VarMOPSO: Multi-Objective Particle Swarm Optimization with Variable Population Size

Javier López, Laura Lanzarini, and Armando De Giusti

III-LIDI(Instituto de Investigación en Informática LIDI)
School of Computer Sciences. National University of La Plata
La Plata, Buenos Aires, Argentina

Abstract. The PSO (Particle Swarm Optimization) metaheuristics, originally defined for solving single-objective problems, has been applied to multi-objective problems with very good results. In its initial conception, the algorithm has a fixed-size population. In this paper, a new variation of this metaheuristics, called VarMOPSO (Variable Multi-Objective Particle Swarm Optimization), characterized by a variable-sized population, is proposed. To this end, the concepts of age and neighborhood are incorporated to be able to modify the size of the population for the different generations. This algorithm was compared with the version that uses fixed-size populations, as well as with other metaheuristics, all of them representative of the state of the art in multi-objective optimization. In all cases, three widely used metrics were considered as quality indicators for Pareto front. The results obtained were satisfactory.

Keywords: Evolutionary Computation, Swarm Intelligence, Particle Swarm Optimization, Multi-objective Function Optimization.

1 Introduction

Problem optimization is a very frequent problem in real life. We are constantly faced with the need of finding solutions that maximize the performance or minimize the consumption of any given scarce resource. There is also a set of real-life problems, known as multiobjective problems, that have more than one objective to maximize or minimize simultaneously. Usually, there is a trade off between these objectives, and a set of compromise solutions to be used during the decision-making process is required [1].

Particle Swarm Optimization (PSO) is a metaheuristic proposed by Kennedy and Eberhart [2], and there are more than 30 variations that allow applying it to multi-objective problems. These variations are generally referred to as MOPSO (Multi-objective Particle Swarm Optimization). Among these, SMPSO and OMOPSO show the best perfomance [3].

In this paper, a new variation of a MOPSO-type algorithm, called VarMOPSO, incorporating the concept of variable population, is proposed. The same as the solution adopted in [5], this algorithm uses an external archive to maintain non-dominated solutions. To calculate particle speed, the velocity constriction equation proposed in [6], based on the constriction factor in [7], is used. Population

A. Kuri-Morales and G.R. Simari (Eds.): IBERAMIA 2010, LNAI 6433, pp. 60–69, 2010.

variation between generations is the result of adding the concept of age to the particles and using an insertion/removal procedure based on the ability of each individual to solve the problem posed. The concept of elitism is used, and only dominated solutions can be removed. These procedures are based on the mono-objective PSO version presented in [8].

It is known that, for PSO, the selection of the leader is crucial when calculating the speed of the particles [9][15][17]. The method proposed here uses a different selection mechanism between the two phases of the algorithm. During the first part of the search, two leaders are selected at random from the external, non-dominated solutions archive, and a binary tournament process based on their crowding value is carried out, in accordance to the mechanism proposed in [5]. During the final part of the process, the previously described mechanism is used 50% of the times, whereas the remaining 50%, the leader with the less crowding value is deterministically selected in order to find new solutions in the less populated areas of Pareto front.

To assess the performance of the method proposed, comparisons with NSGAII [10], SPEA2 [11], and SMPSO [6] algorithms were carried out; the three are representative of the state of the art in multi-objective optimization. To carry out the experiments and obtain the results, the jMetal framework implementation [12] of these algorithms was used, with the added implementation of the method proposed.

This paper is organized as follows: in Section 2, the main concepts related to multi-objective optimization and the main aspects of the PSO metaheuristic are reviewed, a well as the extensions developed for multi-objective optimization. In Section 3, the proposed algorithm is described in detail. Then, in Section 4, detailed information is provided regarding the test functions used and the configuration of each algorithm. Finally, in Section 5, the conclusions and future lines of work are presented.

2 Multi-objective Optimization with PSO

A multi-objective problem implies the simultaneous optimization of several objective functions which are generally at conflict. In this type of problems, the solution consists in a set of optimum solutions. The reason for creating a set of solutions is that no single solution can be considered to be better than the rest when considering all the objective functions, since the best solution for one of the objective functions can be worse for another function.

