On the Effect of Applying a Steady-State Selection Scheme in the Multi-Objective Genetic Algorithm NSGA-II

Antonio J. Nebro and Juan J. Durillo

Abstract. Genetic Algorithms (GAs) are among the most popular techniques to solve multi-objective optimization problems, with NSGA-II being the most well-known algorithm in the field. Although most of multi-objective GAs (MOGAs) use a generational scheme, in the last few years some proposals using a steady-state scheme have been developed. However, studies about the influence of using those selection strategies in MOGAs are scarce. In this chapter we implement a steady-state version of NSGA-II, which is a generational MOGA, and we compare the two versions with a set of four state-of-the-art multi-objective metaheuristics (SPEA2, OMOPSO, AbYSS, and MOCell) attending to two criteria: the quality of the resulting approximation sets to the Pareto front and the convergence speed of the algorithms. The obtained results show that search capabilities of the steady-state version of NSGA-II significantly improves the original version, providing very competitive results in terms of the quality of the obtained Pareto front approximations and the convergence speed.

1 Introduction

Genetic Algorithms (GAs) have been widely applied for solving optimization problems in many areas. Since the appearance of the first multi-objective genetic algorithm (MOGA), the *Multiple Objective Optimization with Vector Evaluated Genetic Algorithm* (VEGA) [21], there has been a growing interest in these kinds of algorithms for problems with two or more objectives. GAs are very popular in multi-objective optimization in part because they can obtain a front of solutions in one single run. Thus, the most well-known algorithms in this field are GAs: NSGA-II [3] and SPEA2 [26]. GAs belong to a family of nature-inspired techniques, the

Antonio J. Nebro · Juan J. Durillo
Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071 Málaga, Spain
e-mail: `antonio@lcc.uma.es,durillo@lcc.uma.es`

evolutionary algorithms, which form part of a broader set of approximation techniques known as *metaheuristics* [11]. Other metaheuristic techniques include particle swarm optimization, ant colony optimization, scatter search, etc.

Based on their selection scheme, there exist two main models of GAs: generational and steady-state. In the generational model, the algorithm creates a population of individuals from an old population using the typical genetic operators (selection, crossover, and mutation); this new population becomes the population of the next generation. On the other hand, a steady-state GA creates typically only one new member which is tested for insertion into the population at each step of the algorithm.

In this chapter our purpose is, taking as starting point a steady-state version of NSGA-II, to study the search enhancements of that scheme over the generational approach of NSGA-II in the context of a comparison against four state-of-the-art multi-objective metaheuristics, namely, SPEA2 [26] (GA), AbYSS [18] (scatter search), MOCell [16] (cellular GA), and OMOPSO [19] (particle swarm optimization). For a broader comparison of these algorithms, we have evaluated them by using test functions from three different benchmarks (ZDT [25], DTLZ [6], and WFG [12]) and we have considered two different criteria. First, we have assessed the quality of the Pareto fronts obtained by those algorithms by applying the additive unary epsilon ($I^1_{\varepsilon}+$) [14], spread (Δ) [3], and hypervolume (HV) [24] quality indicators. Second, we have studied their convergence speed, i.e., the number of function evaluations required by the algorithms to converge towards the optimal Pareto front.

The remainder of this chapter is structured as follows. The next section provides background information about multi-objective optimization. In Section 3 we review previous works in the literature. Section 4 describes the NSGA-II algorithm and its steady-state version. The algorithms used in the comparative study are described in Section 5. The next two sections are devoted to the experimentation and analysis of the obtained results. Finally, Section 8 draws some conclusions and lines of future work.

2 Multi-Objective Optimization Background

In this section, we provide some background on multiobjective optimization. First, we define basic concepts such as Pareto optimality, Pareto dominance, Pareto optimal set, and Pareto front. In these definitions we assume, without loss of generality, the minimization of all the objectives.

A general multiobjective optimization problem (MOP) can be formally defined as follows:

Definition 1 (MOP). Find a vector $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]$ which satisfies the m inequality constraints $g_i(\mathbf{x}) \geq 0, i = 1,2,\ldots,m$, the p equality constraints $h_i(\mathbf{x}) = 0, i = 1,2,\ldots,p$, and minimizes the vector function $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T$, where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

The set of all the values satisfying the constraints defines the *feasible region* Ω and any point $\mathbf{x} \in \Omega$ is a *feasible solution*. As mentioned before, we seek for the *Pareto optima*. Its formal definition is provided next:

Definition 2 (Pareto Optimality). A point $\mathbf{x}^* \in \Omega$ is Pareto Optimal if for every $\mathbf{x} \in \Omega$ and $I = \{1,2,\ldots,k\}$ either $\forall_{i\in I}\,(f_i(\mathbf{x}) = f_i(\mathbf{x}^*))$ or there is at least one $i \in I$ such that $f_i(\mathbf{x}) > f_i(\mathbf{x}^*)$.

This definition states that $\mathbf{x}^*$ is Pareto optimal if no feasible vector $\mathbf{x}$ exists which would improve some criteria without causing a simultaneous worsening in at least one other criterion. Other important definitions associated with Pareto optimality are the following:

Definition 3 (Pareto Dominance). A vector $\mathbf{u} = (u_1,\ldots,u_k)$ is said to dominate $\mathbf{v}=(v_1,\ldots,v_k)$ (denoted by $\mathbf{u} \preccurlyeq \mathbf{v}$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e., $\forall i \in \{1,\ldots,k\},\ u_i \leq v_i \wedge \exists i \in \{1,\ldots,k\} : u_i < v_i$.

Definition 4 (Pareto Optimal Set). For a given MOP $\mathbf{f}(\mathbf{x})$, the Pareto optimal set is defined as $\mathscr{P}^* = \{\mathbf{x} \in \Omega | \neg\exists \mathbf{x}' \in \Omega, \mathbf{f}(\mathbf{x}') \preccurlyeq \mathbf{f}(\mathbf{x})\}$.

Definition 5 (Pareto Front). For a given MOP $\mathbf{f}(\mathbf{x})$ and its Pareto optimal set $\mathscr{P}^*$, the Pareto front is defined as $\mathscr{PF}^* = \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in \mathscr{P}^*\}$.

Pareto dominance relates two solutions and it can be used as a binary operator. Thus, the application of this operator to two solutions in the objective space returns either one solution that *dominates* another or that the solutions do not dominate each other (i.e., they are *non-dominated* solutions). The Pareto optimal set is composed of all those solutions which are non-dominated, and the Pareto front is the correspondence of the Pareto optimal set in the objective space.

Obtaining the Pareto front of a MOP is the main goal of multiobjective optimization. When stochastic techniques such as metaheuristics are applied, the goal is to obtain a finite set of solutions having two properties: *convergence* to the true Pareto front and homogeneous *diversity*. The first property ensures that we are dealing with optimal solutions, while the second one, which refers to obtaining a uniform-spaced set of solutions, indicates that we have carried out an adequate exploration of the search space, so we are not losing valuable information.

3 Related Work

In this section we analyze previous works in the literature which make use of a steady-state scheme in multi-objective GAs. Many MOGAs using such scheme have been proposed in the last few years; here, we focus on the most salient proposals.

One of the first steady-state multi-objective algorithms described in the literature was the *Pareto Converging Genetic Algorithm* (PCGA) [15]. PCGA used a $(\mu + 2)$ scheme and a novel mechanism based on histograms of ranks for assessing convergence to the Pareto front. It was found to produce diverse sampling of the Pareto front without niching and with significantly less computing effort than NSGA, the previous version of NSGA-II. Nevertheless, the algorithms were evaluated using only three test problems and no comparisons with PCGA using a generational scheme were reported.

The *Simple Evolutionary Algorithm for Multi-Objective Optimization* (SEAMO) was proposed in [23]. It was a simple steady-state approach following a $(\mu+1)$ scheme that used only one population and depended entirely on the replacement policy used: no rankings, subpopulations, niches or auxiliary approach were required. Due to the fact that a generational version of SEAMO may not make sense, it was only compared with NSGA-II and SPEA2 using as benchmark the multiple knapsack problem.

Deb et *al.* proposed in [4] a ε-Domination Based Steady State MOEA, which was also evaluated in [5]. This algorithm used a $(\mu+1)$ scheme and was composed of a population and an archive, which used a ε-Domination mechanism. In each generation, one parent from the population and one from the archive were selected to create new offsprings, which were then tested for insertion in both the population and the archive using different strategies. It was compared with several state-of-the-art MOEAS using both bi-objective and three-objective optimization problems. No comparisons with the same algorithm using a generational scheme were reported.

Two multi-objective steady-state algorithms were presented in [1]: the *Objective Exchanging Genetic Algorithm for Design Optimization* (OEGADO), and the *Objective Switching Genetic Algorithm for Design Optimization* (OSGADO). The former proposal consists of several steady-state single-objective optimization GAs which periodically exchange information about the objectives; the second algorithm is also composed of multiple single-objective optimization algorithms, but in this case these algorithms periodically switch the objective they optimize. Both algorithms were compared with NSGA-II using four benchmark academic problems and two engineering problems. In this work, neither OSGADO and OEGADO were evaluated using generational single-objective GAs.

