Self-Adaptation Techniques for Multi-Objective Evolutionary Algorithms

Saúl Zapotecas Martínez,* Edgar G. Yáñez Oropeza
and Carlos A. Coello Coello**

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
México D.F. 07300, MÉXICO
saul.zapotecas@gmail.com
eyanez@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

Abstract. In spite of the success of evolutionary algorithms for dealing with multi-objective optimization problems (the so-called multi-objective evolutionary algorithms (MOEAs)), their main drawback is the fine-tuning of their parameters, which is normally done in an empirical way (using a trial-and-error process for each problem at hand), and usually has a significant impact on their performance. In this paper, we present a self-adaptation methodology that can be incorporated into any MOEA, in order to allow an automatic fine-tuning of parameters, without any human intervention. In order to validate the proposed mechanism, we incorporate it into the NSGA-II, which is a well-known elitist MOEA and we analyze the performance of the resulting approach. The results reported here indicate that the proposed approach is a viable alternative to self-adapt the parameters of a MOEA.

1 Introduction

The design of mechanisms that allow to automate the fine-tuning of the parameters of an evolutionary algorithm (EA) has been subject of a considerable amount of research throughout the years [1, 2]. When dealing with optimization problems having several (often conflicting) objectives (the so-called multi-objective optimization problems), the fine-tuning of parameters gets even more complicated, since we aim to converge to a set of solutions (the so-called Pareto optimal set). Because of such complexity, the design of online and self-adaptation mechanisms have been scarce within the multi-objective evolutionary algorithms (MOEAs) literature (see for example [3–5]).

The main goal of this work is to define a multi-objective evolutionary algorithm that does not require any user-defined parameters. In order to achieve

* The first author acknowledges support from CINVESTAV-IPN and CONACyT to pursue graduate studies at CINVESTAV-IPN.
** The third author acknowledges support from CONACyT project number 103570.

such goal, we define different techniques to self-adapt the main parameters of a state-of-the-art MOEA (the NSGA-II [6]). The resulting approach is then validated using 12 test problems taken from the specialized literature. Results are compared with respect to those obtained with the original NSGA-II. As will be seen, the obtained results are very competitive and indicate that the proposed approach can be a viable alternative to automate the fine-tuning of parameters of a MOEA.

The remainder of this paper is organized as follows. In Section 2, we present the previous related work reported in the specialized literature. In Section 3, we describe in detail our proposed self-adaptation approach. In Section 4, we validate our proposed approach using standard test problems and performance measures reported in the specialized literature. Finally, in Section 5 we present our conclusions and provide some possible paths for future research.

2 Previous Related Work

The interest in reducing the number of parameters of a MOEA, has been studied by relatively few researchers. Apparently, the first attempt to self-adapt the parameters of a MOEA was the one reported by Kursawe [7]. His proposal was to provide individuals in the population of a MOEA with a set of step lengths for each objective function. The aim of Kursawe's work, however, was to be able to deal with dynamic environments rather than automating the fine-tuning of parameters of a MOEA.

Other authors have only focused on the self-adaptation of a single operator. For example, Büche et al. [8] proposed to use Kohonen's self-organizing maps to adapt the step length of a MOEA's mutation operator.

Tan et al. [9] proposed the incrementing multi-objective evolutionary algorithm (IMOEA) with adopts an adaptive population size whose value is computed based on the online discovered trade-off surface and the desired population distribution density. IMOEA relies on a convergence metric that is based on Pareto dominance and a performance measure called "progress ratio", which was proposed by Van Veldhuizen [10]. Additionally, IMOEA also incorporates dynamic niching (i.e., the user does not need to define a niche radius for performing fitness sharing).

Kumar and Rockett [11] proposed the Pareto converging genetic algorithm (PCGA). This MOEA uses a systematic approach based on Pareto rank histograms for assessing convergence towards the Pareto front.

Abbass [3] proposed the self-adaptive Pareto differential evolution (SPDE) which extends a MOEA called Pareto differential evolution (PDE) [12] with self-adaptive crossover and mutation operators. In SPDE, both the crossover and the mutation rate are treated as additional decision variables which are added to the chromosomic string and are affected by the evolutionary process.

Zhu and Leung [13] proposed a parallel multi-objective genetic algorithm which is implemented in an island model and has an asynchronous self-adjustable mechanism. This mechanism uses certain information about the current status

of each island and uses it to focus the search effort towards non-overlapping regions of the search space.