In this context, Pareto's concept of dominance is used. Given a population S, the set of optimum solutions is defined as the subset of S that is formed by all individuals of S that are non-dominated. In the solutions space, it is known as Pareto set, and its image in the space of the objective functions is the Pareto front.

The main purpose of a multi-objective optimization algorithm is finding candidate solutions whose image is as close to the real Pareto front as possible, are evenly distributed and maximize its coverage [14].

PSO is a population heuristics technique where each individual represents a possible solution to the problem and adapts following three factors: its knowledge of the environment (its fitness value), its historical knowledge or previous experiences (its memory), and the historical knowledge or previous experiences of the individuals in its neighborhood (leader) [2].

The PSO metaheuristic has also been used to solve multi-objective problems with good results [6][3][13].

In an MOPSO, the selection of the leader greatly influences both the convergence of the algorithm with Pareto front and the diversity of the solutions found [9]. Elitism is necessary to converge to the Pareto Optimal Front [19]. In most MOPSO implementations, the non-dominated solutions file is a fixed-size external archive. There are different methods to select the non-dominated solutions to be removed when their number exceeds the preset size [3][4][9][19]. When the execution of the algorithm ends, this external archive contains the set of solutions that form the Pareto front obtained. Ideally, this archive is a subset of solutions of the real Pareto front for the optimized problem.

3 varMOPSO: Variable Multi-Objetive Particle Swarm Optimization

The main contribution of our proposal is focused on the variation of the size of the population. This allows the algorithm to adapt the number of individuals to use in both phases, exploration and exploitation, based on the difficulty of the problem. During the exploration phase, which is the first phase of the algorithm, corresponding to the search of promising regions of the parameter space, the population grows until reaching 6 or 7 times the size of the initial population, depending on the problem and the execution. Then, as iterations go by, this number of individuals is gradually reduced until a minimum preset number is reached, which marks the exploitation phase of the algorithm. In the following paragraphs, the concepts and procedures incorporated to the algorithm to change the population size are described.

3.1 Life Time

The particle's life time is one of the main concepts in this work. It defines the permanence of the particle within the population. Such value is expressed in quantity of iterations; once it is over, the particle is removed unless it belongs to the set of non-dominated solutions for the population. This value is closely related to the capacity of each particle and allows the best to remain longer in the population, influencing the behavior of the others.

In order to assess the life time of each population individual, the individuals of a population are grouped according to their rank value in k classes. The rank value indicates the non-domination level of the individual in the population [10]. With the result of this grouping, the method proposed in [8] is applied. For this procedure, elitism is used. The solution is removed from the population only if

it is not a non-dominated solution. This procedure is executed while population size is greater than a minimum, preset value.

3.2 Particle Insertion

The particle insertion procedure compensates the particle elimination, increases the search capacity and the convergence speed. The quantity of incorporated particles at each iteration coincides with the quantity of isolated individuals. An isolated individual is an individual that does not have any neighbor within a calculated r radius, following the criterion adopted in [8].

$$d_i = min\{||x_i – x_j||; \forall j \quad x_i, x_j \in S; x_i \neq x_j\} \; i = 1..n \quad ; \quad r = \sum_{i=1}^{n} d_i/n \qquad (1)$$

The adopted criterion to determine the position of these new individuals was the following: 20% of these new particles receive a position vector of one of the best individuals of the population, but their speed vector is random; the remaining 80% are random, as assessed in [8]. To select the best individual of the population, two particles are randomly selected from the external archive and a binary tournament is carried out. We choose the solution that was located in a less-crowded region within the objective function space [10].

To prevent population size to grow excessively, there is a variable c that regulates population growth. This variable reflects the probability of creating a new particle. At the beginning of the algorithm, the value of this variable is 1.0. If the size of the population grows more than 5% between generations, the value of variable c is decreased by 0.1. When the variable reaches a value of 0, it keeps that value for the remainder of the execution. Thus, the algorithm increases the size of the population during the first iterations, reaching a maximum value. Once the variable c reaches a value of 0, the size of the population decreases gradually until reaching the minimum preset number of individuals.

