Multi-Objective Optimization with a Gaussian PSO algorithm

Susana C. Esquivel and Leticia C. Cagnina
Lab. de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)*
Dpto. de Informática - Universidad Nacional de San Luis
Ejército de los Andes 950 - D5700HHW - San Luis - Argentina
{ *esquivel,lcagnina* }*@unsl.edu.ar*

Abstract

Particle Swarm Optimization is a popular heuristic used to solve suitably and effectively mono-objective problems. In this paper, we present an adaptation of this heuristic to treat unconstrained multi-objective problems. The proposed approach (called G-MOPSO) incorporates a Gaussian update of individuals, Pareto dominance, an elitist policy, and a shake-mechanism to maintain diversity.
In order to validate our algorithm, we use four well-known test functions with different characteristics. Preliminary results are compared with respect to those obtained by a multi-objective evolutionary algorithm representative of the state-of-the-art: NSGA-II. We also compare the results with those obtained by OMOPSO, a multi-objective PSO based algorithm.
The performance of our approach is comparable with the NSGA-II and outperforms the OMOPSO.

Keywords: Multi-objective Optimization, Particle Swarm Optimization, Pareto Optimality.

Resumen

Particle Swarm Optimization es una heurística popular usada para resolver adecuada y efectivamente problemas mono-objetivo. En este artículo, presentamos una primera adaptación de esta heurística para tratar problemas multi-objetivo sin restricciones. La propuesta (llamada G-MOPSO) incorpora una actualización Gaussiana, dominancia Pareto, una política elitista, un archivo externo y un shake-mecanismo para mantener la diversidad. Para validar nuestro algoritmo, usamos cuatro funciones de prueba bien conocidas, con diferentes características. Los resultados preliminares son comparados con los valores obtenidos por un algoritmo evolutivo multi-objetivo representativo del estado del arte en el área: NSGA-II. También comparamos los resultados con los obtenidos por OMOPSO, un algoritmo multi-objetivo basado en la heurística PSO.
La performance de nuestra propuesta es comparable con la de NSGA-II y supera a la de OMOPSO.

Palabras claves: Optimización Multi-Objetivo, Particle Swarm Optimization, Optimalidad Pareto.

*The LIDIC is supported by Universidad Nacional de San Luis and the ANPCyT (National Agency to Promote Science and Technology.

1 INTRODUCTION

Real world problems frequently require that a number of competing objectives have to be traded against one other whilst seeking a viable solution to a given problem. These objectives cannot be met by a single solution. These problem types present different characteristics in the variable and objective function spaces, and solution space.
As can be seen in the mono-objective cases, the use of heuristic to solve optimization problems has experienced a significative grow in the last ten years [7], [19], [4]. One of these heuristics is Particle Swarm Optimization (PSO). Given the promising results reported in the mono-objective optimization domain, to consider the application of PSO to the multi-objective domain, is a natural progression.
In this paper, we present a modified PSO algorithm in order to tackle unconstrained multi-objective problems. This version is simpler than other PSO-based approaches [12], [20], [16], [18], [17] and its performance is comparable with the performance of one of the best evolutionary algorithms of the multi-objective optimization area.
The remainder of the paper is organized as follows: Section 2 gives a brief description of the most relevant previous work. Section 3 reviews the basic concepts of multi-objective optimization and classical PSO algorithm. Section 4 describes our approach. Section 5 presents the test functions taken from the specialized literature to validate our approach, and the metrics used to evaluate the performance of the algorithms. Section 6 explains the experimental design. Section 7 shows and discusses the results obtained. Section 8 shows statistical analysis for the results obtained. Finally, our conclusions and possible future research lines are presented in Section 9.

2 RELATED WORK

Talbi *et al.* proposed a PSO algorithm hybridized with a continuation method. The idea is to combine the global search of PSO with the local search inherent to continuation method. The convergence results show that the performance of their algorithm is good. [18]
Toscano and Coello Coello proposed the use of clustering techniques to improve the performance of a multi-objective PSO. They used Pareto dominance to guide the flight direction of a particle and had a set of sub-swarms to focalize the search. A PSO algorithm is run in each sub-swarm and at some point the sub-swarms exchange information. Their results indicate that the approach is highly competitive with respect to algorithms representative of the state-of-the-art in evolutionary multiobjective optimization. [5]
Becerra *et al.* used different optimization techniques combined to solve hard multi-objective problems. They used the ϵ-constraint method to obtain points near the true Pareto Front and then rough sets were applied to spread the solution on the entire Pareto Front. Their algorithm performs well and can solve some problems that any algorithm can. [15]
Jaeggi *et al.* presented a Tabu Search algorithm for multi-objective optimization enhanced with a novel parameter selection strategy. Two variants are proposed and tested with several standard functions. The results are compared with those of NSGA-II demonstrating the comparability of performance. [11]
Hernandez-Diaz *et al.* proposed a mechanism that can be seen as a variant of ϵ-dominance trying to overcome the main limitation that has: the loss of several nondominated solutions. They tested the mechanism using three algorithms: differential evolution, steady-state genetic algorithm and a variation of differential evolution. In all cases, their mechanism demonstrated the effectiveness in found the best metric results. [8]
Reyes and Coello Coello presented a based-Pareto dominance approach (OMOPSO) with a crowding

