# Handling Multiple Objectives with Particle Swarm Optimization


Carlos A. Coello Coello[†]
Gregorio Toscano Pulido[†]
Maximino Salazar Lechuga[‡]
[†]CINVESTAV-IPN
Sección de Computación
Electrical Engineering Department
México, D.F., 07300, MEXICO
ccoello@cs.cinvestav.mx
gtoscano@computacion.cs.cinvestav.mx
[‡]Universidad Veracruzana
Maestría en Inteligencia Artificial
Xalapa, Veracruz 91000, MEXICO
reactivo@prodigy.net.mx



**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 making it unnecessary to experiment with different parameters. 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.


## 1 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 [6]. 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 PAES [18]), new algorithms that use very small populations (e.g., the micro-GA [5]), and data structures that allow to handle unconstrained external archives (e.g., the dominated tree [11]).

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 [16], 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 [16]. In this paper, we present a proposal, called "multi-objective 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 [4] 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 Non-dominated Sorting Genetic Algorithm-II [10] (NSGA-II), the Pareto Archived Evolution Strategy (PAES) [18], and the micro-genetic algorithm for multiobjective optimization (microGA) [5].

## 2 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(s), $f$ is the objective function, and the set $\Omega$ is the feasible region ($\Omega \in \mathcal{S}$), where $\mathcal{S}$ represents the whole search space.* □

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

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

*the $p$ equality constraints*

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

*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*

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

□

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 Optimal Set):** *For a given MOP* $\vec{f}(x)$*, the Pareto optimal set (*$\mathcal{P}^*$*) is defined as:*

$$\mathcal{P}^* := \{x \in \Omega \mid \neg\exists\, x' \in \Omega \;\; \vec{f}(x') \preceq \vec{f}(x)\}. \tag{7}$$

□

**Definition 5 (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 6 (Pareto Front:):** *For a given MOP* $\vec{f}(x)$ *and Pareto optimal set* $\mathcal{P}^*$*, the Pareto front (*$\mathcal{PF}^*$*) is defined as:*

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

□

In general, it is impossible to find an analytical expression of the line or surface that contains these points and in most cases, it turns out to be impossible to do it. The normal procedure to generate the Pareto front is to compute the points $\Omega$ 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 *non-inferior*, *admissible*, or *efficient* solutions [13]; their corresponding vectors are termed *nondominated*. When plotted in objective space, the nondominated vectors are collectively known as the **Pareto front**.

## 3 Related Work

Kennedy & Eberhart [16] proposed an approach called "particle swarm optimization" (PSO) which was inspired on 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 uses then 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. However, this approach introduces the use of flying potential solutions through hyperspace (used to accelerate convergence) which does not seem to have an analogous mechanism in traditional evolutionary algorithms. Another important difference is the fact that PSO allows individuals to benefit from their past experiences whereas in an evolutionary 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 [16]. Particle swarm optimization seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single-objective optimization [16].

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 [20]**: 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 & Liew [23]**: This algorithm also uses Pareto dominance and combines concepts of evolutionary techniques with the particle swarm. The approach uses crowding to maintain diversity and multilevel sieve to handle constraints (for this, the authors adopt the constraint and objective matrices proposed in some of their previous research [22]).

- **The algorithm of Parsopoulos & Vrahatis [21]**: Unlike the previous proposals, this algorithm adopt 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 [15] 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 [14]**: In this algorithm, only one objective is optimized at a time using a scheme similar to lexicographic ordering [6]. 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.

None of the previous approaches adopt a secondary population (the most common notion of elitism in EMO), which 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 these approaches). Also, none of these techniques was properly validated using test functions and metrics normally adopted by EMO researchers.

We will show that our PSO algorithm for multiobjective optimization incorporates a secondary population with a filtering process that encourages a uniform distribution of solutions along the Pareto front. Our approach is validated against three modern evolutionary multiobjective optimization techniques, adopting standard test functions and metrics from the EMO literature. Our results indicate that our approach is quite competitive in terms of the quality of the solutions that it produces and that it is much faster than any of the other algorithms against which it was compared.

# 4 Description of the Proposed Approach

The analogy of particle swarm optimization with evolutionary algorithms makes evident the notion that using a Pareto ranking scheme [12] 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 towards globally nondominated solutions.