3.3 Particle Removal

This procedure has the purpose of removing those particles that are in overpopulated areas within the parameter space.

First, the number of nearby neighbors for each particle is calculated. This number of nearby neighbors corresponds to the number of particles that surround an individual within a radius r_1, as described by the following equation:

$$r_1 = \sum_{i=1}^{n} (r - ||x_i – x_j||)/n \quad ; \quad \forall j \quad x_i, x_j \in S \; ; \; x_i \neq x_j \quad i = 1..n \qquad (2)$$

All particles with more than 0 nearby neighbors are candidates for removal.

For each individual, an aptitude value normalized between 0 and 1 is calculated considering two factors – the aptitude of the particle to solve the problem and the density of individuals surrounding each solution.

The aptitude value ap of each particle is calculated with the following equation.

$$a_i = (max(ra) – ra_i)/(max(ra) - min(ra)) \quad ; \quad p_i = (pr_i/max(pr))$$

$$ap_i = 0.5 * a_i + 0.5 * p_i \quad ; \quad i = 1..n \tag{3}$$

where

- $max(ra)$ = maximum dominance level in the subpopulation of particles that are candidates for removal;
- ra_i = dominance level of each particle;
- $max(pr)$ = maximum number of neighbors in the subpopulation of particles that are candidates for removal;
- pr_i = number of nearby neighbors of each particle.

The previous calculation always returns a value between 0 and 1. As it can be inferred, the aptitude value depends 50% on the ranking of the particle (non-domination level within the population) and 50% on the number of neighbors within the previously described radius r_1.

Then, for each particle, a random value between 0 and 0.50 is selected. If the aptitude value is lower than this value, the particle is removed; otherwise, it remains in the population. The probability of removal of any given particle is inversely proportional to the calculated aptitude value.

For this procedure, elitism is used. The solution is removed from the population only if it is not a non-dominated solution. This procedure is executed while population size is greater than a minimum, preset value.

Figure 1 shows a detail of the pseudo-code for the proposed algorithm.

```
Pop = CreatePopulation(N) {Initial population is generated}
ComputeFitness(Pop);
ComputeRank(Pop) {computes the nondomination level of each particle}
ComputeLifeTimes(Pop);
AddArchiveLeaders (Ext);
ComputeRadius(Pop, cant, Eliminados) {first time Eliminados is empty}
while no end condition is reached do
   NewPop = CreateNewParticles(Pop, cant);
   ComputeLifeTimes(NewPop)
   Pop = Pop ∪ NewPop;
   ComputeSpeed(Pop);
   ComputeNewPosition(Pop);
   ComputeFitness(Pop);
   AddArchiveLeaders(Ext);
   ComputeRank(Pop);
   Deduct 1 to each particles life time
   Remove the particles with null life time
   ComputeRadius(Pop, cant, Eliminados);
   EliminCrow(Eliminados);
   ComputeLifeTimes(Pop);
end while
Output : Ext
```

Fig. 1. Pseudo-code for the proposed method

The *ComputeLifeTimes* process receives a complete swarm and only computes the life time corresponding to the particles that have null life time.

The *AddArchiveLeaders* adds non-dominated solutions to the external archive, and eliminates dominated solutions from it. If the external archive size is greater than a fixed value, this function uses the crowding distance of NSGA-II [10] to decide which particles should remain in it.

The radius computation, according to equation (1) and (2), is carried out within the *ComputeRadius* process, which receives the complete swarm as parameter and returns the quantity of new particles that should be inserted in the population. This module is the one in charge of avoiding the concentration of several particles in the same place of the search space; for this reason, it also returns, in *Eliminados*, the list of individuals that have really close neighbors. Such particles are eliminated in the *EliminCrow* module, based on their normalized fitness calculated according to (3).