factor for the selection of leaders and two external files: one for storing the leaders being used and the other for saving the final solutions. They used the ϵ-dominance concept to select the particles to put at the final file and they included a scheme to subdivide the swarm into three subsets. A mutation operator also was used. Their results demonstrated be highly competitive with respect to five algorithms representative of the state-of-the-art in multiobjective optimization. [17]
Deb *et al.* proposed a nondominated sorting genetic algorithm (NSGA-II) which alleviates the three main difficulties of multi-objective evolutionary algorithms (with nondominated sorting and sharing): the high computational complexity, the nonelitism and the need for specifying a sharing parameter. A selection operator creates a mating pool by combining the parent and offspring population, and selecting the best solutions for the new population. Their results showed that the algorithm is able to find much better spread of solutions and convergence near the True Pareto front, when is compared with other algorithms. [13]

We select the last two algorithms (OMOPSO and NSGA-II) to compare the performance of our approach. The first because is a based-PSO algorithm with some similar characteristics of our G-MOPSO and the second because is one of the best evolutionary algorithm representative of the multi-objective optimization area.

3 BACKGROUND

3.1 The PSO algorithm: classical model

The Particle Swarm Optimization algorithm (PSO) proposed by J. Kennedy and R. Eberhart [14] in 1995 is based on the behavior of communities that have both social and individual behavior. In the algorithm, population-based, each individual (named *particle*) represents a solution in a $n-$dimensional space. All the particles have knowledge of its previous best experience and know the global best experience (best solution) found by the entire population (*swarm*). Particles update their exploration directions (*flights*) and position at every iteration (*cycle*) of the algorithm using the following equations:

$$v_{i,j} = w \times v_{i,j} + c_1 \times r_1 \times (p_{i,j} - x_{i,j}) + c_2 \times r_2 \times (p_{g,j} - x_{i,j}) \quad (1)$$

$$x_{i,j} = x_{i,j} + v_{i,j} \quad (2)$$

where w is the inertia factor influencing the local and global abilities of the algorithm, $v_{i,j}$ is the velocity of the particle i in the $j-th$ dimension, c_1 and c_2 are weights affecting the cognitive and social factors, respectively. r_1 and $r_2 \sim \mathcal{U}(0,1)$; p_i stands for the best value found by particle i (*pbest*) and p_g denotes the global best found by the entire swarm (*gbest*).

After the velocity is updated, the new position i in its $j-th$ dimension is calculated. This process is repeated for every dimension and for all the particles in the swarm.

In order to use PSO for multi-objective optimization problems, a classical version of PSO was hybridized with some concepts taken from de multi-objective evolutionary algorithms field. This version is described in Section 4.

3.2 Multi-Objective Optimization Problem

The unconstrained Multi-Objective Optimization Problem can be defined as follows [4]:
Definition 1: Find the vector

$$\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$$

which optimizes (minimizes or maximizes) the vector function:

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

The k components of the vector $\vec{f}(\vec{x})$ are the criteria to be considered. The vector $\vec{x}^*$ denotes the optimum solutions.

When there are several objective functions, the concept of optimum changes, because in multi-objective optimization problems the purpose is to find "trade-off" solutions rather than a single solution. The concept of optimum commonly adopted in multi-objective optimization is the one proposed by Vilfredo Pareto in 1986 (and called Pareto optimality).

Definition 2 *Pareto Optimality*: A solution $\vec{x}^* \in \Omega$ is Pareto optimal with respect to Ω if and only if there is no $\vec{x} \in \Omega$ for which $v = \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T$ dominates $v = \vec{f}(\vec{x}^*) = [f_1(\vec{x}^*), f_2(\vec{x}^*), \ldots, f_k(\vec{x}^*)]^T$. That is, $\vec{x}^*$ is Pareto optimal if there exists no feasible vector $\vec{x}$ which would decrease some criteria without causing a simultaneous increase in at least one other criterion.

Definition 3 *Pareto Dominance*: A vector $\vec{x} = (x_1, x_2, \ldots, x_k)$ is said to dominate $\vec{y} = (y_1, y_2, \ldots, y_k)$, denoted by $\vec{x} \preceq \vec{y}$, if and only if $\vec{x}$ is partially less than $\vec{y}$, i.e., $\forall i \in \{1, 2, \ldots, k\}$, $x_i \leq y_i$ and, at least for one i, $x_i < y_i$.

