Handling Multiple Objectives With Particle Swarm Optimization

Carlos A. Coello Coello, *Member, IEEE*, Gregorio Toscano Pulido, and Maximino Salazar Lechuga

***Abstract*—This paper presents an approach in which Pareto dominance is incorporated into particle swarm optimization (PSO) in order to allow this heuristic to handle problems with several objective functions. Unlike other current proposals to extend PSO to solve multiobjective optimization problems, our algorithm uses a secondary (i.e., external) repository of particles that is later used by other particles to guide their own flight. We also incorporate a special mutation operator that enriches the exploratory capabilities of our algorithm. The proposed approach is validated using several test functions and metrics taken from the standard literature on evolutionary multiobjective optimization. Results indicate that the approach is highly competitive and that can be considered a viable alternative to solve multiobjective optimization problems.**

***Index Terms*—Evolutionary multiobjective optimization, multiobjective optimization, multiobjective particle swarm optimization, particle swarm optimization.**

I. Introduction

The USE OF evolutionary algorithms for multiobjective optimization (an area called "evolutionary multiobjective optimization," or EMO for short) has significantly grown in the last few years, giving rise to a wide variety of algorithms [7]. As any other research area, EMO currently presents certain trends. One of them is to improve the efficiency both of the algorithms and of the data structures used to store nondominated vectors. EMO researchers have produced some clever techniques to maintain diversity (e.g., the adaptive grid used by the Pareto Archive Evolutionary Strategy (PAES) [21]), new algorithms that use very small populations (e.g., the microGA [6]), and data structures that allow to handle unconstrained external archives (e.g., the dominated tree [12]).

Particle swarm optimization (PSO) is a relatively recent heuristic inspired by the choreography of a bird flock. PSO has been found to be successful in a wide variety of optimization tasks [19], but until recently it had not been extended to deal with multiple objectives.

PSO seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single-objective optimization [19]. In this paper, we present a proposal, called "multiobjective particle swarm optimization" (MOPSO), which allows the PSO algorithm to be able to deal with multiobjective optimization problems. Our current proposal is an improved version of the algorithm reported in [5], in which we have added a constraint-handling mechanism and a mutation operator that considerably improves the exploratory capabilities of our original algorithm.

MOPSO is validated using several standard test functions reported in the specialized literature and compared against three highly competitive EMO algorithms: the nondominated sorting genetic algorithm-II [11] (NSGA-II), the PAES [21], and the microgenetic algorithm for multiobjective optimization (microGA) [6].

II. Basic Concepts

Definition 1 (Global Minimum): Given a function $f: \Omega \subseteq \mathcal{R}^n \to \mathcal{R}$, $\Omega \neq \emptyset$, for $\vec{x} \in \Omega$ the value $f^* \triangleq f(\vec{x}^*) > -\infty$ is called a global minimum if and only if

$$\forall \vec{x} \in \Omega: \quad f(\vec{x}^*) \leq f(\vec{x}). \tag{1}$$

Then, $\vec{x}^*$ is the global minimum solution, f is the objective function, and the set Ω is the feasible region ($\Omega \in \mathcal{S}$), where $\mathcal{S}$ represents the whole search space.

Definition 2 [General Multiobjective Optimization Problem (MOP)]: Find the vector $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which will satisfy the m inequality constraints

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, \ldots, m \tag{2}$$

the p equality constraints

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

and will optimize the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{4}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

Definition 3 (Pareto Optimality): A point $\vec{x}^* \in \Omega$ is **Pareto optimal** if for every $\vec{x} \in \Omega$ and $I = \{1, 2, \ldots, k\}$ either

$$\forall_{i \in I} \, (f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{5}$$

or, there is at least one $i \in I$ such that

$$f_i(\vec{x}) > f_i(\vec{x}^*) \tag{6}$$

Manuscript received July 8, 2002; revised September 2, 2003. The work of C. A. Coello Coello was supported in part by CONACyT under Project 34 201-A. G. T. Pulido acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the Electrical Engineering Department, CINVESTAV-IPN, México. M. S. Lechuga acknowledges support from CONACyT through a scholarship to pursue graduate studies at the School of Computer Science, University of Birmingham, Birmingham, U.K.

C. A. Coello Coello and G. T. Pulido are with CINVESTAV-IPN, Sección de Computación, Departamento de Ing. Eléctrica, Sección de Computación, México 07300, Mexico (e-mail: ccoello@cs.cinvestav.mx; gtoscano@computacion.cs.cinvestav.mx).

M. S. Lechuga is with the University of Birmingham, School of Computer Science, Edgbaston, Birmingham B15 2TT, U.K. (e-mail: M.S.Lechuga@cs.bham.ac.uk).

Digital Object Identifier 10.1109/TEVC.2004.826067

In words, this definition says that $\vec{x}^*$ is Pareto optimal if there exists no feasible vector $\vec{x}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. The phrase "Pareto optimal" is considered to mean with respect to the entire decision variable space unless otherwise specified.

Definition 4 (Pareto Dominance): A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if u is partially less than 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 5 (Pareto Optimal Set): For a given MOP $\vec{f}(x)$, the Pareto optimal set $(\mathcal{P}^*)$ is defined as

$$\mathcal{P}^* := \left\{ x \in \Omega | \neg \exists \; x' \in \Omega \; \vec{f}(x') \preceq \vec{f}(x) \right\}. \qquad (7)$$

Definition 6 (Pareto Front): For a given MOP $\vec{f}(x)$ and Pareto optimal set $\mathcal{P}^*$, the Pareto front $(\mathcal{PF}^*)$ is defined as

$$\mathcal{PF}^* := \left\{ \vec{u} = \vec{f} = (f_1(x), \ldots, f_k(x)) \, | x \in \mathcal{P}^* \right\}. \qquad (8)$$

In the general case, it is impossible to find an analytical expression of the line or surface that contains these points. The normal procedure to generate the Pareto front is to compute the feasible points Ω and their corresponding $f(\Omega)$. When there is a sufficient number of these, it is then possible to determine the nondominated points and to produce the Pareto front.

Pareto optimal solutions are also termed *noninferior*, *admissible*, or *efficient* solutions [15]; their corresponding vectors are termed *nondominated*.

III. Related Work

Kennedy and Eberhart [19] proposed an approach called PSO, which was inspired by the choreography of a bird flock. The approach can be seen as a distributed behavioral algorithm that performs (in its more general version) multidimensional search. In the simulation, the behavior of each individual is affected by either the best local (i.e., within a certain neighborhood) or the best global individual. The approach then uses the concept of population and a measure of performance similar to the fitness value used with evolutionary algorithms. Also, the adjustments of individuals are analogous to the use of a crossover operator. The approach also introduces the use of flying potential solutions through hyperspace (used to accelerate convergence), which can be seen as a mutation operator. An interesting aspect of PSO is that it allows individuals to benefit from their past experiences (note that in other approaches such as the genetic algorithm, normally the current population is the only "memory" used by the individuals). PSO has been successfully used for both continuous nonlinear and discrete binary single-objective optimization [19]. PSO seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single-objective optimization [19].

In fact, there have been several recent proposals to extend PSO to handle multiobjectives. We will review the most important of them.

- **The algorithm of Moore and Chapman** [24]: This algorithm was presented in an unpublished document and it is based on Pareto dominance. The authors emphasize the importance of performing both an individual and a group search (a cognitive component and a social component). However, the authors did not adopt any scheme to maintain diversity.
- **The swarm metaphor of Ray and Liew** [28]: This algorithm also uses Pareto dominance and combines concepts of evolutionary techniques with the particle swarm. The approach uses crowding to maintain diversity and a multilevel sieve to handle constraints (for this, the authors adopt the constraint and objective matrices proposed in some of their previous research [27]).
- **The algorithm of Parsopoulos and Vrahatis** [26]: Unlike the previous proposals, this algorithm adopts an aggregating function (three types of approaches were implemented: a conventional linear aggregating function, a dynamic aggregating function, and the bang-bang weighted aggregation approach [18] in which the weights are varied in such a way that concave portions of the Pareto front can be generated).
- **Dynamic neighborhood PSO proposed by Hu and Eberhart** [16]: In this algorithm, only one objective is optimized at a time using a scheme similar to lexicographic ordering [7]. Lexicographic ordering tends to be useful only when few objective functions are used (two or three), and it may be sensitive to the ordering of the objectives. The idea of the dynamic neighborhood is, with no doubt quite interesting and is novel in this context.

At the time in which this paper was originally prepared, none of the existing proposals to extend PSO to solve multiobjective optimization problems used a secondary population (the most common notion of elitism in EMO). This may certainly limit the performance of the algorithm, unless a very good diversity maintainance approach is used (unfortunately, this is not the case in most of the approaches available at that time). Also, none of these techniques had been properly validated using test functions and metrics normally adopted by EMO researchers.

Note however, that in more recent papers these ideas have been already incorporated by other authors. The most representative proposals are the following (published after the submission of the original version of this paper).

- Fieldsend and Singh [13] proposed an approach in which they use an unconstrained elite archive (in which a special data structure called "dominated tree" is adopted) to store the nondominated individuals found along the search process. The archive interacts with the primary population in order to define local guides. The approach is compared (using four test functions and two metrics) against an algorithm similar to PAES [21] and with a variation of our original MOPSO [5]. Their approach also uses a "turbulence" operator that is basically a mutation operator that acts on the velocity value used by PSO. The approach seems to have the same problems of our original MOPSO with multifrontal problems such as the fourth example included in this paper. It is important to note that the new version of our MOPSO provided in this paper does not have the problems of the original version in multifrontal problems.

- Hu *et al.* [17] adopted a secondary population (called "extended memory") and introduced some further improvements to their dynamic neighborhood PSO approach [16]. Nevertheless, it is worth indicating that this approach completely fails in generating the true Pareto front of some problems (see [17] for details). Hu *et al.* [17] also compared their algorithm with respect to the strength Pareto evolutionary algorithm (SPEA) [37] using the set coverage metric [34].
- Mostaghim and Teich [25] proposed a sigma method in which the best local guides for each particle are adopted to improve the convergence and diversity of a PSO approach used for multiobjective optimization. They also use a "turbulence" operator, but applied on decision variable space. The idea of the sigma approach is similar to compromise programming [7]. The use of the sigma values increases the selection pressure of PSO (which was already high). This may cause premature convergence in some cases (e.g., in multifrontal problems). In this approach, the authors provide comparisons with SPEA2 [36] and the dominated trees of Fieldsend and Singh [13] using four test functions and the coverage metric.
- Li [23] proposed an approach in which the main mechanisms of the NSGA-II [11] are adopted in a PSO algorithm. The proposed approach showed a very competitive performance with respect to the NSGA-II (even outperforming it in some cases).

The main differences of our approach with respect to the other proposals existing in the literature are:

- We adopt an external (or secondary) repository similar to the adaptive grid of PAES [21] (see Section IV-B). None of the other proposals use such a mechanism in the way adopted in this paper.
- The mutation operator that we use acts both on the particles of the swarm, and on the range of each design variable of the problem to be solved (see Section IV-C). This aims not only to explore remote regions of the search space, but also tries to ensure that the full range of each decision variable is explored.
- We provide an extensive analysis of the impact of the parameters of our MOPSO on its performance. We also compare our MOPSO with respect to three other algorithms (which are representative of the state-of-the-art in evolutionary multiobjective optimization), using three metrics.

IV. Description of the Proposed Approach

The analogy of PSO with evolutionary algorithms makes evident the notion that using a Pareto ranking scheme [14] could be the straightforward way to extend the approach to handle multiobjective optimization problems. The historical record of best solutions found by a particle (i.e., an individual) could be used to store nondominated solutions generated in the past (this would be similar to the notion of elitism used in evolutionary multiobjective optimization). The use of global attraction mechanisms combined with a historical archive of previously found nondominated vectors would motivate convergence toward globally nondominated solutions.

A. Main Algorithm

The algorithm of MOPSO is the following.

1) Initialize the population POP:
   (a) FOR $i = 0$ TO MAX/*MAX $=$ number of particles*/
   (b) Initialize $POP[i]$
2) Initialize the speed of each particle:
   (a) FOR $i = 0$ TO MAX
   (b) $VEL[i] = 0$
3) Evaluate each of the particles in POP.
4) Store the positions of the particles that represent nondominated vectors in the repository REP.
5) Generate hypercubes of the search space explored so far, and locate the particles using these hypercubes as a coordinate system where each particle's coordinates are defined according to the values of its objective functions.
6) Initialize the memory of each particle (this memory serves as a guide to travel through the search space. This memory is also stored in the repository):
   (a) FOR $i = 0$ TO MAX
   (b) $PBESTS[i] = POP[i]$
7) WHILE maximum number of cycles has not been reached DO
   a) Compute the speed of each particle[1] using the following expression:

$$VEL[i] = W \times VEL[i] + R_1 \times (PBESTS[i] - POP[i]) + R_2 \times (REP[h] - POP[i])$$

where W (inertia weight) takes a value of 0.4; R_1 and R_2 are random numbers in the range $[0 \ldots 1]$; $PBESTS[i]$ is the best position that the particle i has had;[2] $REP[h]$ is a value that is taken from the repository; the index h is selected in the following way: those hypercubes containing more than one particle are assigned a fitness equal to the result of dividing any number $x > 1$ (we used $x = 10$ in our experiments) by the number of particles that they contain. This aims to decrease the fitness of those hypercubes that contain more particles and it can be seen as a form of fitness sharing [10]. Then, we apply roulette-wheel selection using these fitness values to select the hypercube from which we will take the corresponding particle. Once the hypercube has been selected, we select randomly a particle within such hypercube. $POP[i]$ is the current value of the particle i.

   b) Compute the new positions of the particles adding the speed produced from the previous step

$$POP[i] = POP[i] + VEL[i]. \tag{9}$$

[1]Each particle has a dimensionality that can vary depending on the problem solved. When we say that we compute the speed of a particle, we refer to computing the speed for each of its dimensions.

[2]We will explain later on how do we define "better" in this context.