Toscano and Coello [4] proposed the micro genetic algorithm 2 (μGA^2) which is a parameterless version of the micro genetic algorithm (μGA) for multi-objective optimization previously introduced by the same authors [14]. The new approach adopts several self-adaptation mechanisms to select the type of encoding (binary or real-numbers), and the type of crossover operator (from several available). For this sake, it executes several μGAs in parallel and performs a comparison of their results. The μGA^2 also incorporates a mechanism based on a performance measure in order to decide when to stop iterating.

Martí et al. [15] proposed a mechanism that gathers information about the solutions obtained so far. This information is accumulated and updated using a discrete Kalman filter and is used to decide when to stop a MOEA.

Zielinski and Laur [16] proposed a mechanism for self-adapting three important parameters in a multi-objective particle swarm optimization: inertia, the cognitive component and the social component. The proposed mechanism is based on a design of experiments technique called evolutionary operation (EVOP). The authors adopt analysis of variance in a two-level factorial design [17] (i.e., two values are considered for each parameter being self-adapted) to determine the effect of each combination of parameters. The information obtained from the analysis of variance allows to determine how should the parameters be modified. The approach defines a measure of "success" based on Pareto dominance, which is used to guide the search.

Trautmann et al. [5] proposed a new convergence criteria for MOEAs. This mechanism consists of analyzing the performace of a MOEA through its iterative process with respect to three well-known performance measures: generational distance [10], hypervolume [18] and spread [6]. In this way, if there is not a significant variance of these performance measures, it is possible to conclude that the MOEA has converged to the real Pareto front and the evolutionary process is consequently stopped.

None of these previous approaches, however, constitutes a full proposal of a self-adaptation framework for MOEAs, which is precisely what we introduce here, with certain specific mechanisms specifically tailored for the NSGA-II.

3 Our Proposed Approach

Our approach consists of two phases. In the first of them, an analysis of variance (ANOVA) [19] of a MOEA, using a certain set of test problems and performance measures is undertaken. This analysis is meant to provide us with the set of parameters to which the MOEA under study is more sensitive. In our case, this analysis was done using the NSGA-II, but any other state-of-the-art MOEA could be adopted instead (e.g., SPEA2 [18]).

In the second phase, we introduce some specific self-adaptation techniques that are used to automatically tune the values of the parameters identified during the first phase.

Next, we will provide a summary of the results obtained from our ANOVA and will also describe the self-adaptation mechanisms that we propose to use.

3.1 Phase 1: Sensitivity Analysis

As indicated before, in order to define the parameters to which the NSGA-II is most sensitive, we performed an analysis of variance. For this analysis, we adopted five problems taken from the Zitzler-Deb-Thiele (ZDT) [20] and from the Deb-Thiele-Laumanns-Zitzler (DTLZ) test suites [21]. The problems were selected in such a way that different features were covered (e.g., non-convexity, disconnected Pareto fronts, etc.) using two and three objectives. The problems chosen for the study are presented next.

- **ZDT3:** The true Pareto front of this problem is disconnected (in two dimensions), consisting of several noncontiguous convex parts.
- **ZDT4:** This problem contains 21^9 false Pareto fronts and, therefore, tests the ability of a MOEA to deal with multifrontality.
- **DTLZ5:** The true Pareto front of this problem is a curve formed by a set of well-distributed solutions.
- **DTLZ6:** The true Pareto front of this problem is unimodal, biased, with a many-to-one mapping and is hard to converge to it.
- **DTLZ7:** The true Pareto front of this problem is disconnected (in three dimensions).

Table 1. Analyzed Parameters

Parameter	**Values**
Population size	100, 200 and 500
Number of generations	100, 200 and 500
Crossover rate	0.5, 0.7 and 1.0
Mutation rate	0.001, 0.1 and 0.3
Encoding	Real and binary
Crossover type	Two-point and uniform crossover for binary encoding SBX and uniform crossover for real numbers encoding
Mutation type	Uniform mutation for binary encoding Parameter-based and boundary mutation for real numbers encoding

In Table 1, we show the parameters and the values that we used for the ANOVA. For each test problem, we performed 20 independent runs using each all the possible combinations of parameters from those indicated in Table 1.

For evaluating the performance of each set of parameters, we used two performance measures: inverted generational distance ($\mathcal{IGD}$) [10] and the multiplicative unary ϵ-indicator ($\mathcal{I}_\epsilon$) [22] (using as reference to the true Pareto front).

The analysis of results led us to conclude that both the crossover rate and the crossover type (for binary encoding) could take a fixed value, since no variation of these parameters had significant effect on the performance of the NSGA-II. Thus, we decided to adopt a crossover rate $P_c = 0.7$ and two-points crossover for binary encoding. Our study indicated that these values produced the best overall performance for the NSGA-II.