Definition 4 *Pareto Optimal Set*: For a given multi-objective problem $\vec{f}(x)$, the Pareto optimal set, denoted by $\mathcal{P}^*$ or $\mathcal{P}_{true}$, is defined as:

$$\mathcal{P}^* = \{x \in \Omega \mid \nexists x^{'} \in \Omega \vec{f}(x^{'}) \preceq \vec{f}(x)\}. \tag{4}$$

Definition 5 *Pareto front*: For a given multi-objective problem $\vec{f}(x)$ and Pareto optimal set $\mathcal{P}^*$, the Pareto front, denoted by $\mathcal{PF}^*$ or $\mathcal{PF}_{true}$, is defined as:

$$\mathcal{PF}^* = \{\vec{y} = \vec{f} = (f_1(x), f_2(x), \ldots, f_k(x)) \mid x \in \mathcal{P}^*\} \tag{5}$$

4 OUR APPROACH

In this section, we describe our PSO-based approach for multi-objective optimization: G-MOPSO (Gaussian Multi-Objective Particle Swarm Optimization). As we stated before, this is a version of the algorithm for unconstrained problems. We extend the classical model described above with the following characteristics.

4.1 Gaussian update

There are some multi-objective problems that require a considerably high number of objective function evaluations in order to obtain a front near the Pareto True front. For help the search process, we combine the equation (2) for updating the particles with the equation (6) as we proposed in a previous work [2]. The algorithm selects the equation to update the position of particles, depending a probability value. For the Gaussian equation (6) the probability of be selected was set in 0.005. That value was empirically found to be the best after performing a series of experiments with all the test functions evaluated. Then, Gaussian equation for updating position is defined as:

$$x_i = N\left(\frac{p_i + p_g}{2}, |p_i - p_g|\right) \tag{6}$$

where x_i is the particle to be updated, N is the Gaussian random generator, p_i is the best position reached by the particle x_i (that is, *pbest*), and p_g is the best position reached by any particle in the swarm (that is, *gbest*).

4.2 External file for elitist policy

The elitist policy was implemented maintaining the best solutions (non-dominated) found in the flight cycles (iterations) in an external archive. This archive has a grid structure as it was proposed by K. Deb [6], and is constructed as follows: each objective is divided into 2^d equal divisions. In this way the entire search space is divided into 2^{dk} unique equal size $k-$dimensional hypercubes (d is a user parameter and k is the number of objective functions). The stored solutions are placed in one of these hypercubes according to their locations in the objective space. The number of solutions in each hypercube is counted. When the archive is full and there is a new non-dominated solution it cannot be included automatically. First, the hypercube that has more non-dominated solutions is found. If the new solution does not belong to that hypercube it is inserted in the archive and at random one of the solutions from the highest covered hypercube is deleted. So, non-dominated solutions are privileged and placed in the archive. When non-dominated solutions compete for a space in the archive, they are evaluated based on how crowded they are in objective function space. The one residing in the least crowded area gets preference. In this manner, we obtain diversity in the non-dominated solutions.

4.3 Mechanism to select the *pbest* and *gbest* particles

The selection of the best particles is different from the mono-objective classical model. Our approach updates *pbest*, the best experience found for a particle, only when the new particle is non-dominated and it dominates the previous *pbest*. In order to select *gbest*, the global best particle, at each iteration we randomly select a non-dominated particle of the external archive, because by definition all the Pareto optimal solutions are equally good.

4.4 Shake-Mechanism

For difficult functions, it is common to have some stagnation problems when we try to obtain solutions close to the Pareto True front. In order to overcome this problem, we incorporate a *shake-mechanism* [1] to maintain diversity into the population. This mechanism is dynamic, that is, applied with a high probability at the beginning of the search process but with a low probability at the end. The probability value is decreased at each iteration of the algorithm. In order to implement the *shake-mechanism*, all the dimensions of the particle are changed by a random value (within the allowable range) depending the current probability. This process is applied to all particles into the swarm.

