A Clustering-Based Coevolutionary Algorithm for Multiobjective Optimization

Margarita Reyes Sierra and Carlos A. Coello Coello
CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Electrical Eng. Department, Computer Science Dept.
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco, México D.F. 07300, MÉXICO

June 9, 2004

Abstract

We propose a new version of a multiobjective coevolutionary algorithm. The main idea of the proposed approach is to concentrate the search effort on promising regions that arise during the evolutionary process as a product of a clustering mechanism applied on the set of decision variables corresponding to the known Pareto front. The proposed approach is validated using several test functions taken from the specialized literature and it is compared with respect to two approaches that are representative of the state-of-the-art in evolutionary multiobjective optimization.

1 Introduction

Despite the considerable volume of research on evolutionary multiobjective optimization [5], little emphasis has been placed on certain algorithmic design aspects such as efficiency [6, 10, 4]. Additionally, the use of coevolutionary mechanisms (which have strong links to game theory [1]) has been scarce in the evolutionary multiobjective optimization literature. As in our original proposal, the main motivation of the work reported here it is precisely to take advantage of some coevolutionary concepts to design a multi-objective evolutionary algorithm (MOEA) that can be more efficient (in terms of fitness function evaluations). The main idea of the proposed algorithm is to obtain information along the evolutionary process as to focus the search in the "promising" sub-regions, and then to use a subpopulation for each of these subregions. At each generation, these different subpopulations (which evolve independently using Fonseca and Fleming's ranking scheme [7]) "cooperate" and "compete" among themselves and from these different processes we obtain a single Pareto front. Each individual contained in the Pareto optimal set has a label that indicates the subpopulation to which it belongs. These labels are used to determine which subpopulations contributed with more solutions. The size of each subpopulation is adjusted based on their contribution

to the current Pareto front (i.e., subpopulations which contributed more are allowed a larger population size and viceversa). The proposed approach uses the adaptive grid proposed in [10] to store the nondominated vectors obtained along the evolutionary process, enforcing a more uniform distribution of such vectors along the Pareto front. This new version of our algorithm performs a clustering analysis on the set of decision variables of the current Pareto front to find the promising regions of the search space. In this way, the number of populations needed does not exceeds the total number of members on the true Pareto front.

2 Statement of the Problem

We are interested in solving problems of the type:

$$\text{minimize } [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \tag{1}$$

subject to:

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

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

where k is the number of objective functions $f_i : R^n \rightarrow R$. We call $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ the vector of decision variables. We thus wish to determine from the set $\mathcal{F}$ of all the vectors that satisfy (2) and (3) to the vectors $x_1^*, x_2^*, \ldots, x_n^*$ that are *Pareto optimal*. We say that a vector of decision variables $\vec{x}^* \in \mathcal{F}$ is *Pareto optimum* if there does not exist another $\vec{x} \in \mathcal{F}$ such that $f_i(\vec{x}) \leq f_i(\vec{x}^*)$ for every $i = 1, \ldots, k$ and $f_j(\vec{x}) < f_j(\vec{x}^*)$ for at least one j. The vectors $\vec{x}^*$ corresponding to the solutions included in the Pareto optimal set are called *nondominated*. The objective function values corresponding to the elements of the Pareto optimal set are called the *Pareto front* of the problem.

3 Coevolution

Coevolution refers to a reciprocal evolutionary change between species that interact with each other. The relationships between the populations of two different species can be described considering all their possible types of interactions. Such interaction can be positive or negative depending on the consequences that such interaction produces on the population. Evolutionary computation researchers have developed several coevolutionary approaches in which normally two or more species relate to each other using any of the possible relationships, mainly competitive (e.g., [11]) or cooperative (e.g., [13]) relationships. Also, in most cases, such species evolve independently through a genetic algorithm. The key issue in these coevolutionary algorithms is that the fitness of an individual in a population depends on the individuals of a different population.