3.2 Self-Adaptation of Parameters

The analysis indicated that the variation of the other parameters of the NSGA-II had a greater impact on performance and, therefore, we incorporated them into our proposed self-adaptation scheme. In Figure 1, we show the general scheme of our self-adaptive MOEA and the corresponding details are presented next.

```
1.  t=0
2.  Initialize the population;
3.  Encode the individuals;
4.  Evaluate the population;
5.  Rank the Population;
6.  while (there are no improvements according to the hypervolume) do
7.      Select the parents // using the same encoding between the individuals;
8.      Perform crossover;
9.      Encode the offspring population;
10.     Evaluate the offspring population;
11.     Join the parent and offspring population;
12.     Perform the elitism procedure;
13.     Performed the Inheritance-Fertilization procedure;
14.     if (t ≥ 100) then
15.         Perform a hypervolume analysis
16.         Add/remove individuals
17.     end if
18.     t = t + 1;
19. end while
```

Fig. 1. Our proposed self-adaptation techniques coupled to the NSGA-II.

Initially, a population of 100 individual is randomly generated. For each individual of the population, the type of encoding to be adopted (real or binary) is randomly assigned. The mutation rate and the individual's chromosome are also randomly initialized using the corresponding encoding. For the individuals with real numbers encoding, it is necessary to define, in a random way, the type of mutation and crossover to be used. This is unnecessary when using binary encoding, as was indicated before (see Section 3.1), since fixed values and operators are adopted in that case.

In this work, we assume that all the test problems use real numbers for their decision variables. When using binary encoding, a decoding is evidently needed to transform the binary numbers of each chromosome into real numbers (an accuracy of eight decimal places is adopted in that case). After doing this, the ranking mechanism of the original NSGA-II is applied.

The tournament selection adopted in our case is different from that of the original NSGA-II, because parents are only selected from individuals that have the same encoding. This way, appropriate crossover and mutation operators are applied to individuals having the same encoding. The specific type of crossover and mutation to be applied are chosen from those available (see Sections 3.4 and 3.5) for each type of encoding. The details about the use of these genetic operators are provided in Sections 3.4 and 3.5, respectively.

Once the offspring population is obtained, both the parents and the offspring populations are merged with the purpose of selecting from them to the best individuals for the next generation. For this task, the crowding comparison operator of the original NSGA-II is adopted to generate a total ordering of the individuals, so that the best half is selected [6].

Our approach introduces an additional step called the inheritance-fertilization procedure. This is a mechanism that we propose for diversifying the population. This procedure is applied at each generation and its details are discussed in Section 3.6.

Finally, the stopping criteria is defined using the hypervolume performance measure [23]. Specifically, what we do is to check if there is a change in the hypervolume value of the individuals in the population. If no significant change is detected after several interations, then the MOEA is stopped. The details of this mechanism are discussed in Section 3.7.

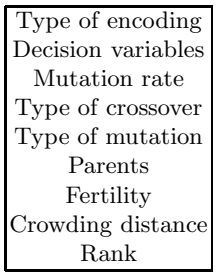

Fig. 2. Definition of each individual in our proposed approach.

3.3 The individual

In evolutionary algorithms, the individual is commonly represented by a single chromosomic string (i.e., haploids are normally adopted). However, in our proposed approach, we adopt diploids, since we simultaneously encode the individual in binary and real-numbers representation. This is a pragmatic solution to deal with the encoding of each individual, since in our approach, the type of representation could change during the self-adaptation process.

In our case, an individual includes the following elements: the type of encoding (real numbers or binary), the decision variables of the problem, the mutation rate, the type of crossover, the type of mutation, the type of encoding, the parents and the fertility. Additionally, each individual also has the parameters from the original NSGA-II (the rank, which relates to Pareto dominance and the crowding distance value, which relates to diversity). Figure 2 shows the parameters contained in each individual. Since each individual has two possible representations (real numbers or binary), the decision variables and the rates of

the operators will be encoded and initialized using the corresponding representation. The type of crossover indicates the crossover operator that was used to generate that individual. This operator will also be used to decide which type of crossover will be used in case the individual is selected for breeding. Similarly, the type of mutation refers to the specific mutation operator that will be applied on the individual that contains it. Since the type of crossover and the type of mutation are already fixed for binary encoding, these parameters are not included in an individual. The parameters called *parents* and *fertility* are used in the inheritance-fertilization procedure which will be explained below.

3.4 Crossover operator

Since the tournament takes place only among individuals with the same type of encoding, the parents selected for breeding will also have the same encoding among themselves.

As indicated before, when using binary encoding, two-points crossover is always adopted in the traditional way [24]. When using real numbers encoding, we have five types of crossover operators available: (1) Simulated Binary Crossover (SBX) [25], (2) simple crossover [26], (3) uniform crossover [27], (4) intermediate crossover [28] and (5) two-points crossover [24].