Therefore, our proposal is based on the idea of having a global repository in which every particle will deposit its flight experiences after each flight cycle. Additionally, the updates to the repository are performed considering a geographically-based system defined in terms of the objective function values of each individual. This technique is inspired on the external file used with the Pareto Archive Evolution Strategy (PAES) [18]. The repository previously mentioned is used by the particles to identify a leader that will guide the search. We implemented a mechanism such that each particle may choose a different guide. Our mechanism is based on the generation of hypercubes which are produced dividing the search space explored.

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:

$$\begin{aligned} VEL[i] = W \times VEL[i] + R_1 \times \\ (PBESTS[i] - POP[i]) + \\ R_2 \times (REP[h] - POP[i]) \end{aligned}$$

   where $W$ (inertia weight) takes a value of 0.4; $R_1$ and $R_2$ are random numbers in the range [0..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 [9]. Then,

[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.

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:

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

(c) Maintain the particles within the search space in case they go beyond its boundaries (avoid generating solutions that do not lie on valid search space).

(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

Additionally, we added a mutation operator that decreases its range of operation over time, such that at the beginning it covers the full range of each variable, and it deacreases according to a a nonlinear function. The number of individuals that are subject to mutation decreases over time, too. This operator was introduced to increase the exploratory power of MOPSO which we found to be limited in some functions in which there were important attractors in certain portions of the Pareto front (e.g., the second test function presented in the next section).

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 within the micro genetic algorithm for multiobjective optimization to handle constraints [5].

# 5 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 [28]:

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. **Error Ratio** (ER): This metric was proposed by Van Veldhuizen [26] to indicate the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set:

$$ER = \frac{\sum_{i=1}^{n} e_i}{n}, \tag{11}$$

   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 $ER = 0$ indicates an ideal behavior, since it would mean that all the vectors generated by our MEA belong to the Pareto optimal set of the problem. This metric addresses the third issue from the list previously provided.

2. **Generational Distance** (GD): The concept of generational distance was introduced by Van Veldhuizen & Lamont [27] as a way of estimating how far are the elements in the set of nondominated vectors found so far from those in the Pareto optimal set and is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{12}$$

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 $GD = 0$ 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.

3. **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 [24] proposed such a metric measuring the range (distance) variance of neighboring vectors in the nondominated vectors found so far. This metric is defined as:

$$S \triangleq \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\overline{d} - d_i)^2}\ , \qquad (13)$$

where $d_i = \min_j(\mid f_1^i(\vec{x}) - f_1^j(\vec{x}) \mid + \mid f_2^i(\vec{x}) - f_2^j(\vec{x}) \mid)$, $i, j = 1, \ldots, n$, $\overline{d}$ 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.

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 was our approach, we decided to compare it against three multiobjective evolutionary algorithms that represent the state-of-the-art:

1. **Nondominated Sorting Genetic Algorithm II**: Proposed by Deb et al. [8, 10], this algorithm is a revised version of the Nondominated Sorting Genetic Algorithm proposed by Srinivas & Deb in 1994 [25]. The original NSGA is based on several layers of classifications of the individuals as suggested by Goldberg [12]. 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

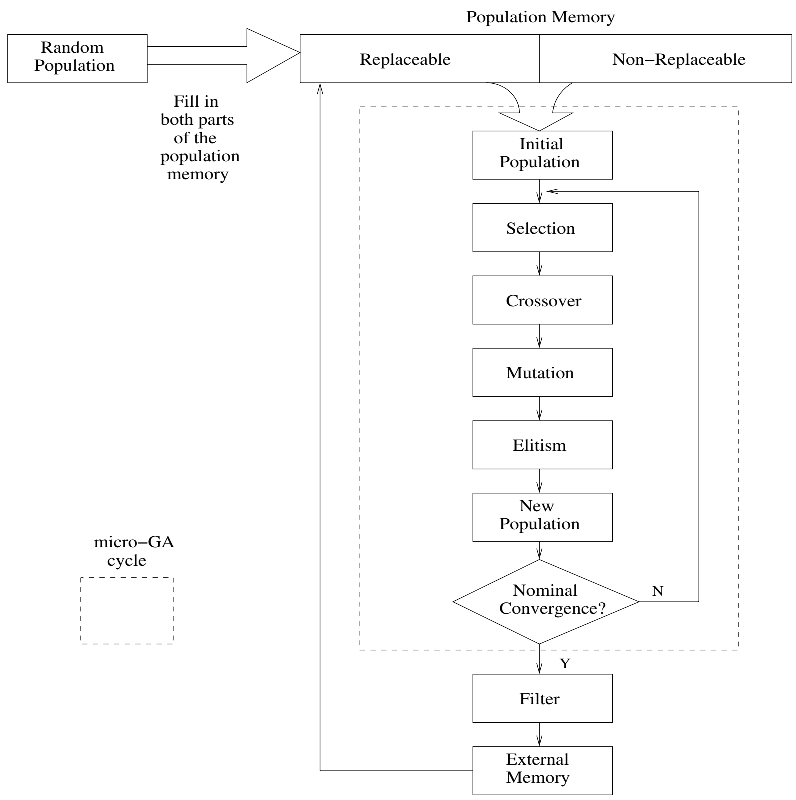



Figure 1: Diagram that illustrates the way in which the micro-GA for multiobjective optimization works.