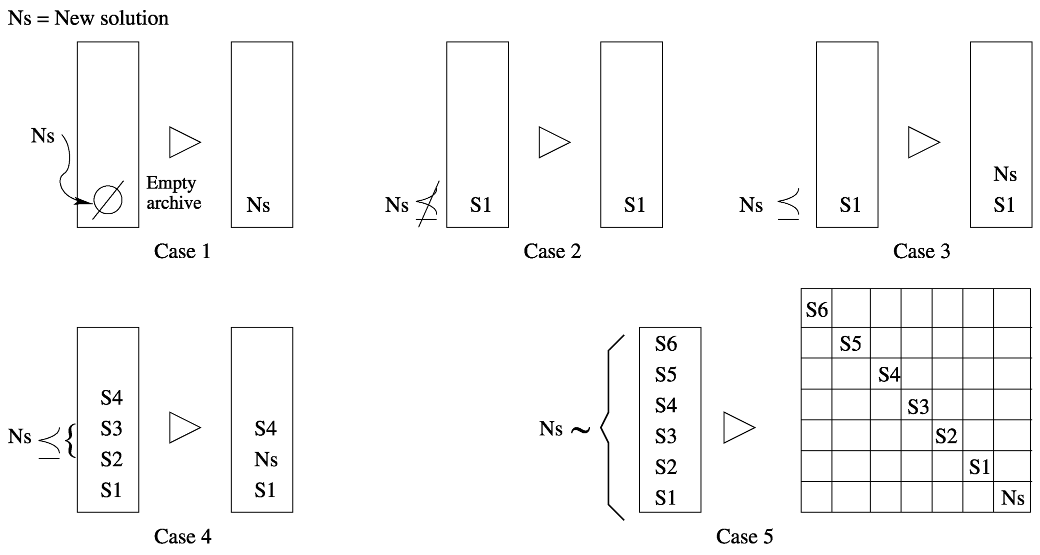


Fig. 1. Possible cases for the archive controller.

c) Maintain the particles within the search space in case they go beyond their boundaries (avoid generating solutions that do not lie on valid search space). When a decision variable goes beyonds its boundaries, then we do two things: 1) the decision variable takes the value of its corresponding boundary (either the lower or the upper boundary) and 2) its velocity is multiplied by (-1) so that it searches in the opposite direction.

d) Evaluate each of the particles in POP.

e) Update the contents of REP together with the geographical representation of the particles within the hypercubes. This update consists of inserting all the currently nondominated locations into the repository. Any dominated locations from the repository are eliminated in the process. Since the size of the repository is limited, whenever it gets full, we apply a secondary criterion for retention: those particles located in less populated areas of objective space are given priority over those lying in highly populated regions.

f) When the current position of the particle is better than the position contained in its memory, the particle's position is updated using

$$PBESTS[i] = POP[i]. \tag{10}$$

The criterion to decide what position from memory should be retained is simply to apply Pareto dominance (i.e., if the current position is dominated by the position in memory, then the position in memory is kept; otherwise, the current position replaces the one in memory; if neither of them is dominated by the other, then we select one of them randomly).

g) Increment the loop counter

8) END WHILE

B. External Repository

The main objective of the external repository (or archive) is to keep a historical record of the nondominated vectors found along the search process. The external repository consists of two main parts: the archive controller and the grid.

We will proceed to discuss each of these two components in more detail.

1) The Archive Controller: The function of the archive controller is to decide whether a certain solution should be added to the archive or not. The decision-making process is the following.

The nondominated vectors found at each iteration in the primary population of our algorithm are compared (on a one-per-one basis) with respect to the contents of the external repository which, at the beginning of the search will be empty. If the external archive is empty, then the current solution is accepted (see case 1, in Fig. 1). If this new solution is dominated by an individual within the external archive, then such a solution is automatically discarded (see case 2, in Fig. 1). Otherwise, if none of the elements contained in the external population dominates the solution wishing to enter, then such a solution is stored in the external archive. If there are solutions in the archive that are dominated by the new element, then such solutions are removed from the archive (see cases 3 and 4, in Fig. 1). Finally, if the external population has reached its maximum allowable capacity, then the adaptive grid procedure is invoked (see case 5, in Fig. 1).

2) The Grid: To produce well-distributed Pareto fronts, our approach uses a variation of the adaptive grid proposed in [21]. The basic idea is to use an external archive to store all the solutions that are nondominated with respect to the contents of the archive. Into the archive, objective function space is divided into regions as shown in Fig. 2. Note that if the individual inserted into the external population lies outside the current bounds of the grid, then the grid has to be recalculated and each individual within it has to be relocated (see Fig. 3).

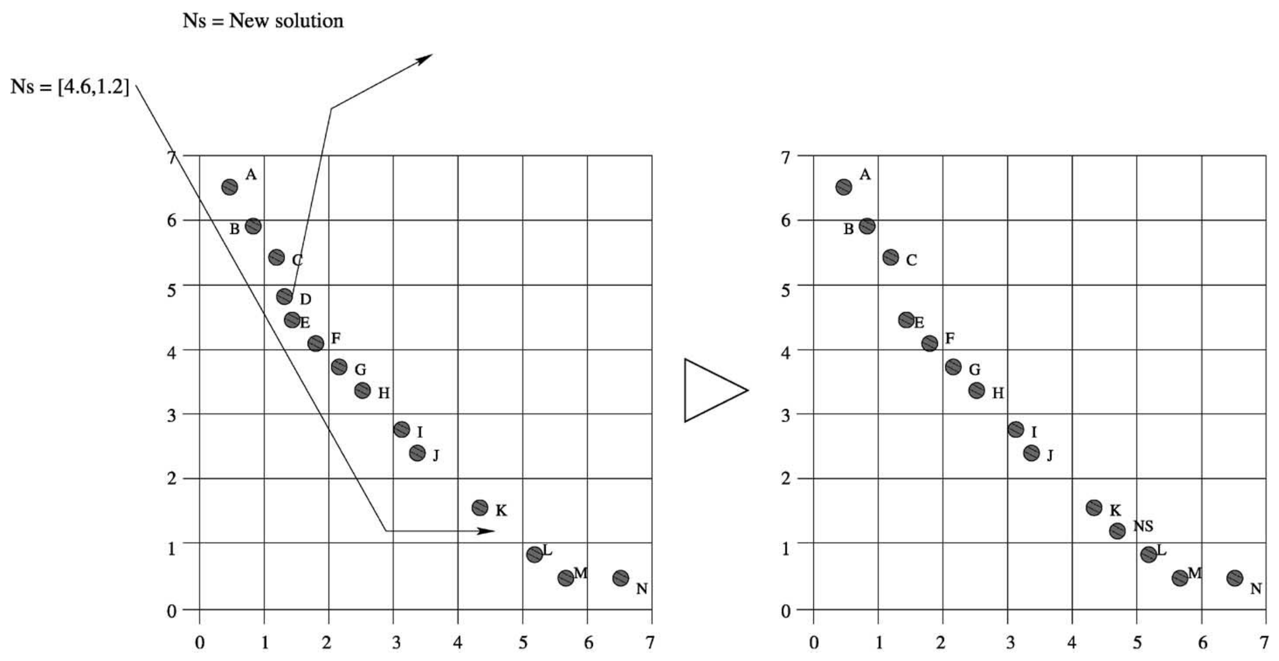


Fig. 2. Graphical representation of the insertion of a new element in the adaptive grid when the individual lies within the current boundaries of the grid.

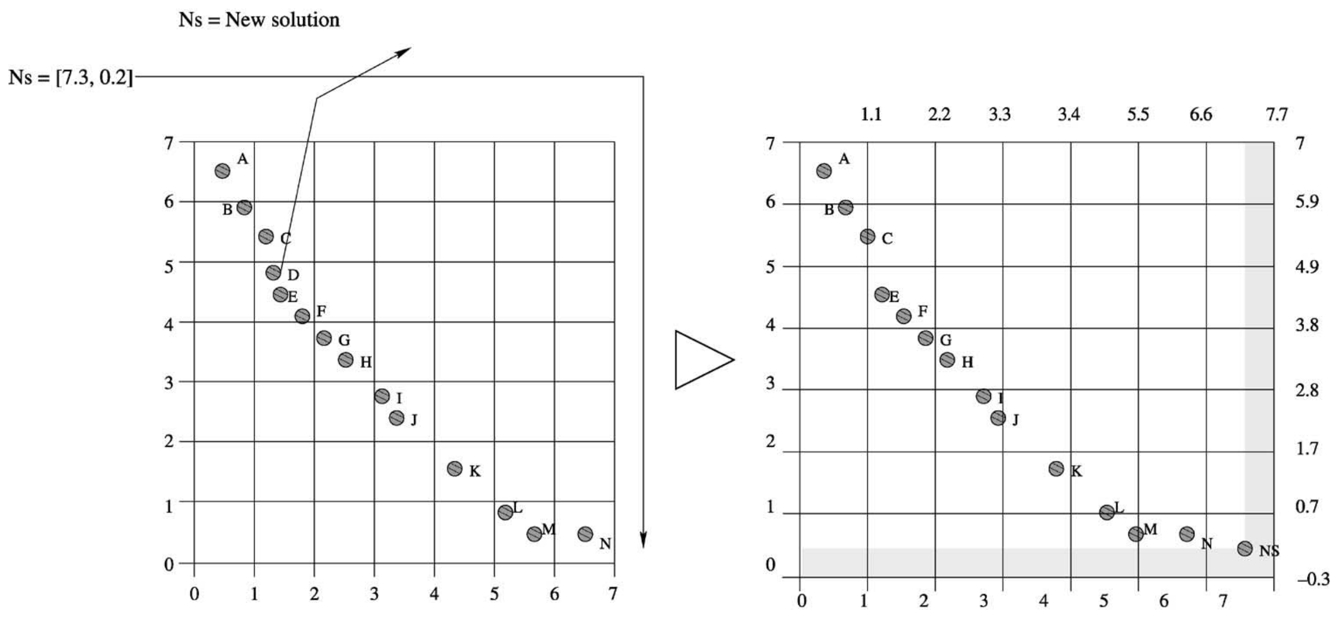


Fig. 3. Graphical representation of the insertion of a new element in the adaptive grid when this lies outside the previous boundaries of the grid.

The adaptive grid is really a space formed by hypercubes.[3] Such hypercubes have as many components as objective functions. Each hypercube can be interpreted as a geographical region that contains an *no* number of individuals. The main advantage of the adaptive grid is that its computational cost is lower than niching (see [21] for a detailed complexity analysis). The only exception would be if the grid had to be updated at each generation. In such a case, the computational complexity of the adaptive grid would be the same as niching [i.e., $O(N^2)$].

[3]Strictly speaking, it is formed by hyperparallelepids when the ranges of the objective functions are not scaled. If scaled, however, we are talking of hypercubes, which is the assumption made in this paper.

The adaptive grid is used to distribute in a uniform way the largest possible amount of hypercubes. In order to achieve this goal, it is necessary to provide and obtain certain information which is problem dependant (i.e., the number of grid subdivisions).

C. Use of a Mutation Operator

This operator deserves a more detailed discussion. PSO is known to have a very high convergence speed. However, such convergence speed may be harmful in the context of multiobjective optimization, because a PSO-based algorithm may converge to a false Pareto front (i.e., the equivalent of a

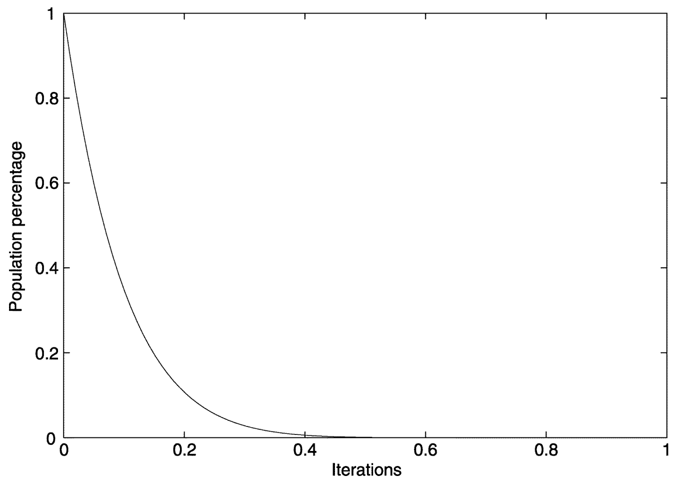


Fig. 4. Behavior of our mutation operator. In the x axis, we show the number of iterations performed by our MOPSO, expressed as a percentage and in the y axis, we show the percentage of the population that is affected by the mutation operator.

local optimum in global optimization). This drawback of PSO is evident in some problems (e.g., in test function 1 described in Section V) in which our original approach did not perform very well. This motivated the development of a mutation operator that tries to explore with all the particles at the beginning of the search. Then, we decrease rapidly (with respect to the number of iterations) the number of particles that are affected by the mutation operator (see Fig. 4). Note that our mutation operator is applied not only to the particles of the swarm, but also to the range of each design variable of the problem to be solved (using the same variation function). What this does is to cover the full range of each design variable at the beginning of the search and then we narrow the range covered over time, using a nonlinear function. From Fig. 4, we can see that at the beginning, all the particles in the population are affected by the mutation operator (as well as the full range of the decision variables). This intends to produce a highly explorative behavior in the algorithm. As the number of iterations increases, the effect of the mutation operator decreases. The pseudocode of our mutation operator is shown in Fig. 5.

The use of mutation operators in PSO is not new. Frans van den Bergh [31], proposed the *randomised particle optimizer* (RPSO) in which the aim was to construct a PSO-based global search algorithm. The RPSO resets the position of an specific particle, at a certain (fixed) number of iterations. Note however, that our approach is not only adding exploratory capabilities to PSO (as in the RPSO), but it also ensures that the full range of every decision variable is explored. Such type of mutation operator is novel (to the authors' best knowledge), at least in the context of PSO approaches used for multiobjective optimization.

D. Handling Constraints

We also added a relatively simple scheme to handle constraints. Whenever two individuals are compared, we check their constraints. If both are feasible, nondominance is directly applied to decide who is the winner. If one is feasible and the other is infeasible, the feasible dominates. If both are infeasible, then the one with the lowest amount of constraint violation dominates the other. This is the same approach that we originally proposed to handle constraints within the microgenetic algorithm for multiobjective optimization (microGA) [6].

V. Comparison of Results

Several test functions were taken from the specialized literature to compare our approach. In order to allow a quantitative assessment of the performance of a multiobjective optimization algorithm, three issues are normally taken into consideration [35].

1) Minimize the distance of the Pareto front produced by our algorithm with respect to the global Pareto front (assuming we know its location).
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.

Based on this notion, we adopted one metric to evaluate each of three aspects previously indicated.

