MOPSO : A Proposal for Multiple Objective Particle Swarm Optimization*

Carlos A. Coello Coello[1] and Maximino Salazar Lechuga[2]
[1]CINVESTAV-IPN
Depto. de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D. F. 07300
ccoello@cs.cinvestav.mx

[2]Maestría en Inteligencia Artificial
LANIA-Universidad Veracruzana
Xalapa, Veracruz, México 91090
msalazar@mail.mia.uv.mx

Abstract

This paper introduces a proposal to extend the heuristic called "particle swarm optimization" (PSO) to deal with multiobjective optimization problems. Our approach uses the concept of Pareto dominance to determine the flight direction of a particle and it maintains previously found nondominated vectors in a global repository that is later used by other particles to guide their own flight. The approach is validated using several standard test functions from the specialized literature. Our results indicate that our approach is highly competitive with current evolutionary multiobjective optimization techniques.

1 Introduction

Most real-world problems are multiobjective in nature, but they tend to be treated as single-objective optimization problems, transforming many of the original objectives into additional constraints. Evolutionary algorithms (EAs) are particularly suitable to solve multi-objective optimization problems (MOPs)

*Technical Report EVOCINV-01-2001, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México, September 2001.

because they are a population-based heuristics and, therefore, allow the generation of several elements of the Pareto optimal set in a single run. In contrast traditional mathematical programming techniques tend to generate a single element of the Pareto optimal set in one run. Additionally, EAs are not susceptible to the shape or continuity of the Pareto front as opposed to traditional methods.

The use of EAs to solve multiobjective problems (an area called evolutionary multiobjective optimization) has generated a considerable volume of research in the last few years (see for example [6, 1, 15]).

The development of an alternative (heuristic-based) multiobjective optimization technique requires to deal with two main problems: how to generate nondominated vectors that approximate the global Pareto front of the problem and how to maintain diversity so that we can generate as many of such vectors as possible. The traditional approach to deal with the first problem has been to use a procedure called Pareto ranking [7] which consists of classifying a population of individuals based on the concept of Pareto dominance. Pareto ranking is $O(kM^2)$, where k is the number of objective functions and M is the size of the population. Additionally, an extra mechanism is required to preserve diversity (some form of fitness sharing [4] is normally adopted). This generally implies the use of another process that is $O(M^2)$.

Knowing the sources of inefficiency of traditional evolutionary multiobjective optimization techniques, several researchers have focused their recent efforts on reducing the checkings for nondominance and in the development of efficient approaches to keep diversity. Regarding the first issue, the main emphasis has been on using an external file that stores nondominated vectors found during the evolutionary process. These vectors are put back into the population at later generations (this can be seen as a form of elitism in the context of multi-objective optimization [8, 16]). Regarding the second issue, the main emphasis has been on using clustering techniques [3] or approaches based on geographical positioning of individuals in an adaptive grid [12].

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. The approach is relatively simple to implement, it is population-based, it uses an external memory (called "repository") and a geographically-based approach to maintain diversity. MOPSO is validated using several standard test functions reported in the specialized literature and compared against two highly competitive EMO algorithms: the Pareto Archived Evolution Strategy [12] (PAES) and the Non-dominated Sorting Genetic Algorithm II [3] (NSGA II).

2 Basic Concepts

Definición 1 (General MOP): *Find the vector* $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ *which will satisfy the* m *inequality constraints:*

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

the p equality constraints

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

and will optimize the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]^T \tag{3}$$

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

Definición 2 (Pareto Optimality:): *A point* $\vec{x}^* \in \Omega$ *(*Ω *is the feasible region) is* **Pareto optimal** *if for every* $\vec{x} \in \Omega$ *and* $I = \{1, 2, \dots, k\}$ *either,*

$$\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{4}$$

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

$$f_i(\vec{x}) > f_i(\vec{x}^*) \tag{5}$$

□

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.

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

Definición 4 (Pareto Optimal Set): *For a given MOP* $\vec{f}(x)$*, the Pareto optimal set (*$\mathcal{P}^*$*) is defined as:*

$$\mathcal{P}^* := \{x \in \Omega \mid \neg\exists\, x' \in \Omega \;\; \vec{f}(x') \preceq \vec{f}(x)\}. \tag{6}$$

□

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

$$\mathcal{PF}^* := \{\vec{u} = \vec{f} = (f_1(x), \dots, f_k(x)) \mid x \in \mathcal{P}^*\}. \tag{7}$$

□

3 Particle Swarm Optimization

Kennedy & Eberhart [9] proposed an approach called "particle swarm optimization" (PSO) which was inspired on the choreography of a bird flock. The idea of this approach is to simulate the movements of a group (or population) of birds which aim to find food. 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 [9, 5, 10, 11].