Emmerich et *al.* presented in [10] the so-called *S metric selection EMOA* (SMS-EMOA), which is a hypervolume based steady-state GA. It has a $(\mu+1)$ scheme as well. The paper included a theoretical analysis in which the advantages of using a steady-state scheme in terms of the complexity of this kind of algorithms were proofed. The algorithm was evaluated using the ZDT benchmark, and it was compared with NSGA-II, SPEA2, and the above described ε-MOEA. No comparisons using a generational scheme were reported.

Srinivasan et *al.* proposed in [22] a new version of the NSGA-II algorithm. This algorithm uses a $(\mu+\lambda)$ scheme, like the original NSGA-II. The main difference was that once all the individuals have been generated, they are considered to update the population in a steady-state model. The new proposal was evaluated using nine benchmark problems and compared with the original NSGA-II algorithm.

Igel et *al.* studied in [13] the effect of two different steady-state schemes, a $(\mu+1)$ and a $(\mu_{<}+1)$, for the *Multi-objective covariance matrix adaption evolution strategy* (MO-CMA-ES). The latter steady-state scheme did not consider all the population for selecting the parents. These different approaches were compared with a

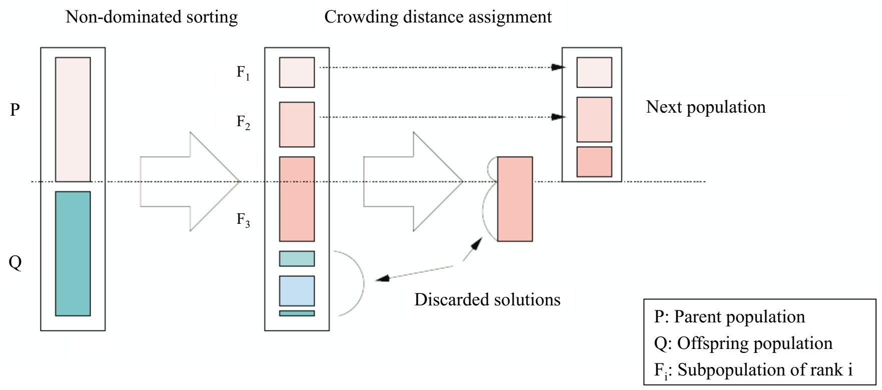


Fig. 1 The NSGA-II procedure, which follows a generational selection scheme

generational scheme, NSGA-II and SPEA2 using a benchmark composed of constrained and unconstrained test functions.

In [9] Durillo et *al.* proposed a steady-state version of the NSGA-II algorithm with a $(\mu + 1)$ scheme and they compared it to the original one. Results showed that by using such a scheme, the algorithm was able to outperform the original one in terms of convergence towards the optimal Pareto front and spread of the resulting fronts of solutions.

Summarizing this section, many of the works in the literature only present new steady-state algorithms and compare them against the state-of-the-art MOGAs; comparisons with the same algorithm using a generational scheme are scarce. Furthermore, many of these proposals are only evaluated using a benchmark composed of small number of problems, and they take into account only the quality of the final front obtained without paying attention to other issues such as the convergence speed of the algorithms.

4 Steady-State NSGA-II

In this section we present the steady-state version of NSGA-II. First, we describe the original (generational) algorithm, and then we go into details related to the steady-state proposal.

The NSGA-II algorithm was proposed by Deb *et al.* [3]. It is based on a ranking procedure, consisting of extracting the non-dominated solutions from a population and assigning them a rank of 1; these solutions are removed and the next group of non-dominated solutions have a rank of 2, and so on. NSGA-II is a generational MOGA, in which a current population is used to create an auxiliary one, the offspring population (see Fig. 1); after that, both populations are combined to obtaining the new current population. The procedure is as follows: the two populations are sorted according to their rank, and the best solutions are chosen to create the new population; in the case of selection between some individuals with the same rank,

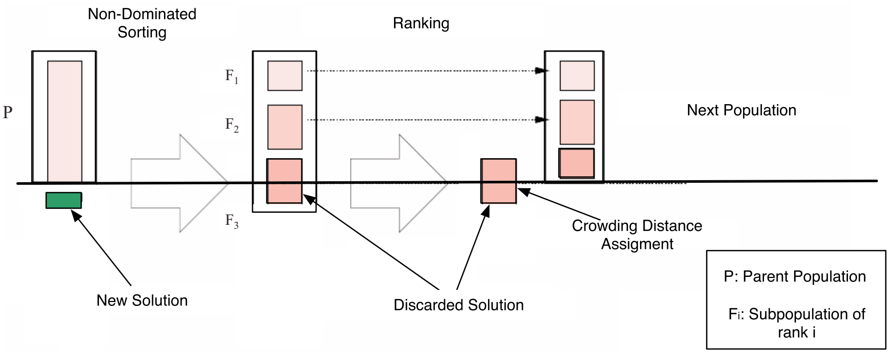


Fig. 2 The NSGA-II$_{ss}$ algorithm. Only one offspring is generated and tested to be inserted at each step.

a density estimation based on the crowding distance to the surrounding individuals of the same rank is used to get the most promising solutions. Typically, both the current and the auxiliary populations have the same size.

A steady-state version of NSGA-II can be easily implemented by using an offspring population of size 1. In this way, the newly generated individual is immediately incorporated into the evolutionary cycle. However, this also means that the ranking and crowding procedures have to be applied each time a new individual is created, so the time required by the algorithm increases notably. The procedure of this version is shown in Fig. 2. In the rest of this work we will refer to the steady-state version as NSGA-II$_{ss}$, and to the original one as NSGA-II$_{gen}$.

5 Description of the Evaluated Algorithms

In this section we briefly describe the four algorithms that we have considered for comparison with the two versions of NSGA-II. We have included SPEA2 because it is, along with NSGA-II, the most popular MOGAs. The other three techniques, OMOPSO, MOCell, and AbYSS are more recent algorithms, and they have been proven to be more effective than NSGA-II and SPEA2 in previous works [17][16][18][20].

The main features of these techniques are described next:

- SPEA2 was proposed by Zitler *et al.* in [26]. In this algorithm, each individual has a fitness value that is the sum of its strength raw fitness plus a density estimation. The algorithm applies the selection, crossover, and mutation operators to fill an archive of individuals; then, the non-dominated individuals of both the original population and the archive are copied into a new population. If the number of non-dominated individuals is greater than the population size, a truncation operator based on the distances to the k-th nearest neighbor is used. This way, the individuals having the minimum distance to any other individual are chosen to be discarded.

- OMOPSO (Optimized MOPSO, Coello *et al.* [19]) is a multi-objective particle swarm optimization algorithm. Its main features include the use of an external archive based on the crowding distance of NSGA-II to filter out leader solutions and the use of mutation operators to accelerate the convergence of the swarm. OMOPSO also has an archive to store the best solutions found during the search. This archive makes use of the concept of ε-dominance to limit the number of solutions stored. Here, we consider the population containing the leaders as the final approximation set.
- AbYSS is an adaptation of the *scatter search* metaheuristic to the multi-objective domain proposed by Nebro *et al.* in [18]. This algorithm uses an external archive similar to the one employed by OMOPSO. AbYSS incorporates operators of the evolutionary algorithms domain, including polynomial mutation and simulated binary (SBX) crossover in the improvement and solution combination methods, respectively.
- MOCell (Nebro *et al.* [16]) is a cellular GA. As OMOPSO and AbYSS, it includes an external archive to store the non-dominated solutions found so far. This archive makes use of the crowding distance of NSGA-II to maintain diversity. MOCell incorporates a feedback procedure: after each interation some random solutions in the current population are replaced by solutions contained in the archive. Here, we have used an asynchronous version of MOCell, called aMOCell4 in [16]. Furthermore, in this version the feedback procedure takes place through the selection operator: one parent is selected from the neighborhood of the current solution, and the other parent is selected randomly from the archive.

We have used the implementation of these algorithms provided by jMetal [8], a Java-based framework aimed at multi-objective optimization problem solving.

6 Experimentation

In this section we explain the benchmark problems used to evaluate the algorithms, the quality indicators used to assess their performance, the criterion used to measure the convergence speed, the parameter settings used, the followed methodology, and the statistical tests carried out.

6.1 Benchmark Problems

Here, we describe the different sets of problems addressed in this work. These problems are well-known, and they have been used in many studies in this area.

The problems families are the following:

- **Zitzler-Deb-Thiele (ZDT):** This benchmark is composed of five bi-objective problems [25]: ZDT1 (convex), ZDT2 (nonconvex), ZDT3 (nonconvex, disconnected), ZDT4 (convex, multimodal), and ZDT6 (nonconvex, nonuniformly spaced). These problems are scalable according to the number of decision variables.