In order to choose the type of operator to be applied to each pair of individuals, we use a $flip(p)$ function, which simulates the tossing of a coin and randomly returns a boolean value (true or false) with a certain pre-defined probability (p). The parameter p defines the probability of this function to return true. In our case, we defined $p = 0.9$. If this function returns true, we use the crossover type of the best parent (in terms of its rank). If this function returns false, then we apply the same function again, but using a probability $p = 0.5$. If this second function returns true again, we choose the type of crossover that was adopted to generate the best parent (in terms of its rank). Otherwise, we choose the type of crossover of the other parent. If both parents have the same rank, we choose the type of crossover in a random manner between them.

The mutation rate is encoded (in binary or as a real number) in the chromosomic string. Thus, the mutation rate can be affected by the crossover operator. When using real numbers encoding, the crossover operator is applied using a $flip(p)$ function with $p = 0.5$ for two-points, simple and uniform crossover. For intermediate recombination, we adopt $k = 0$. SBX is applied in the form suggested in [6].

Finally, each child generated by the crossover operator inherits the type of encoding and the type of crossover from its best parent (in terms of rank).

3.5 Mutation operator

For binary encoding, the mutation rate is defined within the interval $(0.001, 0.3)$ and is also encoded in the chromosome. Mutation is applied to the decision variables first, and then to the mutation rate as well. Then, the type of encoding is mutated (or not) based on the outcome of a $flip(0.5)$ function. If the type of

encoding changes (binary $\mapsto$ real) then the decision variables and the mutation rate are represented using real numbers.

Since there are different types of crossover and mutation operators available (for real-numbers representation), if an individual changes its encoding from binary to real numbers, then we need to define new values for the type of operators to be adopted. In our case, we define such values in a random way.

Since the range of the mutation rate is different for each encoding (in real numbers encoding, the mutation rate is in the range $(1/L, 0, 5)$, where L is the number of decision variables), we use a linear mapping to transform the mutation rate from one encoding to the other (i.e., $(0.001, 0.3) \mapsto (1/L, 0, 5)$). The mutation rate defined for real numbers encoding ensures that at least one decision variable will get mutated. It also guarantees that more than 50% of the decision variables will be mutated. The linear mapping that we adopt for the mutation rate in this case is defined as follows:

$$p_m^{real} = \left(\frac{0.5 - \frac{1}{L}}{0.3 - 0.001} \times (p_m - 0.001) \right) + \frac{1}{L} \tag{1}$$

For perturbing the type of crossover and mutation to be adopted, we perform a similar mapping. Here, we use a mapping defined by $(0.001, 0.3) \mapsto (1/8, 0.8)$. Thus, the mutation rate in this case is defined by:

$$p_m^{parameters} = \left(\frac{0.8 - \frac{1}{8}}{0.3 - 0.001} \times (p_m - 0.001) \right) + \frac{1}{8} \tag{2}$$

where p_m represents the mutation rate of the individual.

If the type of crossover or mutation has to be changed then the new types are defined in a random way. Finally, as in the binary case, an individual can change its type of encoding (real$\mapsto$binary). In this case, the decision variables and mutation rate would be transformed to their equivalent binary representation. The type of crossover and mutation are removed because they are both fixed for binary encoding.

3.6 Inheritance-Fertilization Operator

When the NSGA-II selects the population for the next generation, the parent and offspring populations are merged. The inheritance-fertilization operator identifies the parents and offspring that have been selected to constitute the following generation. Thus, each child has information about who were his parents and viceversa. Once the parents and children have been identified, the mechanism detects parents which have not produce children that had been selected during a certain number of generations (in this work we used a gap of five generations). If this is the case, the parameters of this individual are perturbed.

The mutation of the parameters of each parent is performed according to its encoding, as was indicated before. The same applies to the perturbation of the type of encoding, crossover and mutation (see Section 3.5).

When using the inheritance-fertilization operator, the decision variables are not perturbed. However, the type of encoding can be modified. The aim of this operator is to maintain diversity in the population. The underlying assumption of this operator is that if the children generated by the parents selected in previous generations are not good (in terms of their ranking), is because the genetic operators are not working properly. Thus, they must be modified so that better results can be achieved and that is precisely what the operator does.

3.7 Stopping Criterion and a Varying Population Size

We adopted the hypervolume performance measure [29] to detect when the algorithm has converged (i.e., when no further improvement is found) and we use that as the stopping criterion of our approach. The hypervolume (also known as the S metric or the Lebesgue Measure) of a set of solutions measures the size of the portion of objective space that is dominated by those solutions collectively.