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 [11]. Our hypothesis is that extending PSO to handle multiple objectives will provide us with a very fast algorithm that will also be relatively easy to implement.

4 Description of the Approach

The analogy of particle swarm optimization with evolutionary algorithms makes evident the notion that using a Pareto ranking scheme [7] 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 [13]). 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) [12]. 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:

$$VEL[i] = W \times VEL[i] + R_1 \times (PBESTS[i] - POP[i]) + R_2 \times (REP[h] - POP[i]) \tag{8}$$

   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

[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.

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 [4]. 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:

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

(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 none of them is dominated by the other, then we select one of them randomly).

(g) Increment the loop counter

8. END WHILE

5 Comparison of Results

Several test functions were taken from the specialized literature to compare our approach. To compare our results in a quantitative way we used two criteria:

average running time of the algorithm (using the same number of fitness function evaluations), and the following metric defined in objective space by Zitzler et al. [16]:

$$M_1^* = \frac{1}{|Y'|} \sum_{\mathbf{d}' \in Y'} \min \; \{\| \mathbf{d}' - \bar{\mathbf{d}} \|^*; \bar{\mathbf{d}} \in \bar{Y}\} \tag{11}$$

where: $Y', \bar{Y} \subseteq Y$ are the sets of objective vectors that correspond to a set of pairwise nondominating decision vectors $X', \bar{X} \subseteq X$, respectively, and X corresponds to the decision variables of the problem. It should be obvious that M_1^* gives the average distance to the Pareto optimal set. Therefore, we should aim to minimize this value (see [16] for further details). To compute M_1^*, we generated the global Pareto front for each of the test functions (this is done by enumeration, using a certain coarse representation of the decision variable space), so that we could measure closeness to the desired set of solutions.

Since the main aim of this approach has been to increase efficiency, we compared MOPSO against two very fast algorithms that represent the state of the art in evolutionary multiobjective optimization: the NSGA II [3] and PAES[3] [12].

In the following examples, the NSGA II was run using a population size of 40 (except for test function 4, in which a population size of 80 was used), a crossover rate of 0.8, tournament selection, and a mutation rate of 1/vars, where vars = number of decision variables of the problem. In the following examples, PAES was run using 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 40 particles in the first and the third test functions. In the second test function, only 20 particles were used, and the fourth test function required 80 particles.

To allow a fair comparison of running times, all the experiments were performed on a PC with a Pentium II processor running at 333 MHz, 128 Mb of RAM and a hard drive of 6 Gbytes. Our implementation was compiled using GNU C running under Linux Red Hat release 7.1.

5.1 Test Function 1

Our first test function was proposed by Deb [2]:

$$\text{Min } f_1(x_1, x_2) = x_1 \tag{12}$$

$$\text{Min } f_2(x_1, x_2) = (1 + 10x_2) \times \left[1 - \left(\frac{x_1}{1+10x_2}\right)^\alpha - \frac{x_1}{1+10x_2}\sin(2\pi q x)\right] \tag{13}$$

[3]The source code of the NSGA II and PAES (original versions from their corresponding authors) may be downloaded from the EMOO repository located at `http://www.lania.mx/~ccoello/EMOO/EMOOsoftware.html`