- **Deb-Thiele-Laumanns-Zitzler (DTLZ):** The problems of this family are scalable both in the number of variables and objectives [6]. It is composed of the following seven problems: DTLZ1 (linear), DTLZ2-4 (nonconvex), DTLZ5-6 (degenerate), and DTLZ7 (disconnected).
- **Walking-Fish-Group (WFG):** This set is composed of nine problems, WFG1 - WFG9, that have been constructed using the WFG toolkit [12]. The properties of these problems are detailed in Table 1. They all are scalable both in the number of variables and the number of objectives.

Table 1 Properties of the MOPs created using the WFG toolkit

Problem	Separability	Modality	Bias	Geometry
WFG1	separable	uni	polynomial, flat	convex, mixed
WFG2	non-separable	f_1 uni, f_2 multi	no bias	convex, disconnected
WFG3	non-separable	uni	no bias	linear, degenerate
WFG4	non-separable	multi	no bias	concave
WFG5	separable	deceptive	no bias	concave
WFG6	non-separable	uni	no bias	concave
WFG7	separable	uni	parameter dependent	concave
WFG8	non-separable	uni	parameter dependent	concave
WFG9	non-separable	multi, deceptive	parameter dependent	concave

In this work we have used the bi-objective formulation of the DTLZ and WFG problem families. A total of 21 MOPs are used to evaluate the six metaheuristics.

6.2 *Quality Indicators*

To assess the search capabilities of algorithms on the test problems, two different issues are normally taken into account: the distance between the generated solution set by the proposed algorithm to the optimal Pareto front should be minimized (convergence) and the spread of found solutions should be maximized in order to obtain as smooth and uniform a distribution of solutions as possible (diversity). To measure these two criteria it is necessary to know the exact location of the optimal Pareto front; the benchmark problems used in this work have known Pareto fronts.

The quality indicators can be classified into three categories depending on whether they evaluate the closeness to the Pareto front, the diversity in the solutions obtained, or both [2]. We have adopted one indicator of each type.

- **Unary Epsilon Indicator ($I^1_{\varepsilon+}$).** This indicator was proposed by Zitzler et al. [27] and makes direct use of the principle of Pareto-dominance. Given an approximation set of a problem, A, the $I^1_{\varepsilon+}$ indicator is a measure of the smallest distance one would need to translate every point in A so that it dominates the

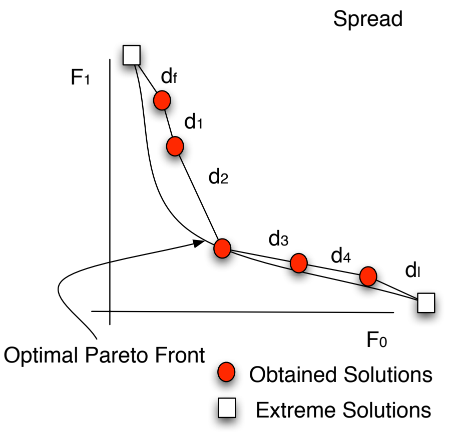


Fig. 3 Calculating the Spread quality indicator

optimal Pareto front of the problem. More formally, given $\mathbf{z^1} = (z_1^1, ..., z_n^1)$ and $\mathbf{z^2} = (z_1^2, ..., z_n^2)$, where n is the number of objectives:

$$I^1_{\varepsilon+}(A) = inf_{\varepsilon \in \mathbb{R}} \left\{ \forall \mathbf{z^2} \in Pareto\ Optimal\ Front\ \exists \mathbf{z^1} \in A : \mathbf{z^1} \prec_\varepsilon \mathbf{z^2} \right\} \quad (1)$$

where, $\mathbf{z^1} \prec_\varepsilon \mathbf{z^2}$ if and only if $\forall 1 \leq i \leq n : z_i^1 < \varepsilon + z_i^2$.

- **Spread (Δ).** The diversity *Spread* indicator [3] measures the extent of spread achieved among the obtained solutions. This indicator (illustrated in Fig. 3) is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \ , \quad (2)$$

where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and d_f and d_l are the Euclidean distances to the *extreme* (bounding) solutions of the optimal Pareto front in the objective space (see [3] for the details). Δ takes a value of zero for an ideal distribution, pointing out a perfect spread out of the solutions in the Pareto front. We apply this indicator after a normalization of the objective function values.

- **Hypervolume (HV).** The *HV* indicator calculates the volume, in the objective space, covered by members of a non-dominated set of solutions Q for problems where all objectives are to be minimized [24]. In the example depicted in Fig. 4, the *HV* is the region enclosed within the discontinuous line, where $Q = \{A, B, C\}$ (in the figure, the grey area represents the objective space that has been explored). Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with a reference point W and the solution i as the diagonal corners of the hypercube. The reference point can be found simply by constructing a vector of worst objective function values. Thereafter, a union of all hypercubes is found and its hypervolume (HV) is calculated:

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right). \quad (3)$$

Algorithms with larger HV values are desirable. Since this indicator is not free from arbitrary scaling of objectives, we have evaluated the metric by using normalized objective function values.

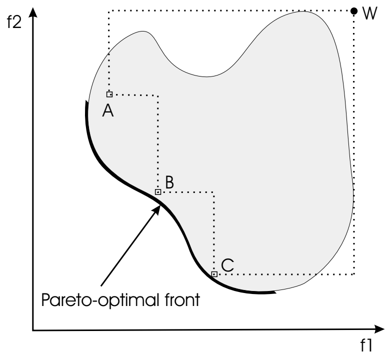


Fig. 4 The hypervolume enclosed by the non-dominated solutions

6.3 Convergence Speed Criterion

Since one of our main interests is to analyze the convergence speed of the analyzed algorithms, it is important to define, first, what we mean by convergence, and to ensure that such definition allows us to measure it in a quantitative and meaningful way. We have studied and defined in [17], a stopping condition based on the *high*

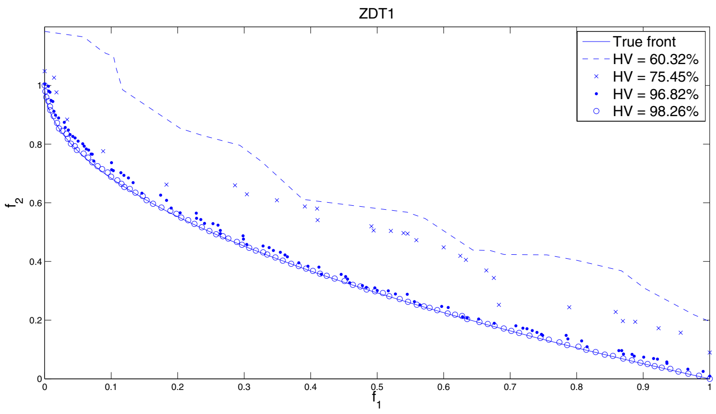


Fig. 5 Fronts with different HV values obtained for problem ZDT1

Table 2 Parameterization (L = individual length)

Parameterization used in NSGA-II [3]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Parameterization used in SPEA2 [26]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Parameterization used in AbYSS [18]	
Population Size	20 individuals
Reference Set Size	10 + 10
Recombination	simulated binary, $p_c = 1.0$
Mutation (local search)	polynomial, $p_m = 1.0/L$
Archive Size	100 individuals
Parameterization used in MOCell [16]	
Population Size	100 individuals (10×10)
Neighborhood	1-hop neighbours (8 surrounding solutions)
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Archive Size	100 individuals
Parameterization used in OMOPSO [19]	
Swarm size	100 particles
Mutation	uniform + non-uniform
Leaders Size	100

quality of the approximation of the Pareto front found. We have used the HV quality indicator for that purpose.

In Fig. 5 we show different approximations to the Pareto front for the problem ZDT1 with different percentages of HV. We can observe that a front with a hypervolume of 98.26% represents a reasonable approximation to the optimal Pareto front in terms of convergence and diversity of solutions. So, we have taken 98% of the hypervolume of the optimal Pareto front as a criterion to decide that a problem has been successfully solved. In this way, we mean by convergence speed the number of function evaluations required to achieve this termination condition. Those algorithms requiring fewer function evaluations can be considered to be more efficient or faster.

6.4 Parameter Settings

We have chosen a set of parameter settings to guarantee a fair comparison among the algorithms. All GAs (NSGA-II, SPEA2, and MOCell) use an internal population of size equal to 100; the size of the archive is also 100 in SPEA2, OMOPSO, AbYSS, and MOCell. OMOPSO has been configured with 100 particles. For AbYSS, both the population and the reference set have a size of 20 solutions. The two versions of NSGA-II share the same parameterization.

In the GAs we have used SBX and polynomial mutation [2] as operators for crossover and mutation operators, respectively. The distribution indices for both operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. The crossover probability is $p_c = 0.9$ and the mutation probability is $p_m = 1/L$, where L is the number of decision variables. AbYSS uses polynomial mutation in the improvement method and SBX in the solution combination method. OMOPSO applies a combination of uniform and non-uniform mutation. A summary of the parameter settings is included in Table 2.

6.5 Methodology

The stopping criterion is to reach $25,000$ function evaluations in the experiments performed for assessing the quality of the obtained solution sets. The quality indicators are computed after the algorithms have finished their executions.

