Micro-MOPSO: A Multi-Objective Particle Swarm Optimizer that Uses a Very Small Population Size

Juan Carlos Fuentes Cabrera and Carlos A. Coello Coello*

CINVESTAV-IPN (Evolutionary Computation Group), Computer Science Department, México
jfuentes@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

Summary. In this chapter, we present a multi-objective evolutionary algorithm (MOEA) based on the heuristic called "particle swarm optimization" (PSO). This multi-objective particle swarm optimizer (MOPSO) is characterized for using a very small population size, which allows it to require a very low number of objective function evaluations (only 3000 per run) to produce reasonably good approximations of the Pareto front of problems of moderate dimensionality. The proposed approach first selects the leader and then selects the neighborhood for integrating the swarm. The leader selection scheme adopted is based on Pareto dominance and uses a neighbors density estimator. Additionally, the proposed approach performs a reinitialization process for preserving diversity and uses two external archives: one for storing the solutions that the algorithm finds during the search process and another for storing the final solutions obtained. Furthermore, a mutation operator is incorporated to improve the exploratory capabilities of the algorithm. The proposed approach is validated using standard test functions and performance measures reported in the specialized literature. Our results are compared with respect to those generated by the Nondominated Sorting Genetic Algorithm II (NSGA-II), which is a MOEA representative of the state-of-the-art in the area.

1 Introduction

The Particle Swarm Optimization (PSO) algorithm is a relatively recent heuristic based on the simulation of social behavior of birds within a flock [9]. Alhough PSO was originally proposed for balancing weights in neural networks, over the years, it has become a popular optimizer in a wide variety of disciplines [14]. In the last few years, a variety of proposals for extending the PSO algorithm to handle multiple objectives have appeared in the specialized literature [27].

*The second author is also associated to the UMI-LAFMIA 3175 CNRS.

This chapter precisely proposes a new multi-objective particle swarm optimizer (MOPSO), called micro-MOPSO because of the very small population size that it adopts. Such a small population size combined with a good mechanism to preserve diversity allows us to produce reasonably good approximations of the Pareto front of several test problems of moderate dimensionality (up to 30 decision variables), while performing only 3,000 objective function evaluations. To the best of the authors' knowledge, this is the first micro-MOPSO ever proposed, and its main aim is to serve as a good alternative for applications in which only a very small implementation can be stored (e.g., when the multi-objective evolutionary algorithm needs to be placed into a microcontroller) or when only a low number of objective function evaluations can be afforded (e.g., in aeronautical engineering problems).

The organization of the rest of the chapter is the following. In Section 2, we define the problem of our interest. The Particle Swarm Optimization algorithm is introduced in Section 3. The previous related work is provided in Section 4. Section 5 describes our approach including the leader selection mechanism, the neighbors selection mechanism, the reinitialization process, and the mutation operator adopted. In Section 6, we present the performance measures, our experimental setup and the results obtained. Finally, Section 7 presents our conclusions and some possible paths for future research.

2 Basic Concepts

We are interested in solving problems of the type:

$$\text{Find } \mathbf{x} \text{ wich optimizes } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \tag{1}$$

subject to:

$$g_i(\mathbf{x}) \leq 0 \quad i = 1, 2, \ldots, n \tag{2}$$

$$h_i(\mathbf{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables, f_i, $i = 1, ..., k$ are the objective functions and g_i, h_j, $i = 1, ..., m$, $j = 1, ..., p$ are the constraint functions of the problem.

In multi-objective optimization problems the aim is to find good compromises (trade-offs). To understand the concept of optimality, we will introduce first a few definitions.

Definition 1. Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$, we say that $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for $i = 1, ..., k$, and that $\mathbf{x}$ **dominates** $\mathbf{y}$ (denoted by $\mathbf{x} \prec \mathbf{y}$) if $\mathbf{x} \leq \mathbf{y}$ and $\mathbf{x} \neq \mathbf{y}$.

Definition 2. We say that a vector of decision variables $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ is **nondominated** with respect to $\mathcal{X}$, if there does not exist another $\mathbf{x}' \in \mathcal{X}$ such that $\mathbf{f}(\mathbf{x}') \prec \mathbf{f}(\mathbf{x})$.

Definition 3. We say that a vector of decision variables $\mathbf{x}^* \in \mathcal{F} \subset \mathbb{R}^n$ ($\mathcal{F}$ is the feasible region) is **Pareto optimal** if it is nondominated with respect to $\mathcal{F}$.

Definition 4. The **Pareto Optimal Set** $\mathcal{P}^*$ is defined by:

$$\mathcal{P}^* = \{\mathbf{x} \in \mathcal{F} | \mathbf{x} \text{ is Pareto optimal}\}$$

Definition 5. The **Pareto Front** $\mathcal{PF}^*$ is defined by:

$$\mathcal{PF}^* = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^k | \mathbf{x} \in \mathcal{P}^*\}$$

We thus wish to determine the Pareto optimal set from the set $\mathcal{F}$ of all the decision variable vectors that satisfy (2) and (3).

In general, when solving a multi-objective optimization problem with an evolutionary algorithm, we aim to produce as many elements of the Pareto optimal set as possible. Additionally, such solutions should be as uniformly distributed along the Pareto front as possible.

In spite of the existence of a variety of algorithms for solving multi-objective optimization problems in the operations research literature, such approaches normally become inefficient and even inappropriate when facing high-dimensional, discontinuous, and highly nonlinear problems [21]. Furthermore, there exist problems with a very large search space, which can also be very difficult to explore because of its shape. In all of these cases, the use of heuristic techniques is fully justified, since they will normally provide a reasonably good approximation of the Pareto optimal set within a reasonable time, although, without guaranteeing convergence [7]. One of such heuristic techniques is PSO, which is described next.