get more copies than the rest of the population. This allows to 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 [18]. 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. **Micro-Genetic Algorithm for Multiobjective Optimization**: This approach was introduced by Coello Coello & Toscano Pulido [3, 5]. A micro-genetic algorithm is a GA with a small population and a reinitialization process. The way in which the micro-GA works is illustrated in Figure 1. First, a random population is generated. This random population feeds the population memory, which is divided in two parts: a replaceable and a non-replaceable portion. The non-replaceable 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 micro-GA.

   The population of the micro-GA 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 (non-replaceable portion) and evolved individuals (replaceable portion). During each cycle, the micro-GA undergoes conventional genetic operators. After the micro-GA finishes one cycle, two nondominated vectors are chosen[3] 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

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

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 micro-GA uses then three forms of elitism: (1) it retains nondominated solutions found within the internal cycle of the micro-GA, (2) it uses a replaceable memory whose contents is partially "refreshed" at certain intervals, and (3) it replaces the population of the micro-GA by the nominal solutions produced (i.e., the best solutions found after a full internal cycle of the micro-GA). Therefore, the micro-GA is another example of how elitism can play a vital role to improve the performance of an evolutionary algorithm used for multiobjective optimization.

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 micro-GA 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 non-replaceable memory of 0.05, a population size (for the micro-GA 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 0.05[4], and 30 divisions for the adaptive grid. Our implementation uses a real-numbers representation, but a binary representation could be also adopted if needed. These parameters were kept in all the examples, and we only changed the total number of fitness function evaluations[5] 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 micro-GA is available from the EMOO repository located at: `http://delta.cs.cinvestav.mx/~ccoello/EMOO`. The source code of MOPSO may be obtained by emailing to `ccoello@cs.cinvestav.mx`.

## 5.1 Test Function 1

For our first example, we used the following problem proposed by Kita [17]:

Maximize $F = (f_1(x,y), f_2(x,y))$, where

[4]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.

[5]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.

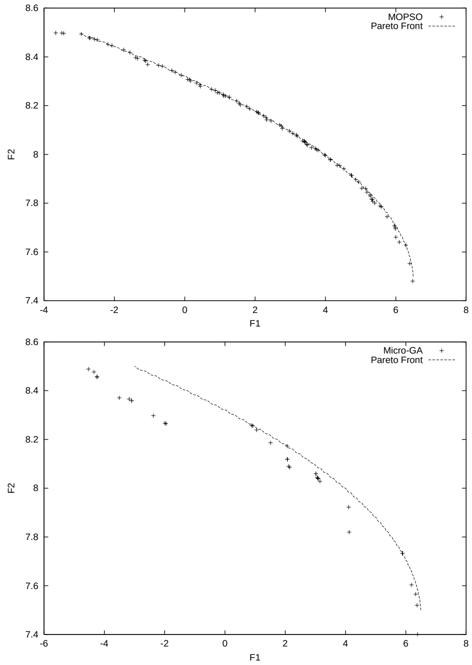



Figure 2: Pareto fronts produced by our MOPSO (left) and the micro-GA (right) for the first test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

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

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: $x, y \geq 0$.

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

Figures 2 and 3 show the graphical results produced by the PAES, the NSGA-II, the micro-GA and our MOPSO in the first test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables 1, 2, 3 and 4 show the comparison of results among the four algorithms considering the metrics previously described. It can be seen that the average performance of