1) **Generational distance** (GD): The concept of generational distance was introduced by Van Veldhuizen and Lamont [33] as a way of estimating how far the elements are in the set of nondominated vectors found so far from those in the Pareto optimal set and is defined as
$$\mathrm{GD} = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{11}$$
where n is the number of vectors in the set of nondominated solutions found so far and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the Pareto optimal set. It should be clear that a value of $\mathrm{GD} = 0$ indicates that all the elements generated are in the Pareto optimal set. Therefore, any other value will indicate how "far" we are from the global Pareto front of our problem. This metric addresses the first issue from the list previously provided.
2) **Spacing** (SP): Here, one desires to measure the spread (distribution) of vectors throughout the nondominated vectors found so far. Since the "beginning" and "end" of the current Pareto front found are known, a suitably defined metric judges how well the solutions in such front are distributed. Schott [29] proposed such a metric measuring the range (distance) variance of neighboring vectors in the nondominated vectors found so far. This metric is defined as
$$S \triangleq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d} - d_i)^2} \tag{12}$$
where $d_i = \min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|)$, i, $j = 1, \ldots, n$, $\overline{d}$ is the mean of all d_i, and n is the number of nondominated vectors found so far. A value of zero for this metric indicates all members of the Pareto front currently available are equidistantly spaced. This metric addresses the second issue from the list previously provided.
3) **Error ratio** (ER): This metric was proposed by Van Veldhuizen [32] to indicate the percentage of solutions (from

```
% particle = particle to be mutated
% dims = number of dimensions (i.e., decision variables)
% currentgen = current iteration
% totgen = total number of iterations
% mutrate = mutation rate
function Mutation-Operator(particle,dims,currentgen,totgen,mutrate)
  begin
    if flip($(1 - currentgen/totgen)^{5/mutrate}$) then
    begin
      wichdim=random(0,dims-1)
      mutrange=(upperbound[wichdim]-lowerbound[wichdim])*$(1 - currentgen/totgen)^{5/mutrate}$
      ub=particle[wichdim]+mutrange
      lb=particle[wichdim]-mutrange
      if lb < lowerbound[wichdim] then lb=lowerbound[wichdim]
      if ub > upperbound[wichdim] then ub=upperbound[wichdim]
      particle[wichdim]=RealRandom(lb,ub)
    end if
end function
```

Fig. 5. Pseudocode of our mutation operator. The variation of our mutation operator is graphically shown in Fig. 4.

the nondominated vectors found so far) that are not members of the true Pareto optimal set

$$\mathrm{ER} = \frac{\sum_{i=1}^{n} e_i}{n} \tag{13}$$

where n is the number of vectors in the current set of nondominated vectors available, $e_i = 0$ if vector i is a member of the Pareto optimal set, and $e_i = 1$, otherwise. It should then be clear that $\mathrm{ER} = 0$ indicates an ideal behavior, since it would mean that all the vectors generated by our algorithm belong to the Pareto optimal set of the problem. This metric addresses the third issue from the list previously provided.

Additionally, times were also evaluated (using the same hardware platform and the exact same environment for each of the algorithms) in order to establish if our MOPSO algorithm was really faster than the other techniques as we hypothesized.

In order to know how competitive our approach was, we decided to compare it against three multiobjective evolutionary algorithms that are representative of the state-of-the-art.

1) **Nondominated Sorting Genetic Algorithm II**: Proposed by Deb *et al.* [9], [11], this algorithm is a revised version of the nondominated sorting genetic algorithm proposed by Srinivas and Deb [30]. The original NSGA is based on several layers of classifications of the individuals as suggested by Goldberg [14]. Before selection is performed, the population is ranked on the basis of nondomination: all nondominated individuals are classified into one category (with a dummy fitness value, which is proportional to the population size, to provide an equal reproductive potential for these individuals). Then, this group of classified individuals is ignored and another layer of nondominated individuals is considered. The process continues until all individuals in the population are classified. Since individuals in the first front have the maximum fitness value, they always get more copies than the rest of the population. This allows a search for nondominated regions, and results in convergence of the population toward such regions.

   The NSGA-II is more efficient (computationally speaking) than the original NSGA, uses elitism and a crowded comparison operator that keeps diversity without specifying any additional parameters (the original NSGA used fitness sharing). This algorithm uses $(\mu + \lambda)$-selection as its elitist mechanism.

2) **Pareto Archived Evolution Strategy**: This algorithm was introduced by Knowles and Corne [21]. PAES consists of a $(1 + 1)$ evolution strategy (i.e., a single parent that generates a single offspring) in combination with a historical archive that records some of the nondominated solutions previously found. This archive is used as a reference set against which each mutated individual is being compared. Such a historical archive is the elitist mechanism adopted in PAES. However, an interesting aspect of this algorithm is the procedure used to maintain diversity which consists of a crowding procedure that divides objective space in a recursive manner. Each solution is placed in a certain grid location based on the values of its objectives (which are used as its "coordinates" or "geographical location"). A map of such grid is maintained, indicating the number of solutions that reside in each grid location. Since the procedure is adaptive, no extra parameters are required (except for the number of divisions of the objective space).

3) **Microgenetic Algorithm for Multiobjective Optimization**: This approach was introduced by Coello Coello and Toscano Pulido [3], [6]. A microgenetic algorithm is a GA with a small population and a reinitialization process. The

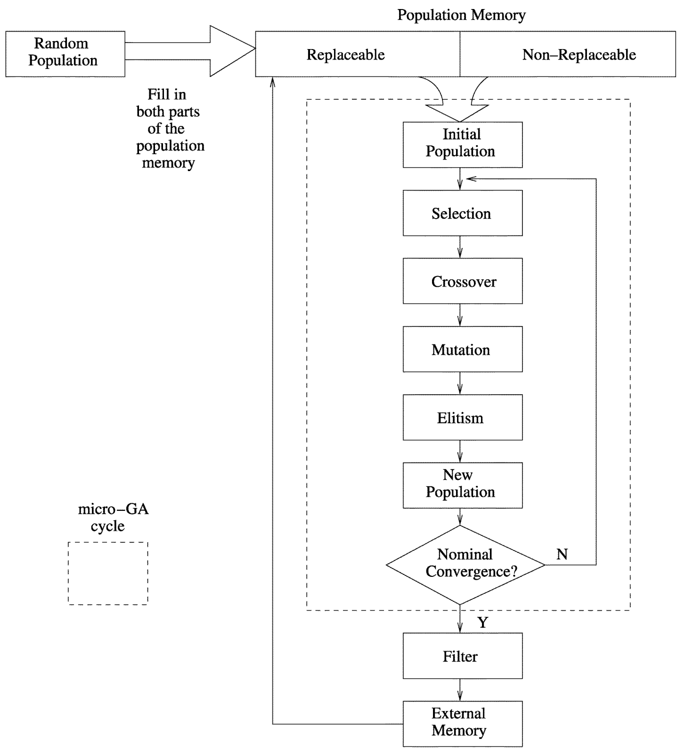


Fig. 6. Diagram that illustrates the way in which the microGA for multiobjective optimization works.

way in which the microGA works is illustrated in Fig. 6. First, a random population is generated. This random population feeds the population memory, which is divided in two parts: a replaceable and a nonreplaceable portion. The nonreplaceable portion of the population memory never changes during the entire run and is meant to provide the required diversity for the algorithm. In contrast, the replaceable portion experiences changes after each cycle of the microGA.

The population of the microGA at the beginning of each of its cycles is taken (with a certain probability) from both portions of the population memory so that there is a mixture of randomly generated individuals (nonreplaceable portion) and evolved individuals (replaceable portion). During each cycle, the microGA undergoes conventional genetic operators. After the microGA finishes one cycle, two nondominated vectors are chosen[4] from the final population and they are compared with the contents of the external memory (this memory is initially empty). If either of them (or both) remains as nondominated after comparing it against the vectors in this external memory, then they are included there (i.e., in the external memory). This is the historical archive of nondominated vectors. All dominated vectors contained in the external memory are eliminated.

The microGA uses then three forms of elitism: 1) it retains nondominated solutions found within the internal cycle of the microGA; 2) it uses a replaceable memory whose contents is partially "refreshed" at certain intervals; and 3) it replaces the population of the microGA by the nominal solutions produced (i.e., the best solutions found after a full internal cycle of the microGA).

[4]This is assuming that there are two or more nondominated vectors. If there is only one, then this vector is the only one selected.

In the following examples, the NSGA-II was run using a population size of 100, a crossover rate of 0.8 (uniform crossover was adopted), tournament selection, and a mutation rate of $1/L$, where L = chromosome length (binary representation was adopted). The microGA used a crossover rate of 0.9, an external memory of 100 individuals, a number of iterations to achieve nominal convergence of two, a population memory of 50 individuals, a percentage of nonreplaceable memory of 0.05, a population size (for the microGA itself) of four individuals, and 25 subdivisions of the adaptive grid. The mutation rate was set to $1/L$ (L = length of the chromosomic string). PAES was run using an adaptive grid with a depth of five, a size of the archive of 100, and a mutation rate of $1/L$, where L refers to the length of the chromosomic string that encodes the decision variables. MOPSO used a population of 100 particles, a repository size of 100 particles, a mutation rate of 0.5,[5] and 30 divisions for the adaptive grid. Our implementation uses a real-numbers representation, but a binary representation could also be adopted if needed. These parameters were kept in all the examples, and we only changed the total number of fitness function evaluations[6] but the same value was adopted for all the algorithms in each of the examples presented next. The source code of the NSGA-II, PAES and the microGA is available from the EMOO repository [4].[7]

In all the following examples, we report the results obtained from performing 30 independent runs of each algorithm compared. In all cases, the best average results obtained with respect to each metric are shown in **boldface**.

A. Test Function 1

For our first example, we used the following problem proposed by Kita [20]:

$$\text{Maximize } F = (f_1(x,y), f_2(x,y)), \quad \text{where}$$
$$f_1(x,y) = -x^2 + y, \quad f_2(x,y) = \frac{1}{2}x + y + 1$$

subject to

$$0 \geq \frac{1}{6}x + y - \frac{13}{2}, \quad 0 \geq \frac{1}{2}x + y - \frac{15}{2}, \quad 0 \geq 5x + y - 30$$

and $x, y \geq 0$. The range adopted in our case is $0 \leq x, y \leq 7$.

In this example, the total number of fitness function evaluations was set to 5000.

Figs. 7 and 8 show the graphical results produced by our MOPSO, the microGA, the NSGA-II, and PAES in the first test function chosen. The true Pareto front of the problem is shown as a continuous line. The solutions displayed correspond to the

[5]This value was determined after performing an extensive set of experiments. Note, however, that the performance of MOPSO can improve in some of the test functions presented if a higher mutation rate is adopted. This, however, also increases the computational cost of the approach.

[6]The total number of fitness function evaluations was empirically determined based on the complexity of the test function adopted. However, at the end of the paper some guidelines are provided regarding how to determine the parameters of the MOPSO for an arbitrary problem.

[7]The source code of MOPSO may be obtained by e-mailing to ccoello@cs.cinvestav.mx.

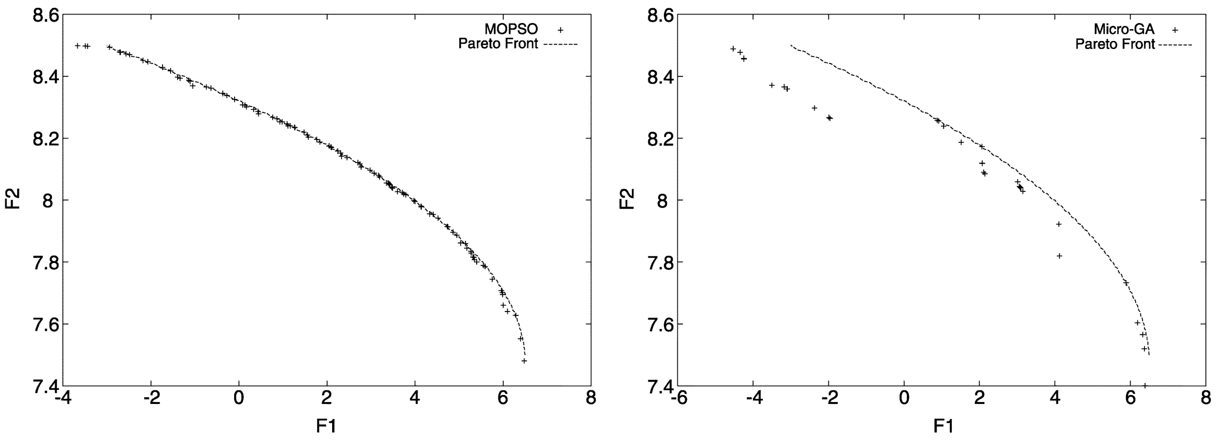


Fig. 7. Pareto fronts produced by our MOPSO (left) and the microGA (right) for the first test function. The true Pareto front is shown as a continuous line.

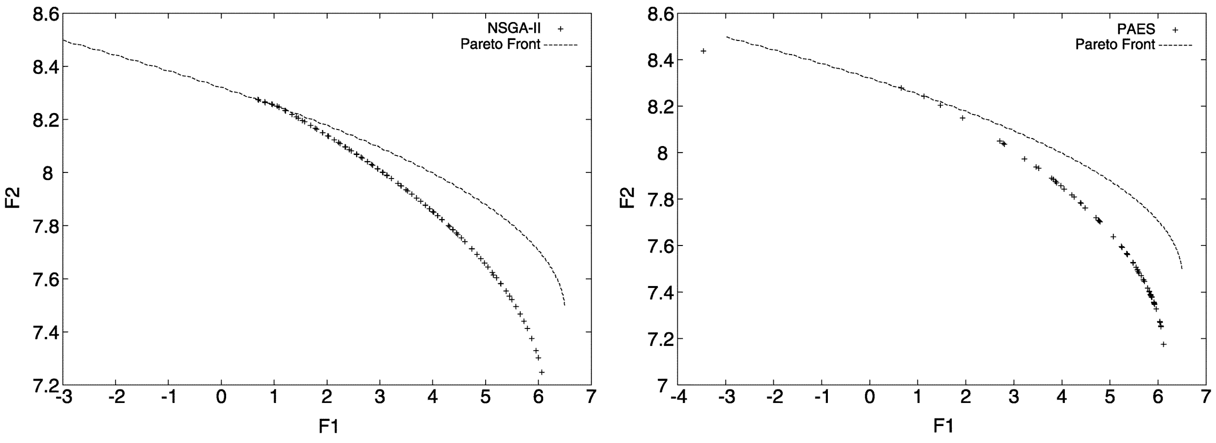


Fig. 8. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the first test function.