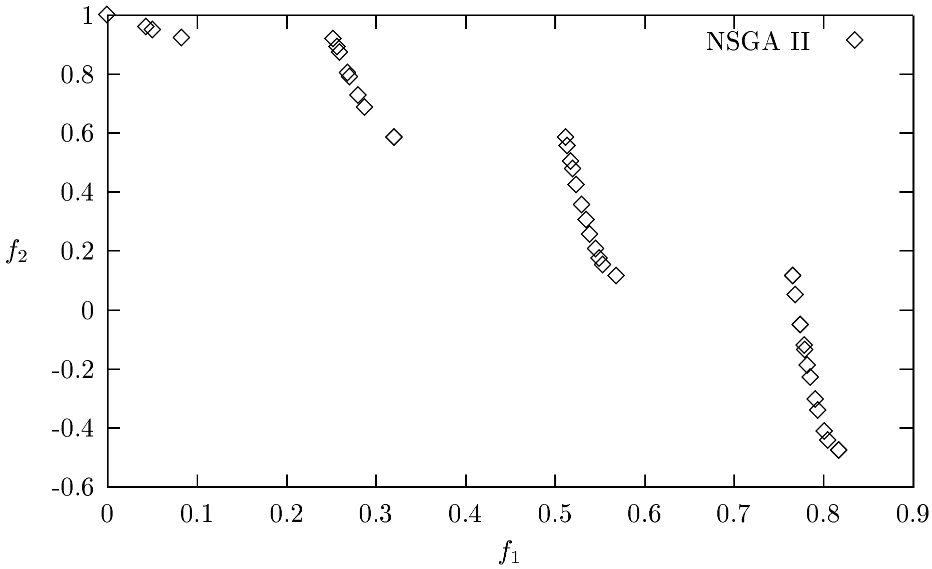

Figure 1: Pareto front produced by the NSGA-II for the first test function.

where:

$$0 \leq x_1, x_2 \leq 1 \tag{14}$$

and $q = 4$, $\alpha = 2$. This problem has its Pareto front disconnected and consistent on 4 Pareto curves.

Figures 1, 2 and 3 show the Pareto fronts produced by the NSGA-II, PAES and MOPSO, respectively for the first test function. For this example, the three algorithms performed 4040 evaluations of the fitness function. The average running time of each algorithm (over 20 runs) were the following: 0.928 seconds for the NSGA II, 2.0915 seconds for PAES and only 0.1065 seconds for our MOPSO. The average values of the metric M_1^* were the following: 0.002235304 for the NSGA II (with a standard deviation of 0.000298614), 0.008877235 for PAES (with a standard deviation of 0.015277404), and 0.002173076 for MOPSO (with a standard deviation of 0.00014726).

5.2 Test Function 2

Our second test function was proposed by Schaffer [14]:

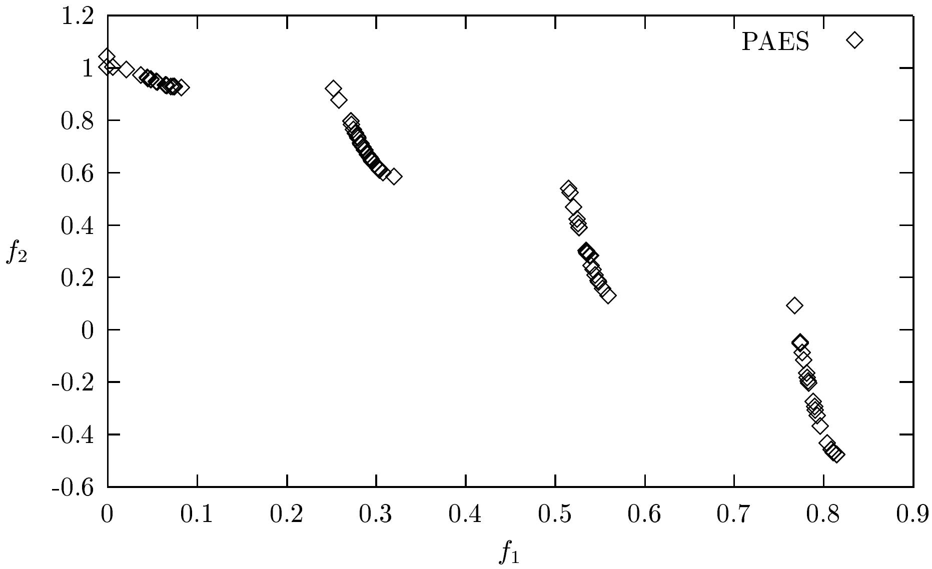

Figure 2: Pareto front produced by PAES for the first test function.