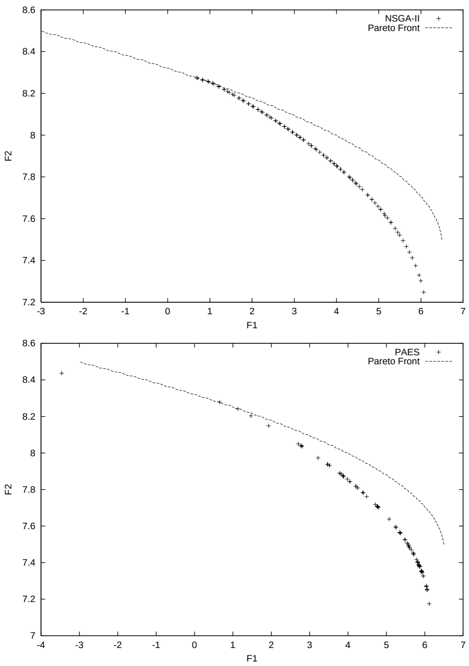



Figure 3: Pareto fronts produced by the NSGA-II (left) and PAES (right) for the first test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

| **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 1: Results of the Error Ratio 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 2: Results of the Generational Distance 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 3: Results of the Spacing metric 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 |

Table 4: Computational time (in seconds) required by each algorithm for the first test function.

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 except for MOPSO, none of the algorithms was able to cover the full Pareto front.[6] This is then an example in which a metric may be misleading, since the fact that the spacing metric provides a good value if the nondominated vectors produced by the algorithm are not part of the true Pareto front of the problem [26]. Also, it is important to notice the very high speed of MOPSO, which requires almost half of the time than the micro-GA. This is remarkable if we consider that the NSGA-II and the micro-GA are algorithms that are normally considered "very fast" approaches.

# 6 Test Function 2

Our second test function was proposed by Kursawe [19]:

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

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

[6] When using our original implementation of MOPSO [4], we ran into the same problem. Such a behavior motivated the development of the mutation operator reported in this paper.

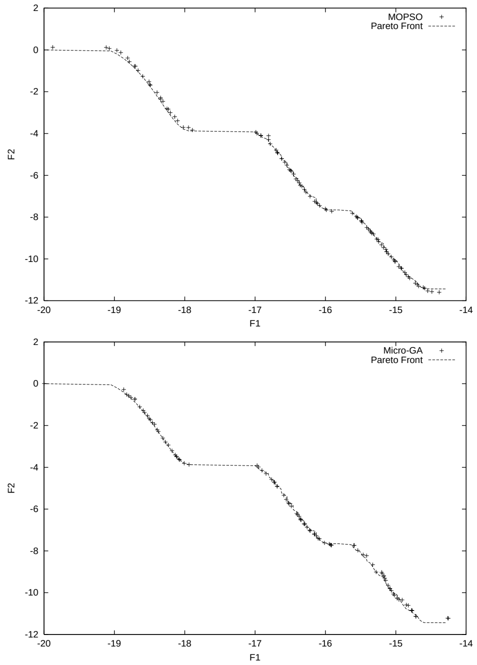



Figure 4: Pareto fronts produced by our MOPSO (left) and the micro-GA (right) for the second test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

where:

$$-5 \leq x_1, x_2, x_3 \leq 5 \tag{16}$$

In this example, the total number of fitness function evaluations was set to 12000.

Figures 4 and 5 show the graphical results produced by the PAES, the NSGA-II, the micro-GA and our MOPSO in the second test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables 5, 6, 7 and 8 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 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

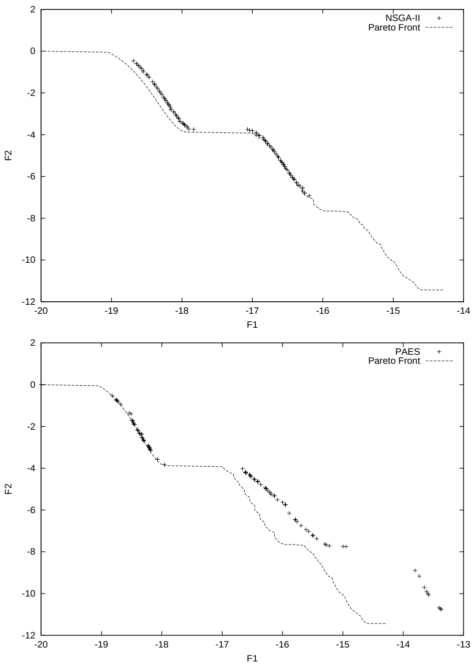



Figure 5: Pareto fronts produced by the NSGA-II (left) and PAES (right) for the second test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

| ER | MOPSO | NSGA-II | microGA | PAES |
|---|---|---|---|---|
| Best | 0.18 | 0.06 | 0.18 | 0.10 |
| Worst | 0.28 | 1.01 | 0.36 | 0.68 |
| Average | 0.24 | 0.56 | 0.27 | 0.27 |
| Median | 0.25 | 0.495 | 0.245 | 0.245 |
| Std. Dev. | 0.03539 | 0.384516 | 0.053947 | 0.10489 |