TABLE I
RESULTS OF THE ERROR RATIO METRIC FOR THE FIRST TEST FUNCTION

ER	MOPSO	NSGA-II	microGA	PAES
Best	0.08	0.75	0.734694	0.93
Worst	0.27	0.99	1.01639	1.01
Average	**0.132532**	0.8965	0.927706	0.993
Median	0.14	0.92	0.936365	1.01
Std. Dev.	0.045007	0.067143	0.068739	0.025361

TABLE II
RESULTS OF THE GENERATIONAL DISTANCE METRIC FOR THE FIRST TEST FUNCTION

GD	MOPSO	NSGA-II	microGA	PAES
Best	0.002425	0.003885	0.00513	0.011321
Worst	0.476815	0.678449	0.912065	0.919167
Average	**0.036535**	0.084239	0.150763	0.193173
Median	0.007853	0.011187	0.089753	0.033289
Std. Dev.	0.104589	0.165244	0.216558	0.249653

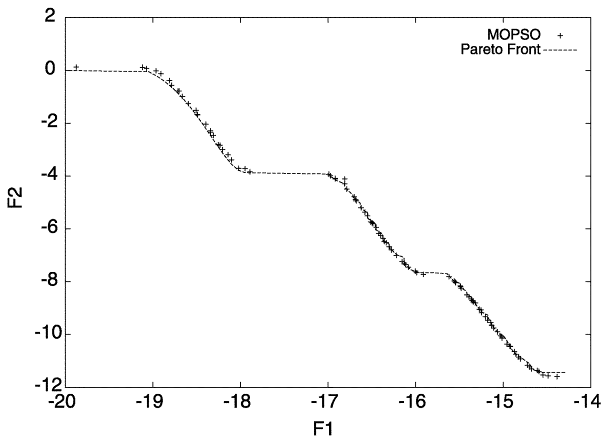
TABLE III
RESULTS OF THE SPACING METRIC FOR THE FIRST TEST FUNCTION

SP	MOPSO	NSGA-II	microGA	PAES
Best	0.043982	0.001032	0.06561	0.006669
Worst	0.538102	1.48868	1.64386	0.432865
Average	0.109452	**0.098486**	0.31502	0.110103
Median	0.06748	0.027173	0.129744	0.081999
Std. Dev.	0.110051	0.32738	0.421742	0.099598

TABLE IV
COMPUTATIONAL TIME (IN SECONDS) REQUIRED BY EACH ALGORITHM FOR THE FIRST TEST FUNCTION

time	MOPSO	NSGA-II	microGA	PAES
Best	0.007	0.989	0.362	7.369
Worst	0.272	1.135	0.513	8.331
Average	**0.2512**	1.08815	0.41895	7.77665
Median	0.264	1.1085	0.418	7.805
Std. Dev.	0.0575569	0.043216	0.036957	0.288103

median result with respect to the generational distance metric. Tables I, II, III, and IV show the comparison of results among the four algorithms considering the metrics previously described. It can be seen that the average performance of MOPSO is the best with respect to the error ratio (by far), and to the generational distance. With respect to spacing it places slightly below the NSGA-II, but with a lower standard deviation. By looking at the Pareto fronts of this test function, it is easy to notice that



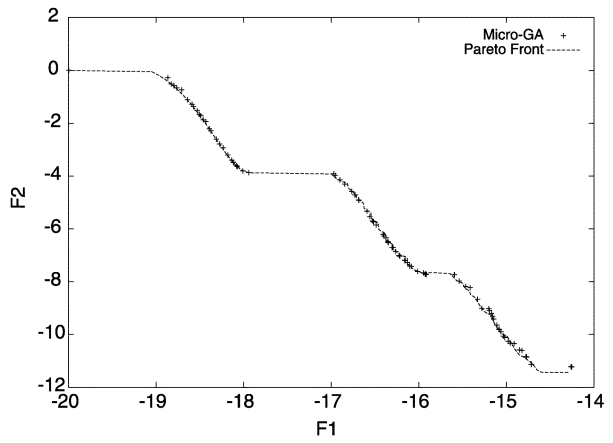
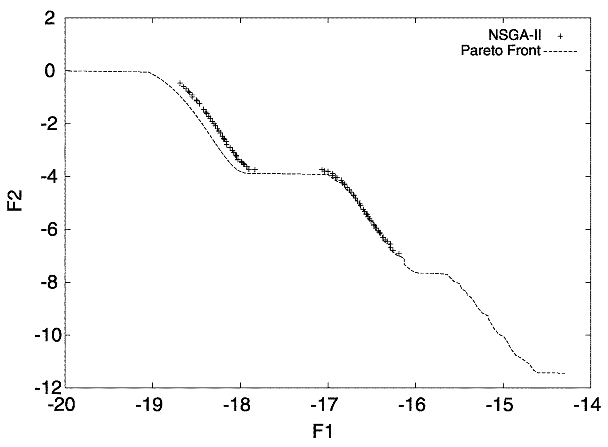


Fig. 9. Pareto fronts produced by our MOPSO (left) and the microGA (right) for the second test function. The true Pareto front is shown as a continuous line.



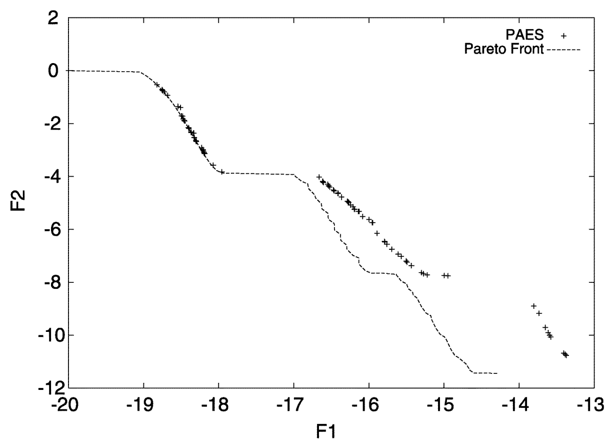


Fig. 10. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the second test function.

except for MOPSO, none of the algorithms were able to cover the full Pareto front.[8] This is then an example in which a metric may be misleading, since the fact that the spacing metric provides a good value is irrelevant if the nondominated vectors produced by the algorithm are not part of the true Pareto front of the problem [32]. Also, it is important to notice the very high speed of MOPSO, which requires almost half of the time than the microGA. This is remarkable if we consider that the NSGA-II and the microGA are algorithms that are normally considered "very fast" approaches.

VI. Test Function 2

Our second test function was proposed by Kursawe [22]

Minimize

$$f_1(\vec{x}) = \sum_{i=1}^{n-1} \left(-10 \exp \left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right) \quad (14)$$

Minimize

$$f_2(\vec{x}) = \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5 \sin(x_i)^3 \right) \quad (15)$$

[8]When using our original implementation of MOPSO [5], we ran into the same problem. Such a behavior motivated the development of the mutation operator reported in this paper.

TABLE V
Results of the Error Ratio Metric for the Second Test Function

ER	MOPSO	NSGA-II	microGA	PAES
Best	0.18	0.06	0.18	0.10
Worst	0.37	1.01	0.36	0.68
Average	**0.2535**	0.56	0.27	0.27
Median	0.255	0.495	0.245	0.245
Std. Dev.	0.04082	0.384516	0.053947	0.10489

where

$$-5 \leq x_1, x_2, x_3 \leq 5 \quad (16)$$

In this example, the total number of fitness function evaluations was set to 12 000.

Figs. 9 and 10 show the graphical results produced by our MOPSO, the microGA, the NSGA-II, and PAES in the second test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables V, VI, VII, and VIII show the comparison of results among the four algorithms considering the metrics previously described. It can be seen that the average performance of MOPSO is the best with respect to the error ratio, and it is slightly below the microGA with respect to

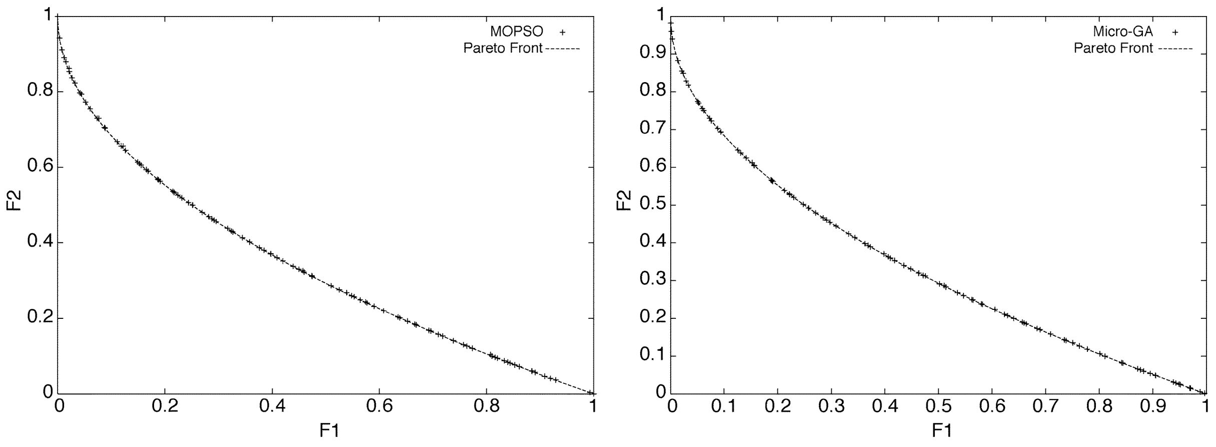


Fig. 11. Pareto fronts produced by our MOPSO (left) and the microGA (right) for the third test function. The true Pareto front is shown as a continuous line.

TABLE VI
RESULTS OF THE GENERATIONAL DISTANCE METRIC FOR THE SECOND TEST FUNCTION

GD	MOPSO	NSGA-II	microGA	PAES
Best	0.00745	0.006905	0.006803	0.01467
Worst	0.00960	0.103095	0.010344	0.157191
Average	**0.008450**	0.029255	0.008456	0.54914
Median	0.00845	0.017357	0.008489	0.049358
Std. Dev.	0.00051	0.02717	0.000987	0.030744

TABLE VII
RESULTS OF THE SPACING METRIC FOR THE SECOND TEST FUNCTION

SP	MOPSO	NSGA-II	microGA	PAES
Best	0.06187	0.018418	0.071686	0.064114
Worst	0.118445	0.065712	0.203127	0.340955
Average	0.09747	**0.036136**	0.128895	0.197532
Median	0.10396	0.036085	0.126655	0.186632
Std. Dev.	0.01675	0.010977	0.029932	0.064114

TABLE VIII
COMPUTATIONAL TIME (IN SECONDS) REQUIRED BY EACH ALGORITHM FOR THE SECOND TEST FUNCTION

time	MOPSO	NSGA-II	microGA	PAES
Best	0.155	2.181	0.295	0.938
Worst	0.168	2.693	0.345	1.39
Average	**0.162158**	2.426789	0.32695	1.12615
Median	0.161	2.461	0.3325	1.121
Std. Dev.	0.003468	0.171008	0.014813	0.105224

TABLE IX
RESULTS OF THE ERROR RATIO METRIC FOR THE THIRD TEST FUNCTION

ER	MOPSO	NSGA-II	microGA	PAES
Best	0.19	0.0	0.02	0.06
Worst	0.55	1.01	1.04545	1.01
Average	0.3335	0.35	**0.2568**	0.4485
Median	0.3	0.20	0.19	0.24
Std. Dev.	0.09388	0.396153	0.256456	0.381993

the generational distance. With respect to spacing it does considerably worse than the NSGA-II, but the graphical solutions show that the NSGA-II is not able to cover the entire Pareto front of the problem, whereas the MOPSO does it. This makes the value of this metric irrelevant, since some of the solutions produced by the NSGA-II are not part of the true Pareto front of the problem. Also, note that MOPSO is 15 times faster than the NSGA-II in this test function.

VII. TEST FUNCTION 3

Our third test function was proposed by Deb [8]

$$\text{Minimize } f_1(x_1, x_2) = x_1 \tag{17}$$
$$\text{Minimize } f_2(x_1, x_2) = g(x_1, x_2) \cdot h(x_1, x_2) \tag{18}$$

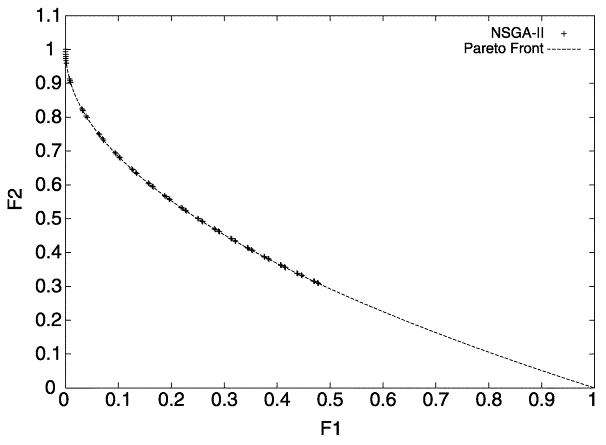
where

$$g(x_1, x_2) = 11 + x_2^2 - 10 \cdot \cos(2\pi x_2) \tag{19}$$
$$h(x_1, x_2) = \begin{cases} 1 - \sqrt{\frac{f_1(x_1, x_2}{g(x_1, x_2}}, & \text{if } f_1(x_1, x_2) \le g(x_1, x_2) \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

and $0 \le x_1 \le 1$, $-30 \le x_2 \le 30$.

In this example, the total number of fitness function evaluations was set to 4000.

Figs. 11 and 12 show the graphical results produced by our MOPSO, the microGA, the NSGA-II, and PAES in the third test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables IX, X, XI, and XII show the comparison of results among the four algorithms considering the metrics previously described. It can be seen that the average performance of MOPSO plays second with respect to



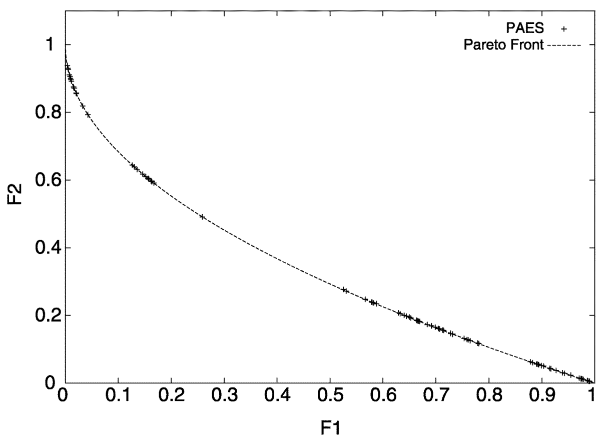
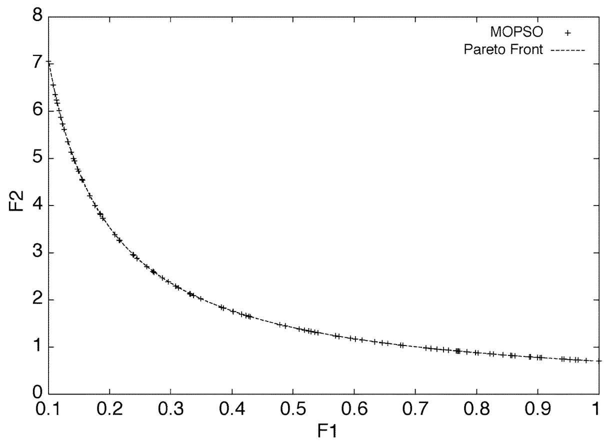


Fig. 12. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the third test function.



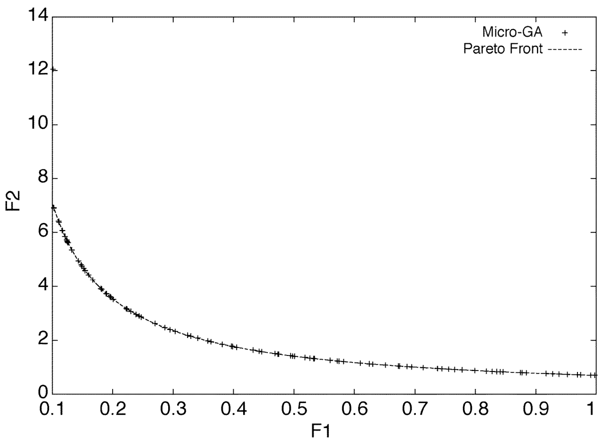


Fig. 13. Pareto fronts produced by our MOPSO (left) and the microGA (right) for the fourth test function. The true Pareto front is shown as a continuous line.