3 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm was introduced by Russell Eberhart and James Kennedy in 1995 [9]. PSO is a population-based search algorithm based on the simulation of social behavior of birds within a flock [14, 10]. In PSO, each individual (particle) of the population (swarm) adjusts its trajectory according to its own flight experience and the flying experience of the other particles within its topological neighborhood in the search space. In the PSO algorithm, the particles' positions and velocities are randomly initialized at the beginning of the search, and then they are iteratively updated,

based on their previous positions and those of each particle's neighbors. Our proposed approach implements equations (4) and (5), proposed in [31] for computing the velocity and the position of a particle.

$$v_{id} = w \times v_{id} + c_1 r_1 (pb_{id} - x_{id}) + c_2 r_2 (lb_{id} - x_{id}) \tag{4}$$

$$x_{id} = x_{id} + v_{id} \tag{5}$$

where c_1 and c_2 are both positive constants, r_1 and r_2 are random numbers generated from a uniform distribution in the range [0,1], and w is the inertia weight that is generated in the range (0,1].

There are two versions of the PSO algorithm: the **global** version and the **local** version. In the global version, the neighborhood consists of all the particles of the swarm and the best particle of the population is called the "global best" (*gbest*). In contrast, in the local version, the neighborhood is a subset of the population and the best particle of the neighborhood is called "local best" (*lbest*).

4 Related Work

In order to extend the PSO algorithm to handle multi-objective optimization problems, the leader selection mechanism must be modified so that it incorporates the concept of Pareto optimality. Additionally, a mechanism for preserving diversity (normally called "density estimator") must be incorporated.

Moore and Chapman [22] introduced the first extension of PSO for handling multi-objective problems (MOPSO) in an unpublished manuscript. Since then, a wide variety of additional proposals have been introduced [12, 6, 23]. In [27], the authors present a survey of this area, in which a taxonomy of MOPSOs is also introduced. Based on that taxonomy, our approach can be classified as a Pareto-based MOPSO. Next, we will review the most representative Pareto-based MOPSOs that have been reported in the specialized literature, since they constitute the previous work related to our proposal.

In [12], Fieldsend and Singh proposed an approach that uses an elite external archive. This approach adopts a special data structure called "dominated tree" for storing the nondominated particles found along the search process. The archive interacts with the population in order to define leaders. This algorithm also adopts a "turbulence" operator that acts on the velocity value.

Coello Coello et al. [6] proposed a MOPSO that uses Pareto dominance to determine the flight direction of a particle. The algorithm uses a global archive for storing the nondominated solutions found during the search and also adopts a mutation operator that generates solutions within ranges of the decision variables that have not been previously used. In more recent work,

Toscano Pulido and Coello Coello [34] proposed another MOPSO that divides the population into several swarms adopting clustering techniques.

Mostaghim and Teich [24] proposed another Pareto-based MOPSO, which adopts the so-called sigma method, in which the best local guides for each particle are adopted to improve the convergence and diversity. A mutation operator (called "turbulence") is also adopted in this case. In further work, Mostaghim and Teich [23] studied the influence of a relaxed form of Pareto dominance called ϵ-dominance [18] on MOPSO methods. In more recent work, Mostaghim and Teich [25] proposed a new method called *covering*MOPSO (cvMOPSO), which works in two phases. In the first phase, a MOPSO algorithm is run with a restricted archive size and the goal is to obtain a good approximation of the Pareto front. In the second phase, the nondominated solutions obtained from the first phase are considered as the input archive of the cvMOPSO. The particles in the population of the cvMOPSO are divided into subswarms around each nondominated solution after the first generation. The task of the subswarms is to cover the gaps between the nondominated solutions obtained from the first phase.

Li in [19] proposed an approach that incorporates the main mechanisms of the NSGA-II [8] into the PSO algorithm. In this algorithm, once a particle has updated its position, all the **pbest** positions of the swarm and all the new positions recently obtained are combined in just one set. Then, the approach selects the best solutions among them. The algorithm also randomly selects the leader based on two mechanisms: a niche count and a crowding distance.

In [3], Bartz-Beielstein et al. proposed an approach that starts from the idea of introducing elitism into a MOPSO, via an external archive. The authors analyzed different methods for selecting and deleting particles from the archive, aiming to generate a satisfactory approximation of the Pareto front.

Several other MOPSOs exist (see for example [38, 26, 1, 32, 35, 28]). However, none of them adopts a small population size, as our proposed approach. The use of small population sizes is unusual in the evolutionary algorithms literature in general, because its use normally speeds up the loss of diversity, and tends to generate premature convergence. However, in the genetic algorithms literature, it is known that the use of very small population sizes is possible, if an appropriate reinitialization process is adopted (such approaches are called micro-genetic algorithms (micro-GAs) [16, 4] and they use population sizes no larger than five individuals). Krishnakumar [16] proposed the first implementation of a micro-GA, and we are only aware of one multi-objective micro-GA, which was introduced in [5, 4]. This approach uses a population size of four individuals, and three forms of elitism: (1) an external archive that adopts the adaptive grid from the Pareto Archived Evolution Strategy (PAES) [15], (2) a population memory, in which randomly generated individuals are replaced by evolved individuals, and (3) a mechanism that retains the two best solutions generated by each run of the micro-GA. In a further paper, Coello Coello and Pulido introduced the micro-GA^2 [33], which avoids the many parameters (eight) required by the original micro-GA for multi-objective op-

timization. The micro-GA^2 uses online adaptation and it performs a parallel strategy to adapt the crossover operator and the type of encoding (binary or real numbers) to be used.

To the authors' best knowledge, the approach described in this chapter is the first MOPSO in using a very small population size (i.e., a micro-MOPSO). The only micro-PSO that we are aware of is the (single-objective) micro-PSO for constrained optimization problems, which was developed by the same authors of this chapter [13].

5 The Micro-MOPSO

Our proposed micro-MOPSO is based on the global version of the PSO algorithm. It uses two external archives: one (called *auxiliary*) for storing the nondominated solutions that the algorithm finds throughout the search and another (called *final*) for storing the final nondominated solutions obtained. Our proposed algorithm performs the nondominated sorting introduced in [8] and uses a crowding distance for selecting leaders. As indicated before, our approach first selects the leader and then selects the neighborhood for creating the swarm. Our micro-MOPSO also uses a reinitialization process for preserving diversity and a mutation operator is incorporated to improve the overall exploratory capabilities of this heuristic (see Algorithm 1). The leader selection mechanism, the external archives together with the reinitialization process and the mutation operator adopted, are all described next in more detail.