Table 5: Results of the Error Ratio metric for the second test function.

| GD | MOPSO | NSGA-II | microGA | PAES |
|---|---|---|---|---|
| Best | 0.007494 | 0.006905 | 0.006803 | 0.01467 |
| Worst | 0.010299 | 0.103095 | 0.010344 | 0.157191 |
| Average | 0.008889 | 0.029255 | 0.008456 | 0.54914 |
| Median | 0.008821 | 0.017357 | 0.008489 | 0.049358 |
| Std. Dev. | 0.000778 | 0.02717 | 0.000987 | 0.030744 |

Table 6: Results of the Generational Distance metric for the second test function.

| SP | MOPSO | NSGA-II | microGA | PAES |
|---|---|---|---|---|
| Best | 0.058507 | 0.018418 | 0.071686 | 0.064114 |
| Worst | 0.119415 | 0.065712 | 0.203127 | 0.340955 |
| Average | 0.088491 | 0.036136 | 0.128895 | 0.197532 |
| Median | 0.091109 | 0.036085 | 0.126655 | 0.186632 |
| Std. Dev. | 0.01869 | 0.010977 | 0.029932 | 0.064114 |

Table 7: Results of the Spacing metric 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 8: Computational time (in seconds) required by each algorithm for the second test function.

the NSGA-II in this test function.

# 7 Test Function 3

Our third test function was proposed by Deb [7]:

$$\text{Minimize } f_1(x_1, x_2) = x_1 \tag{17}$$

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

where:

$$g(x_1, x_2) = 11 + x_2^2 - 10 \cdot \cos(2\pi x_2) \tag{19}$$

$$h(x_1, x_2) = \begin{cases} 1 - \sqrt{\frac{f_1(x_1, x_2}{g(x_1, x_2}} & \text{if } f_1(x_1, x_2) \leq g(x_1, x_2) \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

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

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

Figures 6 and 7 show the graphical results produced by the PAES, the NSGA-II, the micro-GA and our MOPSO in the third test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables 9, 10, 11 and 12 show the comparison of results among the four algorithms considering the

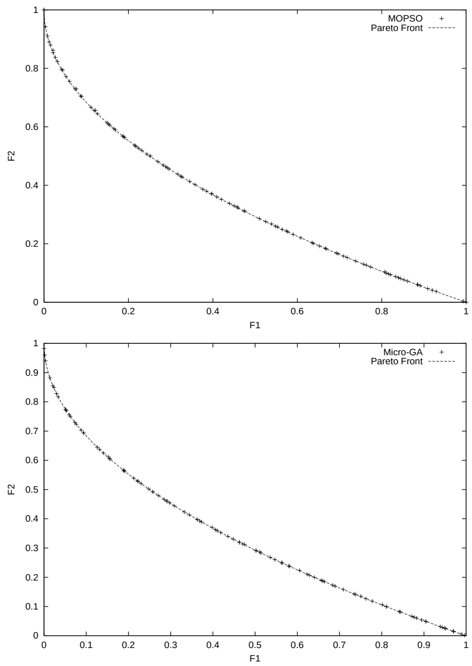


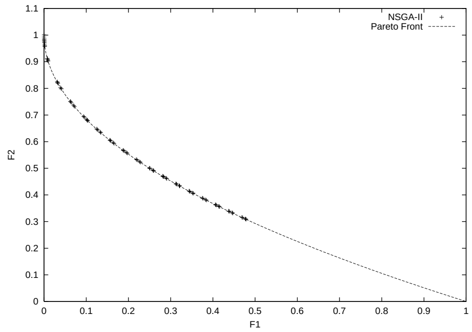

Figure 6: Pareto fronts produced by our MOPSO (left) and the micro-GA (right) for the third test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

| **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 |

Table 9: Results of the Error Ratio metric for the third test function.

| **GD** | **MOPSO** | **NSGA-II** | **microGA** | **PAES** |
|---|---|---|---|---|
| Best | $8.61 \times 10^{-5}$ | 0.000133 | $8.74 \times 10^{-5}$ | 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. | $2.55 \times 10^{-5}$ | 0.045429 | 0.181155 | 0.441303 |

Table 10: Results of the Generational Distance metric for the third test function.