TABLE X
RESULTS OF THE GENERATIONAL DISTANCE METRIC FOR THE THIRD TEST FUNCTION

GD	MOPSO	NSGA-II	microGA	PAES
Best	8.61×10^{-5}	0.000133	8.74×10^{-5}	0.000114
Worst	0.000191	0.163146	0.811403	1.99851
Average	**0.000118**	0.023046	0.047049	0.163484
Median	0.000111	0.000418	0.000236	0.058896
Std. Dev.	2.55×10^{-5}	0.045429	0.181155	0.441303

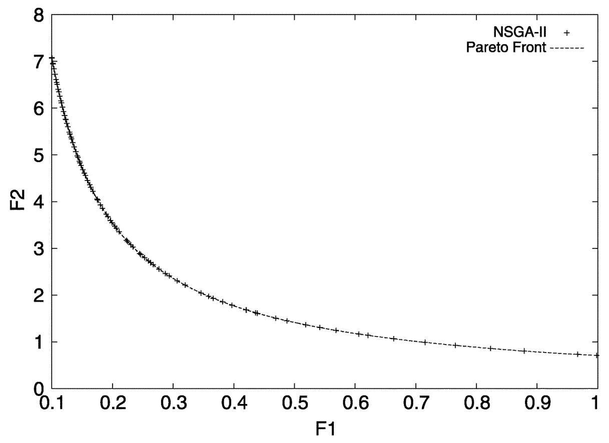
TABLE XI
RESULTS OF THE SPACING METRIC FOR THE THIRD TEST FUNCTION

SP	MOPSO	NSGA-II	microGA	PAES
Best	0.00727	0.000205	0.007596	0.009164
Worst	0.018676	0.010234	5.56727	19.8864
Average	0.010392	**0.00369**	0.341659	1.114617
Median	0.009542	0.002094	0.2995	0.018755
Std. Dev.	0.002782	0.003372	1.247561	4.434594

TABLE XII
COMPUTATIONAL TIME (IN SECONDS) REQUIRED BY EACH ALGORITHM FOR THE THIRD TEST FUNCTION

time	MOPSO	NSGA-II	microGA	PAES
Best	0.054	0.624	0.282	0.311
Worst	0.085	0.725	0.345	1.089
Average	**0.0721**	0.69355	0.30115	0.71525
Median	0.0725	0.6975	0.2995	0.73
Std. Dev.	0.00784	0.020028	0.016255	0.29049

the error ratio, but it is the best with respect to the generational distance. With respect to spacing it does considerably worse than the NSGA-II. However, graphical results again indicate that the NSGA-II does not cover the full Pareto front (it only covers about half of it). Since the nondomianted vectors found by the NSGA-II are clustered together, the spacing metric provides very good results. MOPSO, on the other hand, covers the entire Pareto front and is 10 times faster than the NSGA-II in this test function.



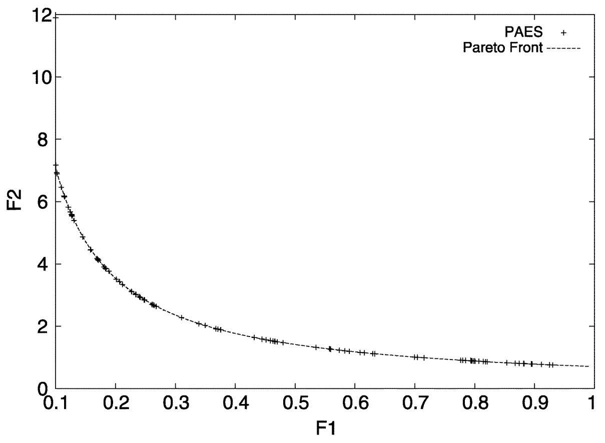


Fig. 14. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the fourth test function.

TABLE XIII
RESULTS OF THE ERROR RATIO METRIC FOR THE FOURTH TEST FUNCTION

ER	MOPSO	NSGA-II	microGA	PAES
Best	0.0	0.02	0.08	0.02
Worst	1.01	1.01	1.01	1.01
Average	0.25658	0.41450	**0.25200**	0.48900
Median	0.045	0.115	0.16	0.28
Std. Dev.	0.400658	0.459387	0.231576	0.438117

TABLE XV
RESULTS OF THE SPACING METRIC FOR THE FOURTH TEST FUNCTION

SP	MOPSO	NSGA-II	microGA	PAES
Best	0.04007	0.026086	0.030267	0.047844
Worst	0.58185	0.061422	0.817642	0.664676
Average	0.08358	**0.037447**	0.213584	0.194767
Median	0.05494	0.035529	0.06301	0.070365
Std. Dev.	0.11821	0.009238	0.250586	0.204687

TABLE XIV
RESULTS OF THE GENERATIONAL DISTANCE METRIC FOR THE FOURTH TEST FUNCTION

GD	MOPSO	NSGA-II	microGA	PAES
Best	0.00043	0.0007	0.000465	0.000453
Worst	0.18531	0.208467	0.183501	0.221671
Average	**0.03273**	0.044236	0.043466	0.194767
Median	0.00051	0.000856	0.050042	0.070365
Std. Dev.	0.06062	0.07368	0.048212	0.204687

TABLE XVI
COMPUTATIONAL TIME (IN SECONDS) REQUIRED BY EACH ALGORITHM FOR THE FOURTH TEST FUNCTION

time	MOPSO	NSGA-II	microGA	PAES
Best	0.214	1.426	0.136	0.777
Worst	0.379	1.653	0.158	1.236
Average	0.27675	1.5871	**0.1437**	1.0544
Median	0.2535	1.633	0.142	1.0605
Std. Dev.	0.054883	0.073463	0.005667	0.094123

VIII. TEST FUNCTION 4

Our fourth test function was also proposed by Deb [8]

$$\text{Minimize } f_1(x_1, x_2) = x_1 \quad (21)$$

$$\text{Minimize } f_2(x_1, x_2) = \frac{g(x_2)}{x_1} \quad (22)$$

$$g(x_2) = 2.0 - \exp\left\{-\left(\frac{x_2 - 0.2}{0.004}\right)^2\right\} - 0.8\exp\left\{-\left(\frac{x_2 - 0.6}{0.4}\right)^2\right\} \quad (23)$$

and $0.1 \leq x_1 \leq 1.0$, $0.1 \leq x_2 \leq 1.0$.

In this example, the total number of fitness function evaluations was set to 10 000.

Figs. 13 and 14 show the graphical results produced by the PAES, the NSGA-II, the microGA, and our MOPSO in the fourth test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables XIII, XIV, XV, and XVI show the comparison of results among the four algorithms considering the metrics previously described. It can be seen that the average performance of MOPSO is the best with respect to the generational distance, and it is slightly below the microGA with respect to the error ratio. With respect to spacing it does considerably worse than the NSGA-II, but the NSGA-II is

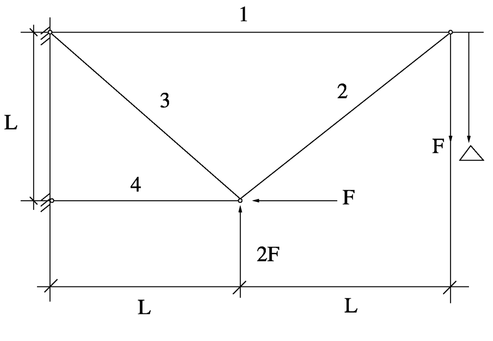


Fig. 15. Plane truss used for the fifth test function. The structural volume and the joint displacement (Δ) are to be optimized.

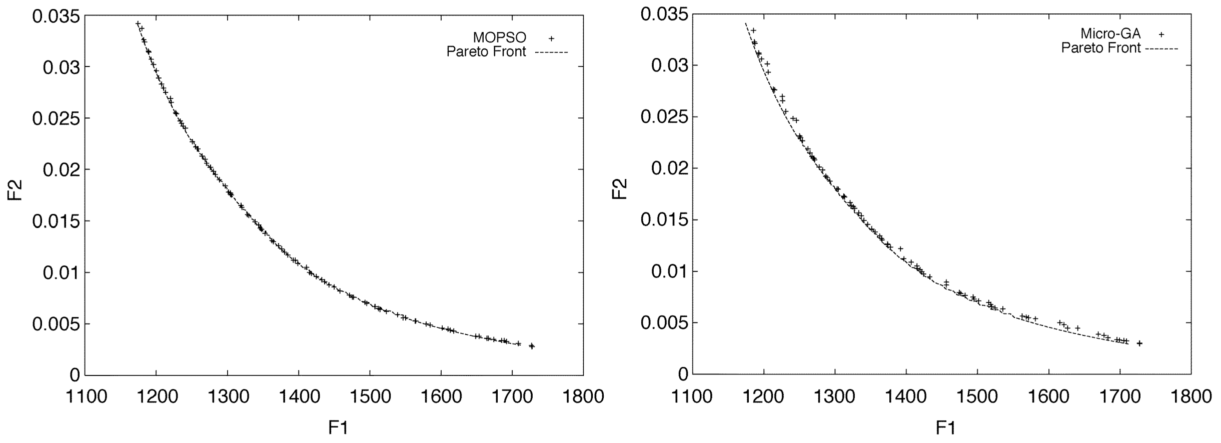


Fig. 16. Pareto fronts produced by our MOPSO (left) and the microGA (right) for the fifth test function. The true Pareto front is shown as a continuous line.

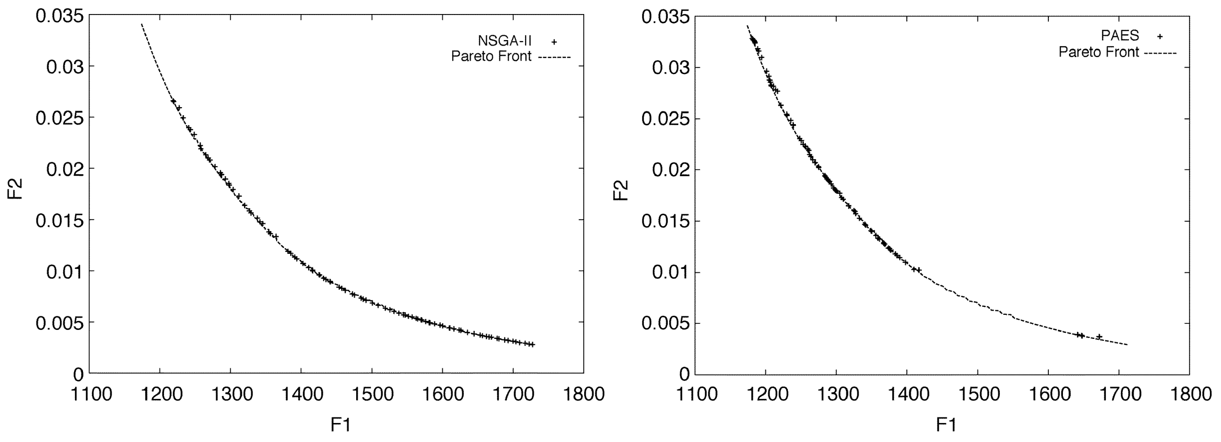


Fig. 17. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the fifth test function.

again unable to cover the full Pareto front of the problem (i.e., the spacing metric becomes misleading in this case). Note that this is the only example (from those presented in this paper) in which the microGA is faster than MOPSO (it requires about half the CPU time of MOPSO).

IX. TEST FUNCTION 5

Our fifth test function is to optimize the four-bar plane truss shown in Fig. 15. The problem is the following [2]:

Minimize

$$f_1(\mathbf{x}) = L(2x_1 + \sqrt{2x_2} + \sqrt{x_3} + x_4) \tag{24}$$

$$f_2(\mathbf{x}) = \frac{FL}{E}\left(\frac{2}{x_2} + \frac{2\sqrt{2}}{x_2} - \frac{2\sqrt{(2)}}{x_3} + \frac{2}{x_4}\right) \tag{25}$$

such that

$$\left(\frac{F}{\sigma}\right) \le x_1 \le 3 \times \left(\frac{F}{\sigma}\right)$$
$$\sqrt{2}\left(\frac{F}{\sigma}\right) \le x_2 \le 3 \times \left(\frac{F}{\sigma}\right)$$
$$\sqrt{2}\left(\frac{F}{\sigma}\right) \le x_3 \le 3 \times \left(\frac{F}{\sigma}\right)$$
$$\left(\frac{F}{\sigma}\right) \le x4 \le 3 \times \left(\frac{F}{\sigma}\right)$$

TABLE XVII
RESULTS OF THE ERROR RATIO METRIC FOR THE FIFTH TEST FUNCTION

ER	MOPSO	NSGA-II	microGA	PAES
Best	0.07	0.11	0.53	0.09
Worst	0.35	0.96	1.01351	0.39
Average	0.22550	0.35200	0.89836	**0.21600**
Median	0.235	0.23	0.989689	0.22
Std. Dev.	0.063534	0.26365	0.163529	0.078967

TABLE XVIII
RESULTS OF THE GENERATIONAL DISTANCE METRIC FOR THE FIFTH TEST FUNCTION

GD	MOPSO	NSGA-II	microGA	PAES
Best	0.306646	0.250678	0.263584	0.149926
Worst	0.462209	0.462011	7.11616	6.04106
Average	0.374129	**0.360182**	0.91025	0.973388
Median	0.3679	0.368241	0.383652	0.232008
Std. Dev.	0.042228	0.047016	1.705316	1.821166

where $F = 10$ kN, $E = 2 \times 10^5$ kN/cm^2, $L = 200$ cm $\sigma = 10$ kN/cm^3. Using these values, the ranges adopted for the decision variables are the following: $0.05 \le x_1 \le 0.15$,

TABLE XIX
RESULTS OF THE SPACING METRIC FOR THE FIFTH TEST FUNCTION

SP	MOPSO	NSGA-II	microGA	PAES
Best	2.13706	1.80346	2.20101	5.95558
Worst	2.97969	2.7608	70.1234	28.4336
Average	2.530339	**2.363556**	8.274281	3.231429
Median	2.501015	2.40616	3.11687	1.812675
Std. Dev.	0.227512	0.255117	16.83111	5.95558

TABLE XX
COMPUTATIONAL TIME (IN SECONDS) REQUIRED BY EACH ALGORITHM FOR THE FIFTH TEST FUNCTION

time	MOPSO	NSGA-II	microGA	PAES
Best	0.133	1.814	0.151	2.259
Worst	0.152	2.083	0.178	2.562
Average	**0.1443**	1.9317	0.1657	2.38285
Median	0.144	1.9055	0.168	2.376
Std. Dev.	0.05823	0.090874	0.008221	0.074412

TABLE XXI
RESULTS OF EXPERIMENT 1 FOR THE FIRST TEST FUNCTION

	ER		GD		SP	
	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)
Best	0.49	0.08	0.00349891	0.00242521	0.04247	0.04398
Worst	1.11111	0.27	1.82596	0.476815	3.01586	0.53810
Average	0.85441	**0.13253**	0.45625	**0.03653**	0.72153	**0.10945**
Median	0.89483	0.14	0.08953	0.00785	0.2008595	0.0674799
Std. Dev.	0.23121	0.04501	0.60545	0.10459	0.88902	0.11005

MOPSO (no) refers to the version of our approach without mutation and **MOPSO (yes)** refers to the version with mutation. ER = error ratio, GD = generational distance, and SP = spacing.