Final archive

Since the micro-MOPSO uses a very small population size (only five particles), it needs an external archive for reporting the final solutions that it has found (this is called the *final archive*). Our algorithm uses this archive for selecting leaders (see Section 5.1). The upper bound of the final archive (FAB) is a parameter that needs to be set by the user.

At each iteration, and after updating the particles' positions, the nondominated solutions are obtained from the population of the micro-MOPSO. These solutions enter into the final archive and remain there only if no other solution dominates them during the entire evolutionary process. If a solution dominates another solution stored in the final archive, the stored solution is deleted. When the maximum capacity of the archive is reached, then the algorithm applies the crowding distance [8] as the criterion to prune the contents of the archive.

Auxiliary Archive

The micro-MOPSO uses another external archive called *auxiliary archive*, which is used for storing the solutions that the algorithm finds along the

Algorithm 1: Pseudocode of the Micro-MOPSO

Input: *Number of particles, number of generations, number of reinitialization particles (pr), number of reinitialization generations (gr), auxiliary archive bound (AAB), final archive bound (FAB), mutation rate.*
Output: *Nondominated solutions (final archive)*
begin
  Initialize the final archive (empty);
  Initialize the auxiliary archive (empty);
  for $i = 1$ *to Number of particles* **do**
    Initialize position and velocity randomly;
  end
  Store the swarm into the auxiliary archive;
  Get the set nondominated solutions;
  Store the nondominated solutions into the final archive;
  $cont = 1$;
  repeat
  until *Maximum number of generations* ;
  Select the leader;
  Select the neighborhood;
  if *cont == number of generations for reinitialization* **then**
    Reinitialization process;
    $cont = 1$;
  end
  for $i = 1$ *to Number of particles* **do**
    Update the velocity;
    Compute the actual position;
    if x_i *dominates to* xpb_i **then**
      for $d = 1$ *to Number of dimensions* **do**
        $xpb_{id} = x_{id}$; // Update the **pbest**;
      end
    end
  end
  Performs mutation;
  Store the swarm into the auxiliary archive;
  if *auxiliary archive length* $> AAB$ **then**
    Filter out the auxiliary archive;
  end
  Get the nondominated solutions;
  Store the nondominated solutions into the final archive;
  if *final archive length* $> FAB$ **then**
    Filter out the final archive;
  end
  cont =cont +1;
  Report the nondominated solutions (final archive)
end

search process. Our algorithm uses it for selecting the neighbor in the swarm. The upper bound of the auxiliary archive (AAB) is a parameter that needs to be set by the user.

At each iteration and after updating the particles' positions, the swarm is stored into the auxiliary archive. When the maximum limit imposed on the size of the archive is reached, the algorithm performs nondominated sorting for keeping the solutions located into the first five fronts or the best AAB solutions (see Figure 1).

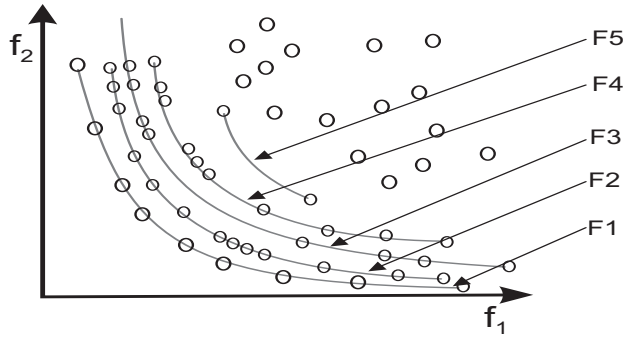
5.1 Leader Selection

As we said before, the micro-MOPSO uses the final archive for selecting leaders. The idea is to have better diversity for the selection process. Algorithm 2 shows the mechanism for selecting a leader. The leader is selected from a subset of the final archive members that have the best crowding distances (i.e., the best spread). The size of the subset of leaders is a percentage (defined by the user) of the total population size.

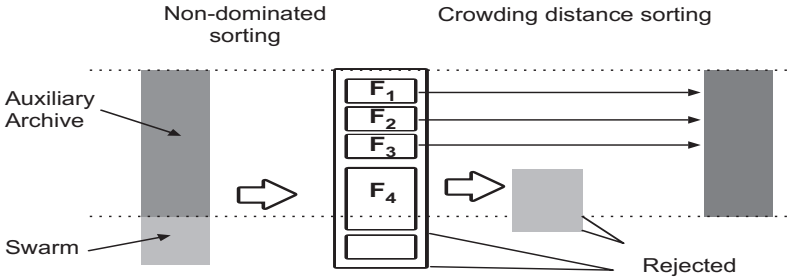
Algorithm 2: Pseudocode of the leader selection mechanism adopted in our micro-MOPSO.

Input: *Final archive FA, final archive maximum limit (FAB), population percentage for selecting leader (PPS).*
Output: *Leader*
begin
  for $i = 1$ *to Final archive length* **do**
    Compute the crowding distance CD_i;
  end
  Sort the final archive members (according to CD);
  Get a percentage of particles (pps) from the population based on their CD values;
  Choose a particle in a random way;
  Make this particle the leader;
end

The use of the crowding distance for selecting leaders allows the micro-MOPSO to select nondominated solutions that are located in less crowded areas of the Pareto front. The use of diversity information allows us to discriminate from among several potential leaders (i.e., all the nondominated solutions produced so far). The potential disadvantage of this selection method is that, in the presence of local Pareto fronts, a good diversity maintenance mechanism is required, in order to avoid stagnation. In our case, our mutation operator is responsible for such diversity maintenance, but other mechanisms could also be adopted (e.g., niching [29]).



