successfully achieve the defined goals.

We have indicated a number of open research lines in the discussion section. Additionally, to reduce the computing time of the meta-optimization, we have set in Experiments 1 and 2 a number of function evaluations which is lower than the used when validating the configurations; we are interested in studying to what extent the number of evaluations can be reduced in the search while the resulting NSGA-II designs are able of solving the problems efficiently.

Acknowledgments

This work has been partially funded by the Spanish Ministry of Science and Innovation via Grant PID2020-112540RB-C41 (AEI/FEDER, UE). José F. Aldana-Martín is supported by Grant PRE2021-098594 (Spanish Ministry of Science, Innovation and Universities).

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
2. Huang, C., Li, Y., Yao, X.: A survey of automatic parameter tuning methods for metaheuristics. IEEE Trans. Evol. Comput. **24** (2020) 201–216
3. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3** (2016) 43–58
4. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: Paramils: An automatic algorithm configuration framework. J. Artif. Int. Res. **36** (2009) 267–306
5. Yi, W., Qu, R., Jiao, L., Niu, B.: Automated design of metaheuristics using reinforcement learning within a novel general search framework. IEEE Transactions on Evolutionary Computation (2022) 1–1
6. Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., Hutter, F.: Smac3: A versatile bayesian optimization package for hyperparameter optimization. Journal of Machine Learning Research **23** (2022) 1–9
7. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Automatic component-wise design of multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation **20** (2016) 403–417
8. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T. In: Automatic Configuration of Multi-objective Optimizers and Multi-objective Configuration. Springer International Publishing, Cham (2020) 69–92
9. Nebro, A.J., López-Ibáñez, M., Barba-González, C., García-Nieto, J.: Automatic configuration of NSGA-II with jMetal and irace. Genetic and Evolutionary Computation Conference (2019) 1374–1381
10. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7** (2003) 117–132
11. Durillo, J.J., Nebro, A.J.: jMetal: A java framework for multi-objective optimization. Advances in Engineering Software **42** (2011) 760–771
12. Nebro, A.J., Durillo, J.J., Vergne, M.: Redesigning the jMetal multi-objective optimization framework. Genetic and Evolutionary Computation Conference (2015) 1093–1100
13. Durillo, J.J., Nebro, A.J., Luna, F., Alba, E.: On the effect of the steady-state selection scheme in multi-objective genetic algorithms. In Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.K., Sevaux, M., eds.: Evolutionary Multi-Criterion Optimization, Berlin, Heidelberg, Springer Berlin Heidelberg (2009) 183–197
14. Nebro, A.J., Galeano-Brajones, J., Luna, F., Coello Coello, C.A.: Is nsga-ii ready for large-scale multi-objective optimization? Mathematical and Computational Applications **27** (2022)
15. Doblas, D., Nebro, A.J., López-Ibáñez, M., García-Nieto, J., Coello Coello, C.A.: Automatic design of multi-objective particle swarm optimizers. In Dorigo, M., Hamann, H., López-Ibáñez, M., García-Nieto, J., Engelbrecht, A., Pinciroli, C., Strobel, V., Camacho-Villalón, C., eds.: Swarm Intelligence, Cham, Springer International Publishing (2022) 28–40
16. Durillo, J.J., Nebro, A.J., Luna, F., Alba, E.: A study of master-slave approaches to parallelize nsga-ii. In: 2008 IEEE International Symposium on Parallel and Distributed Processing. (2008) 1–8
17. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8** (2000) 173–195
18. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multi-Objective Optimization. In Abraham, A., Jain, L., Goldberg, R., eds.: Evolutionary Multiobjective Optimization. Theoretical Advances and Applications. Springer (2001) 105–145
19. Ishibuchi, H., Pang, L.M., Shang, K.: A new framework of evolutionary multi-objective algorithms with an unbounded external archive. In Giacomo, G.D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J., eds.: ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020). Volume 325 of Frontiers in Artificial Intelligence and Applications., IOS Press (2020) 283–290
20. Huband, S., Hingston, P., Barone, L., While, L.: A review of multi-objective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation **10** (2006) 477–506
21. Nebro, A., Durilllo, J., García-Nieto, J., Coello Coello, C., Luna, F., Alba, E.: SMPSO: a new pso-based metaheuristic for multi-objective optimization. In: 2009 IEEE Symposium on Computational Intelligence in Multi-criteria decision-making (MCDM 2009), IEEE Computer Society (2009) 66–73