The number of generations and the size of the population play an important role in MOEAs. However, it is well-known that it is unnecessary to have an extremely large population to perform a better search. It is possible to use a modest population size, as long as we have a good mechanism to maintain diversity and we run the MOEA during a sufficiently large number of generations [3]. These aspects were taken into account for the design of the strategy that is explained next.

Intially, the population size is set to 100 individuals. After 100 generations, we start applying hypervolume at each generation. If after 30 generations, there is no improvement in the hypervolume, then 100 new individuals (randomly generated) are added to the population. If some improvement is detected, then the counter is reset so that we start counting again 30 more generations, and repeat this process until no improvement is detected.

Once the 100 new individuals have been added, we continue with the second phase at which we run our MOEA during 20 more generations and check again for improvements in the hypervolume. If no improvement is detected, then we generate 300 new (random) individuals. On the contrary, if some improvement is detected, then, we reduce the population size from 200 to 100. We keep the best half, using Pareto ranking and the crowding comparison operator of the original NSGA-II. Then, the counter is reset again and the search continues, aiming to find 30 consecutive generations without any improvement, before adding 100 new individuals.

Once the population reaches 500 individuals, we enter the third stage. At that point, we run the MOEA for 40 generations. If no improvement in the hypervolume is detected, we consider that the algorithm has converged and we stop the execution of our MOEA. However, if there is an improvement in the hypervolume during these 40 generations, we remove 300 individuals from the population using the same procedure indicated before. In this case, the counter is reset to 30 generations, so that we try to obtain 60 consecutive generations without any improvements in the hypervolume before stopping the execution of the MOEA. Since the hypervolume requires a reference vector, we use the

same in all cases, to avoid any errors in its calculation. The complete process is graphically depicted in Figure 3.

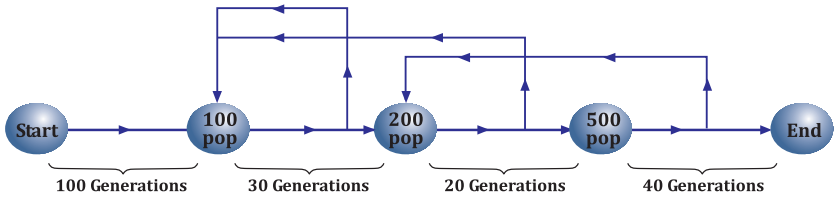

Fig. 3. Graphical illustration of the stopping criterion and the adaptive population size mechanisms

4 Experimental results

In order to validate the performance of our proposed approach, we compared its results with respect to those obtained by the original NSGA-II using twelve problems taken form ZDT [20] (ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6) and the DTLZ [21] (DTLZ1, DTLZ2, DTLZ3,DTLZ4, DTLZ5, DTLZ6 and DTLZ7) test suites. We adopted three performance measures to assess our results: Inverted Generational Distance ($\mathcal{IGD}$) [10], the multiplicative unary ϵ-indicator ($\mathcal{I}_\epsilon$) [22] and Spread ($\mathcal{S}$) [6].

4.1 Experimental Setup

We performed 20 independent runs per problem per approach. Since our approach does not require any extra parameters, we define only the parameters for the NSGA-II: population size $N = 500$, crossover probability $P_c = 0.7$, mutation probability $P_{mr} = 0.1$. For the genetic operators (SBX and PBM) we used a crossover index $\eta_c = 1$ and a mutation index $\eta_m = 50$.

Since the stopping criteria of our approach does not have a fixed number of generations, in order to define the number of generations to be performed by the original NSGA-II we experimented with two different criteria:

1. **Average number of evaluations:** In this case, we used the average (over all runs) number of objective function evaluations performed by our self-adaptive approach to set the number of generations[1] of the NSGA-II.
2. **Average number of generations:** In this case, we used instead the average number of generations (over all runs) performed.

In Tables 2 and 3, we show the average of number of objective function evaluations and the number of generations in which our proposed approach was stopped.

Table 2. Number of generations for the ZDT test suite.

Problem	$ZDT1$	$ZDT2$	$ZDT3$	$ZDT4$	$ZDT6$
Evaluations Average	238	241	353	611	858
Generations Average	630	643	960	1676	2016

Table 3. Number of generations for DTLZ test suite.

Problem	$DTLZ1$	$DTLZ2$	$DTLZ3$	$DTLZ4$	$DTLZ5$	$DTLZ6$	$DTLZ7$
Evaluations Average	138	115	170	106	90	99	117
Generations Average	379	324	497	289	248	250	306

Thus, in order to obtain the number of generations during which the original NSGA-II would run, we used either the average of number fitness function evaluations (this variant was called NSGA-II$_{eval}$) or the average of number of generations (this variant was called NSGA-II$_{gen}$).