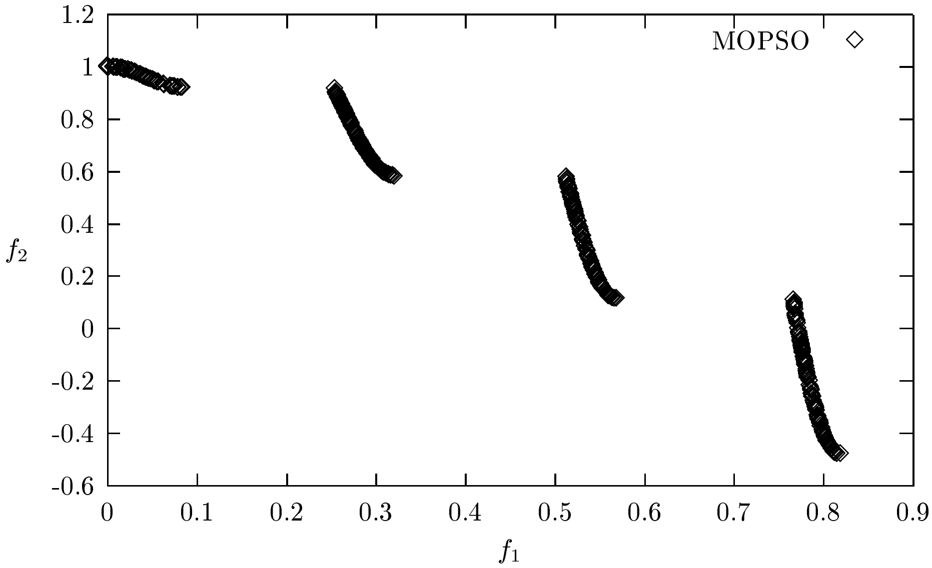

Figure 3: Pareto front produced by MOPSO for the first test function.

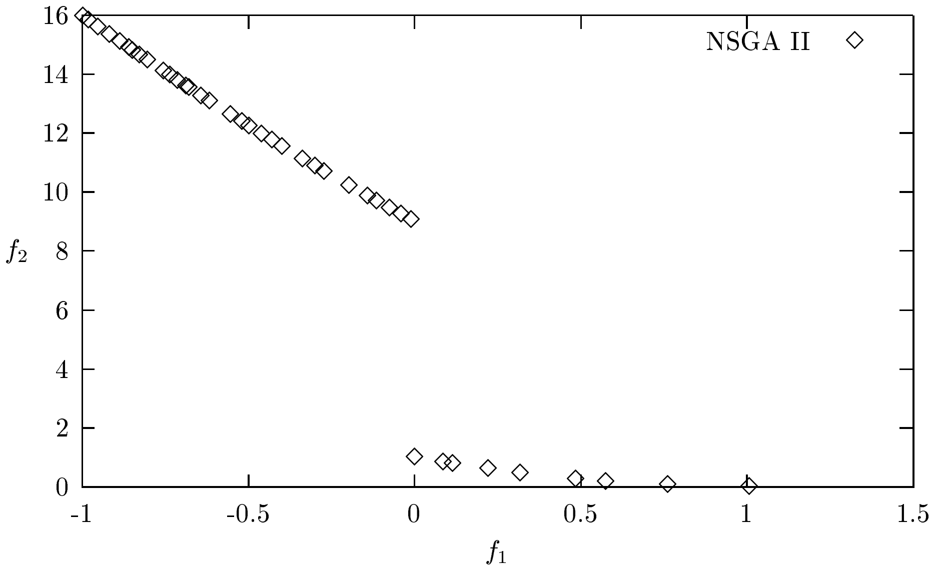

Figure 4: Pareto front produced by the NSGA-II for the second test function.

$$\text{Min } f_1(x) = \begin{cases} -x & \text{if } x \le 1 \\ -2+x & \text{if } 1 < x \le 3 \\ 4-x & \text{if } 3 < x \le 4 \\ -4+x & \text{if } x > 4 \end{cases} \tag{15}$$

$$\text{Min } f_2(x) = (x-5)^2 \tag{16}$$

where: $-5 \le x \le 10$.

This problem also has a disconnected Pareto front. Figures 4, 5 and 6 show the Pareto fronts produced by the NSGA-II, PAES and MOPSO, respectively for the second test function. For this example, the three algorithms performed 1240 evaluations of the fitness function. The average running time of each algorithm (over 20 runs) were the following: 0.3605 seconds for the NSGA II, 0.3275 seconds for PAES and only 0.071 seconds for our MOPSO. The average values of the metric M_1^* were the following: 0.001829892 for the NSGA II (with a standard deviation of 0.000434949), 0.061678751 for PAES (with a standard deviation of 0.116312867), and 0.001275433 for MOPSO (with a standard deviation of 0.0001265695).