In the experiments carried out to study the convergence speed, the stopping criterion is to reach either $1,000,000$ function evaluations or a front with 98% HV of the optimal Pareto front. If an algorithm stops according to the first condition, we consider that it has failed to solve the problem. In these experiments, the HV is measured at every 100 function evaluations.

6.6 Statistical Tests

Since we are dealing with stochastic algorithms we have made 100 independent runs of each experiment, and we show the median, $\tilde{x}$, and interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively. The following statistical analysis has been performed throughout this work [7]. Firstly, a Kolmogorov-Smirnov test was performed in order to check whether the values of the results follow a normal (gaussian) distribution or not. If the distribution is normal, the Levene test checks for the homogeneity of the variances. If samples have equal variance (positive Levene test), an ANOVA test is done; otherwise a Welch test is performed. For non-gaussian distributions, the non-parametric Kruskal-Wallis test is used to compare the medians of the algorithms. Fig. 6 summarizes the statistical analysis.

We always consider in this work a confidence level of 95% (i.e., significance level of 5% or p-value under 0.05) in the statistical tests, which means that the differences are unlikely to have occurred by chance with a probability of 95%. Successful tests are marked with '+' symbols in the last column in all the tables containing the results; conversely, '-' means that no statistical confidence was found (p-value $>$ 0.05). For the sake of better understanding, the best result for each problem has a gray colored background and the second best one has a clearer gray background.

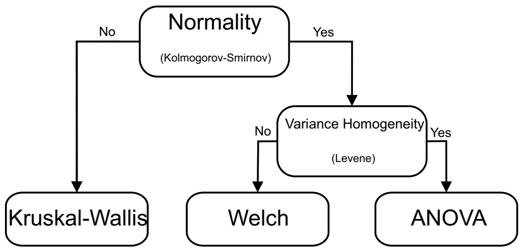


Fig. 6 Statistical analysis performed in this work

7 Results

This section is devoted to presenting and discussing the results of the experiments. We start with the analysis of the obtained values of the $I^1_{\varepsilon+}$, Δ, and HV quality indicators; after that, we focus on the convergence speed.

7.1 Quality Assessment

The results after applying the $I^1_{\varepsilon+}$ indicator are provided in Table 3. Our main interest is to focus on the two versions of NSGA-II, not to determine the best algorithm in the comparisons. We can observe that NSGA-II$_{gen}$ only achieves the best (lower) values in two out of the 21 problems composing the whole benchmark, while NSGA-II$_{ss}$ gets four best and eleven second best results. With the exceptions of problems WFG1 and WFG8, it is clear that the steady-state scheme in NSGA-II allows to improve the convergence of the obtained fronts.

If we make a ranking of the algorithms according to the convergence indicator, considering the best and second best values, it would be headed by MOCell followed by NSGA-II$_{ss}$; after them, OMOPSO, NSGA-II$_{gen}$, AbYSS, and SPEA2. In Fig. 7

Table 3 Median and interquartile range of the $I^1_{\varepsilon+}$ quality indicator

Problem	NSGA-II$_{gen}$ $\bar{x}_{IQR}$	NSGA-II$_{ss}$ $\bar{x}_{IQR}$	SPEA2 $\bar{x}_{IQR}$	AbYSS $\bar{x}_{IQR}$	MOCell $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	
ZDT1	$1.37e-02_{3.0e-03}$	$5.81e-03_{6.3e-04}$	$8.69e-03_{1.1e-03}$	$7.72e-03_{1.8e-03}$	$6.23e-03_{4.1e-04}$	$5.77e-03_{3.8e-04}$	+
ZDT2	$1.28e-02_{2.3e-03}$	$5.79e-03_{5.5e-04}$	$8.73e-03_{1.4e-03}$	$7.10e-03_{1.6e-03}$	$5.57e-03_{3.0e-04}$	$5.64e-03_{3.0e-04}$	+
ZDT3	$8.13e-03_{1.9e-03}$	$5.24e-03_{5.4e-04}$	$9.72e-03_{1.9e-03}$	$6.10e-03_{3.1e-01}$	$5.66e-03_{7.5e-04}$	$6.16e-03_{1.2e-03}$	+
ZDT4	$1.49e-02_{3.0e-03}$	$9.78e-03_{2.6e-03}$	$3.42e-02_{7.9e-02}$	$1.14e-02_{4.2e-03}$	$8.17e-03_{2.3e-03}$	$7.40e+00_{4.5e+00}$	+
ZDT6	$1.47e-02_{2.8e-03}$	$7.02e-03_{7.6e-04}$	$2.42e-02_{5.2e-03}$	$5.06e-03_{3.9e-04}$	$6.53e-03_{5.6e-04}$	$4.67e-03_{3.3e-04}$	+
DTLZ1	$7.13e-03_{1.6e-03}$	$4.62e-03_{1.9e-03}$	$5.89e-03_{2.8e-03}$	$5.85e-03_{5.5e-03}$	$4.02e-03_{1.5e-03}$	$1.54e+01_{1.4e+01}$	+
DTLZ2	$1.11e-02_{2.7e-03}$	$5.13e-03_{3.6e-04}$	$7.34e-03_{1.1e-03}$	$5.39e-03_{4.6e-04}$	$5.09e-03_{2.8e-04}$	$5.23e-03_{2.9e-04}$	+
DTLZ3	$1.04e+00_{1.2e+00}$	$9.63e-01_{1.4e+00}$	$2.28e+00_{1.9e+00}$	$1.66e+00_{1.6e+00}$	$7.91e-01_{1.0e+00}$	$7.87e+01_{7.5e+01}$	+
DTLZ4	$1.13e-02_{9.9e-01}$	$5.24e-03_{9.9e-01}$	$7.66e-03_{9.9e-01}$	$5.39e-03_{3.0e-04}$	$5.74e-03_{9.9e-01}$	$5.55e-03_{4.5e-04}$	+
DTLZ5	$1.05e-02_{2.5e-03}$	$5.14e-03_{3.4e-04}$	$7.47e-03_{1.2e-03}$	$5.36e-03_{5.2e-04}$	$5.08e-03_{3.2e-04}$	$5.27e-03_{2.8e-04}$	+
DTLZ6	$4.39e-02_{3.4e-02}$	$3.07e-02_{2.5e-02}$	$3.03e-01_{5.3e-02}$	$9.50e-02_{4.7e-02}$	$4.16e-02_{3.8e-02}$	$5.18e-03_{4.1e-04}$	+
DTLZ7	$1.04e-02_{2.8e-03}$	$5.13e-03_{4.1e-04}$	$9.09e-03_{1.4e-03}$	$5.51e-03_{9.6e-04}$	$5.19e-03_{1.0e-03}$	$5.39e-03_{4.8e-04}$	+
WFG1	$3.52e-01_{4.6e-01}$	$4.98e-01_{5.3e-01}$	$9.92e-01_{2.1e-01}$	$1.05e+00_{5.1e-01}$	$4.49e-01_{5.0e-01}$	$1.13e+00_{2.1e-01}$	+
WFG2	$7.10e-01_{7.0e-01}$	$7.10e-01_{7.0e-01}$	$7.10e-01_{6.9e-01}$	$7.11e-01_{1.6e-03}$	$7.10e-01_{3.7e-04}$	$9.51e-03_{9.0e-04}$	+
WFG3	$2.00e+00_{5.8e-04}$	$2.00e+00_{4.3e-04}$	$2.00e+00_{1.1e-03}$	$2.00e+00_{1.6e-03}$	$2.00e+00_{5.2e-04}$	$2.00e+00_{2.0e-05}$	+
WFG4	$3.26e-02_{6.7e-03}$	$1.52e-02_{1.5e-03}$	$2.52e-02_{4.0e-03}$	$1.49e-02_{7.7e-04}$	$1.51e-02_{7.3e-04}$	$4.33e-02_{5.6e-03}$	+
WFG5	$8.41e-02_{8.3e-03}$	$6.41e-02_{1.5e-03}$	$7.27e-02_{2.9e-03}$	$6.39e-02_{7.5e-04}$	$6.35e-02_{6.5e-04}$	$6.36e-02_{6.6e-04}$	+
WFG6	$4.14e-02_{1.6e-02}$	$2.50e-02_{2.8e-02}$	$3.11e-02_{1.4e-02}$	$7.84e-02_{5.9e-02}$	$3.65e-02_{5.4e-02}$	$1.43e-02_{6.7e-04}$	+
WFG7	$3.47e-02_{8.1e-03}$	$1.51e-02_{1.5e-03}$	$2.54e-02_{3.0e-03}$	$1.55e-02_{1.1e-03}$	$1.49e-02_{7.5e-04}$	$1.52e-02_{7.6e-04}$	+
WFG8	$3.38e-01_{2.3e-01}$	$5.08e-01_{2.2e-01}$	$5.11e-01_{1.9e-01}$	$5.13e-01_{7.4e-02}$	$5.08e-01_{5.3e-02}$	$5.09e-01_{2.0e-03}$	+
WFG9	$3.73e-02_{7.5e-03}$	$1.80e-02_{3.7e-03}$	$2.92e-02_{5.9e-03}$	$2.21e-02_{6.0e-03}$	$1.94e-02_{3.6e-03}$	$2.55e-02_{2.7e-03}$	+