4.5 The pseudo-code of our approach: G-MOPSO

Figure 1 shows a pseudo-code of G-MOPSO. In that, the particle swarm is initialized with random values corresponding to the ranges (depending on the test functions) and the velocities are initialized with zero values (lines 2-3). Then, the swarm is evaluated using the corresponding objective functions (line 4). Next, the fitness vectors are updated (line 5). As we are dealing with multi-objective optimization, these vectors store the values of each decision variable, in which the particles obtained the best values in a Pareto sense (that is, for each function). At this stage of the algorithm these vectors are filled with the results of the initial particle evaluations. Analogously, these values are copied in the *pbest* vectors (line 6) and all non-dominated particles are inserted in the grid, i.e. in the external file (line 7). Global *gbest* particle is randomly selected (line 8) from the external file.
The flight cycles start at line 9, the velocity of each particle is updated, and its position is also updated using the corresponding equation (lines 10-18). The keeping operation is carried out to maintain the particles into the allowable range values (line 19). Then the *shake-mechanism* is applied (line 20), the

```
1.  G−MOPSO{
2.      Init_Pop();
3.      Init_Velocity();
4.      Evaluate_Pop();
5.      Update_Fbest();
6.      Update_Pbest();
7.      Insert_nodom();
8.      Gbestpos = rnd(0,nodomfileSize)
9.      for(i=1 to MAXCYCLES){
10.         for(j=0 to MAXPARTICLES){
11.             Update_Velocity with eq.(1);
12.             Update_Particle:
13.             if rnd(0,1) > 0.005
14.                Update with eq.(2)
15.             else
16.                Gaussian update with eq.(6)
17.             end
18.         }
19.         Keeping();
20.         Shake_Mechanism();
21.         Evaluate_Pop();
22.         Update_Fbest();
23.         Update_Pbest();
24.         Insert_nodom();
25.         Gbestpos = rnd(0,nodomfileSize)
26.     }
27.     Print_Statistics();
28.     Generate_Outfile();
29. }
```

Figure 1: Pseudo-code G-MPSO.

particles evaluated and, *pbest* vectors updated (lines 21-23).
As the particles were moved in the search space because they changed positions, the dominance of each particle (line 24) is verified and, if appropriate, they are inserted in the grid. Then the new *gbest* is randomly selected (line 25). The cycle is executed until the condition is false and at this point we print the statistics and generate an output file, which contains the non-dominated particles (lines 27 and 28).

5 TEST FUNCTIONS AND METRICS

5.1 Test functions

In order to validate our approach, we selected the following four well-known unconstrained test functions [3]. Each one was selected because has different characteristics:

Viennet3: Proposed by Viennet, it is an (unconstrained) three objective function that has its $\mathcal{P}_{true}$ disconnected and unsymmetric, and its $\mathcal{PF}_{true}$ is connected. It is defined as:

$$F = (f_1(x,y), f_2(x,y), f_3(x,y)) \text{ with } -3 \leq x, y \leq 3$$

$$f_1(x,y) = 0.5 * (x^2 + y^2) + sin(x^2 + y^2)$$

$$f_2(x,y) = \frac{(3x - 2y + 4)^2}{8} + \frac{(x - y + 1)^2}{27} + 15$$

$$f_3(x,y) = \frac{1}{(x^2 + y^2 + 1)} - 1.1e^{(-x^2 - y^2)}$$

Fonseca2: Proposed by Fonseca and Fleming, it is an (unconstrained) two objective function that has its $\mathcal{P}_{true}$ connected, and its $\mathcal{PF}_{true}$ is concave. It is defined as:

$$F = (f_1(x,y), f_2(x,y)) \text{ with } -4 \leq x, y \leq 4$$

$$f_1(x,y) = 1 - exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2)$$

$$f_2(x,y) = 1 - exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2)$$

Viennet2: Proposed by Viennet *et al.*, it is an (unconstrained) three objective function that has its $\mathcal{P}_{true}$ connected, and its $\mathcal{PF}_{true}$ is disconnected. It is defined as:

$$F = (f_1(x,y), f_2(x,y), f_3(x,y)) \text{ with } -4 \leq x, y \leq 4$$

$$f_1(x,y) = \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3$$

$$f_2(x,y) = \frac{(x+y-3)^2}{36} + \frac{(-x+y+2)^2}{8} - 17$$

$$f_3(x,y) = \frac{(x+2y-1)^2}{175} + \frac{(-x+2y)^2}{17} - 13$$

Schaffer: Proposed by Jones *et al.*, it is an (unconstrained) two objective function that has its $\mathcal{P}_{true}$ connected, and its $\mathcal{PF}_{true}$ is convex. It is defined as:

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

$$f_1(x,y) = x^2$$

$$f_2(x,y) = (x-2)^2$$

5.2 Metrics

The performance analysis of multi-objective optimization algorithms is made assigning a measure of quality to an approximation set, that is the Pareto front obtained by the algorithm. There are several performance indicators having different properties and measuring different aspects of the solution set. Normally, two issues are taken into account: minimize the distance of the Pareto front obtained with respect to the Pareto True front, and maximize the distribution of solutions so we have vectors as smooth and uniform as possible. For that, usually the metrics can be classified into three categories depending if they evaluate the closeness to the Pareto True front, the diversity of solutions obtained, or both factors [6]. We select one metric of each type: Generational Distance, Spread and Hypervolume [4], respectively.