4.2 Discussion of Results

The results obtained by NSGA-II$_{eval}$, NSGA-II$_{gen}$ and NSGA-II$_{self_adap}$ (our self-adaptive approach) are summarized in Tables 4 to 9. Each table displays both, the mean and the standard deviation (σ) of each performance measure, for each test problem. The best results are shown in **boldface**.

$\mathcal{IGD}$ performance measure In Tables 4 and 7, we can see that our proposed approach outperforms the NSGA-II in most of the test problems adopted with respect to $\mathcal{IGD}$. However, in five of the twelve adopted problems (ZDT4, DTLZ1, DTLZ2, DTLZ4 and DTLZ5), our algorithm was outperformed by the original NSGA-II. Nonetheless, according the Wilcoxon rank-sum test [30], the original NSGA-II is not significantly better than our proposed approach (NSGA-II$_{self_adap}$) using a significance level of 0.05.

$\mathcal{I}_\epsilon$ performance measure In Tables 5 and 8), we can see that our proposed approach outperforms the NSGA-II in seven of the twelve test problems adopted with respect to $\mathcal{I}_\epsilon$. Although for ZDT1, ZDT3, ZDT4, DTLZ1 and DTLZ5 the original NSGA-II is better, our algorithm is not significantly worse.

$\mathcal{S}$ performance measure Tables 6 and 9 show that our proposed approach was outperformed by the NSGA-II in most of the test problems adopted (seven out of twelve) with respect to spread. However, we do not consider this to be

[1] Knowing the total number of objective function of evaluations and the population size, it is straightforward to obtain the total number of generations.

a major drawback, since our self-adaptation mechanisms were focused on convergence rather than on spreading solutions along the Pareto front and in terms of convergence, we found better results in most cases. We believe that the use of additional individuals in the population, in order to maintain diversity is the main reason why our proposed approach does not reach the same quality of results with respect to spread as the original NSGA-II.

Table 4. Results of $\mathcal{IGD}$ for the ZDT test suite.

	$NSGA-II_{eval}$		$NSGA-II_{gen}$		$NSGA-II_{self\_adap}$	
	average	(σ)	*average*	(σ)	*average*	(σ)
$ZDT1$	0.000058	(0.000002)	0.000058	(0.000002)	**0.000056**	(0.000001)
$ZDT2$	0.000059	(0.000002)	0.000059	(0.000002)	**0.000058**	(0.000003)
$ZDT3$	0.000132	(0.000007)	0.000132	(0.000006)	**0.000127**	(0.000007)
$ZDT4$	**0.000083**	(0.000004)	0.000084	(0.000004)	0.000093	(0.000004)
$ZDT6$	0.002230	(0.000191)	0.002230	(0.000191)	**0.000018**	(0.000002)

Table 5. Results of $\mathcal{I}_\epsilon$ for the ZDT test suite.

	$NSGA-II_{eval}$		$NSGA-II_{gen}$		$NSGA-II_{self\_adp}$	
	average	(σ)	*average*	(σ)	*average*	(σ)
$ZDT1$	1.002235	(0.000322)	**1.002152**	(0.000225)	1.002214	(0.000373)
$ZDT2$	1.001938	(0.000440)	1.001967	(0.000317)	**1.001792**	(0.000346)
$ZDT3$	1.001863	(0.000393)	**1.001830**	(0.000226)	1.002024	(0.000609)
$ZDT4$	1.001709	(0.000316)	**1.001674**	(0.000324)	1.001916	(0.000420)
$ZDT6$	1.140147	(0.011069)	1.140147	(0.011069)	**1.001409**	(0.000171)

Table 6. Results of $\mathcal{S}$ for the ZDT test suite.

	$NSGA-II_{eval}$		$NSGA-II_{gen}$		$NSGA-II_{self\_adp}$	
	average	(σ)	*average*	(σ)	*average*	(σ)
$ZDT1$	0.65008	(0.040356)	0.663761	(0.034663)	**0.542132**	(0.029852)
$ZDT2$	0.666593	(0.037943)	0.662773	(0.035102)	**0.539112**	(**0.025966**)
$ZDT3$	0.746819	(0.036011)	0.732614	(0.030007)	**0.612249**	(**0.021919**)
$ZDT4$	**0.556536**	(0.019120)	0.578010	(0.041313)	0.801464	(0.044060)
$ZDT6$	**0.788426**	(0.024690)	0.796272	(0.024848)	0.856744	(0.046337)

5 Conclusions and Future Research

In this paper, we have presented self-adaptation mechanisms for a MOEA, aiming to have an approach that does not require any manual fine-tuning of its parameters. It is important to emphasize, however, that the parameters are not removed. Instead, we use mechanisms that automatically define them using information gathered during the search, so that no user intervention is required.