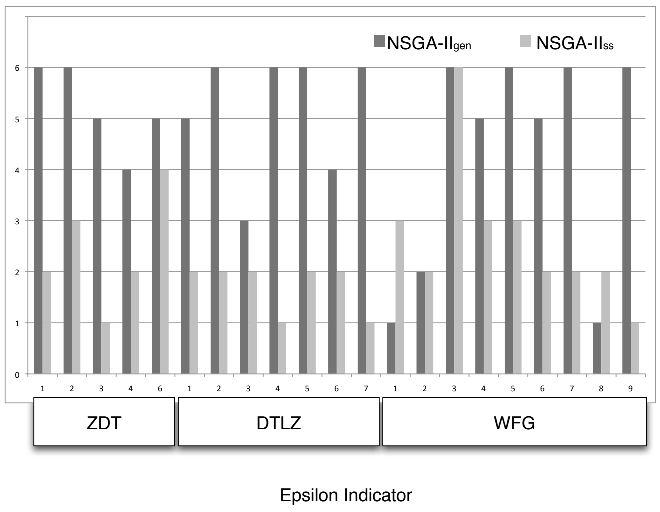


Fig. 7 Positions of NSGA-II$_{gen}$ (left columns) and NSGA-II$_{ss}$ (right columns) in the ranking of the $I^1_{\varepsilon+}$ indicator

we include the positions of each NSGA-II version if we rank the six compared algorithms per individual problem. We can observe clearly that while NSGA-II$_{gen}$ is in the sixth position in many problems, NSGA-II$_{ss}$ is ranked in the first or second positions in all but six instances.

The values of the Δ indicator are included in Table 4. From the table we can see that the steady-state version of NSGA-II yields better (lower) values than its generational counterpart in all the 21 benchmark problems. However, it is unable to

Table 4 Median and interquartile range of the Δ quality indicator

	NSGA-II$_{gen}$	NSGA-II$_{ss}$	SPEA2	AbYSS	MOCell	OMOPSO	
Problem	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	
ZDT1	$3.70e-01_{4.2e-02}$	$7.52e-02_{1.5e-02}$	$1.52e-01_{2.2e-02}$	$1.05e-01_{2.0e-02}$	$7.64e-02_{1.3e-02}$	$7.34e-02_{1.7e-02}$	+
ZDT2	$3.81e-01_{4.7e-02}$	$7.80e-02_{1.3e-02}$	$1.55e-01_{2.7e-02}$	$1.07e-01_{1.8e-02}$	$7.67e-02_{1.4e-02}$	$7.29e-02_{1.5e-02}$	+
ZDT3	$7.47e-01_{1.8e-02}$	$7.03e-01_{3.5e-03}$	$7.10e-01_{7.5e-03}$	$7.09e-01_{9.7e-03}$	$7.04e-01_{6.2e-03}$	$7.08e-01_{6.4e-03}$	+
ZDT4	$4.02e-01_{5.8e-02}$	$1.27e-01_{2.9e-02}$	$2.72e-01_{1.6e-01}$	$1.27e-01_{3.5e-02}$	$1.10e-01_{2.8e-02}$	$8.85e-01_{4.6e-02}$	+
ZDT6	$3.56e-01_{3.6e-02}$	$1.05e-01_{1.5e-02}$	$2.28e-01_{2.5e-02}$	$8.99e-02_{1.4e-02}$	$9.33e-02_{1.3e-02}$	$2.95e-01_{1.1e+00}$	+
DTLZ1	$4.03e-01_{6.1e-02}$	$1.18e-01_{4.0e-02}$	$1.81e-01_{9.8e-02}$	$1.40e-01_{1.7e-01}$	$1.05e-01_{3.6e-02}$	$7.74e-01_{1.3e-01}$	+
DTLZ2	$3.84e-01_{3.8e-02}$	$1.10e-01_{1.6e-02}$	$1.48e-01_{1.6e-02}$	$1.09e-01_{1.9e-02}$	$1.08e-01_{1.7e-02}$	$1.27e-01_{2.0e-02}$	+
DTLZ3	$9.53e-01_{1.6e-01}$	$9.52e-01_{3.4e-01}$	$1.07e+00_{1.6e-01}$	$7.55e-01_{4.5e-01}$	$7.45e-01_{5.5e-01}$	$7.68e-01_{9.3e-02}$	+
DTLZ4	$3.95e-01_{6.4e-01}$	$1.13e-01_{9.0e-01}$	$1.48e-01_{8.6e-01}$	$1.08e-01_{1.8e-02}$	$1.23e-01_{9.0e-01}$	$1.23e-01_{1.9e-02}$	+
DTLZ5	$3.79e-01_{4.0e-02}$	$1.11e-01_{1.6e-02}$	$1.50e-01_{1.9e-02}$	$1.10e-01_{2.0e-02}$	$1.09e-01_{1.7e-02}$	$1.25e-01_{1.9e-02}$	+
DTLZ6	$8.64e-01_{3.0e-01}$	$1.81e-01_{5.3e-02}$	$8.25e-01_{9.3e-02}$	$2.31e-01_{6.3e-02}$	$1.50e-01_{4.3e-02}$	$1.03e-01_{2.1e-02}$	+
DTLZ7	$6.23e-01_{2.5e-02}$	$5.19e-01_{1.9e-03}$	$5.44e-01_{1.3e-02}$	$5.19e-01_{1.3e-03}$	$5.19e-01_{2.9e-02}$	$5.20e-01_{3.7e-03}$	+
WFG1	$7.18e-01_{5.4e-02}$	$5.81e-01_{5.8e-02}$	$6.51e-01_{4.8e-02}$	$6.66e-01_{5.8e-02}$	$5.81e-01_{9.4e-02}$	$1.15e+00_{1.2e-01}$	+
WFG2	$7.93e-01_{1.7e-02}$	$7.47e-01_{1.0e-02}$	$7.53e-01_{1.3e-02}$	$7.46e-01_{4.3e-03}$	$7.47e-01_{2.2e-03}$	$7.60e-01_{2.7e-03}$	+
WFG3	$6.12e-01_{3.6e-02}$	$3.71e-01_{7.2e-03}$	$4.39e-01_{1.2e-02}$	$3.73e-01_{8.7e-03}$	$3.64e-01_{6.3e-03}$	$3.65e-01_{6.9e-03}$	+
WFG4	$3.79e-01_{3.9e-02}$	$1.40e-01_{2.0e-02}$	$2.72e-01_{2.5e-02}$	$1.36e-01_{2.1e-02}$	$1.36e-01_{2.2e-02}$	$3.94e-01_{5.2e-02}$	+
WFG5	$4.13e-01_{5.1e-02}$	$1.38e-01_{1.5e-02}$	$2.79e-01_{2.3e-02}$	$1.31e-01_{2.1e-02}$	$1.32e-01_{2.2e-02}$	$1.36e-01_{2.0e-02}$	+
WFG6	$3.90e-01_{4.2e-02}$	$1.23e-01_{3.2e-02}$	$2.49e-01_{3.1e-02}$	$1.45e-01_{4.3e-02}$	$1.27e-01_{4.0e-02}$	$1.19e-01_{1.9e-02}$	+
WFG7	$3.79e-01_{4.6e-02}$	$1.11e-01_{1.9e-02}$	$2.47e-01_{1.8e-02}$	$1.17e-01_{3.0e-02}$	$1.07e-01_{1.8e-02}$	$1.29e-01_{1.7e-02}$	+
WFG8	$6.45e-01_{5.5e-02}$	$5.63e-01_{5.7e-02}$	$6.17e-01_{8.1e-02}$	$5.86e-01_{7.1e-02}$	$5.57e-01_{4.2e-02}$	$5.42e-01_{3.6e-02}$	+
WFG9	$3.96e-01_{4.1e-02}$	$1.52e-01_{2.1e-02}$	$2.92e-01_{2.0e-02}$	$1.50e-01_{2.2e-02}$	$1.44e-01_{1.7e-02}$	$2.03e-01_{2.0e-02}$	+

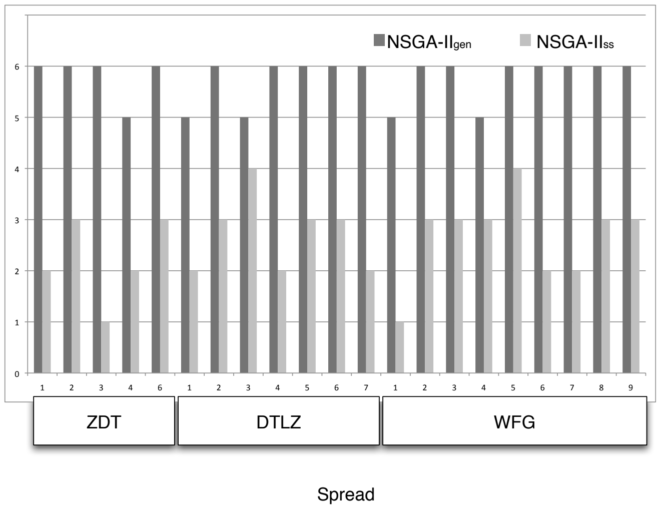


Fig. 8 Positions of NSGA-II$_{gen}$ (left columns) and NSGA-II$_{ss}$ (right columns) in the ranking of the Δ indicator