4 Experiments

To assess the performance of varMOPSO, the ZDT (Zitzler-Deb-Thiele) [18] and DTLZ (Deb-Thiele-Laumanns-Zitzler) [20] sets of functions, configured with 2 objectives, were used. The experiments were carried out for the NSGA II, SPEA2, SMPSO and varMOPSO algorithms. The following indicators were calculated: additive unary epsilon indicator, spread, and hypervolume [1][14][16], to compare the quality of the Pareto fronts obtained. These indicators are representative to measure the general convergence to the real Pareto front, the coverage of the Pareto front obtained, and the uniformity in the distribution of solutions reached[14] [16]. To develop the algorithm, as well as to carry out the experiment and generate the results, jMetal Framework [12] was used, given its ease of use and its good documentation.

Table 1. Hipervolumen. Median and Interquartile

	NSGAII	SPEA2	varMOPSO	SMPSO
ZDT1	$6.59e-01_{3.8e-04}$	$6.60e-01_{3.7e-04}$	$6.62e-01_{3.3e-04}$	$6.62e-01_{1.1e-04}$
ZDT2	$3.26e-01_{2.7e-04}$	$3.26e-01_{6.7e-04}$	$3.29e-01_{2.5e-01}$	$3.29e-01_{1.6e-04}$
ZDT3	$5.15e-01_{2.0e-04}$	$5.14e-01_{4.9e-04}$	$5.16e-01_{3.0e-04}$	$5.15e-01_{6.1e-04}$
ZDT4	$6.54e-01_{6.5e-03}$	$6.51e-01_{7.4e-03}$	$6.62e-01_{4.0e-04}$	$6.61e-01_{2.4e-04}$
ZDT6	$3.89e-01_{1.9e-03}$	$3.79e-01_{4.6e-03}$	$4.01e-01_{1.2e-04}$	$4.01e-01_{1.2e-04}$
DTLZ1	$4.87e-01_{7.8e-03}$	$4.85e-01_{5.7e-03}$	$4.95e-01_{2.0e-04}$	$4.94e-01_{2.9e-04}$
DTLZ2	$2.11e-01_{3.1e-04}$	$2.12e-01_{1.8e-04}$	$2.12e-01_{2.6e-04}$	$2.12e-01_{1.7e-04}$
DTLZ3	$0.00e+00_{3.8e-02}$	$0.00e+00_{0.0e+00}$	$2.12e-01_{1.6e-04}$	$2.12e-01_{1.3e-01}$
DTLZ4	$2.09e-01_{2.1e-01}$	$2.10e-01_{2.1e-01}$	$2.10e-01_{2.6e-04}$	$2.10e-01_{1.2e-04}$
DTLZ5	$2.11e-01_{3.3e-04}$	$2.12e-01_{1.5e-04}$	$2.12e-01_{2.8e-04}$	$2.12e-01_{1.5e-04}$
DTLZ6	$1.73e-01_{3.9e-02}$	$8.12e-03_{1.2e-02}$	$2.12e-01_{2.1e-01}$	$2.12e-01_{6.9e-05}$
DTLZ7	$3.33e-01_{2.3e-04}$	$3.34e-01_{2.7e-04}$	$3.34e-01_{5.4e-05}$	$3.34e-01_{1.1e-04}$

Table 2. Statistical Significance for Hypervolume in ZDT Functions

	SPEA2					varMOPSO					SMPSO				
NSGAII	▽	–	▲	–	▲	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
SPEA2						▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
varMOPSO											–	–	▲	▲	–

Table 3. Statistical Significance for Hypervolume in DTLZ Functions

	SPEA2							varMOPSO							SMPSO						
NSGAII	–	▽	–	▽	▽	▲	▽	▽	▽	▽	▽	▽	–	▽	▽	▽	▽	▽	▽	▽	▽
SPEA2								▽	▽	▽	▽	▽	–	▽	▽	▽	▽	▽	▽	▽	▽
varMOPSO															▲	–	▲	▲	▲	▽	▲