Case a)



Case b)

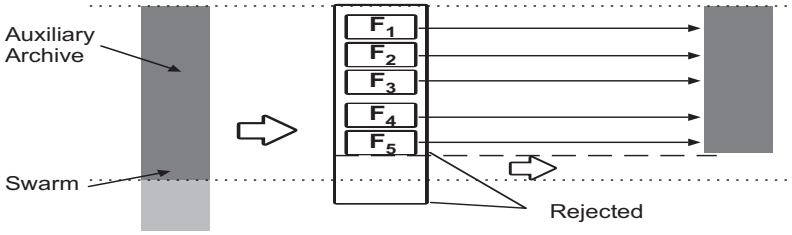


Fig. 1. An example in which the auxiliary archive exceeds its maximum allowable limit. The process finds the particles of the first nondominated front for all the archive members. If the front length is lower than the maximum limit, the front is kept into the archive. Then, in order to find the individuals in the next front, the solutions of the first front are temporarily disregarded and the above procedure is repeated until five fronts are found. Case a) When the front is kept into the archive and it exceeds the allowable limit, a crowding distance is computed (just for the front) in orden to filter out solutions, and the following fronts are deleted. Case b) The micro-MOPSO keeps into the auxiliary archive at most five fronts.

Neighbors selection mechanism

In contrast with others MOPSOs, the micro-MOPSO first selects the leader and then uses the auxiliary archive for selecting the neighborhood for creating the swarm. Algorithm 3 shows the mechanism for selecting the neighbors which have the smallest Euclidian distance.

Algorithm 3: Pseudocode for selecting the neighborhood

```
begin
    Input: Leader gbest, auxiliary archive.
    Output: Neighborhood
    for i = 1 to auxiliary archive length do
        Compute the Euclidean distance (ED) from the ith particle to the
        leader particle (gbest) in objective function space;
    end
    Sort the auxiliary archive (according to the ED values);
    Choose the N − 1 particles closest to gbest;
    Create a swarm with the N − 1 particles chosen plus the leader (gbest);
end
```

This mechanism allows the micro-MOPSO to be explorative at the beginning of the search and exploitative at the end.

5.2 Reinitialization Process

The micro-MOPSO uses a reinitialization process similar to the one proposed in [13]. The mechanism is the following: after certain number of iterations (replacement generations **rg**), the algorithm identifies a certain number of nondominated solutions (**swarm**) and replaces them by randomly generated particles (**rp**). The rationale for mixing evolved and randomly generated particles is to avoid premature convergence.

5.3 Mutation Operator

Although the original PSO algorithm has no mutation operator, the addition of such a mutation operator is a relatively common practice nowadays in the specialized literature. The main motivation for adding this operator is to improve the performance of PSO as an optimizer, and to improve the overall exploratory capabilities of this heuristic [2]. In our proposed approach, we implemented the mutation operator originally developed by Michalewicz for genetic algorithms [20]. It is worth noticing that this mutation operator has been used before in PSO, but in the context of unconstrained multimodal optimization [11]. This operator varies the value added or substracted to a solution during the actual mutation, depending on the current iteration number

(at the beginning of the search, large changes are allowed, and they become very small towards the end of the search). We apply the mutation operator in the particle's position, for all its dimensions:

$$x_{id} = \begin{cases} x_{id} + \Delta(t, UB - x_{id}) & \text{if } R = 0 \\ x_{id} - \Delta(t, x_{id} - LB) & \text{if } R = 1 \end{cases} \tag{6}$$

where t is the current iteration number, UB is the upper bound on the value of the particle's dimension, LB is the lower bound on the particle's dimension, R is a randomly generated bit (zero and one both have a 50% probability of being generated) and $\delta(t, y)$ returns a value in the range $[0, y]$. $\delta(t, y)$ is defined by:

$$\Delta(t, y) = y * (1 - r^{1-(\frac{t}{T})^b}) \tag{7}$$

where r is a random number generated from a uniform distribution in the range [0,1], T is the maximum number of iterations and b is a tunable parameter that defines the non-uniformity level of the operator. In this approach, the b parameter is set to 5 as suggested in [20].

6 Experiments and Results

When an evolutionary algorithm is used for solving multi-objective problems, three issues have to take into consideration for evaluating the performance of the algorithm [39]:

1. Minimize the distance of the Pareto front produced by the algorithm with respect to the true Pareto front.
2. Maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible.
3. Maximize the number of elements of the Pareto optimal set found.

We adopt three performance measures for evaluating the performance of the micro-MOPSO and for comparing it with respect to the NSGA-II that is an algorithm representative of the state-of-the-art in the area.

6.1 Performance Measures

In order to assess the performance of our proposed approach and compare it with respect to the NSGA-II, we adopted three performance measures that have been commonly used in the specialized literature: error ratio, inverted generational distance, and spacing. Each of them is briefly described next.

Error Ratio

The error ratio (ER) performance measure reports the number of vectors in the Pareto front found that are not members of the true Pareto front [36]. This is mathematically represented by:

$$ER = \frac{\sum_{i=1}^{n} e_i}{n} \tag{8}$$

$$e_i = \begin{cases} 1 \text{ if the } i\text{th vector is member of the true Pareto front} \\ 0 \text{ otherwise} \end{cases} \tag{9}$$

Inverted Generational Distance

The generational distance (GD) reports how far, on average, the Pareto front found by an algorithm is from the true Pareto front. Mathematically, it is defined by:

$$GD = \frac{(\sum_{i=1}^{n} d_p^i)^{1/p}}{n} \tag{10}$$

where n is the number of vectors that the algorithm found, for $p = 2$, d_i is the Euclidean distance (in objective space) between each member, i, of the Pareto front found and the closest member in the true Pareto front.

The inverted generational distance (IGD) reports how far, on average, the true Pareto front is from the Pareto front found by an algorithm. This intends to reduce some of the problems that occur with the original generational distance metric when an algorithm generates very few nondominated solutions.