Spread: is a diversity metric that measures the extent of spread achieved among the Pareto front obtained. It is defined as:

$$Spr = \Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - d'|}{d_f + d_l + (N-1)d'}$$

where d_i is the Euclidean distance between consecutive solutions, d' is the mean of these distances and, d_f and d_l are the Euclidean distances to the extremes (bounding solutions of the Pareto True front) in the objective space. A zero-value indicates an ideal distribution (perfect spread).

Generational Distance: measures how far the elements are in the Pareto front obtained, from those in the Pareto True front. It is defined as:

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

where n is the number of vectors in the Pareto front obtained and d_i is the Euclidean distance (measured in the objective space) between each of these solutions and the nearest member of the Pareto True front. A zero-value indicates that all solutions found are in the Pareto True front set.

Hypervolume: calculates the volume Q (in the objective space) covered by each solution of the Pareto front obtained. Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with a reference point w and the solution i as the diagonal corners of the hypercube. The reference point can simply be found using the vector with worst objective function value. Then, a union of all hypercubes is found and its hypervolume calculated as:

$$Hyp = volume\left(\bigcup_{i=1}^{|Q|} v_i\right)$$

Algorithms with large values of Hyp are desirable. The metric is only defined for minimizing problems with three objective functions as the maximum.

6 EXPERIMENTAL DESIGN

The experiments were designed to evaluate the performance of G-MOPSO. We use the test functions described before. For each function, we obtained the Pareto front with our algorithm and, the value of the metrics (maximum, minimum, mean and deviation standard). To compare the performance of our algorithm, we also run other two algorithms: NSGA-II (one of the state-of-the-art-representative) and OMOPSO (a PSO-based with some similar characteristics that G-MOPSO) both described previously. To run NSGA-II and OMOPSO we downloaded the codes, compiled and run with the JMetal-NEO free on-line toolkit [10]. JMetal is a Java-based framework that facilities the development, experimentation and study of metaheuristics for multi-objective optimization problems.
All the algorithms performed 50 independent runs with 25,000 evaluations ($iterations \times individuals$) of the objective function for each tested function, in order to compare them on the basis of the same amount of computational effort. The parameter setting for all the algorithms are summarized in Table 1. For NSGA-II and OMOPSO we used the parameter settings proposed by the original authors. The parameter values for G-MOPSO were empirically derived from a set of previous experiments. The entry 'Divisions', in Table 1, indicates the number of hypercubes in the grid used to maintain diversity and, 'Upd eq prob.' the probabilities of update of particles used to select the equation (2) and (6), respectively.

7 RESULTS AND DISCUSSION

The values shown in Table 2 correspond to mean and standard deviation values calculated over the 50 runs performed in each experiment. The best value reached for each function is marked with boldface.

The performance of G-MOPSO across all functions studied outperforms OMOPSO in all metrics. These results are notable because G-MOPSO, although share some characteristics with OMOPSO, is

Table 1: Parameter Settings.

Parameters	G-M0PSO	NSGA-II	OMOPSO
Iterations	5000	250	250
Extern file size	100	100	100
Crossover prob.	-	0.9(SBX)	-
Mutation prob.	-	0.5	0.5
eta	-	-	0.0075
Individuals	5	100	100
Divisions	4	-	-
C1=C2/W	1.5/0.5	-	-
SHAKE prob.	0.95	-	-
Upd eq prob.	0.995/0.005	-	-