$0.070\,710\,678 \leq x_2 \leq 0.15$, $0.070\,710\,678 \leq x_3 \leq 0.15$, $0.05 \leq x_4 \leq 0.15$.

In this example, the total number of fitness function evaluations was set to 8000.

Figs. 16 and 17 show the graphical results produced by the PAES, the NSGA-II, the microGA, and our MOPSO in the fifth test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables XVII, XVIII, XIX, and XX show the comparison of results among the four algorithms considering the metrics previously described. In this case, MOPSO is the second best with respect to the three metrics, but only marginally. Note however, that only MOPSO covers the entire Pareto front of the problem. Furthermore, in terms of CPU time, MOPSO is about 14 times faster than the NSGA-II and its remarkable speed is only comparable to the microGA which, however, has a much poorer performance with respect to the three metrics adopted.

X. SENSITIVITY ANALYSIS

We performed an extensive analysis of the impact of the parameters of our MOPSO on its performance. In this paper, we used the five test functions previously described, as well as the three metrics adopted before. We performed four experiments.

TABLE XXII
RESULTS OF EXPERIMENT 1 FOR THE SECOND TEST FUNCTION

	ER		GD		SP	
	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)
Best	0.863636	0.18	0.04077	0.00745	0.01395	0.06187
Worst	1.2	0.37	0.22021	0.00960	3.09499	0.118445
Average	1.01719	**0.2535**	0.08285	**0.00845**	0.35047	**0.09747**
Median	1.01	0.255	0.06083	0.00845	0.18558	0.10396
Std. Dev.	0.05959	0.04082	0.05241	0.00051	0.67227	0.01675

MOPSO (no) refers to the version of our approach without mutation and **MOPSO (yes)** refers to the version with mutation. ER = error ratio, GD = generational distance, and SP = spacing.

TABLE XXIII
RESULTS OF EXPERIMENT 1 FOR THE THIRD TEST FUNCTION

	ER		GD		SP	
	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)
Best	0.15	0.19	8.27×10^{-5}	8.61×10^{-5}	0.00579	0.00727
Worst	0.59	0.55	0.00809	0.00019	0.08049	0.01867
Average	**0.31**	0.3335	0.00059	**0.00012**	0.01309	**0.01039**
Median	0.295	0.3	0.00013	0.00011	0.00905	0.00954
Std. Dev.	0.10458	0.09388	0.00178	2.55×10^{-5}	0.01603	0.00279

MOPSO (no) refers to the version of our approach without mutation and **MOPSO (yes)** refers to the version with mutation. ER = error ratio, GD = generational distance, and SP = spacing.

TABLE XXIV
RESULTS OF EXPERIMENT 1 FOR THE FOURTH TEST FUNCTION

	ER		GD		SP	
	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)
Best	0.01	0.0	0.00044	0.00043	0.03889	0.04007
Worst	1.01	1.01	0.21435	0.18531	0.42262	0.58185
Average	0.76298	**0.25658**	0.08694	**0.03273**	0.14773	**0.08358**
Median	1.01	0.045	0.09038	0.00051	0.11322	0.05494
Std. Dev.	0.43352	0.40066	0.06455	0.06062	0.10974	0.11821

MOPSO (no) refers to the version of our approach without mutation and **MOPSO (yes)** refers to the version with mutation. ER = error ratio, GD = generational distance, and SP = spacing.

TABLE XXV
RESULTS OF EXPERIMENT 1 FOR THE FIFTH TEST FUNCTION

	ER		GD		SP	
	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)	MOPSO (no)	MOPSO (yes)
Best	0.04	0.07	0.41863	0.30665	0.42447	2.13706
Worst	1.07143	0.35	80.1206	0.46221	162.678	2.97969
Average	0.57479	**0.2255**	18.22747	**0.37413**	29.67843	**2.53034**
Median	0.58852	0.235	5.65958	0.3679	11.00947	2.50102
Std. Dev.	0.49477	0.06353	23.19053	0.04223	41.83782	0.22751

MOPSO (no) refers to the version of our approach without mutation and **MOPSO (yes)** refers to the version with mutation. ER = error ratio, GD = generational distance, and SP = spacing.

TABLE XXVI
RESULTS OF EXPERIMENT 2 FOR THE FIRST TEST FUNCTION

ERROR RATIO

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.28	0.54	0.39	0.08	0.05	0.06
Worst	1.01	1	0.980769	0.27	0.18	0.18
Average	0.45415	0.72810	0.75574	0.13253	**0.11729**	0.1198
Median	0.40707	0.72	0.79545	0.14	0.125	0.12
Std. Dev.	0.18113	0.11255	0.13408	0.04501	0.04026	0.03559

GENERATIONAL DISTANCE

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.00356	0.00512	0.00563	0.00242	0.00235	0.00243
Worst	1.77886	0.22063	0.21974	0.47682	0.07275	0.15988
Average	0.10774	0.05913	0.05340	0.03653	**0.01609**	0.02372
Median	0.00509	0.02902	0.02986	0.00785	0.00668	0.01322
Std. Dev.	0.39607	0.06338	0.05917	0.10459	0.01812	0.03579

SPACING

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.05675	0.05905	0.05817	0.04398	0.04239	0.04835
Worst	0.21458	1.0411	1.1096	0.53810	0.72031	0.66442
Average	0.09535	0.29755	0.31191	**0.10945**	0.12982	0.17126
Median	0.08607	0.17277	0.19864	0.06748	0.06147	0.10217
Std. Dev.	0.04063	0.30204	0.30952	0.11005	0.16127	0.18638

We analyze the effect of the number of divisions on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

1) **Experiment 1**: We compared MOPSO with mutation versus MOPSO without mutation using the original set of parameters (those adopted in Section V).
2) **Experiment 2**: We varied the number of divisions of the adaptive grid used in the secondary population. We performed runs using 5, 10, 20, 30, 40, and 50 divisions.
3) **Experiment 3**: We modified the number of particles of the swarm, and the number of iterations in order to perform the same number of evaluations of the objective functions as in the original experiments. We performed runs using 5, 25, 75, and 100 particles. All the other parameters were left as defined in Section V.
4) **Experiment 4**: We modified the size of the particle repository (the secondary population of our algorithm). We performed runs using 100, 150, 200, and 250 particles. All the other parameters were left as defined in Section V.

The results obtained from each experiment are discussed in the following sections.

A. *Experiment 1*

This experiment was designed to determine if the mutation operator adopted really played an important role in our MOPSO. We compared MOPSO without mutation versus MOPSO with mutation using the same test functions and metrics described in Section V. The following is a summary of the results obtained.

1) **Test Function 1**: In this case, the use of the mutation operator clearly improved the results of all the metrics adopted (see Table XXI).
2) **Test Function 2**: Again, the use of mutation produced a significant improvement for our MOPSO with respect to all the metrics adopted (see Table XXII).
3) **Test Function 3**: This is an interesting example, because the version without mutation produced a slightly better average result for the error ratio metric (0.31 versus 0.3335). However, as can be seen in Table XXIII, the use of mutation produces improvements (although such improvements tend to be marginal as well) for the two other metrics. We attribute this behavior to the fact that this test function has a search space considerably easier to explore (i.e., less accidented) than the others adopted in this paper. That is why we believe that mutation does not produce an important difference in this case.
4) **Test Function 4**: Again, the use of mutation produced a significant improvement for our MOPSO with respect to the three metrics adopted (see Table XXIV).
5) **Test Function 5**: Once more, the use of mutation produced a significant improvement for our MOPSO with respect to the three metrics adopted (see Table XXV). In fact, this is the example where the improvements produced by the mutation operator are more evident.

TABLE XXVII
RESULTS OF EXPERIMENT 2 FOR THE SECOND TEST FUNCTION

ERROR RATIO

Cell divisions	5	10	20	30	40	50
Best	0.2	0.19	0.18	0.18	0.18	0.14
Worst	0.42	0.35	0.37	0.33	0.29	0.23
Average	0.30318	0.285	0.2535	0.24762	0.22	**0.20510**
Median	0.3	0.295	0.255	0.24	0.22	0.21
Std. Dev.	0.06381	0.04261	0.04082	0.03638	0.02956	0.02437

GENERATIONAL DISTANCE

Cell divisions	5	10	20	30	40	50
Best	0.00697	0.00768	0.00745	0.00780	0.00769	0.00801
Worst	0.01075	0.00934	0.00960	0.01309	0.00931	0.00956
Average	0.00849	**0.00828**	0.00845	0.00889	0.00857	0.00865
Median	0.00862	0.00819	0.00845	0.00851	0.00841	0.00868
Std. Dev.	0.00093	0.00042	0.00051	0.00122	0.00046	0.00040

SPACING

Cell divisions	5	10	20	30	40	50
Best	0.07320	0.06548	0.06187	0.06295	0.08090	0.07978
Worst	0.14142	0.12332	0.11845	0.13972	0.13736	0.14486
Average	0.10787	0.09961	0.09747	**0.08995**	0.11222	0.11151
Median	0.11129	0.10683	0.10396	0.09610	0.11287	0.10892
Std. Dev.	0.02134	0.01827	0.01675	0.02116	0.01319	0.015523

We analyze the effect of the number of divisions on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

1) Conclusions From Experiment 1: Based on the analysis performed in this experiment, we conclude that the use of mutation turns out to be beneficial in most cases, and only marginally harmful when the problem is very simple to solve. Note however, that since the difference is almost negligible, we recommend to use mutation with our MOPSO in all cases.

B. Experiment 2

As indicated before, in this experiment, we varied the number of divisions of the adaptive grid used in the secondary population. We performed runs using 5, 10, 20, 30, 40, and 50 divisions to see the effect of this parameter in the performance of our MOPSO. The following is a summary of the results obtained.

1) **Test Function 1**: In this case, we found that the results improved as we increase the number of divisions, but after reaching 40 divisions, the results started degrading again. In Table XXVI, we can see that in this test function the best average results were obtained with 40 divisions for both the error ratio and the generational distance metrics and with 30 divisions for the spacing metric. However, in all cases, the difference between the average results obtained with 30 and 40 divisions is marginal.
2) **Test Function 2**: In this case, the difference in the results is only marginal in most cases regardless of the number of divisions adopted. In Table XXVII, we can see that 50 divisions produced the best average result with respect to error ratio, 10 divisions produced the best average result with respect to generational distance, and 30 divisions produced the best average result with respect to spacing.
3) **Test Function 3**: In this case, a number of divisions greater or equal than 30 provided the best results with respect to all the metrics adopted. In Table XXVIII, we can see that 40 divisions provided the best average result with respect to error ratio, 30 divisions produced the best average result with respect to generational distance, and 40 divisions produced the best average result with respect to spacing.
4) **Test Function 4**: This was an atypical function in which a number of divisions of 10 provided the best average results with respect to all the metrics adopted (see Table XXIX). However, a number of divisions of 30 was the second best in all cases.
5) **Test Function 5**: In this case, 30 divisions provided the best average results with respect to both generational distance and spacing (see Table XXX). With respect to error ratio, 20 divisions provided a better average result, but the difference with respect to 30 divisions is very small.

1) Conclusions From Experiment 2: From the results obtained from this experiment, we can see that in most cases a value greater or equal than 30 divisions provided good results.

TABLE XXVIII
RESULTS OF EXPERIMENT 2 FOR THE THIRD TEST FUNCTION

ERROR RATIO

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.21	0.27	0.2	0.19	0.09	0.17
Worst	0.68	0.72	0.57	0.55	0.42	0.46
Average	0.38909	0.47718	0.36158	0.33350	**0.29263**	0.30316
Median	0.34	0.5	0.35	0.3	0.31	0.31
Std. Dev.	0.14866	0.14029	0.08668	0.09388	0.08054	0.06351

GENERATIONAL DISTANCE

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	7.98×10^{-5}	8.69×10^{-5}	8.60×10^{-5}	8.61×10^{-5}	8.46×10^{-5}	7.46×10^{-5}
Worst	0.00177	0.02428	0.01489	0.00019	0.00022	0.00023
Average	0.00033	0.00242	0.00091	**0.00012**	0.00013	0.00013
Median	0.00016	0.00022	0.00012	0.00011	0.00013	0.00010
Std. Dev.	0.00045	0.00618	0.00339	2.55×10^{-5}	4.41×10^{-5}	5.53×10^{-5}

SPACING

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.00805	0.00712	0.00505	0.00727	0.00739	0.00520
Worst	0.01868	0.15427	0.14811	0.01868	0.01613	0.01731
Average	0.01203	0.01871	0.01724	0.01039	**0.01013**	0.01174
Median	0.01132	0.01059	0.01044	0.00954	0.00932	0.01116
Std. Dev.	0.00267	0.03210	0.03176	0.00279	0.00249	0.00349

We analyze the effect of the number of divisions on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

We noted that even in those cases in which a lower number of divisions was better (e.g., 10 divisions), 30 divisions remained as a competitive value, we concluded that the value of 30 was the most suitable for this parameter of our MOPSO.

C. Experiment 3

As indicated before, in this experiment, we varied the number of particles of the swarm (or primary population) of our MOPSO. We also had to vary the number of iterations as to maintain the same (total) number of fitness function evaluations of the original experiments. We performed runs using 5, 25, 75, and 100 particles. The following is a summary of the results obtained.

1) **Test Function 1**: In this case, we obtained mixed results. With respect to error ratio, the use of 100 particles provided the best average result. However, with respect to both generational distance and spacing, the use of only five particles provided the best average results. However, it is worth noting in Table XXXI that the differences obtained when increasing the number of particles are not too big. The explanation for these results has to do with the characteristics of this problem. In this case, a swarm of smaller size is better because it uses a larger number of cycles and, therefore, has a better chance of converging to the true Pareto front of this problem (which is difficult to reach by most algorithms). With fewer particles, may be also easier to obtain a better (i.e., more uniform) distribution of solutions.
2) **Test Function 2**: Table XXXII shows that, in this case, the use of 100 particles provides the best average results with respect to both error ratio and generational distance. With respect to spacing, the use of 25 particles provided a better average result, but the difference with respect to the use of 100 particles is negligible.
3) **Test Function 3**: This is another case with mixed results. We can see in Table XXXIII that the use of 25 particles provided the best average result with respect to error ratio, the use of 50 particles provided the best average result with respect to generational distance and the use of five particles provided the best average result with respect to spacing. Note however, that in the case of both generational distance and spacing, the use of 100 particles provided very competitive results.
4) **Test Function 4**: We can see in Table XXXIV that in this case, 100 particles provided the best average results with respect to both error ratio and generational distance. With respect to spacing, the use of 75 particles provided the best average result, although the results obtained with 100 particles are not too different.