4 Previous Work

There are very few references in the literature in which coevolutionary concepts are used to solve multiobjective optimization problems. We will review the main ones in

this section.

Parmee and Watson [12] proposed a collaborative scheme in which they use one population to optimize each of the objective functions of a problem. The method is really created to converge to a single (ideal) trade-off solution. However, through the use of penalties the algorithm can maintain diversity in the population. These penalties relate to variability in the decision variables' values. The authors also store solutions produced during the evolutionary process so that the user can analyze the historical paths traversed by the algorithm.

Barbosa and Barreto [2] proposed a cooperative approach for solving a graph layout generation problem. The approach uses two populations (a separate genetic algorithm is used for each of them and information is exchanged through a shared fitness function): a graph layout population (i.e., individuals that contain the coordinates of all vertices in the graph) and a population of weights (i.e., individuals that contain, each one, a set of weights to be applied on the different aesthetic objectives imposed on the problem). Each of the solutions produced by the system are presented to a user who ranks them based on (subjective) preferences. This ranking is used to determine fitness of the population of weights.

Keerativuttitumrong et.al [9] proposed a cooperative scheme in which one population is defined for each decision variable of the problem. The evolution of each of these populations is controlled through Fonseca and Fleming's MOGA [7]. In order to evaluate an individual in any population, individuals from the other populations must be selected in order to complete a solution (this is because each population only encodes one decision variable).

Coello and Reyes [3] proposed a collaborative and cooperative scheme in which, after a stage that explores the whole search space, the search space is splitted based on a simple analysis of the current Pareto front. After that, a population is assigned to each one of the subparts. The non-dominated individuals of each populations cooperate and compete to form a single Pareto front. The sources (individuals) of each population depend on the corresponding contribution to the current Pareto front.

5 Description of Our Approach

As in [3], the main idea of our approach is to focus the search efforts only towards the promising regions of the search space. Such "promising" regions are determined using clustering analysis of the current Pareto front. The evolutionary process of our approach is divided in two main stages. The first stage takes place during the first quarter of the total of generations. After that, in the second stage (the rest of the generations) we perform what we call a *checkpoint* in specific moments that will be mentioned later.

First Stage. During the first stage, the algorithm is allowed to explore all of the search space, by using a population of individuals which are selected using Fonseca and Fleming's Pareto ranking scheme [7]. Additionally, the approach uses the adaptive grid proposed in [10]. At the end of this first stage, the algorithm analyses the current Pareto front (stored in the adaptive grid) in order to determine the promising regions of the search space.

```
1. gen = 0
2. populations = 1
3. while (gen < Gmax) {
        if (gen ≥ Gmax/4)
4.          if (gen = Gmax/4, Gmax/2, 3Gmax/4 or
            ∃ x ∈ pop_zero : x ∈ current Pareto front)
            {
5.              check_active_populations()
6.              clustering_algorithm()
7.              construct_new_subpopulations()
            }
8.      for (i = 1; i ≤ populations; i++)
9.          if (population i contributes
            to the current Pareto front)
10.             evolve_and_compete(i)
11.     elitism()
12.     reassign_resources()
13.     gen++ }
```

Figure 1: Pseudocode of our algorithm.

In this new version, we perform a clustering analysis on the set of values of the decision variables corresponding to the current Pareto front. The aim is to determine the promising regions of the search space (line 6, Figure 1). This analysis is performed independently for each decision variable. Once that we know the clusters of the values corresponding to each one of the decision variables, we proceed to form a set of new populations. This process is illustrated in Figure 2. A cluster is a set of values, so for each cluster of each variable, we obtain the limits that bound that cluster. Once that we know the limits of each cluster, we have a set of intervals for each variable. Then, a set of sub-regions is created in the following way. For each point in the current Pareto front, we proceed to locate the interval on each variable to which it belongs. This process give us a region in the search space. For each point in the current Pareto front we first check if it belongs to any region already located. If the point belongs to an existing region, we continue with the next point. Otherwise, we proceed to create the corresponding region. And so on. After that, we assign a new population to each region created, i.e., those that have individuals in the current Pareto front (line 7, Figure 1). In this way, in the worst case we will have as many populations as points in the current Pareto front. Finally, we use one extra population (call *population zero*) that continues searching for good solutions on the whole search space. This population is initialized with an 80% of points of the current Pareto front and a 20% of random points (with the aim of generate intermediate points on the current Pareto front while adding diversity).
Second Stage. When reaching the end of the first stage, the algorithm consists of a certain number of populations looking each at different regions of the search space. At each generation, the evolution of all the populations takes place independently and,