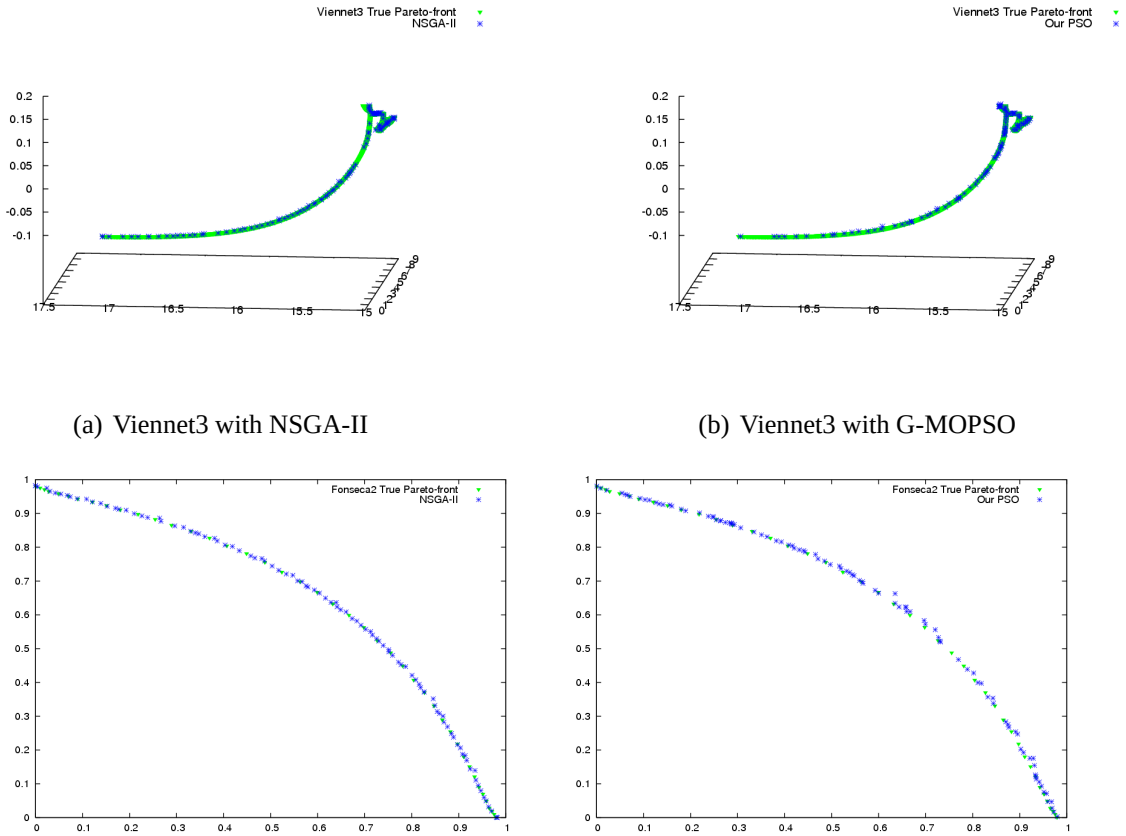
Table 2: Values of the Metrics

Functions	**NSGA-II**		**G-MOPSO**		**OMOPSO**	
	Spread					
	mean	***stdDev.***	***mean***	***stdDev.***	***mean***	***stdDev.***
Viennet3	**0.4179**	0.0390	0.6993	0.0565	0.7454	0.1662
Fonseca2	**0.3727**	0.0362	0.5834	0.0669	0.7383	0.2459
Viennet2	**0.7674**	0.0462	0.8395	0.0438	0.9487	0.1736
Schaffer	**0.2721**	0.0312	0.7611	0.0705	0.9425	0.1807
	Generational Distance					
Viennet3	**2.50E-4**	4.92E-5	4.67E-4	2.50E-4	0.0447	0.0329
Fonseca2	**0.0012**	3.88E-5	**0.0012**	5.82E-5	0.1570	0.0626
Viennet2	1.90E-4	5.32E-5	**1.33E-4**	4.84E-5	0.0479	0.0668
Schaffer	2.40E-4	1.62E-5	**2.34E-4**	1.19E-5	91.1728	179.9310
	Hypervolume					
Viennet3	**0.8325**	7.50E-4	0.8304	0.0013	0.5786	0.1349
Fonseca2	**0.3055**	4.22E-4	0.3008	0.013	0.0335	0.0499
Viennet2	0.9956	9.18E-5	**0.9958**	1.60E-4	0.9573	0.0310
Schaffer	**0.8294**	1.14E-4	0.8260	0.0011	0.0503	0.1663

simpler. The G-MOPSO metrics results are comparable with those of NSGA-II. For the spread metric, NSGA-II is better than our algorithm, although the values reached by G-MOPSO are not very different in some cases. For the generational distance metric, NSGA-II is better in Viennet3 and Fonseca2, but G-MOPSO outperforms NSGA-II in Viennet2 and Schaffer functions. The hypervolume values of NSGA-II are almost the same obtained by G-MOPSO.
Figures 2 and 3 show the Pareto fronts obtained by NSGA-II and G-MOPSO, with respect to the true Pareto fronts (obtained by enumeration). The graphics correspond to the fronts that obtained the minimum value of spread in the total of runs. We do not include the graphics of OMOPSO fronts because considerate the metrics results obtained by OMOPSO are not comparable with G-MOPSO neither NSGA-II.

8 STATISTICAL ANALYSIS

To analyze the performance of our algorithm we used a statistical test. We do an analysis of variance between NSGA-II, OMOPSO and G-MOPSO using the mean values of the metrics obtained at the 50 independent runs. We analyze each metric separately. We apply the Kruskal-Wallis [9] nonparametric one-way analysis because the values (the sample) do not have a normal distribution (determined with the Shapiro-Wilk normality test) neither same variances (determined with the Bartlett test of homogeneity of variances).