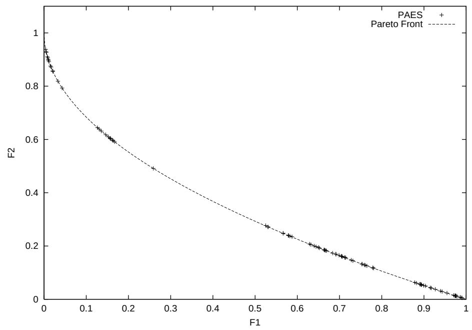




Figure 7: Pareto fronts produced by the NSGA-II (left) and PAES (right) for the third test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

| **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 11: Results of the Spacing metric 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 |

Table 12: Computational time (in seconds) required by each algorithm for the third 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 | 0.256583 | 0.4145 | 0.252 | 0.489 |
| Median | 0.045 | 0.115 | 0.16 | 0.28 |
| Std. Dev. | 0.400658 | 0.459387 | 0.231576 | 0.438117 |

Table 13: Results of the Error Ratio metric for the fourth test function.

metrics previously described. It can be seen that the average performance of MOPSO plays second with respect to 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.

# 8 Test Function 4

Our fourth test function was also proposed by Deb [7]:

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

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

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

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 10000.

Figures 8 and 9 show the graphical results produced by the PAES, the NSGA-II, the micro-GA and our MOPSO in the fourth test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables 13, 14, 15 and 16 show the comparison of results among the four algorithms considering the metrics previously described. It can be seen that the average performance of

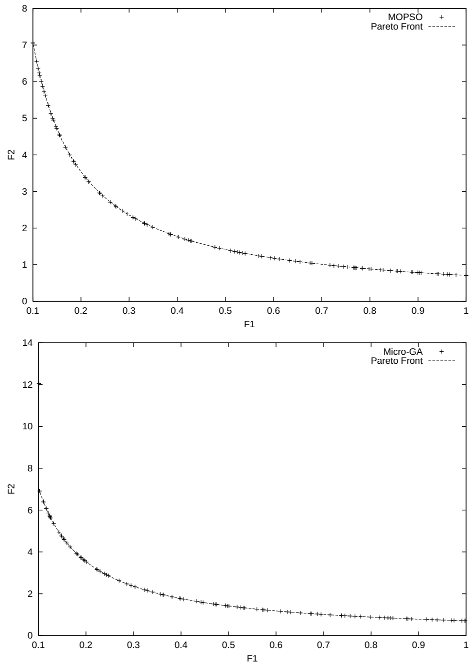



Figure 8: Pareto fronts produced by our MOPSO (left) and the micro-GA (right) for the fourth test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

| **GD** | **MOPSO** | **NSGA-II** | **microGA** | **PAES** |
|---|---|---|---|---|
| Best | 0.000428 | 0.0007 | 0.000465 | 0.000453 |
| Worst | 0.185306 | 0.208467 | 0.183501 | 0.221671 |
| Average | 0.032725 | 0.044236 | 0.043466 | 0.194767 |
| Median | 0.00051 | 0.000856 | 0.050042 | 0.070365 |
| Std. Dev. | 0.060622 | 0.07368 | 0.048212 | 0.204687 |

Table 14: Results of the Generational Distance metric for the fourth test function.

| **SP** | **MOPSO** | **NSGA-II** | **microGA** | **PAES** |
|---|---|---|---|---|
| Best | 0.040069 | 0.026086 | 0.030267 | 0.047844 |
| Worst | 0.58185 | 0.061422 | 0.817642 | 0.664676 |
| Average | 0.083579 | 0.037447 | 0.213584 | 0.194767 |
| Median | 0.054945 | 0.035529 | 0.06301 | 0.070365 |
| Std. Dev. | 0.118214 | 0.009238 | 0.250586 | 0.204687 |

Table 15: Results of the Spacing metric for the fourth test function.

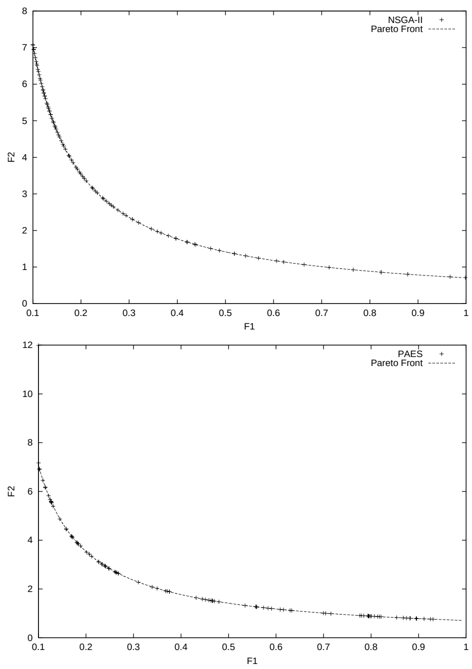



Figure 9: Pareto fronts produced by the NSGA-II (left) and PAES (right) for the fourth test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

| **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 |

Table 16: Computational time (in seconds) required by each algorithm for the fourth test function.