Table 4. SPREAD. Median and IQR

	NSGAII	SPEA2	varMOPSO	SMPSO
ZDT1	$3.69e-01_{5.8e-02}$	$1.50e-01_{2.3e-02}$	$8.32e-02_{2.5e-02}$	$7.83e-02_{1.2e-02}$
ZDT2	$3.71e-01_{3.1e-02}$	$1.56e-01_{1.8e-02}$	$8.10e-02_{3.8e-01}$	$7.53e-02_{1.9e-02}$
ZDT3	$7.47e-01_{2.6e-02}$	$7.08e-01_{8.6e-03}$	$7.09e-01_{1.2e-02}$	$7.09e-01_{9.8e-03}$
ZDT4	$3.93e-01_{4.4e-02}$	$2.82e-01_{1.1e-01}$	$8.50e-02_{1.3e-02}$	$9.04e-02_{1.5e-02}$
ZDT6	$3.64e-01_{4.9e-02}$	$2.26e-01_{2.1e-02}$	$9.77e-02_{6.8e-01}$	$8.80e-02_{8.9e-01}$
DTLZ1	$3.93e-01_{5.7e-02}$	$1.84e-01_{7.0e-02}$	$6.82e-02_{1.3e-02}$	$7.29e-02_{2.3e-02}$
DTLZ2	$3.74e-01_{5.3e-02}$	$1.51e-01_{1.9e-02}$	$1.30e-01_{1.4e-02}$	$1.25e-01_{2.2e-02}$
DTLZ3	$8.95e-01_{1.5e-01}$	$1.05e+00_{1.6e-01}$	$1.13e-01_{2.5e-02}$	$1.41e-01_{7.7e-01}$
DTLZ4	$3.97e-01_{6.2e-01}$	$1.54e-01_{8.6e-01}$	$1.14e-01_{3.7e-02}$	$1.23e-01_{2.7e-02}$
DTLZ5	$3.80e-01_{4.1e-02}$	$1.52e-01_{2.0e-02}$	$1.30e-01_{2.6e-02}$	$1.28e-01_{1.2e-02}$
DTLZ6	$8.67e-01_{3.6e-01}$	$8.16e-01_{1.2e-01}$	$1.09e-01_{3.6e-01}$	$1.01e-01_{2.7e-02}$
DTLZ7	$6.31e-01_{2.3e-02}$	$5.47e-01_{1.2e-02}$	$5.19e-01_{1.3e-03}$	$5.19e-01_{2.4e-03}$

Table 5. Statistical Significance for Spread in ZDT Functions

	SPEA2					varMOPSO					SMPSO				
NSGAII	▽	▽	▽	▽	▽	▽	▽	▽	▽	–	▽	▽	▽	▽	▽
SPEA2						▽	▽	–	▽	–	▽	▽	–	▽	–
varMOPSO											–	–	–	–	–

Table 6. Statistical Significance for Spread in DTLZ Functions

	SPEA2							varMOPSO							SMPSO						
NSGAII	▽	▽	▲	▽	▽	–	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
SPEA2								▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
varMOPSO															–	–	▲	–	–	–	–

For the first three algorithms, a fixed population of 100 individuals was used. In the case of varMOPSO, the initial population included 100 particles. The size of the external archive was 100 individuals in all cases.

In NSGA II [10] and SPEA2 [11] SBX and polynomial mutation were used as operators for the crossover and mutation operators, respectively. The crossover probability is 0.9 and the mutation probability is 1/L, where L is the number of