(a) Viennet3 with NSGA-II

(b) Viennet3 with G-MOPSO

(c) Fonseca2 with NSGA-II

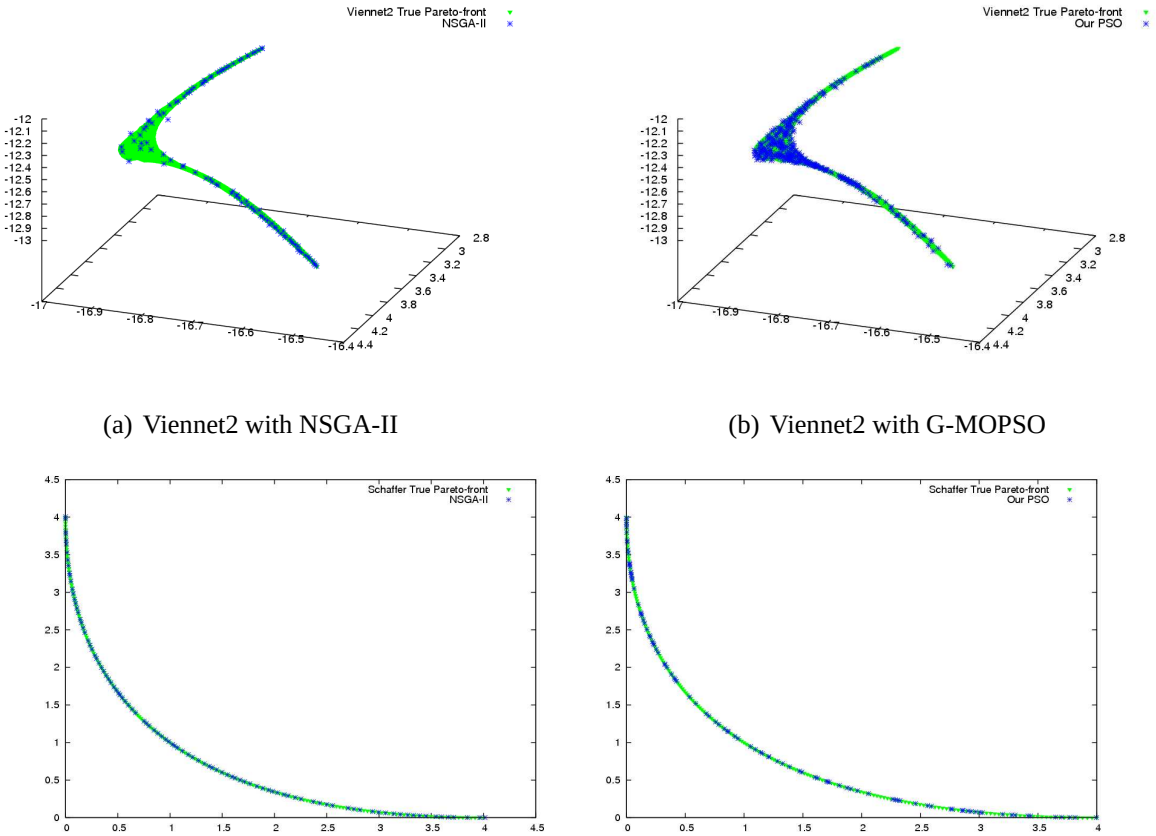
(d) Fonseca2 with G-MOPSO

Figure 2: Pareto Fronts obtained with the algorithms.

The Kruskal-Wallis test returns the *p-value* for the null hypothesis for all samples. If the *p-value* is zero or near, that suggests that at least one sample is significantly different (or *statistically significative*) than the other samples. Usually, if *p-value* is less than 0.05, we declare that the results are significative.
Table 3 shows the *p-value* for each function and each metric. The results indicate that values reached with G-MOPSO are statistically significative from those of NSGA-II and OMOPSO. Only for GD metric for Schaffer problem, the *p-value* was higher than 0.05 so for that metric, G-MOPSO do not obtained result statistical difference of NSGA-II. The test shows that G-MOPSO is statistically significant of OMOPSO, and if we observe Table 2, we can conclude that the performance of our algorithm is higher. With respect a NSGA-II we conclude that, although some results are almost similar, the statistical analysis says that there are significative difference with G-MOPSO. This last result was expected because NSGA-II actually is one of best evolutionary algorithm in the field of multi-objective optimization.