5.3 Test Function 3

Our third example was proposed by Deb [2]:

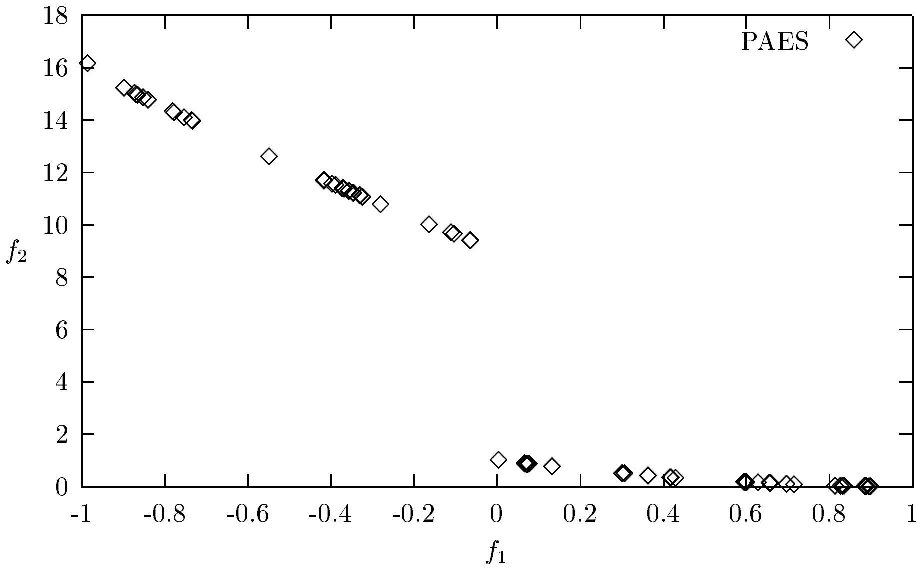

Figure 5: Pareto front produced by PAES for the second test function.

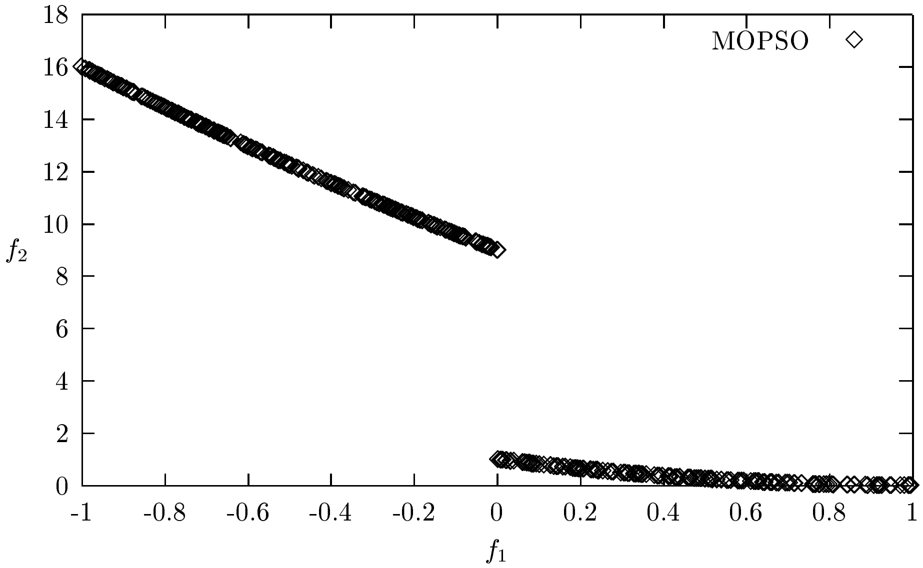

Figure 6: Pareto front produced by MOPSO for the second test function.

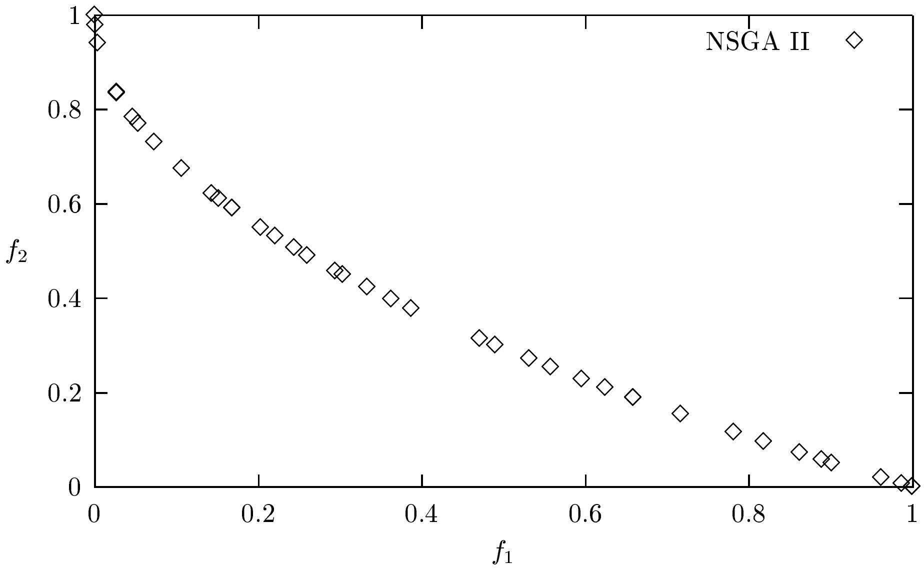

Figure 7: Pareto front produced by the NSGA-II for the third test function.