Table 7. Results of $\mathcal{IGD}$ for the DTLZ test suite.

	$NSGA-II_{eval}$		$NSGA-II_{gen}$		$NSGA-II_{self\_adp}$	
	$average$	(σ)	$average$	(σ)	$average$	(σ)
$DTLZ1$	0.005666	(0.006462)	**0.001938**	(0.003133)	0.002161	(0.003941)
$DTLZ2$	0.000172	(0.000004)	**0.000168**	(0.000004)	0.000174	(0.000005)
$DTLZ3$	0.250904	(0.098975)	0.055570	(0.032845)	**0.012097**	(0.013000)
$DTLZ4$	0.000547	(0.000009)	**0.000542**	(0.000008)	0.000558	(0.000030)
$DTLZ5$	**0.000015**	(0.000002)	0.000023	(0.000002)	0.000037	(0.000013)
$DTLZ6$	0.015200	(0.002084)	0.002186	(0.000930)	**0.000094**	(0.000034)
$DTLZ7$	0.000817	(0.000125)	0.000579	(0.000062)	**0.000389**	(0.000016)

Table 8. Results of $\mathcal{I}_\epsilon$ for the DTLZ test suite.

	$NSGA-II_{eval}$		$NSGA-II_{gen}$		$NSGA-II_{self\_adp}$	
	$average$	(σ)	$average$	(σ)	$average$	(σ)
$DTLZ1$	1.1904444	(0.181980)	**1.066070**	(0.082606)	1.079437	(0.123285)
$DTLZ2$	1.049461	(0.006330)	1.047081	(0.006184)	**1.046300**	(0.005485)
$DTLZ3$	10.176407	(3.861924)	2.667639	(0.900274)	**1.396945**	(0.380866)
$DTLZ4$	1.042668	(0.004336)	1.040609	(0.004801)	**1.039888**	(0.004084)
$DTLZ5$	**1.001418**	(0.000338)	1.001947	(0.000312)	1.002976	(0.001318)
$DTLZ6$	1.732627	(0.082842)	1.103957	(0.040262)	**1.008537**	(0.004132)
$DTLZ7$	1.081822	(0.021139)	1.057326	(0.011657)	**1.026261**	(0.003132)

Table 9. Results of S for the DTLZ test suite.

	$NSGA-II_{eval}$		$NSGA-II_{gen}$		$NSGA-II_{self\_adp}$	
	$average$	(σ)	$average$	(σ)	$average$	(σ)
$DTLZ1$	0.974938	(0.190992)	0.576810	(0.087172)	**0.546923**	(0.160001)
$DTLZ2$	0.429743	(0.022495)	**0.429741**	(0.017543)	0.490002	(0.034583)
$DTLZ3$	1.188861	(0.085993)	0.955311	(0.117838)	**0.931206**	(0.336853)
$DTLZ4$	**0.412915**	(0.019106)	0.424239	(0.018287)	0.453508	(0.036211)
$DTLZ5$	**0.523047**	(0.092002)	0.717033	(0.018537)	0.746702	(0.018931)
$DTLZ6$	0.972776	(0.067786)	**0.782302**	(0.025047)	0.804278	(0.013795)
$DTLZ7$	0.478671	(0.038158)	**0.440922**	(0.030344)	0.531645	(0.021466)

Our results indicate that our proposed approach is able to outperform the original NSGA-II in several test problems, with respect to convergence, with the advantage of not requiring any empirical fine-tuning of parameters. Thus, we believe that our proposal can be a viable alternative for end-users who want to apply an out-of-the-box NSGA-II in a certain application, without having much knowledge about evolutionary computation techniques.

Since self-adaptation mechanisms are, in general, hard to define (particularly in the context of MOEAs), in order to simplify things, we tailored most of the mechanisms described here to the specific selection scheme and density estimator adopted by the NSGA-II. However, as part of our future work, we are interested in defining more general versions of some of the self-adaptation mechanisms introduced here, so that they are applicable to more than one MOEA. We are also interested in adding self-adaptation mechanisms that improve the spread of solutions produced by our approach. In that regard, the use of archiving techniques may be useful.