Spacing

The spacing (S) performance measure numerically describes the spread of the vectors in the Pareto front found by the algorithm [30]. It measures the distance variance of neighboring vectors in the Pareto front found. The equations 11 and 12 define this performance measure.

$$E = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\overline{d} - d_i)^2} \tag{11}$$

$$d_i = min_{i,j \neq i} (\sum_{m=1}^{M} |f_i^m - f_j^m|) \tag{12}$$

6.2 Experiments

For assessing the performance of the micro-MOPSO, we used five test functions from the ZDT (Zitzler-Deb-Thiele) benchmark described in [39]. Additionally, we adopted Kursawe's test function [17] and Viennet's test funtion [37]. These test functions contain characteristics that make them difficult to solve using MOEAs.

We performed thirty independent runs for each test function and we compared our results with respect to the NSGA-II [8]. The number of objective function evaluations performed was set to 3000 in all cases.

For obtaining the results reported next, we adopted the following parameters, which were obtained after numerous experiments:

- w = random number from a uniform distribution in the range [0,1]
- $C_1 = C_2 = 1.8$
- population size = 5 particles
- number of generations = 600
- final archive bound (FAB) = 100
- auxiliary archive bound (AAB)= 200
- population percent for leader selection (pps) = 0.20
- number of replacement generations = 100
- number of replacement particles = 2
- mutation rate = 0.1

6.3 Results

The summary of the results obtained are shown in Table 1. This table shows the mean values and the standard deviations for each of the three performance measures adopted over the 30 independent runs performed.

The results show that our micro-MOPSO produced the best mean values in almost all cases. The graphical results shown in Figures 2 to 8, clearly reflect that our proposed micro-MOPSO outperforms the NSGA-II. In test functions ZDT1, ZDT2, ZDT3 and ZDT6, we can see that the NSGA-II is far away from the true Pareto front, whereas our micro-MOPSO has already converged to the true Pareto front after performing only 3000 fitness function evaluations. Figures 7 and 8 show that our micro-MOPSO can cover most of the Pareto front even when it is discontinuous (ZDT3 and Kursawe). The results and the figures show that the spread of solutions of our micro-MOPSO is evidently good enough for almost all the test functions (except for ZDT4 in which both algorithms have a poor performance) in spite of the low number of objective function evaluations performed.

	ER				DGI				S			
	micro-MOPSO		**N**SGA-II		**m**icro-MOPSO		**N**SGA-II		**m**icro-MOPSO		**N**SGA-II	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
ZDT1	**0.53**	0.4796	1.0	0.0	**0.0003**	0.0002	0.0151	0.002	**0.005**	0.0004	0.031	0.009
ZDT2	**0.496**	0.334	1.0	0.0	**0.0009**	0.0036	0.0339	0.006	**0.0048**	0.0005	0.048	0.0182
ZDT3	**0.935**	0.244	1.0	0.0	**0.002**	0.0043	0.0216	0.002	**0.0085**	0.009	0.034	0.009
ZDT4	1.0	0.0	1.0	0.0	1.585	0.790	**0.594**	0.1782	**0.0001**	0.0004	5.321	3.837
ZDT6	**0.133**	0.001	1.0	0.0	**0.00006**	0.0	0.0570	0.004	**0.033**	0.149	0.124	0.085
Kur	0.954	0.392	**0.837**	0.093	**0.003**	0.0007	0.004	0.0005	**0.082**	0.028	0.10	0.025
Vnt	**0.592**	0.44	0.6	0.36	**0.0006**	0.0005	0.002	0.004	**0.04**	0.02	0.061	0.010

Table 1. Comparison of results between our micro-MOPSO and the NSGA-II. σ refers to the standard deviation over the 30 runs performed.

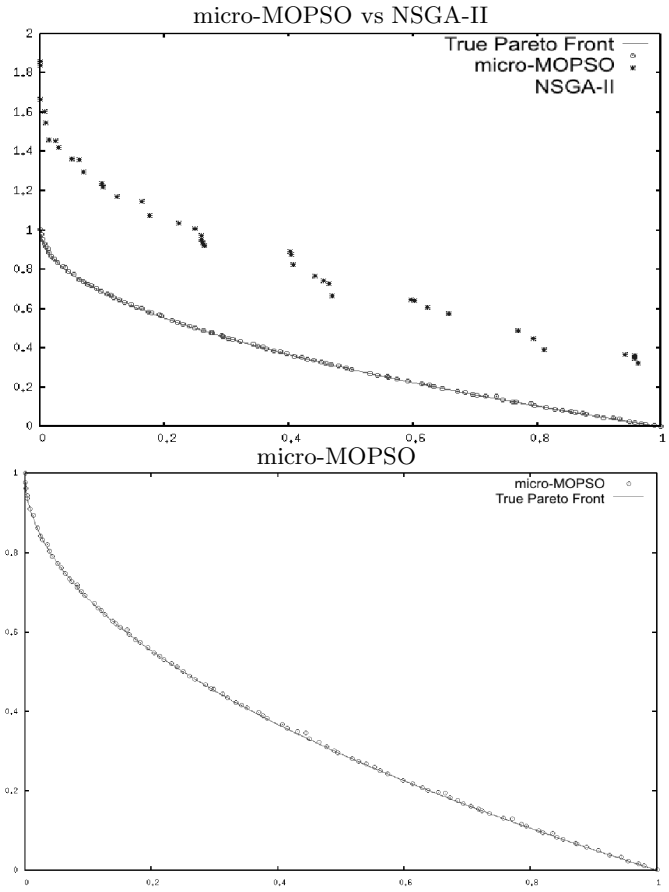


Fig. 2. Graphical comparison of the Pareto fronts found by our micro-MOPSO (bottom) and the NSGA-II (top) for ZDT1.