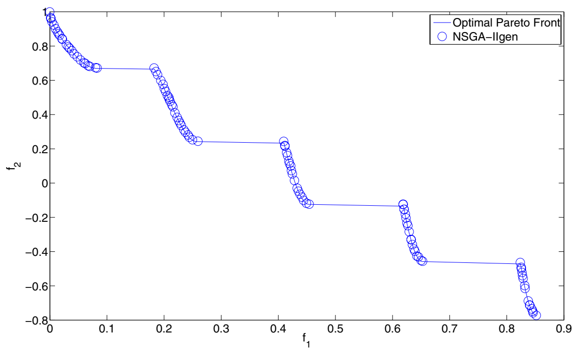
outperform MOCell and AbYSS, the two best algorithms taking into account this indicator except for the two problems ZDT3 and WFG1.

As before, we include in Fig. 8 the positions of the two versions when ranking the six compared algorithms per problem. The improvements of the steady-state version over the generational one are evident: the latter occupies the last rank position in most of the problems, while the former has two fourth positions as its worst results.

To illustrate the differences between the two versions of NSGA-II, we include in Fig. 9 the approximation sets to the Pareto front obtained by them when solving problem ZDT3. This problem is characterized by having a discontinuos Pareto front. We can observe how the steady-state version produces a front having a uniform spread of the solutions (Fig. 9 - bottom), while the generational one generates a not-so-uniform front.

The results of the *HV* quality indicator are included in Table 5. Those cells containg the symbol "–" mean that the *HV* has a value of 0, meaning that the obtained solution sets are out of the limits of the optimal Pareto front. We observe, first, that NSGA-II$_{ss}$ yield better (higher) values than NSGA-II$_{gen}$ in 18 out the 21 problems. Second, a ranking of the algorithms considering the number of best and second best values would be led by NSGA-II$_{ss}$, because although OMOPSO yields six best values (five in the case of NSGA-II$_{ss}$), it has only a single second best result, while NSGA-II$_{ss}$ has eight. The ranking per problem is included in Fig. 10, which shows that NSGA-II$_{ss}$ has a ranking greater than three in only three problems.

As the *HV* measures both convergence and diversity, and considering the other two indicators, we can conclude that NSGA-II$_{ss}$ not only outperforms NSGA-II$_{gen}$, but it is also a competitive technique based on the convergence of the produced Pareto fronts of all the evaluated algorithms.



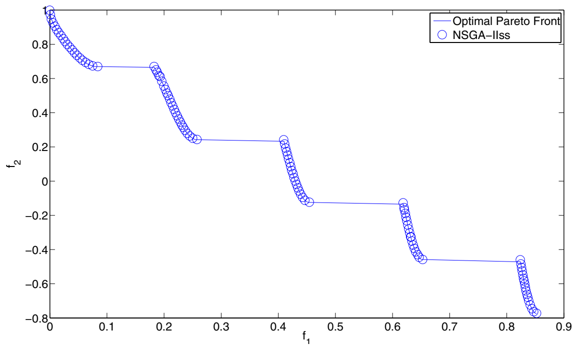


Fig. 9 Pareto fronts obtained by NSGA-II$_{gen}$ (top) and NSGA-II$_{ss}$ (bottom) when solving problem ZDT3

To conclude this section we would like to remark, first, that the obtained results in the experiments carried out have statistical significance, as it can be observed in the '+' symbols in the last column in the three tables. Second, it has to be pointed out that the use of the steady-states scheme has a computational cost that has to be taken into account. Specifically, we have measured the running times of the two versions of NSGA-II when solving all the problems, and NSGA-II$_{ss}$ is about 10 times slower than the generational algorithm: the mean time is about 1.2 seconds per execution, in the case of the original NSGA-II, and roughly 12.5 seconds the steady-state version. We have used the profiling tool provided by the Java IDE Netbeans 6.1 to analyze the execution of the two algorithms when solving the ZDT1 problem. The profiling report has shown that the computing time required to evaluate the problem is less than 1% of the total execution time. So, although the diferences between the two algorithms are important in the context of our study, we must note that in a real scenario the computing time of the algorithms can become negligible. As an example, if evaluating the objective functions of a problem requires 1 second, as we carry out 25,000 evaluations, the total time would be 25,000 seconds (6,94 hours), so it is obvious that the running times of the algorithms would not be relevant in this situation.

Table 5 Median and interquartile range of the *HV* quality indicator

Problem	NSGA-II$_{gen}$ $\bar{x}_{IQR}$	NSGA-II$_{ss}$ $\bar{x}_{IQR}$	SPEA2 $\bar{x}_{IQR}$	AbYSS $\bar{x}_{IQR}$	MOCell $\bar{x}_{IQR}$	OMOPSO $\bar{x}_{IQR}$	
ZDT1	$6.59e-01_{4.4e-04}$	$6.62e-01_{1.4e-04}$	$6.60e-01_{3.9e-04}$	$6.61e-01_{3.2e-04}$	$6.61e-01_{2.5e-04}$	$6.61e-01_{3.2e-04}$	+
ZDT2	$3.26e-01_{4.3e-04}$	$3.28e-01_{1.6e-04}$	$3.26e-01_{8.1e-04}$	$3.28e-01_{2.8e-04}$	$3.28e-01_{4.3e-04}$	$3.28e-01_{2.4e-04}$	+
ZDT3	$5.15e-01_{2.3e-04}$	$5.16e-01_{8.0e-05}$	$5.14e-01_{3.6e-04}$	$5.16e-01_{3.5e-03}$	$5.15e-01_{3.1e-04}$	$5.15e-01_{8.8e-04}$	+
ZDT4	$6.56e-01_{4.5e-03}$	$6.57e-01_{4.0e-03}$	$6.51e-01_{1.2e-02}$	$6.55e-01_{6.0e-03}$	$6.59e-01_{3.0e-03}$	–	+
ZDT6	$3.88e-01_{2.3e-03}$	$3.96e-01_{1.1e-03}$	$3.79e-01_{3.6e-03}$	$4.00e-01_{1.9e-04}$	$3.97e-01_{1.1e-03}$	$4.01e-01_{7.1e-05}$	+
DTLZ1	$4.88e-01_{5.5e-03}$	$4.89e-01_{6.5e-03}$	$4.89e-01_{6.2e-03}$	$4.86e-01_{1.7e-02}$	$4.91e-01_{3.8e-03}$	–	+
DTLZ2	$2.11e-01_{3.1e-04}$	$2.12e-01_{4.3e-05}$	$2.12e-01_{1.7e-04}$	$2.12e-01_{6.5e-05}$	$2.12e-01_{4.5e-05}$	$2.12e-01_{2.8e-04}$	+
DTLZ3	–	–	–	–	–	–	+
DTLZ4	$2.09e-01_{2.1e-01}$	$2.11e-01_{2.1e-01}$	$2.10e-01_{2.1e-01}$	$2.11e-01_{5.9e-05}$	$2.11e-01_{2.1e-01}$	$2.10e-01_{4.0e-04}$	+
DTLZ5	$2.11e-01_{3.5e-04}$	$2.12e-01_{3.7e-05}$	$2.12e-01_{1.7e-04}$	$2.12e-01_{6.8e-05}$	$2.12e-01_{3.1e-05}$	$2.12e-01_{3.0e-04}$	+
DTLZ6	$1.75e-01_{3.6e-02}$	$1.73e-01_{2.8e-02}$	$9.02e-03_{1.4e-02}$	$1.11e-01_{4.1e-02}$	$1.61e-01_{4.2e-02}$	$2.12e-01_{5.0e-05}$	+
DTLZ7	$3.33e-01_{2.1e-04}$	$3.34e-01_{3.9e-05}$	$3.34e-01_{2.2e-04}$	$3.34e-01_{7.8e-05}$	$3.34e-01_{9.5e-05}$	$3.34e-01_{2.2e-04}$	+
WFG1	$5.23e-01_{1.3e-01}$	$4.90e-01_{1.9e-01}$	$3.85e-01_{1.1e-01}$	$2.27e-01_{1.3e-01}$	$4.95e-01_{1.7e-01}$	$1.60e-01_{9.0e-02}$	+
WFG2	$5.61e-01_{2.8e-03}$	$5.62e-01_{2.6e-03}$	$5.62e-01_{2.8e-03}$	$5.61e-01_{1.1e-03}$	$5.62e-01_{2.9e-04}$	$5.64e-01_{6.8e-05}$	+
WFG3	$4.41e-01_{3.2e-04}$	$4.42e-01_{1.8e-04}$	$4.42e-01_{2.0e-04}$	$4.42e-01_{5.9e-04}$	$4.42e-01_{1.9e-04}$	$4.42e-01_{2.2e-05}$	+
WFG4	$2.17e-01_{4.9e-04}$	$2.19e-01_{2.4e-04}$	$2.18e-01_{3.0e-04}$	$2.19e-01_{2.0e-04}$	$2.19e-01_{2.3e-04}$	$2.06e-01_{1.7e-03}$	+
WFG5	$1.95e-01_{3.6e-04}$	$1.96e-01_{6.7e-05}$	$1.96e-01_{1.8e-04}$	$1.96e-01_{6.3e-05}$	$1.96e-01_{6.9e-05}$	$1.96e-01_{6.3e-05}$	+
WFG6	$2.03e-01_{9.0e-03}$	$2.03e-01_{1.9e-02}$	$2.04e-01_{8.6e-03}$	$1.71e-01_{3.3e-02}$	$1.95e-01_{3.4e-02}$	$2.10e-01_{1.1e-04}$	+
WFG7	$2.09e-01_{3.3e-04}$	$2.11e-01_{1.4e-04}$	$2.10e-01_{2.4e-04}$	$2.11e-01_{1.7e-04}$	$2.11e-01_{1.3e-04}$	$2.10e-01_{1.0e-04}$	+
WFG8	$1.47e-01_{2.1e-03}$	$1.48e-01_{1.6e-03}$	$1.47e-01_{2.2e-03}$	$1.44e-01_{3.2e-03}$	$1.47e-01_{2.2e-03}$	$1.46e-01_{1.1e-03}$	+
WFG9	$2.37e-01_{1.7e-03}$	$2.40e-01_{1.9e-03}$	$2.39e-01_{2.3e-03}$	$2.38e-01_{3.6e-03}$	$2.39e-01_{2.6e-03}$	$2.37e-01_{5.8e-04}$	+