References

1. Ágoston Endre Eiben, Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation **3**(2) (1999) 124–141
2. Lobo, F.G., Lima, C.F., Michalewicz, Z., eds.: Parameter Setting in Evolutionary Algorithms. Springer, Berlin, Germany (2007) ISBN 978-3-540-69431-1.
3. Abbass, H.A.: The Self-Adaptive Pareto Differential Evolution Algorithm. In: Congress on Evolutionary Computation (CEC'2002). Volume 1., Piscataway, New Jersey, IEEE Service Center (2002) 831–836
4. Toscano Pulido, G., Coello Coello, C.A.: The Micro Genetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimization. In Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L., eds.: Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, Faro, Portugal, Springer. Lecture Notes in Computer Science. Volume 2632 (2003) 252–266
5. Trautmann, H., Ligges, U., Mehnen, J., Preuss, M.: A Convergence Criterion for Multiobjective Evolutionary Algorithms Based on Systematic Statistical Testing. In Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N., eds.: Parallel Problem Solving from Nature–PPSN X. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Germany (2008) 825–836
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6**(2) (2002) 182–197
7. Kursawe, F.: A Variant of Evolution Strategies for Vector Optimization. In Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature. 1st Workshop, PPSN I. Volume 496 of Lecture Notes in Computer Science Vol. 496., Berlin, Germany, Springer-Verlag (1991) 193–197
8. Büche, D., Guidati, G., Stoll, P., Kourmoursakos, P.: Self-Organizing Maps for Pareto Optimization of Airfoils. In Merelo Guervós, J.J., Adamidis, P., Beyer, H.G., nas, J.L.F.V., Schwefel, H.P., eds.: Parallel Problem Solving from Nature—PPSN VII, Granada, Spain, Springer-Verlag. Lecture Notes in Computer Science No. 2439 (2002) 122–131
9. Tan, K., Lee, T., Khor, E.: Evolutionary Algorithms with Dynamic Population Size and Local Exploration for Multiobjective Optimization. IEEE Transactions on Evolutionary Computation **5**(6) (2001) 565–588
10. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1999)
11. Kumar, R., Rockett, P.: Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimizations by Steady-State Evolution: A Pareto Converging Genetic Algorithm. Evolutionary Computation **10**(3) (2002) 283–314
12. Abbass, H.A., Sarker, R., Newton, C.: PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems. In: Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001). Volume 2., Piscataway, New Jersey, IEEE Service Center (2001) 971–978
13. Zhu, Z.Y., Leung, K.S.: Asynchronous Self-Adjustable Island Genetic Algorithm for Multi-Objective Optimization Problems. In: Congress on Evolutionary Computation (CEC'2002). Volume 1., Piscataway, New Jersey, IEEE Service Center (2002) 837–842

14. Coello Coello, C.A., Toscano Pulido, G.: A Micro-Genetic Algorithm for Multiobjective Optimization. In Zitzler, E., Deb, K., Thiele, L., Coello, C.A.C., Corne, D., eds.: First International Conference on Evolutionary Multi-Criterion Optimization. Springer-Verlag. Lecture Notes in Computer Science No. 1993 (2001) 126–140
15. Martí, L., García, J., Berlanga, A., Molina, J.M.: A Cumulative Evidential Stopping Criterion for Multiobjective Optimization Evolutionary Algorithms. In Thierens, D., ed.: 2007 Genetic and Evolutionary Computation Conference (GECCO'2007). Volume 1., London, UK, ACM Press (2007) 911
16. Zielinski, K., Laur, R.: Adaptive Parameter Setting for a Multi-Objective Particle Swarm Optimization Algorithm. In: 2007 IEEE Congress on Evolutionary Computation (CEC'2007), Singapore, IEEE Press (2007) 3019–3026
17. Myers, R.H., Montgomery, D.C.: Response Surface Methodology-Process and Product Optimization Using Designed Experiments. John Wiley and Sons (2002)
18. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation **3**(4) (1999) 257–271
19. Lindman, H.R.: Analysis of variance in complex experimental designs. SIAM Rev. **18**(1) (1976) 134–137
20. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation **8**(2) (2000) 173–195
21. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In Abraham, A., Jain, L., Goldberg, R., eds.: Evolutionary Multiobjective Optimization. Theoretical Advances and Applications. Springer, USA (2005) 105–145
22. 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**(2) (2003) 117–132
23. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (1999)
24. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading, Massachusetts (1989)
25. Deb, K., Agrawal, R.B.: Simulated Binary Crossover for Continuous Search Space. Complex Systems **9** (1995) 115–148
26. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Third edn. Springer-Verlag (1996)
27. Syswerda, G.: Uniform Crossover in Genetic Algorithms. In Schaffer, J.D., ed.: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, Morgan Kaufmann Publishers (1989) 2–9
28. Schwefel, H.P.: Evolution and Optimum Seeking. John Wiley & Sons, New York (1995)
29. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In Eiben, A.E., ed.: Parallel Problem Solving from Nature V, Amsterdam, Springer-Verlag (1998) 292–301
30. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6) (1945) 80–83