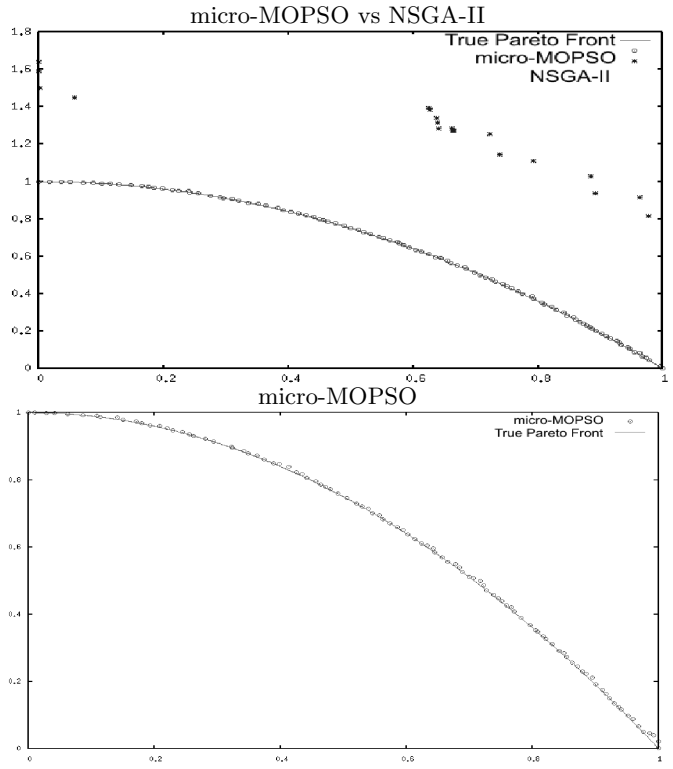


Fig. 3. Graphical comparison of the Pareto fronts found by our micro-MOPSO (bottom) and the NSGA-II (top) for ZDT2.

7 Conclusions and Future Work

We have proposed the use of a PSO algorithm with a very small population size (only five particles) for solving unconstrained multi-objective optimization problems. The proposed approach first selects the leader and then selects the neighborhood for constructing the swarm around the leader. It also uses a mutation operator, a reinitialization process, and a mechanism based on the crowding distance for selecting leaders. Our proposed approach was able to provide a better spread of the solutions obtained, as well as a faster convergence, when compared to the NSGA-II, in several test functions, in which only 3000 objective functions were performed. These results are very encouraging and indicate that our proposed approach could be a viable alternative for solving problems in which the evaluations of the objective functions are very expensive (computationally speaking).

As part of our future work, we are interested in incorporating a constraint-handling mechanism into our approach. We are also interested in adopting

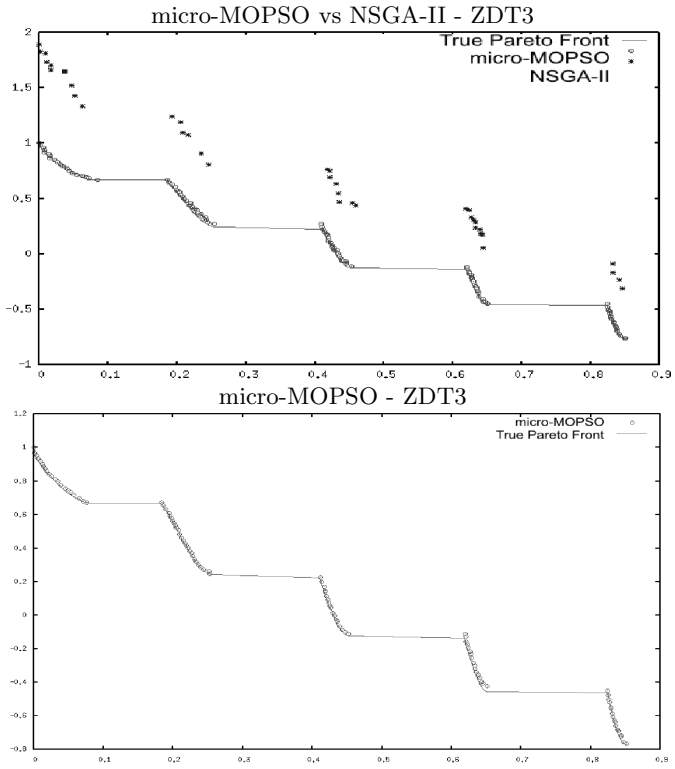


Fig. 4. Graphical comparison of the Pareto fronts found by our micro-MOPSO (right) and the NSGA-II (left) for ZDT3.

clustering techniques in order to provide a better spread of solutions. Additionally, we also want to study the sensitivity of our micro-MOPSO to its parameters, with the aim of finding a set of parameters (or a self-adaptation mechanism) that allows us to improve the performance and the robustness of our approach. Finally, we are also interested in experimenting with other types of reinitialization processes, since they could improve the convergence rate of our algorithm (i.e., we could reduce the number of objective function evaluations performed) as well as the quality of the results achieved.

Acknowledgements

The second author acknowledges support from CONACyT project no. 45683-Y.

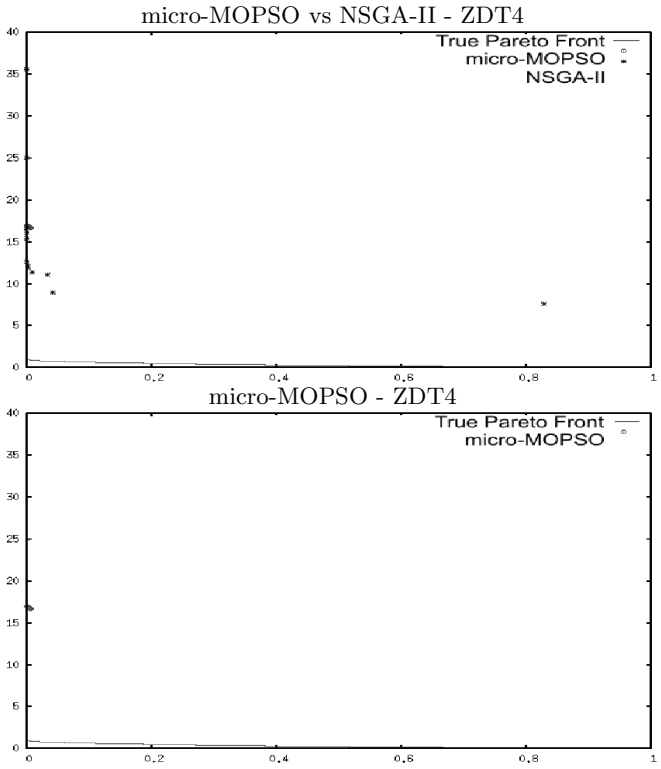


Fig. 5. Graphical comparison of the Pareto fronts found by our micro-MOPSO (right) and the NSGA-II (left) for ZDT4.