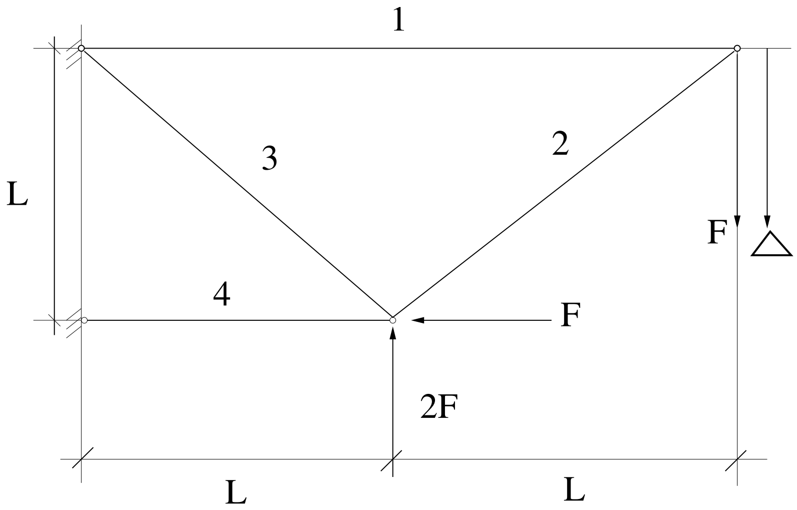



Figure 10: Plane truss used for the fifth test function. The structural volume and the joint displacement ($\Delta$) are to be optimized.

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 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 micro-GA is faster than MOPSO (it requires about half the CPU time of MOPSO).

# 9 Test Function 5

Our fifth test function is to optimize the four-bar plane truss shown in Figure 10. 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

$(F/\sigma) \leq x_1 \leq 3 \times (F/\sigma)$
$\sqrt{2}(F/\sigma) \leq x_2 \leq 3 \times (F/\sigma)$
$\sqrt{2}(F/\sigma) \leq x_3 \leq 3 \times (F/\sigma)$
$(F/\sigma) \leq x4 \leq 3 \times (F/\sigma)$

where:

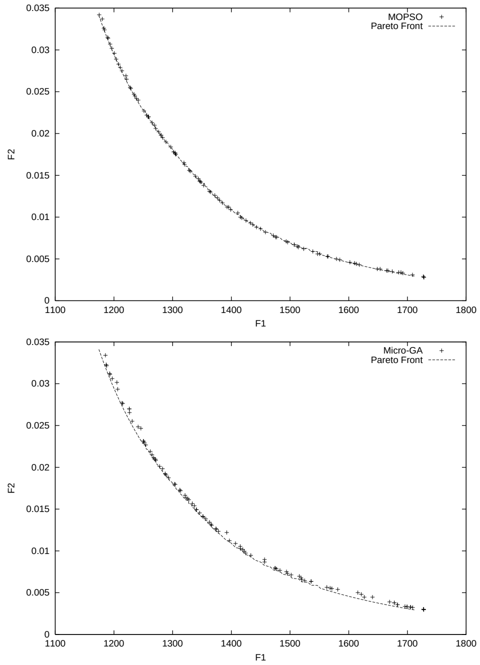



Figure 11: Pareto fronts produced by our MOPSO (left) and the micro-GA (right) for the fifth test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

$$F = 10kN, \; E = (2)10^5 kN/cm^2, \; L = 200cm \;\; \sigma = 10kN/cm^3$$

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

Figures 11 and 12 show the graphical results produced by the PAES, the NSGA-II, the micro-GA and our MOPSO in the fifth test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables 17, 18, 19 and 20 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 micro-GA which, however, has a much poorer performance with respect to the three metrics adopted.