TABLE XXIX
RESULTS OF EXPERIMENT 2 FOR THE FOURTH TEST FUNCTION

ERROR RATIO

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.01	0.0	0.0	0.0	0.0	0.03
Worst	1.01	1.01	1.01	1.01	1.01	1.01
Average	0.58631	**0.18100**	0.34600	0.25658	0.66142	0.85493
Median	0.93299	0.04	0.06	0.045	0.98958	1.01
Std. Dev.	0.48109	0.35683	0.45142	0.40066	0.44671	0.34145

GENERATIONAL DISTANCE

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.00049	0.00040	0.00039	0.00043	0.00044	0.00045
Worst	0.591756	0.164373	0.156151	0.185306	0.303403	0.390608
Average	0.11665	**0.02338**	0.04609	0.03273	0.10639	0.15231
Median	0.13933	0.00054	0.00051	0.00051	0.12305	0.14304
Std. Dev.	0.13692	0.05597	0.06643	0.06062	0.09447	0.09062

SPACING

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.04078	0.04012	0.03507	0.04007	0.04370	0.05807
Worst	0.56897	0.10237	0.57664	0.58185	0.16255	0.19214
Average	0.10366	**0.06207**	0.08479	0.08358	0.08959	0.09852
Median	0.08074	0.05985	0.06026	0.05494	0.09173	0.08426
Std. Dev.	0.11193	0.01436	0.11626	0.11821	0.03121	0.03650

We analyze the effect of the number of divisions on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

5) **Test Function 5**: In this problem, we have again mixed results. We can see in Table XXXV that the use of 50 particles provided the best average result with respect to error ratio, the use of 100 particles provided the best average result with respect to generational distance, and the use of 5 particles provided the best average result with respect to spacing. However, once again, the average results obtained with 100 particles remain competitive in terms of both error ratio and spacing.

1) Conclusions From Experiment 3: Although the results obtained from this experiment seem inconclusive, we argue that the use of 100 particles is a reasonable choice if nothing is known about the problem to be solved (we obtained competitive results in most cases when adopting this value).

D. Experiment 4

As indicated before, in this final experiment, we modified the size of the particle repository (the secondary population of our algorithm). This parameter refers to the expected number of points (i.e., nondominated vectors) that our algorithm will find. We performed runs using 100, 150, 200, and 250 particles. The following is a summary of the results obtained.

1) **Test Function 1**: As we can see in Table XXXVI, in this case, a value of 100 for the size of the repository provided the best average results with respect to both error ratio and spacing. A value of 150 provided a better average result with respect to generational distance.
2) **Test Function 2**: As we can see in Table XXXVII, a value of 250 for the size of the repository provided in this case the best average results with respect to both error ratio and spacing. A value of 200 provided the best average result with respect to generational distance.
3) **Test Function 3**: As we can see in Table XXXVIII, a value of 250 for the size of the repository provided in this case the best average results with respect to all the metrics considered.
4) **Test Function 4**: In Table XXXIX, we can see that again, a value of 250 for the size of the repository provided in this case the best average results with respect to all the metrics considered.
5) **Test Function 5**: In Table XL, we can see that again, a value of 250 for the size of the repository provided in this case the best average results with respect to all the metrics considered.

1) Conclusions From Experiment 4: We can see that in this case, a value of 250 for the size of the repository provided the best average results in most problems. Note however, that as we increase the size of the external repository, the search effort required to converge to a good (and well-distributed) approxima-

TABLE XXX
RESULTS OF EXPERIMENT 2 FOR THE FIFTH TEST FUNCTION

ERROR RATIO

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.1	0.13	0.11	0.07	0.17	0.15
Worst	0.45	0.61	0.31	0.35	0.32	0.38
Average	0.2325	0.2515	**0.216**	0.2255	0.242	0.2245
Median	0.22	0.24	0.21	0.235	0.24	0.205
Std. Dev.	0.08303	0.11301	0.04946	0.06353	0.03736	0.06320

GENERATIONAL DISTANCE

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	0.23876	0.31383	0.30815	0.30665	0.31681	0.31458
Worst	0.85455	1.71286	0.60339	0.46221	2.45391	2.09210
Average	0.42063	0.47934	0.41192	**0.37413**	0.60619	0.47138
Median	0.40279	0.38630	0.38164	0.36790	0.40450	0.37482
Std. Dev.	0.12363	0.32707	0.08635	0.04223	0.50437	0.38414

SPACING

Cell divisions	**5**	**10**	**20**	**30**	**40**	**50**
Best	2.31593	2.65061	2.15975	2.13706	1.98569	1.93643
Worst	6.625	7.84865	3.61638	2.97969	8.98066	18.6239
Average	3.47514	3.42916	2.75623	**2.53034**	3.12892	3.20783
Median	3.31644	3.15755	2.69052	2.50102	2.24099	2.46743
Std. Dev.	0.88237	1.09678	0.41661	0.22751	2.23027	3.63706

We analyze the effect of the number of divisions on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

tion of the true Pareto front tends to increase as well. This fact is, however, evident mainly in problems in which reaching the true Pareto front is particularly difficult (e.g., the first test function from Section V). That is one of the main reasons why we decided to adopt a value of 100 for this parameter. Additionally, in the specialized literature, a size of 100 for the external population has been a common practice [7]. Nevertheless, as we saw in this analysis, our MOPSO can improve its results when using a larger repository (although, in some cases, the improvement is only marginal).

E. Parameters Recommended

Based on the experimental study conducted, we found that the following values for the parameters of our MOPSO provide the most competitive results:

- **Number of particles**: This is equivalent to the population size of a genetic algorithm. Obviously, a larger number of particles involves a higher computational cost. We recommend to use 100 particles.
- **Number of cycles**: This parameter is related to the number of particles. The relationship tends to be inversely proportional (i.e., to larger number of particles, smaller number of cycles, and *vice versa*). We recommend to use between 80 and 120. The number of cycles is related to the complexity of the problem (i.e., more difficult problems may require more cycles). However, it is important to keep in mind that for a constant number of particles, as we increase the number of cycles, the computational cost of the method also increases. If nothing is known about the problem, we suggest to use 100 cycles (adopting 100 particles for the swarm).
- **Number of divisions**: It allows us to determine the number of hypercubes that will be generated in objective function space. We recommend to use 30 divisions, since this value provided good results in most cases (see Section X).
- **Size of the repository**: This parameter is used to delimit the maximum number of nondominated vectors that can be stored in the repository. The value of this parameter will determine the quality of the Pareto front produced. We recommend to use 250 particles (see Section X). However, since it is normally common practice that multiobjective evolutionary algorithms that use an external memory similar to our own use only a size of 100, this value may be an alternative to facilitate (indirect) comparisons that other authors wish to perform. That is the reason why we adopted such value in the study presented in this paper.

TABLE XXXI
RESULTS OF EXPERIMENT 3 FOR THE FIRST TEST FUNCTION

ERROR RATIO

Particles	5	25	50	75	100
Best	0.24	0.42424	0.37	0.47	0.08
Worst	0.6	0.82759	0.77778	0.73	0.27
Average	0.5075	0.54346	0.57114	0.57197	**0.13253**
Median	0.53	0.535	0.595	0.57288	0.14
Std. Dev.	0.08540	0.08065	0.09173	0.06391	0.04501

GENERATIONAL DISTANCE

Particles	5	25	50	75	100
Best	0.00126	0.00259	0.00287	0.00259	0.00243
Worst	0.03149	0.01526	0.05831	0.07369	0.47682
Average	**0.00634**	0.00712	0.02057	0.00978	0.03653
Median	0.00432	0.00563	0.01294	0.00503	0.00785
Std. Dev.	0.00747	0.00387	0.01834	0.01596	0.10459

SPACING

Particles	5	25	50	75	100
Best	0.02307	0.04656	0.04960	0.04983	0.04398
Worst	0.22192	0.18141	0.52492	0.71090	0.53810
Average	**0.06898**	0.07565	0.18725	0.10842	0.10945
Median	0.05786	0.06587	0.13833	0.06655	0.06748
Std. Dev.	0.03924	0.03112	0.15247	0.14883	0.11005

We analyze the effect of the number of particles of the swarm on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XXXII
RESULTS OF EXPERIMENT 3 FOR THE SECOND TEST FUNCTION

ERROR RATIO

Particles	5	25	50	75	100
Best	0.15	0.17	0.21	0.18	0.18
Worst	0.31	0.34	0.34	0.3	0.2476212
Average	0.24313	0.2485	0.259	0.2465	**0.2400**
Median	0.235	0.25	0.26	0.25	0.03637
Std. Dev.	0.04228	0.05153	0.03878	0.03645	0.33

GENERATIONAL DISTANCE

Particles	5	25	50	75	100
Best	0.00787	0.00796	0.00782	0.00809	0.00780
Worst	0.01177	0.01369	0.01019	0.01061	0.00888
Average	0.00894	0.00938	0.00871	0.00891	**0.00850**
Median	0.00863	0.00864	0.00860	0.00888	0.00121
Std. Dev.	0.00088	0.00156	0.00070	0.00062	0.01308

SPACING

Particles	5	25	50	75	100
Best	0.06121	0.05637	0.06087	0.05174	0.06295
Worst	0.11862	0.11761	0.11587	0.11817	0.08995
Average	0.09271	**0.08284**	0.08601	0.08617	0.09609
Median	0.09778	0.08467	0.09026	0.08768	0.02115
Std. Dev.	0.01657	0.02083	0.01847	0.01960	0.13971

We analyze the effect of the number of particles of the swarm on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XXXIII
RESULTS OF EXPERIMENT 3 FOR THE THIRD TEST FUNCTION

ERROR RATIO

Particles	5	25	50	75	100
Best	0	0.02	0.08	0.09	0.19
Worst	1.01	0.21	0.24	0.36	0.55
Average	0.16600	**0.09800**	0.16500	0.20750	0.33350
Median	0.115	0.095	0.17	0.19	0.3
Std. Dev.	0.21117	0.05258	0.04583	0.07326	0.09388

GENERATIONAL DISTANCE

Particles	5	25	50	75	100
Best	0.00007	0.00007	0.00008	0.00009	0.00009
Worst	0.05353	0.00071	0.00018	0.00021	0.00019
Average	0.00279	0.00014	**0.00011**	0.00013	0.00012
Median	0.00011	0.00010	0.00011	0.00011	0.00011
Std. Dev.	0.01194	0.00014	0.00003	0.00004	0.00003

SPACING

Particles	5	25	50	75	100
Best	0.00002	0.00532	0.00667	0.00708	0.00727
Worst	0.01354	0.04091	0.01238	0.01290	0.01868
Average	**0.00904**	0.01111	0.00960	0.00919	0.01039
Median	0.00952	0.00970	0.00956	0.00914	0.00954
Std. Dev.	0.00300	0.00718	0.00129	0.00125	0.00279

We analyze the effect of the number of particles of the swarm on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XXXIV
RESULTS OF EXPERIMENT 3 FOR THE FOURTH TEST FUNCTION

ERROR RATIO

Particles	5	25	50	75	100
Best	0.01	0	0	0	0
Worst	1.01	1.01	1.01	1.01	1.01
Average	0.86500	0.71200	0.39766	0.49818	**0.25658**
Median	1.01	1.01	0.1	0.341804	0.045
Std. Dev.	0.35432	0.46706	0.45827	0.48586	0.40066

GENERATIONAL DISTANCE

Particles	5	25	50	75	100
Best	0.00048	0.00046	0.00045	0.00045	0.00043
Worst	0.14645	0.14212	0.37819	0.16711	0.18531
Average	0.10816	0.09075	0.05838	0.06060	**0.03273**
Median	0.12709	0.12435	0.00059	0.00127	0.00051
Std. Dev.	0.04866	0.06091	0.09647	0.06872	0.06062

SPACING

Particles	5	25	50	75	100
Best	0.04904	0.03780	0.03579	0.02803	0.04007
Worst	0.11174	0.09069	0.36344	0.17196	0.58185
Average	0.07935	0.06785	0.07502	**0.06702**	0.08358
Median	0.08319	0.07096	0.05443	0.06242	0.05494
Std. Dev.	0.01712	0.01521	0.07090	0.02956	0.11821

We analyze the effect of the number of particles of the swarm on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XXXV
RESULTS OF EXPERIMENT 3 FOR THE FIFTH TEST FUNCTION

ERROR RATIO

Particles	**5**	**25**	**50**	**75**	**100**
Best	0.12	0.08	0.14	0.15	0.07
Worst	1.01	0.35	0.41	0.31	0.35
Average	0.31950	0.22700	**0.21250**	0.22200	0.22550
Median	0.26	0.24	0.205	0.215	0.235
Std. Dev.	0.23887	0.07292	0.05999	0.04948	0.06353

GENERATIONAL DISTANCE

Particles	**5**	**25**	**50**	**75**	**100**
Best	0.31270	0.29963	0.33139	0.29349	0.30665
Worst	0.50351	1.34638	0.60252	4.62551	0.46221
Average	0.37668	0.40763	0.39296	0.56956	**0.37413**
Median	0.35866	0.35980	0.36643	0.35580	0.36790
Std. Dev.	0.05112	0.22259	0.06636	0.95523	0.04223

SPACING

Particles	**5**	**25**	**50**	**75**	**100**
Best	1.80515	1.75635	1.87571	2.10319	2.13706
Worst	2.91673	3.30144	3.17307	28.43200	2.97969
Average	**2.38988**	2.59673	2.52440	3.80488	2.53034
Median	2.35045	2.62423	2.54916	2.53711	2.50102
Std. Dev.	0.29995	0.34920	0.29610	5.79950	0.22751

We analyze the effect of the number of particles of the swarm on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XXXVI
RESULTS OF EXPERIMENT 4 FOR THE FIRST TEST FUNCTION

ERROR RATIO

Expected points	**100**	**150**	**200**	**250**
Best	0.08	0.488722	0.49629	0.45323
Worst	0.27	0.66	0.78461	0.70270
Average	**0.13253**	0.58440	0.57143	0.56506
Median	0.14	0.584846	0.5509705	0.553914
Std. Dev.	0.04501	0.04158	0.07332	0.06467