Table 7. EPSILON. Median and IQR

	NSGAII	SPEA2	varMOPSO	SMPSO
ZDT1	$1.26e-02_{2.1e-03}$	$8.88e-03_{6.3e-04}$	$5.76e-03_{4.7e-04}$	$5.60e-03_{1.5e-04}$
ZDT2	$1.32e-02_{2.6e-03}$	$9.04e-03_{1.2e-03}$	$5.63e-03_{5.4e-01}$	$5.58e-03_{2.9e-04}$
ZDT3	$7.98e-03_{1.8e-03}$	$9.90e-03_{2.5e-03}$	$5.45e-03_{9.1e-04}$	$5.73e-03_{1.2e-03}$
ZDT4	$1.62e-02_{3.9e-03}$	$3.47e-02_{5.6e-02}$	$5.93e-03_{6.0e-04}$	$6.13e-03_{5.8e-04}$
ZDT6	$1.43e-02_{2.5e-03}$	$2.42e-02_{5.0e-03}$	$4.76e-03_{4.2e-04}$	$4.68e-03_{3.3e-04}$
DTLZ1	$7.69e-03_{2.3e-03}$	$6.14e-03_{2.8e-03}$	$2.91e-03_{2.1e-04}$	$3.03e-03_{1.8e-04}$
DTLZ2	$1.18e-02_{3.3e-03}$	$6.92e-03_{1.1e-03}$	$5.05e-03_{3.1e-04}$	$5.28e-03_{1.7e-04}$
DTLZ3	$9.34e-01_{1.5e+00}$	$2.36e+00_{1.2e+00}$	$5.34e-03_{3.6e-04}$	$5.51e-03_{7.0e-01}$
DTLZ4	$1.08e-02_{9.9e-01}$	$8.60e-03_{9.9e-01}$	$5.42e-03_{4.3e-04}$	$5.39e-03_{3.7e-04}$
DTLZ5	$1.06e-02_{2.1e-03}$	$7.68e-03_{1.3e-03}$	$5.06e-03_{3.8e-04}$	$5.22e-03_{3.1e-04}$
DTLZ6	$4.39e-02_{3.0e-02}$	$3.07e-01_{5.6e-02}$	$5.44e-03_{8.2e-01}$	$5.10e-03_{5.4e-04}$
DTLZ7	$1.03e-02_{2.1e-03}$	$9.52e-03_{2.1e-03}$	$5.03e-03_{4.0e-04}$	$5.09e-03_{4.7e-04}$

Table 8. Statistical Significance for Epsilon in ZDT Functions

	SPEA2					varMOPSO					SMPSO				
NSGAII	▽	▽	▲	▲	▲	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
SPEA2						▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
varMOPSO											▽	–	–	–	–

Table 9. Statistical Significance for Epsilon in DTLZ Functions

	SPEA2							varMOPSO							SMPSO						
NSGAII	▽	▽	▲	–	▽	▲	–	▽	▽	▽	▽	▽	–	▽	▽	▽	▽	▽	▽	▽	▽
SPEA2								▽	▽	▽	▽	▽	–	▽	▽	▽	▽	▽	▽	▽	▽
varMOPSO															–	▲	▲	–	▲	▽	–

decision variables. In SMPSO [6], polynomial mutation was used. The mutation probability is 1/L where L is the number of decision variables. In varMOPSO, a minimum population of 10 individuals and k=4 classes for calculating life time were used.

In every case, 25,000 function assessments were used, and each algorithm was executed 25 times for each problem.

Table 1 sows the results of the hypervolume indicator for all the algorithms in all the problems. The table includes median and interquartile range (IQR) values. The best results are highlighted in dark gray, and the second best results in light gray. Tables 2 and 3 indicate if the results are statistically significant for the two groups of test functions.

In each table, a ▽ symbol implies a p-value < 0.05, indicating that the null hypothesis (both distributions have the same median) is rejected; otherwise, a ▲ or − are used. Similarly, Tables 4, 5 and 6 show the same information for the Spread indicator, and Tables 7, 8 and 9 show the Median and interquartile range of the (additive) Epsilon (I_ε) indicator.