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

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

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

This problem has 60 local Pareto fronts to which the population could be easily attracted.

Figures 7, 8 and 9 show the Pareto fronts produced by the NSGA-II, PAES and MOPSO, respectively, for the third test function. For this example, the three algorithms performed 3240 evaluations of the fitness function. The average running time of each algorithm (over 20 runs) were the following: 0.927 seconds for the NSGA II, 1.2005 seconds for PAES and only 0.3035 seconds for our MOPSO. The average values of the metric M_1^* were the following: 0.023737708 for the NSGA II (with a standard deviation of 0.099354268), 0.333610493 for

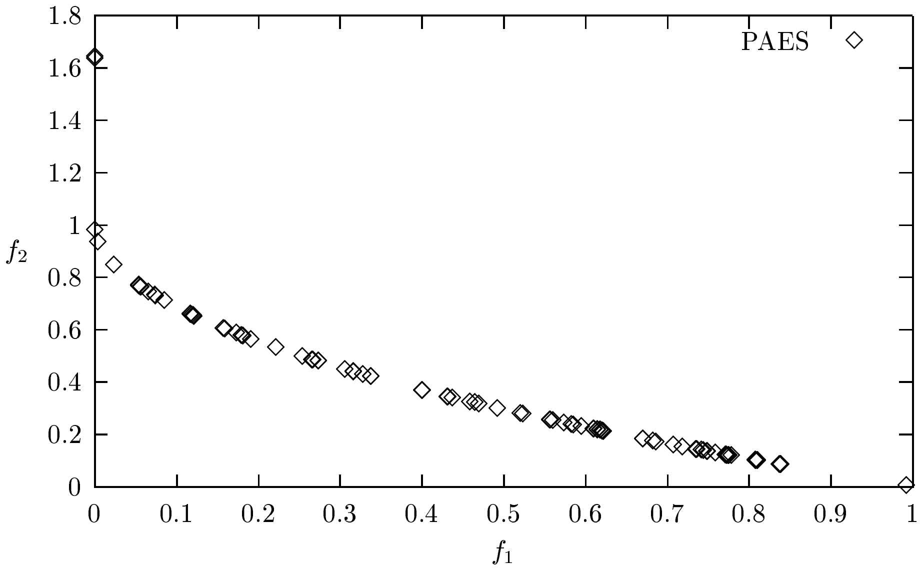

Figure 8: Pareto front produced by PAES for the third test function.

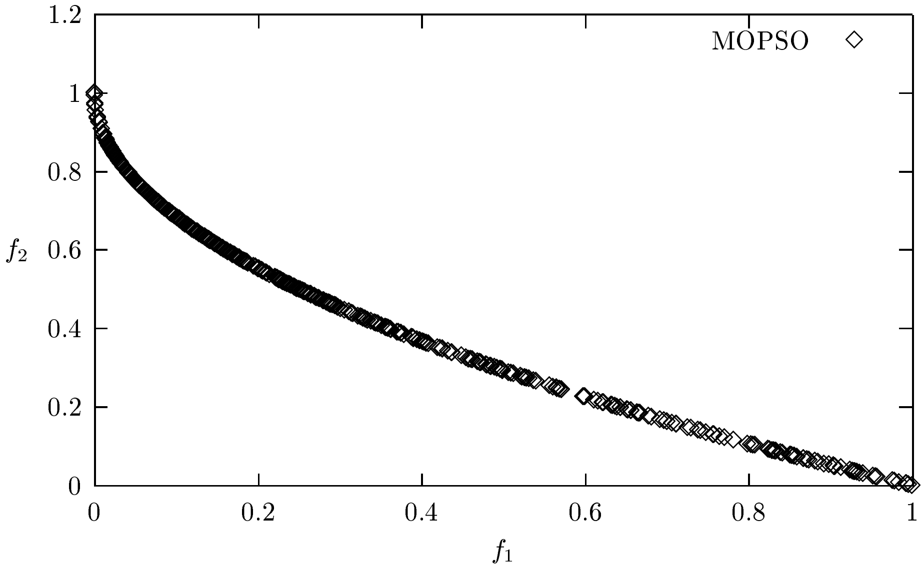

Figure 9: Pareto front produced by MOPSO for the third test function.