7.2 Convergence Speed

In the previous section we have shown that the steady-state version of NSGA-II performed better than the original algorithm in most of the tested problems. In this section we analyze the obtained results when measuring the convergence speed.

Table 6 contains the number of evaluations required by the algorithms to reach a Pareto front with 98% of the *HV* of the optimal Pareto front. There are cases

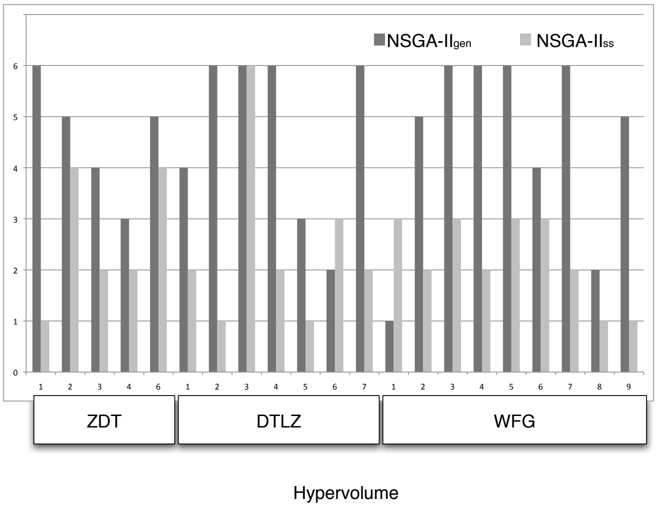


Fig. 10 Positions of NSGA-II$_{gen}$ (left columns) and NSGA-II$_{ss}$ (right columns) in the ranking of the *HV* indicator

Table 6 Median and *IQR* of the number of evaluations computed by the algorithms

Problem	NSGA-II$_{gen}$ $\tilde{x}_{IQR}$	NSGA-II$_{ss}$ $\tilde{x}_{IQR}$	SPEA2 $\tilde{x}_{IQR}$	AbYSS $\tilde{x}_{IQR}$	MOCell $\tilde{x}_{IQR}$	OMOPSO $\tilde{x}_{IQR}$	
ZDT1	1.430e+04 $_{8.0e+02}$	1.160e+04 $_{9.0e+02}$	1.600e+04 $_{1.1e+03}$	1.370e+04 $_{1.6e+03}$	1.300e+04 $_{1.2e+03}$	6.800e+03 $_{2.0e+03}$	+
ZDT2	2.460e+04 $_{1.6e+03}$	1.770e+04 $_{1.3e+03}$	2.480e+04 $_{1.9e+03}$	1.710e+04 $_{2.8e+03}$	1.170e+04 $_{4.0e+03}$	8.900e+03 $_{3.6e+03}$	+
ZDT3	1.280e+04 $_{8.5e+02}$	1.095e+04 $_{1.0e+03}$	1.520e+04 $_{1.0e+03}$	1.270e+04 $_{2.0e+03}$	1.300e+04 $_{1.3e+03}$	9.850e+03 $_{2.7e+03}$	+
ZDT4	2.245e+04 $_{5.9e+03}$	1.985e+04 $_{5.4e+03}$	2.520e+04 $_{6.0e+03}$	2.285e+04 $_{1.1e+04}$	1.635e+04 $_{5.0e+03}$	–	+
ZDT6	2.930e+04 $_{1.4e+03}$	2.280e+04 $_{1.2e+03}$	3.335e+04 $_{1.0e+03}$	1.560e+04 $_{1.2e+03}$	2.090e+04 $_{1.3e+03}$	2.800e+03 $_{1.5e+03}$	+
DTLZ1	2.495e+04 $_{8.4e+03}$	2.225e+04 $_{8.6e+03}$	2.400e+04 $_{7.5e+03}$	2.375e+04 $_{1.2e+04}$	2.015e+04 $_{7.7e+03}$	1.000e+06 $_{4.7e+04}$	+
DTLZ2	8.150e+03 $_{1.2e+03}$	5.300e+03 $_{7.0e+02}$	7.400e+03 $_{8.0e+02}$	4.700e+03 $_{9.0e+02}$	5.600e+03 $_{9.0e+02}$	8.200e+03 $_{3.1e+03}$	+
DTLZ3	1.127e+05 $_{5.3e+04}$	8.270e+04 $_{3.5e+04}$	1.000e+05 $_{3.0e+04}$	1.194e+05 $_{7.5e+04}$	6.735e+04 $_{2.3e+04}$	–	+
DTLZ4	8.650e+03 $_{1.3e+03}$	5.500e+03 $_{7.0e+02}$	7.800e+03 $_{5.0e+05}$	4.800e+03 $_{7.5e+02}$	1.000e+06 $_{9.9e+05}$	1.255e+04 $_{3.8e+03}$	+
DTLZ5	8.300e+03 $_{1.4e+03}$	5.150e+03 $_{6.0e+02}$	7.500e+03 $_{7.0e+02}$	4.650e+03 $_{8.0e+02}$	5.800e+03 $_{8.5e+02}$	8.450e+03 $_{2.9e+03}$	+
DTLZ6	1.000e+06 $_{9.7e+05}$	–	1.000e+06 $_{9.7e+05}$	–	–	4.100e+03 $_{1.5e+03}$	+
DTLZ7	1.360e+04 $_{9.0e+02}$	1.060e+04 $_{9.0e+02}$	1.585e+04 $_{1.1e+03}$	1.060e+04 $_{1.7e+03}$	1.110e+04 $_{1.6e+05}$	6.150e+03 $_{2.6e+03}$	+
WFG1	4.315e+04 $_{5.4e+04}$	3.715e+04 $_{1.5e+04}$	1.096e+05 $_{7.7e+05}$	–	4.160e+04 $_{1.7e+04}$	–	+
WFG2	1.700e+03 $_{4.0e+02}$	1.400e+03 $_{5.0e+02}$	2.000e+03 $_{7.0e+02}$	1.850e+03 $_{2.4e+03}$	1.400e+03 $_{8.0e+02}$	1.800e+03 $_{4.0e+02}$	+
WFG3	–	–	–	–	–	–	-
WFG4	2.050e+04 $_{8.8e+03}$	8.200e+03 $_{2.9e+03}$	1.280e+04 $_{4.6e+03}$	6.750e+03 $_{2.4e+03}$	1.050e+04 $_{3.1e+03}$	2.233e+05 $_{1.3e+05}$	+
WFG5	–	–	–	–	–	–	-
WFG6	–	1.000e+06 $_{9.8e+05}$	1.000e+06 $_{4.8e+04}$	–	1.000e+06 $_{5.5e+05}$	7.300e+03 $_{1.2e+03}$	+
WFG7	1.686e+05 $_{2.5e+05}$	1.035e+04 $_{2.6e+03}$	1.775e+04 $_{5.4e+03}$	9.600e+03 $_{3.4e+03}$	1.215e+04 $_{3.4e+03}$	1.495e+04 $_{2.6e+03}$	+
WFG8	–	–	–	–	–	–	+
WFG9	–	1.000e+06 $_{9.4e+05}$	–	–	–	8.935e+04 $_{4.9e+04}$	+

in which a "–" is reported when the algorithms have computed 1,000,000 function evaluations without producing a solution set with the desired *HV* value.