References

1. Julio E. Alvarez-Benitez, Richard M. Everson, and Jonathan E. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 459–473, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.
2. Paul S. Andrews. An investigation into mutation operators for particle swarm optimization. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation, (CEC'2006)*, pages 3789–3796, Vancouver, Canada, July 2006.
3. Thomas Bartz-Beielstein, Philipp Limbourg, Konstantinos E. Parsopoulos, Michael N. Vrahatis, Jörn Mehnen, and Karlheinz Schmitt. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1780–1787, Canberra, Australia, December 2003. IEEE Press.
4. Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective optimization using a micro-genetic algorithm. In L. Spector, E.D. Goodman, A. Wu,

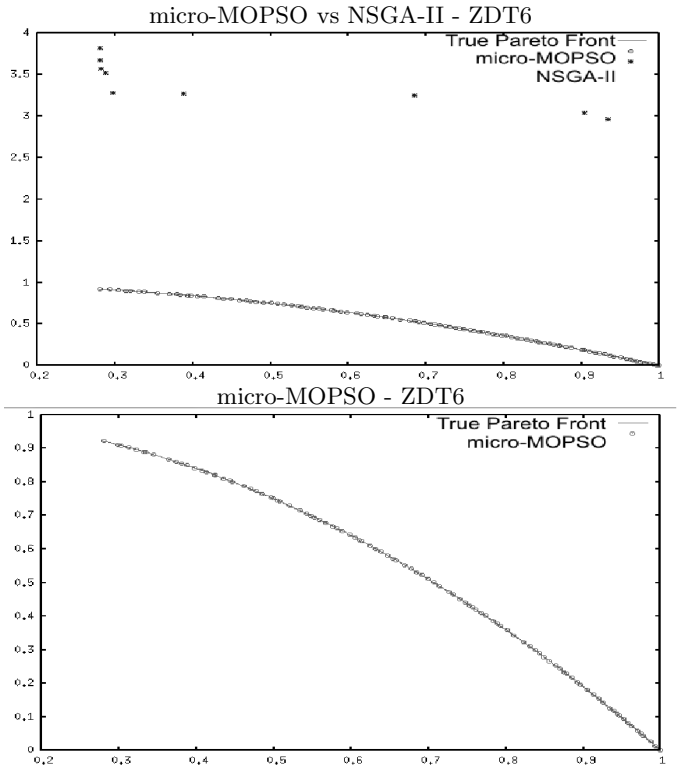


Fig. 6. Graphical comparison of the Pareto fronts found by our micro-MOPSO (right) and the NSGA-II (left) for ZDT6.

W. Langdon, H.M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk M., Garzon, and E. Burke, editors, *Genetic and Evolutionary Computation Conference, GECCO,2001*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

5. Carlos A. Coello Coello and Gregorio Toscano Pulido. A Micro-Genetic Algorithm for Multiobjective Optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.
6. Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.
7. Carlos A. Coello Coello David A. Van Veldhuizen and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, USA, second edition, 2007.
8. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

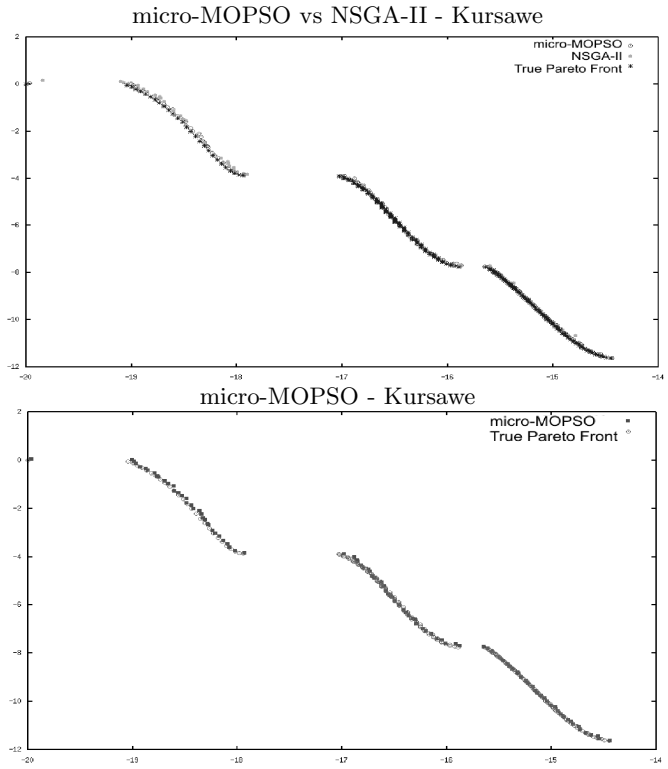


Fig. 7. Graphical comparison of the Pareto fronts found by our micro-MOPSO (right) and the NSGA-II (left) for Kursawe's test function.