9 CONCLUSIONS AND FUTURE WORK

The performance of our approach G-MOPSO turned out to be satisfactory. The aim was to determine if this version was able to obtain results at least comparable to those obtained with a well-known multi-objective algorithm, and our algorithm did. We also compare these results with an existent PSO-based algorithm and our version (simpler) obtained better results. For that, we conclude that G-



(a) Viennet2 with NSGA-II

(b) Viennet2 with G-MOPSO

(c) Schaffer with NSGA-II

(d) Schaffer with G-MOPSO

Figure 3: Pareto Fronts obtained with the algorithms.

Table 3: Kruskal-Wallis' p-values.

Functions	**G-MOPSO vs NSGA-II**			**G-MOPSO vs OMOPSO**		
	Spr	GD	Hyp	Spr	GD	Hyp
Viennet3	2.2E-16	1.8E-9	7.9E-13	0.039	2.2E-16	2.2E-16
Fonseca2	2.2E-16	8.7E-4	2.2E-16	2.0E-4	2.2E-16	2.2E-16
Viennet2	1.1E-10	1.94E-6	6.5E-7	1.3E-5	2.2E-16	2.2E-16
Schaffer	2.2E-16	**0.089**	2.2E-16	8.7E-14	2.2E-16	2.2E-16

MOPSO is a promising approach to multi-objective optimization because this study of performance was satisfactory.
In our future work we will pretend enhance the results of this version and incorporate a mechanism to handle constraints, so we also can study its performance using constrained multi-objective problems.

REFERENCES

[1] L. Cagnina, S. Esquivel, and C. Coello Coello. A bi-population pso with a shake-mechanism for solving constrained numerical optimization. In *IEEE Congress on Evolutionary Computation - CEC2007*, pages 670–676, Singapore, 2007.

[2] L. C. Cagnina, S. C. Esquivel, and C. A. Coello Coello. A particle swarm optimizer for constrained numerical optimization. In *9th International Conference - Parallel problem Solving from Nature - PPSN IX*, pages 910–919, Reykjavik, Island, 2006.

[3] C. Coello Coello, G. Lamont, and D. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Kluwer Academic Publishers, 2002. ISBN 0-306-46762-3.

[4] C. Coello Coello, G. Lamont, and D. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007. ISBN 978-0-387-33254-3.

[5] C. A. Coello Coello and G. Toscano Pulido. Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. In *Genetic and Evolutionary Computation Conference*, volume 3102 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

[6] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. J. Wiley & Sons., England, 2001.

[7] M. Ehrgott. *Multicriteria Optimization*. Springer, Berlin, second edition, 2005. ISBN 3-540-21398-8.

[8] A. G. Hernandez-Diaz, L. V. Santana-Quintero, C. A. Coello Coello, and J. Molina. Pareto-adaptive ϵ-dominance. Technical report, EVOCINV-02-2006, Mexico, 2006.

[9] M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*. Wiley, 1973.

[10] http://sourceforge.net/projects/jmetal.

[11] D. M. Jaeggi, G. T. Parks, T. Kipouros, and P. J. Clarkson. The development of a multi-objective tabu search algorithm for continuous optimisation problems. *European Journal of Operational Research*, 185:1192–1212, 2008.

[12] S. Janson and D. Merkle. A new multi-objective particle swarm optimization algorithm using clustering applied to automated docking. *Lecture Notes in Computer Science*, 3636, 2005.

[13] S. Agrawal K. Deb, A. Pratap and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: Nsga-ii. In *IEEE Transaction on Evolutionary Computation*, pages 182–197, 2002.

[14] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ., 1995. IEEE Service Center.

[15] R. Landa Becerra, C. A. Coello Coello, A. G. Hernandez-Diaz, R. Caballero, and J. Molina. Alternative techniques to solve hard multi-objective optimization problems. In *GECCO 2007*, England, UK, 2007.

[16] S. Mostaghim and J. Teich. Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *2004 Congress on Evolutionary Computation*, pages 1404–1411, Portland, Oregon, USA, 2004.

[17] M. Reyes Sierra and C. A. Coello Coello. Improving pso-based multi-objective optimization using crowing, mutation and epsilon-dominance. In *Third International Conference on Evolutionary MultiCriterion Optimization. EMO 2005*, volume 3410 of LNCS. Springer, 2005.

[18] O. Schutze, C. Coello Coello, S. Mostaghim, E. Talbi, and M. Dellnitz. Hybridizing evolutionary strategies with continuation methods for solving multi-objective problems. *Engineering Optimization*, 00(00):1–28, 2007.

[19] K. Tan, E. Khor, and T. Lee. *Multiobjective evolutionary algorithms and applications*. Springer-Verlag, London, second edition, 2005. ISBN 1-85233-836-9.

[20] Z. Xiao-Hua, M. Hong-Yun, and J. Li-Cheng. Intelligent particle swarm optimization in multiobjective optimization. In *2005 Congress on Evolutionary Computation*, pages 714–719, Scotland, UK, 2005.