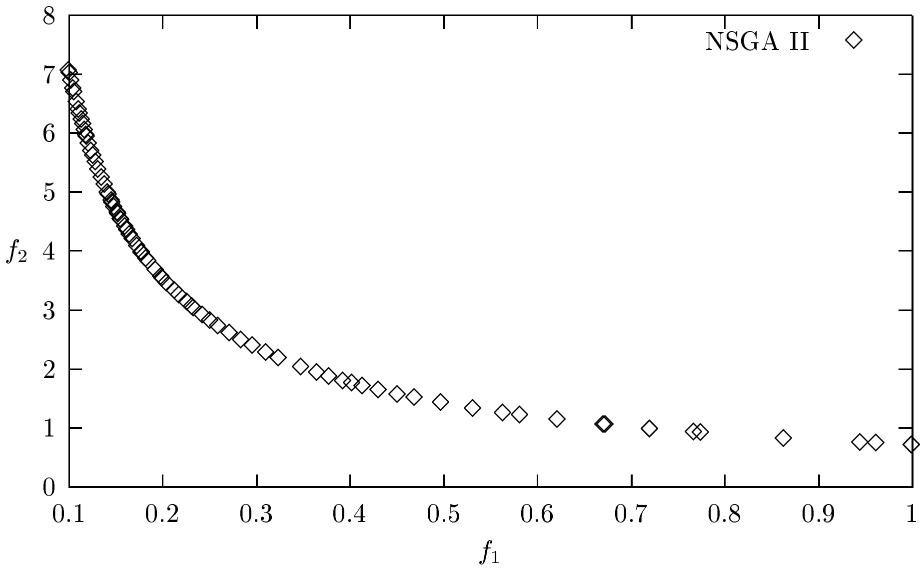

Figure 10: Pareto front produced by the NSGA-II for the fourth test function.

PAES (with a standard deviation of 0.598214005), and 0.000782244 for MOPSO (with a standard deviation of 0.000077611).

5.4 Test Function 4

Our fourth example was proposed by Deb [2]:

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

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

where:

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

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

Figures 10, 11 and 12 show the Pareto fronts produced by the NSGA-II, PAES and MOPSO, respectively, for the fourth test function. For this example, the three algorithms performed 12080 evaluations of the fitness function. The

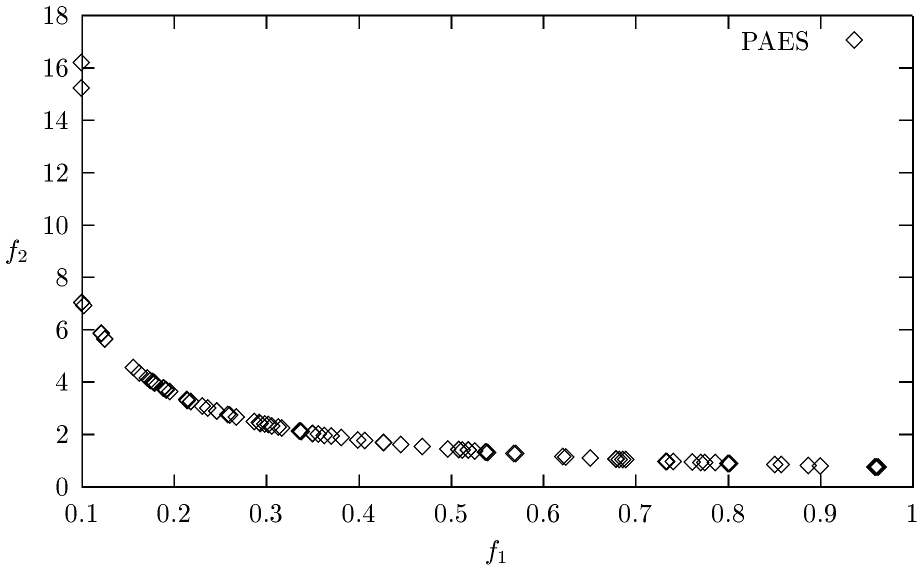

Figure 11: Pareto front produced by PAES for the fourth test function.