The results show that NSGA-II$_{ss}$ requires a fewer number of evaluations than NSGA-II$_{gen}$ in all the problems but DTLZ6, which means that the steady-approach makes NSGA-II converge faster. If we consider all the problems, we see that NSGA-II$_{ss}$ is the fastest only in two out of the 21 problems, but it is the second fastest in 10 problems. This can be clearly observed in the ranking per problem in Fig. 11, which

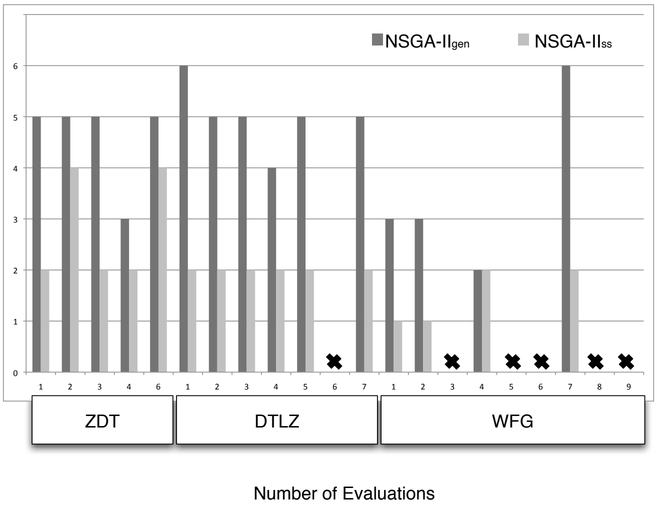
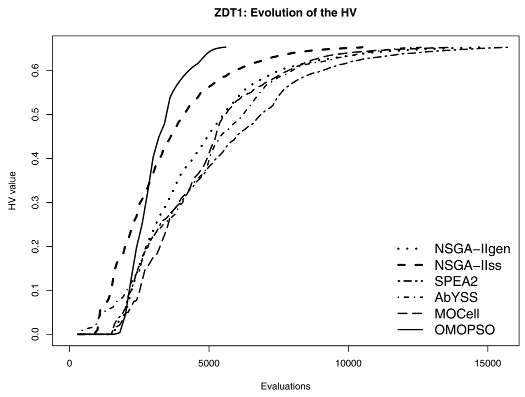


Fig. 11 Positions of NSGA-II$_{gen}$ (left columns) and NSGA-II$_{ss}$ (right columns) in the ranking of the convergence speed



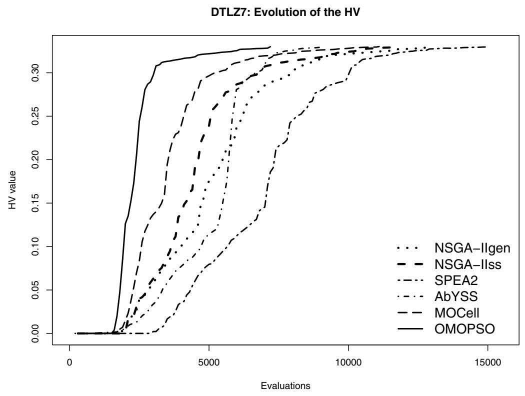


Fig. 12 Evolution of the HV value during the different generations carried out in the problem ZDT1 (top) and DTLZ7 (bottom)

allows us to conclude that the steady-state approach makes NSGA-II improve from being the second slowest algorithm in the comparison (fifth position in most of the problems) to be second one in terms of convergence speed.

We include in Fig. 12 a trace of the evolution of the value of the *HV* when solving problems ZDT1 and DTLZ7 in a single run. The values have been recorded at each 100 function evaluations. Focusing on ZDT1, Fig. 12 - top shows that NSGA-II$_{ss}$ is the second algorithm in achieving the desired *HV* value after OMOPSO, the fastest metaheuristic on this problem. The trace of the DTLZ7 problem (Fig. 12 - bottom) reveals that NSGA-II$_{ss}$ has been the fourth fastest algorithm in the monitored execution. In both cases, we can observe that the NSGA-II$_{ss}$ converges faster than the original algorithm.

8 Conclusions and Future Work

In this chapter we have studied the effect of applying a steady-state selection scheme to NSGA-II, the reference algorithm in multi-objective optimization. Both the original and the steady-state versions have been evaluated using a benchmark composed of 21 bi-objective problems for comparing the performance of the algorithms in terms of the quality of the obtained solutions sets and their converging speed towards the optimal Pareto front. We have compared the two versions with a set of four state-of-the-art multi-objective optimizers (SPEA2, AbYSS, MOCell, and OMOPSO) to have an insight on the search improvements of the steady-state scheme in NSGA-II.

The obtained results have shown that, in the context of the problems, with the quality indicators and the parameter settings considered, the use of a steady-state scheme has improved the results obtained by the generational NSGA-II in most of the problems. Furthermore, it has also shown to be very competitive taking account of the quality of the obtained approximation sets and the convergence speed of the other state-of-the-art algorithms.

Some future research topics along this line are related to the study and application of steady-state scheme to other multi-objective algorithms and to solve benchmarks composed of rotated problems and with more than two objectives.

Acknowledgements. Authors acknowledge funds from the "Consejería de Innovación, Ciencia y Empresa", Junta de Andalucía under contract P07-TIC-03044 DIRICOM project, `http://diricom.lcc.uma.es`.

Juan J. Durillo is supported by grant AP-2006-003349 from the Spanish Ministry of Education and Science.

References

1. Chafekar, D., Xuan, J., Rasheed, K.: Constrained Multi-objective Optimization Using Steady State Genetic Algorithms. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 813–824. Springer, Heidelberg (2003)
2. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
4. Deb, K., Mohan, M., Mishra, S.: Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 222–236. Springer, Heidelberg (2003)
5. Deb, K., Mohan, M., Mishar, S.: Evaluating the ε-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. Evolutionary Computation 13(4), 501–525 (2005)
6. 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, pp. 105–145. Springer, Heidelberg (2005)
7. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. 7, 1–30 (2006)

8. Durillo, J.J., Nebro, A.J., Luna, F., Dorronsoro, B., Alba, E.: jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Tech. Rep. ITI-2006-10, Dept. de Lenguajes y Ciencias de la Computación, University of Málaga (2006)
9. Durillo, J.J., Nebro, A.J., Luna, F., Alba, E.: A Study of Master-Slave Approaches to Parallelize NSGA-II. In: IEEE Iinternational Symposium on Parallel and Distributed Processing - IPDPS 2008, pp. 1–8 (2008)
10. Emmerich, M., Beume, N., Naujoks, B.: An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)
11. Glover, F.W., Kochenberger, G.A.: Handbook of Metaheuristics. Kluwer Academic Publishers, Dordrecht (2003)
12. Huband, S., Hingston, P., Barone, L., While, R.L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans Evolutionary Computation 10(5), 477–506 (2006)
13. Igel, C., Suttorp, T., Hansen, N.: Steady-state Selection and Efficient Covariance Matrix Update in the Multi-objective CMA-ES. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 171–185. Springer, Heidelberg (2007)
14. Knowles, J., Thiele, L., Zitzler, E.: A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Tech. Rep. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2006)
15. 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), 283–314 (2002)
16. Nebro, A.J., Durillo, J.J., Luna, F., Dorronsoro, B., Alba, E.: Design issues in a multiobjective cellular genetic algorithm. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 126–140. Springer, Heidelberg (2007)
17. Nebro, A.J., Durillo, J.J., Coello Coello, C.A., Luna, F., Alba, E.: A Study of Convergence Speed in Multi-objective Metaheuristics. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 171–185. Springer, Heidelberg (2008)
18. Nebro, A.J., Luna, F., Alba, E., Dorronsoro, B., Durillo, J.J., Beham, A.: AbYSS: Adapting Scatter Search to Multiobjective Optimization. IEEE Transactions on Evolutionary Computation 12(4), 439–457 (2008)
19. Reyes-Sierra, M., Coello Coello, C.A.: Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 509–519. Springer, Heidelberg (2005)
20. Reyes-Sierra, M., Coello Coello, C.A.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)
21. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Grefenstette, J.J. (ed.) Proceedings of the 1st International Conference on Genetic Algorithms, pp. 93–100 (1985)
22. Srinivasan, D., Rachmawati, L.: An Efficient Multi-objective Evolutionary Algorithm with Steady-State Replacement Model. In: Genetic and Evolutionary Computation - GECCO 2006, pp. 715–722 (2006)

23. Valenzuela, C.L.: A Simple Evolutionary Algorithm for Multi-Objective Optimization (SEAMO). In: IEEE Congress on Evolutionary Computation - CEC 2002, pp. 717–722 (2002)
24. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
25. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
26. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)
27. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7, 117–132 (2003)