9. R. Eberhart and J. Kennedy. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press, 1995.
10. Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons Ltd, England, 2005.
11. S.C. Esquivel and C.A. Coello Coello. On the use of particle swarm optimization with multimodal functions. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'2003)*, pages 1130–1136, Canberra, Australia, 2003. IEEE Press.
12. Jonathan E. Fieldsend and Sameer Singh. A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. In *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, pages 37–44, Birmingham, UK, September 2002.
13. Juan Carlos Fuentes Cabrera and Carlos A. Coello Coello. Handling Constraints in Particle Swarm Optimization using a Small Population Size. In Alexander Gelbukh and Angel F. Kuri Morales, editors, *Mexican International Conference on Artificial Intelligence. Sixth International Conference, MICAI 2007*, pages

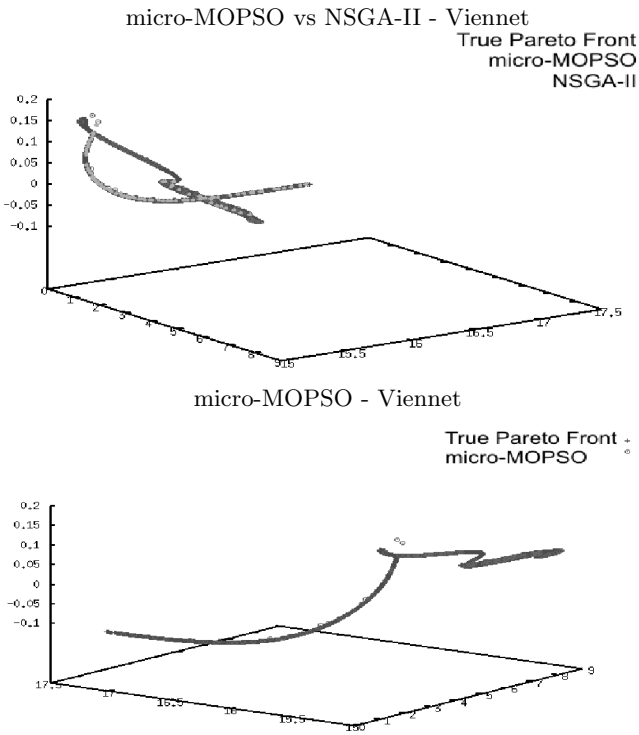


Fig. 8. Graphical comparison of the Pareto fronts found by our micro-MOPSO (right) and the NSGA-II (left) for Viennet's test function.

41–51, Aguascalientes, Mexico, November 2007. Springer. Lecture Notes in Artificial Intelligence. Volume 4827.

14. James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.
15. Joshua Knowles and David Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 98–105, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.
16. Kalmanje Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. *SPIE Proceedings: Intelligent Control and Adaptive Systems*, 1196:289–296, 1989.
17. Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science Vol. 496*, pages 193–197, Berlin, Germany, October 1991. Springer-Verlag.
18. Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining Convergence and Diversity in Evolutionary Multi-objective Optimization.

Evolutionary Computation, 10(3):263–282, Fall 2002.
19. Xiaodong Li. A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I*, pages 37–48. Springer. Lecture Notes in Computer Science Vol. 2723, July 2003.
20. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.
21. Kaisa M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1999.
22. J. Moore and R. Chapman. Application of particle swarm to multiobjective optimization, 1999.
23. Sanaz Mostaghim and Jürgen Teich. The Role of ε-dominance in Multi Objective Particle Swarm Optimization Methods. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1764–1771, Canberra, Australia, December 2003. IEEE Press.
24. Sanaz Mostaghim and Jürgen Teich. Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 26–33, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.
25. Sanaz Mostaghim and Jürgen Teich. Covering Pareto-optimal Fronts by Subswarms in Multi-objective Particle Swarm Optimization. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 1404–1411, Portland, Oregon, USA, June 2004. IEEE Service Center.
26. Margarita Reyes Sierra and Carlos A. Coello Coello. Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 505–519, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.
27. Margarita Reyes-Sierra and Carlos A. Coello Coello. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.
28. Margarita Reyes Sierra and Carlos A. Coello Coello. A Study of Techniques to Improve the Efficiency of a Multi-Objective Particle Swarm Optimizer. In Shengxiang Yang, Yew Soon Ong, and Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 269–296. Springer, 2007. ISBN 978-3-540-49772-1.
29. Isabella Schoeman and Andries Engelbrecht. Niching for Dynamic Environments using Particle Swarm Optimization. In Tzai-Der Wang, Xiaodong Li, Shu-Heng Chen, Xufa Wang, Hussein Abbass, Hitoshi Iba, Guoliang Chen, and Xin Yao, editors, *Simulated Evolution and Learning, 6th International Conference, SEAL 2006*, pages 134–141. Springer. Lecture Notes in Computer Science Vol. 4247, Hefei, China, October 2006.
30. Jason R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.
31. Y. Shi and R.C. Eberhart. A modified particle swarm optimizer. In *Proceedings of the 1998 IEEE Congress on Evolutionary Computation*, pages 69–73. IEEE Press, 1998.

32. C. H. Tan, C. K. Goh, K. C. Tan, and A. Tay. A Cooperative Coevolutionary Algorithm for Multiobjective Particle Swarm Optimization. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 3180–3186, Singapore, September 2007. IEEE Press.
33. Gregorio Toscano Pulido and Carlos A. Coello Coello. The Micro Genetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimization. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 252–266, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.
34. Gregorio Toscano Pulido and Carlos A. Coello Coello. Using clustering techniques to improve the performance of a particle swarm optimizer. In Kalyanmoy Deb et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2004)*, pages 225–237, Seattle,USA, June 2004. Springer. Lecture Notes in Computer Science Vol.3102.
35. Praveen Kumar Tripathi, Sanghamitra Bandyopadhyay, and Sankar Kumar Pal. Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. *Information Sciences*, 177(22):5033–5049, November 2007.
36. David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithm Test Suites. In Janice Carroll, Hisham Haddad, Dave Oppenheim, Barrett Bryant, and Gary B. Lamont, editors, *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357, San Antonio, Texas, 1999. ACM.
37. Rémy Viennet, Christian Fontiex, and Ivan Marc. Multicriteria Optimization Using a Genetic Algorithm for Determining a Pareto Set. *International Journal of Systems Science*, 27(2):255–260, 1996.
38. Zhang Xiao-hua, Meng Hong-yun, and Jiao Li-cheng. Intelligent Particle Swarm Optimization in Multiobjective Optimization. In *2005 Congress on Evolutionary Computation*, pages 714–719, Edinburgh, Scotland, UK, September 2005. IEEE Press.
39. Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.