GENERATIONAL DISTANCE

Expected points	**100**	**150**	**200**	**250**
Best	0.00243	0.00201	0.00207	0.00212
Worst	0.47682	0.09917	0.11489	0.09834
Average	0.03653	**0.01485**	0.02021	0.01976
Median	0.00785	0.00456	0.00457	0.01088
Std. Dev.	0.10459	0.02482	0.02983	0.02552

SPACING

Expected points	**100**	**150**	**200**	**250**
Best	0.04398	0.03174	0.02928	0.03643
Worst	0.53810	1.03857	1.23382	0.81578
Average	**0.10945**	0.13679	0.17168	0.18083
Median	0.06748	0.04995	0.05892	0.07891
Std. Dev.	0.11005	0.24057	0.27305	0.20594

We analyze the effect of the size of the external repository (i.e., the secondary population of our MOPSO) on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XXXVII
RESULTS OF EXPERIMENT 4 FOR THE SECOND TEST FUNCTION

ERROR RATIO

Expected points	**100**	**150**	**200**	**250**
Best	0.18	0.106667	0.16	0.16
Worst	0.37	0.273333	0.265	0.295
Average	0.25350	0.22567	0.21644	**0.21601**
Median	0.255	0.233333	0.2175	0.21
Std. Dev.	0.04082	0.03543	0.02666	0.03609

GENERATIONAL DISTANCE

Expected points	**100**	**150**	**200**	**250**
Best	0.00745	0.00635	0.00500	0.00520
Worst	0.00960	0.00868	0.00606	0.00643
Average	0.00845	0.00695	**0.00562**	0.00577
Median	0.00845	0.00686	0.00562	0.00581
Std. Dev.	0.00051	0.00049	0.00023	0.00033

SPACING

Expected points	**100**	**150**	**200**	**250**
Best	0.06187	0.04532	0.03444	0.03382
Worst	0.11845	0.08894	0.08050	0.08372
Average	0.09747	0.06817	0.06134	**0.06098**
Median	0.10396	0.07113	0.06701	0.06824
Std. Dev.	0.01675	0.01655	0.01512	0.01632

We analyze the effect of the size of the external repository (i.e., the secondary population of our MOPSO) on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XXXVIII
RESULTS OF EXPERIMENT 4 FOR THE THIRD TEST FUNCTION

ERROR RATIO

Expected points	**100**	**150**	**200**	**250**
Best	0.19	0.233333	0.245	0.216
Worst	0.55	0.44	0.415	0.44
Average	0.33350	0.34133	0.32400	**0.32169**
Median	0.3	0.3566665	0.325	0.316
Std. Dev.	0.09388	0.06301	0.03932	0.05781

GENERATIONAL DISTANCE

Expected points	**100**	**150**	**200**	**250**
Best	0.00009	0.00007	0.00007	0.00005
Worst	0.00019	0.00026	0.00056	0.00015
Average	0.00012	0.00012	0.00012	**0.00008**
Median	0.00011	0.00011	0.00008	0.00007
Std. Dev.	0.00003	0.00005	0.00013	0.00003

SPACING

Expected points	**100**	**150**	**200**	**250**
Best	0.00727	0.00577	0.00436	0.00342
Worst	0.01868	0.00765	0.00905	0.00447
Average	0.01039	0.00641	0.00513	**0.00400**
Median	0.00954	0.00638	0.00469	0.00406
Std. Dev.	0.00279	0.00051	0.00113	0.00033

We analyze the effect of the size of the external repository (i.e., the secondary population of our MOPSO) on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XXXIX
RESULTS OF EXPERIMENT 4 FOR THE FOURTH TEST FUNCTION

ERROR RATIO				
Expected points	**100**	**150**	**200**	**250**
Best	0	0	0.005	0.008
Worst	1.01	1.00667	1.00505	1.004
Average	0.25658	0.46418	0.36650	**0.22180**
Median	0.045	0.2433335	0.03	0.028
Std. Dev.	0.40066	0.47132	0.48078	0.40142
GENERATIONAL DISTANCE				
Expected points	**100**	**150**	**200**	**250**
Best	0.00043	0.00035	0.00033	0.00031
Worst	0.18531	0.11823	0.11565	0.08754
Average	0.03273	0.04547	0.03478	**0.01751**
Median	0.00051	0.00051	0.00039	0.00034
Std. Dev.	0.06062	0.05263	0.04832	0.03523
SPACING				
Expected points	**100**	**150**	**200**	**250**
Best	0.04007	0.02697	0.02113	0.01599
Worst	0.58185	0.59909	0.05147	0.03218
Average	0.08358	0.06911	0.02944	**0.02232**
Median	0.05494	0.03955	0.02798	0.02167
Std. Dev.	0.11821	0.12519	0.00774	0.00476

We analyze the effect of the size of the external repository (i.e., the secondary population of our MOPSO) on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

TABLE XL
RESULTS OF EXPERIMENT 4 FOR THE FIFTH TEST FUNCTION

ERROR RATIO				
Expected points	**100**	**150**	**200**	**250**
Best	0.07	0.153333	0.105	0.124
Worst	0.35	0.313333	0.395	0.3
Average	0.22550	0.21767	0.22775	**0.20685**
Median	0.235	0.22	0.205	0.208
Std. Dev.	0.06353	0.04544	0.07819	0.05028
GENERATIONAL DISTANCE				
Expected points	**100**	**150**	**200**	**250**
Best	0.30665	0.23684	0.20810	0.18967
Worst	0.46221	1.57960	0.35337	0.24430
Average	0.37413	0.37884	0.26050	**0.21672**
Median	0.36790	0.29926	0.25143	0.21570
Std. Dev.	0.04223	0.29656	0.03392	0.01455
SPACING				
Expected points	**100**	**150**	**200**	**250**
Best	2.13706	1.53664	1.25815	1.04094
Worst	2.97969	8.68894	1.93296	1.25219
Average	2.53034	2.18673	1.47291	**1.12607**
Median	2.50102	1.83570	1.39545	1.10992
Std. Dev.	0.22751	1.54941	0.19517	0.06414

We analyze the effect of the size of the external repository (i.e., the secondary population of our MOPSO) on the three metrics adopted in our study (i.e., error ratio, generational distance, and spacing).

XI. Conclusion and Future Work

We have presented a proposal to extend PSO to handle multiobjective problems. The proposed algorithm is relatively easy to implement and it improves the exploratory capabilities of PSO by introducing a mutation operator whose range of action varies over time. This also makes unnecesary to perform a fine tuning on the inertia weights used by the expression adopted to compute the velocity of each particle (in our experiments, we found that our approach was highly sensitive to the values of such inertia weights). The proposed approach was validated using the standard methodology currently adopted in the evolutionary multiobjective optimization community. The results indicate that our approach is a viable alternative since it has an average performance highly competitive with respect to some of the best multiobjective evolutionary algorithms known to date. In fact, MOPSO was the only algorithm from those adopted in our study that was able to cover the full Pareto front of all the functions used. Additionally, the exceptionally low computational times required by our approach make it a very promising approach to problems in which the computational cost is a vital issue (e.g., engineering optimization).

One aspect that we would like to explore in the future is the use of a crowding operator to improve the distribution of nondominated solutions along the Pareto front [11]. This would improve the capabilities of the algorithm to distribute uniformly the nondominated vectors found. We are also considering the possibility of extending this algorithm so that it can deal with dynamic functions [1]. Finally, it is desirable to study in more detail the parameters fine tuning required by the algorithm, as to provide a more solid basis to define them.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments that greatly helped us to improve the contents of this paper.

References

[1] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Norwell, MA: Kluwer, 2002.

[2] F. Y. Cheng and X. S. Li, "Generalized center method for multiobjective engineering optimization," *Eng. Opt.*, vol. 31, pp. 641–661, 1999.

[3] C. A. Coello Coello and G. Toscano, "A micro-genetic algorithm for multiobjective optimization," in *Lecture Notes in Computer Science no. 1993*, E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, Eds. Berlin, Germany: Springer-Verlag, 2001, Proc. 1st Int. Conf. Evolutionary Multi-Criterion Optimization, pp. 126–140.

[4] C. A. Coello Coello. EMOO Repository. [Online]. Available: http://delta.cs.cinvestav.mx/~ccoello/EMOO/

[5] C. A. Coello Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. Congr. Evolutionary Computation (CEC'2002)*, vol. 1, Honolulu, HI, May 2002, pp. 1051–1056.

[6] C. A. Coello Coello and G. T. Pulido, "Multiobjective optimization using a micro-genetic algorithm," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO'2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds., San Francisco, CA, 2001, pp. 274–282.

[7] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Norwell, MA: Kluwer, 2002, ISBN 0-3064-6762-3.

[8] K. Deb, "Multi-objective genetic algorithms: problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, pp. 205–230, Fall 1999.

[9] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Parallel Problem Solving From Nature VI Conf.*, 2000, pp. 849–858.

[10] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed., San Mateo, CA, June 1989, pp. 42–50.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 182–197, Apr. 2002.

[12] R. M. Everson, J. E. Fieldsend, and S. Singh, "Full elite sets for multi-objective optimization," in *Proc. 5th Int. Conf. Adaptive Computing Design Manufacture (ACDM 2002)*, vol. 5, I. C. Parmee, Ed., Devon, U.K., Apr. 2002, pp. 343–354.

[13] J. E. Fieldsend and S. Singh, "A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence," in *Proc. 2002 U.K. Workshop on Computational Intelligence*, Birmingham, U.K., Sept. 2002, pp. 37–44.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[15] J. Horn, "Multicriterion decision making," in *Handbook of Evolutionary Computation*, T. Bäck, D. Fogel, and Z. Michalewicz, Eds. London, U.K.: Oxford Univ. Press, 1997, vol. 1, pp. F1.9:1–F1.9:15.

[16] X. Hu and R. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proc. Congr. Evolutionary Computation (CEC'2002)*, vol. 2, Honolulu, HI, May 2002, pp. 1677–1681.

[17] X. Hui, R. C. Eberhart, and Y. Shi, "Particle swarm with extended memory for multiobjective optimization," in *Proc. 2003 IEEE Swarm Intelligence Symp.*, Indianapolis, IN, Apr. 2003, pp. 193–197.

[18] Y. Jin, T. Okabe, and B. Sendhoff, "Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how?," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO'2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds., San Francisco, CA, 2001, pp. 1042–1049.

[19] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.

[20] H. Kita, Y. Yabumoto, N. Mori, and Y. Nishikawa, "Multi-objective optimization by means of the thermodynamical genetic algorithm," in *Parallel Problem Solving From Nature—PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, Sept. 1996, Lecture Notes in Computer Science, pp. 504–512.

[21] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evol. Comput.*, vol. 8, pp. 149–172, 2000.

[22] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Lecture Notes in Computer Science*, H. P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer-Verlag, Oct 1991, vol. 496, Proc. Parallel Problem Solving From Nature, 1st Workshop, PPSN I, pp. 193–197.

[23] X. Li *et al.*, "A nondominated sorting particle swarm optimizer for multiobjective optimization," in *Lecture Notes in Computer Science*, vol. 2723, Proc. Genetic and Evolutionary Computation—GECCO 2003—Part I, E. Cantú-Paz *et al.*, Eds.. Berlin, Germany, July 2003, pp. 37–48.

[24] J. Moore and R. Chapman, *Application of Particle Swarm to Multiobjective Optimization*: Dept. Comput. Sci. Software Eng., Auburn Univ., 1999.

[25] S. Mostaghim and J. Teich, "Strategies for finding good local guides in Multi-Objective Particle Swarm Optimization (MOPSO)," in *Proc. 2003 IEEE Swarm Intelligence Symp.*, Indianapolis, IN, Apr. 2003, pp. 26–33.

[26] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proc. 2002 ACM Symp. Applied Computing (SAC'2002)*, Madrid, Spain, 2002, pp. 603–607.

[27] T. Ray, T. Kang, and S. K. Chye, "An evolutionary algorithm for constrained optimization," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO'2000)*, D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds., San Francisco, CA, 2000, pp. 771–777.

[28] T. Ray and K. M. Liew, "A swarm metaphor for multiobjective design optimization," *Eng. Opt.*, vol. 34, no. 2, pp. 141–153, Mar. 2002.

[29] J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," M.S. thesis, Dept. Aeronautics and Astronautics, Massachusetts Inst. Technol., Cambridge, MA, May 1995.

[30] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, Fall 1994.

[31] F. van den Bergh, "An analysis of particle swarm optimization," Ph.D. dissertation, Faculty of Natural and Agricultural Sci., Univ. Petoria, Pretoria, South Africa, Nov. 2002.

[32] D. A. Van Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyzes, and new innovations," Ph.D. dissertation, Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH, May 1999.

[33] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH, Tech. Rep. TR-98-03, 1998.

[34] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Ph.D. dissertation, Swiss Fed. Inst. Technol. (ETH), Zurich, Switzerland, Nov. 1999.

[35] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Summer 2000.

[36] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," in *Proc. EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control With Applications to Industrial Problems*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., Athens, Greece, Sept. 2001.

[37] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 257–271, Nov. 1999.

Carlos A. Coello Coello (M'99) received the B.Sc. degree in civil engineering from the Universidad Autónoma de Chiapas, México, and the M.Sc. and Ph.D. degrees in computer science from Tulane University, New Orleans, LA, in 1991, 1993, and 1996, respectively.

He is currently an Associate Professor (CINVESTAV-3B Researcher) with the Electrical Engineering Department, CINVESTAV-IPN, Mexico City, México. He has authored and coauthored over 100 technical papers and several book chapters. He has also coauthored the book *Evolutionary Algorithms for Solving Multi-Objective Problems* (Norwell, MA: Kluwer, 2002). His major research interests are: evolutionary multiobjective optimization, constraint-handling techniques for evolutionary algorithms, and evolvable hardware.

Dr. Coello Coello has served in the program committees of over 30 international conferences and has been technical reviewer for over 30 international journals including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION in which he also serves as an Associate Editor. He also Chairs the Task Force on Multiobjective Evolutionary Algorithms of the IEEE Neural Networks Society. He is a Member of the Association for Computing Machinery (ACM) and the Mexican Academy of Sciences.

Gregorio Toscano Pulido received the B.Sc. degree in computer science from the Instituto Tecnológico de Mérida, in Mérida, Yucatán, México, and the M.Sc. degree in artificial intelligence from the Universidad Veracruzana, Xalapa, Veracruz, México, in 1999 and 2002, respectively. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN), México.

His current research interests include evolutionary computation and multiobjective optimization.

Maximino Salazar Lechuga received the B.Sc. degree in computer science and the M.Sc. degree in artificial intelligence from the Universidad Veracruzana, Xalapa, Veracruz, México, in 2000 and 2002, respectively. He is currently the recipient of a scholarship from the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) to pursue the Ph.D. degree in computer science at the University of Birmingham, Birmingham, U.K.

His main research interests are evolutionary computation, artificial life, emergent behaviors, and neural networks.