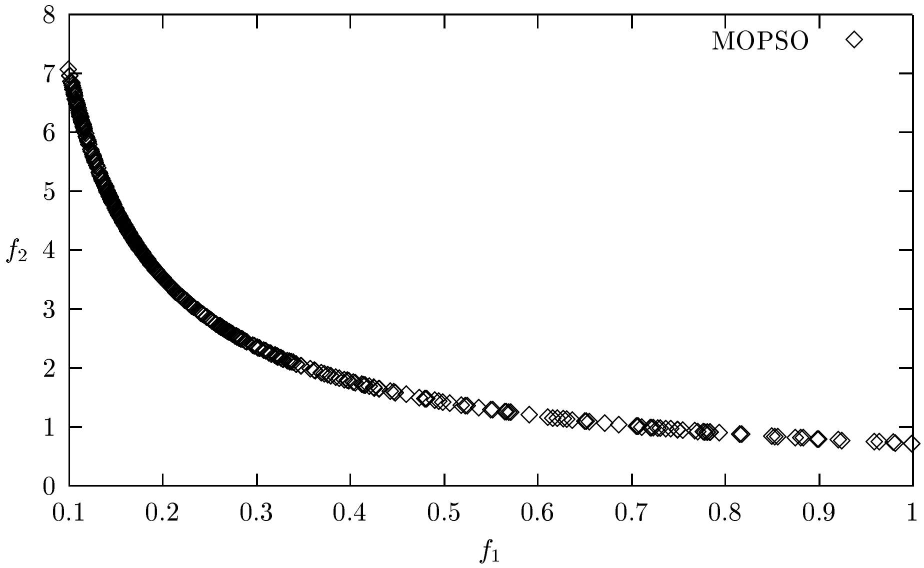

Figure 12: Pareto front produced by MOPSO for the fourth test function.

average running time of each algorithm (over 20 runs) were the following: 4.1855 seconds for the NSGA II, 1.8685 seconds for PAES and only 1.516 seconds for our MOPSO. The average values of the metric M_1^* were the following: 0.5474575 for the NSGA II (with a standard deviation of 0.730824461), 0.31079663 for PAES (with a standard deviation of 0.365548992), and 0.293114235 for MOPSO (with a standard deviation of 0.345811402).

6 Discussion of Results

In all the test functions used (we tested our approach with others that were not included in this paper due to space limitations), MOPSO performed better than the NSGA II and PAES both in terms of the metric used (it produced lower values of M_1^* with lower standard deviations) and in terms of the amount of time (it required less time than the other approaches). Furthermore, the graphical representation of the Pareto fronts produced in each case also shows that MOPSO had a better spread along the Pareto front than the other algorithms ant is was also closer to the desired Pareto fronts (see Figures 3, 6, 9, and 12).

Finally, we want to mention briefly what are the parameters used by MOPSO and some of the suggested guidelines to fine tune them:

- **Number of particles**: This is equivalent to the population size of a genetic algorithm. We recommend using between 20 and 80 particles. We normally use 40 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 used a value of 500 in our experiments.

7 Conclusions and Future Work

It is important to indicate that PSO is an unconstrained search technique. Therefore, it is necessary to develop an additional mechanism to deal with constrained multiobjective optimization problems. We have focused this paper only on the generation of nondominated vectors and in the mechanism to maintain diversity, but we have not dealt with constrained test functions. The extension to handle constrained problems is, however, not too difficult to develop and it is currently under way.

A sensitivity analysis is also under way, so that we can determine the role of each of the parameters used in the performance of the algorithm.

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 Maestría en Inteligencia Artificial of LANIA and the Universidad Veracruzana.

References

[1] Carlos A. Coello Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, August 1999.

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

[3] 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.

[4] 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.

[5] Russell C. Eberhart and Yuhui Shi. Comparison between Genetic Algorithms and Particle Swarm Optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A.E. Eibe, editors, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pages 611–619. Springer-Verlag, March 1998.

[6] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

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

[8] Hisao Ishibuchi and Tadahiko Murata. Multi-Objective Genetic Local Search Algorithm. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.

[9] James Kennedy and Russell C. Eberhart. Particle Swarm Optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, New Jersey, 1995. IEEE Service Center.

[10] James Kennedy and Russell C. Eberhart. A Discrete Binary Version of the Particle Swarm Algorithm. In *Proceedings of the 1997 IEEE Conference on Systems, Man, and Cybernetics*, pages 4104–4109, Piscataway, New Jersey, 1997. IEEE Service Center.

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

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

[13] Marco Laumanns, Eckart Zitzler, and Lothar Thiele. On the Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective 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 181–196. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[14] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[15] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 7(3):1–26, 2000.

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