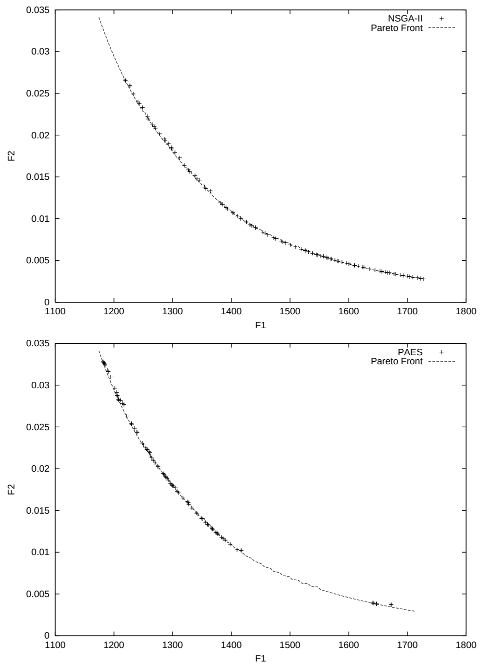



Figure 12: Pareto fronts produced by the NSGA-II (left) and PAES (right) for the fifth test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

| **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 | 0.2255 | 0.352 | 0.898355 | 0.216 |
| Median | 0.235 | 0.23 | 0.989689 | 0.22 |
| Std. Dev. | 0.063534 | 0.26365 | 0.163529 | 0.078967 |

Table 17: Results of the Error Ratio 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 |

Table 18: Results of the Generational Distance 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 19: Results of the Spacing metric 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 20: Computational time (in seconds) required by each algorithm for the fifth test function.

# 10 Parameters Fine-Tuning

Although the set of parameters adopted for MOPSO in the experiments reported in this paper have been found to be appropriate in a variety of problems, it is useful to provide some guidelines regarding how to fine tune them in case it becomes necessary:

- **Number of particles**: This is equivalent to the population size of a genetic algorithm. We recommend using between 20 and 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 viceversa). We recommend to use between 80 and 120.
- **Number of divisions**: It allows us to determine the number of hypercubes that will be generated in objective function space. We recommend to use between 30 and 50 divisions.
- **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 between 100 and 200 particles.

## 11 Conclusions and Future Work

We have presented a proposal to extend particle swarm optimization 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 in 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. 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].

## Acknowledgements

The first author gratefully acknowledges support from CONACyT through project 34201-A. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the Electrical Engineering Department at CINVESTAV-IPN.

## References

[1] Jürgen Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer Academic Publishers, Boston/Dordecht/London, 2002.

[2] F.Y. Cheng and X.S. Li. Generalized Center Method for Multiobjective Engineering Optimization. *Engineering Optimization*, 31:641–661, 1999.

[3] Carlos A. Coello Coello and Gregorio Toscano. A Micro-Genetic Algorithm for Multiobjective Optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[4] Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.

[5] Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[6] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic Publishers, Boston, 2002. ISBN 0-3064-6762-3.

[7] Kalyanmoy Deb. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230, Fall 1999.

[8] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858. Springer, 2000.

[9] Kalyanmoy Deb and David E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.

[10] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[11] Richard M. Everson, Jonathan E. Fieldsend, and Sameer Singh. Full Elite Sets for Multi-Objective Optimisation. In I.C. Parmee, editor, *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002)*, volume 5, pages 343–354, University of Exeter, Devon, UK, April 2002. Springer-Verlag.

[12] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[13] Jeffrey Horn. Multicriterion Decision making. In Thomas Bäck, David Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, volume 1, pages F1.9:1–F1.9:15. IOP Publishing Ltd. and Oxford University Press, 1997.

[14] Xiaohui Hu and Russell Eberhart. Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1677–1681, Piscataway, New Jersey, May 2002. IEEE Service Center.

[15] Yaochu Jin, Tatsuya Okabe, and Bernhard Sendhoff. Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How? In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 1042–1049, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[16] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

[17] Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, pages 504–512. Springer-Verlag, Berlin, Germany, September 1996.

[18] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[19] Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.

[20] Jacqueline Moore and Richard Chapman. Application of Particle Swarm to Multiobjective Optimization. Department of Computer Science and Software Engineering, Auburn University, 1999.

[21] K.E. Parsopoulos and M.N. Vrahatis. Particle Swarm Optimization Method in Multiobjective Problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002)*, pages 603–607, Madrid, Spain, 2002. ACM Press.

[22] Tapabrata Ray, Tai Kang, and Seow Kian Chye. An Evolutionary Algorithm for Constrained Optimization. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, 2000. Morgan Kaufmann.

[23] Tapabrata Ray and K.M. Liew. A Swarm Metaphor for Multiobjective Design Optimization. *Engineering Optimization*, 34(2):141–153, March 2002.

[24] J. R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

[25] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, fall 1994.

[26] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[27] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[28] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.