5 Conclusions and Future Work

After an analysis of the results shown in the previous tables, the following conclusions can be drawn:

- Both MOPSO algorithms (SMPSO and varMOPSO) are capable of generating better-quality Pareto fronts, considering the three metrics used and the set of selected tests, than the NSGA II and SPEA2 algorithms.
- No significant differences are found between the performance of the SMPSO and varMOPSO algorithms. In some functions and indicators, the first algorithm presents the best results, whereas in others, the second algorithm offers a better performance; although the differences are not statistically significant considering a p-value < 0.05 (except for the Hypervolume and Epsilon indicators in the DTLZ6 function, where SMPSO shows better results).
- In general, varMOPSO shows a greater dispersion than SMPSO, but most of the time the former got the best individual result for the evaluated metrics in the different test functions.

Based on the discussion presented in this paper, we consider that our proposal is an interesting option for solving multi-objective problems. The method is based on an extension of the varPSO algorithm for multi-objective problems, incorporating improvements studied in state-of-the-art MOPSO algorithms.

The use of variable population algorithms is a rather unexplored alternative for multi-objective optimization. By changing the size of the population, a better adaptation to the problem at hand is achieved, and the possibilities of finding a balance between exploration and exploitation increase, which is desirable in any optimization algorithm. We believe that there is still much that can be done in the field of variable population MOEAs (multiobjective evolutionary algorithms).

References

1. Zitzler, E., Laumanns, M., Bleuler, S.: A Tutorial on Evolutionary Multiobjective Optimization, Swiss Federal Institute of Technology (ETH) Zurich. In: GECCO 2009 (2009)
2. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, vol. IV, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
3. Durillo, J.J., García Nieto, J., Nebro, A.J., Coello Coello, C.A., Luna, F., Alba, E.: Multi-objective particle swarm optimizers: An experimental comparison. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 495–509. Springer, Heidelberg (2009)
4. Raquel, C., Naval, P.: An effective use of crowding distance in multiobjective particle swarm optimization. In: Beyer, H. (ed.) 2005 Conference on Genetic and Evolutionary Computation, GECCO 2005, pp. 257–264. ACM, New York (2005)
5. Sierra, R., Coello, C.: Improving PSO-Based Multiobjective Optimization Using Crowding, Mutation and *epsilon*-Dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005)

6. Nebro, A.J., Durillo, J.J., García-Nieto, J., Coello Coello, C.A., Luna, F., Alba, E.: SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization. In: IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009), pp. 66–73 (March 2009)
7. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
8. Lanzarini, L., Leza, V., De Giusti, A.: Particle Swarm Optimization with Variable Population Size. In: 9th International Conference on Artificial Intelligence and Soft Computing Zakopane, Poland, pp. 438–449 (2008) ISBN: 978-3-540-69572-1
9. Sierra, R., Coello, C.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)
10. 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)
11. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Giannakoglou, K., et al. (eds.) EUROGEN 2001, Athens, Greece, pp. 95–100 (2001)
12. Durillo, J.J., Nebro, A.J., Luna, F., Dorronsoro, B., Alba, E.: jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Technical Report ITI-2006-10, Depto.de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos (December 2006)
13. Abido: Two-Level of Nondominated Solutions Approach to Multiobjective Particle Swarm Optimization. In: Genetic And Evolutionary Computation Conference Proceedings, pp. 726–733 (2007) ISBN:978-1-59593-697-4
14. Zitzler: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD Thesis, Swiss Federal Institute of Technology (ETH) Zurich (November 1999)
15. Shi, E.: Parameter Selection in Particle Swarm Optimization. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998)
16. Knowles, J.D., Tiele, S., Zitzler, E.: A Tutorial on the Perfomance Assessment of Stochastic Multiobjective Optimizers, TIK - Report 214, ETH Zurich (2006)
17. Van den Bergh: An Analysis of Particle Swarm Optimizers. Ph.D. dissertation. Department Computer Science. University Pretoria, South Africa (2002)
18. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
19. Mostaghim, S., Teich, J.: Covering Pareto optimal Fronts by Subswarms in Multiobjective Particle Swarm Optimization. In: Congress on Evolutionary Computation (2), 1404 (2004)
20. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Theoretical Advances and Applications, pp. 105–145. Springer, Heidelberg (2001)