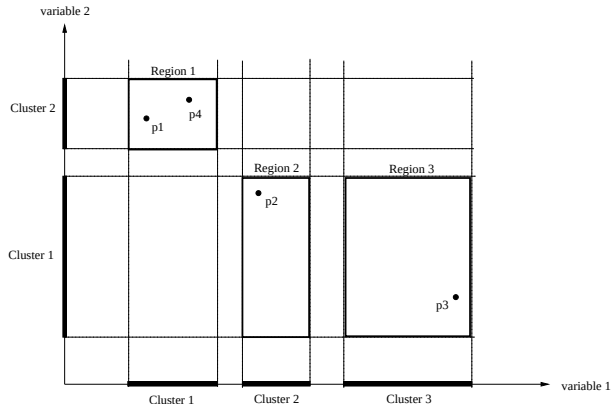

Figure 2: Mechanism used to locate the promising regions of the search space. A population will be assigned to each located promising region.

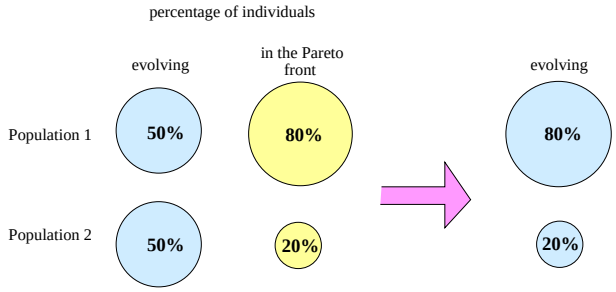

Figure 3: Resources reassignment: Each population is assigned or removed individuals such that its final size is proportional to its contribution to the current Pareto front.

later on, the nondominated elements from each population are sent to the adaptive grid where they "cooperate" and "compete" in order to conform a single Pareto front. After this, we count the number of individuals that each of the populations contributed to the current Pareto front. Our algorithm is *elitist* (line 11, Figure 1), because after the first generation of the second stage, all the populations that do not provide any individual to the current Pareto front are automatically eliminated and the sizes of the other populations are properly adjusted (line 12, Figure 1). Each population is assigned or removed individuals such that its final size is proportional to its contribution to the current Pareto front. These individuals to be added or removed are randomly generated/chosen. This process is illustrated in Figure 3. Thus, populations compete with each other to get as many extra individuals as possible. Note that it is, however, possible that the sizes of the populations "converge" to a constant value once their contribution to the current Pareto front no longer changes.

Checkpoint. During the second stage, we perform a *checkpoint* in specific moments

of the evolution process (line 4, Figure 1). The checkpoint takes place as before (see Figure 1) [3], but also when the population zero includes a new individual in the adaptive grid, that is, in the current Pareto front. When the checkpoint happens, we perform a check on the current populations in order to determine how many (and which) of them can continue (i.e., those populations which continue contributing individuals to the current Pareto front, which are the "good" populations) (line 5, Figure 1). As at the end of the first stage, we perform again the clustering analysis on the set of values of the decision variables corresponding to the current Pareto front, and proceed to form a set of new populations. The non-dominated individuals from the "good" populations are kept. All the good individuals are distributed across the newly generated populations. The elitist process continues takes place and the size of each population will be adjusted based on the same criteria as before. Note however, that we define a minimum population size and this size is enforced for all populations after each checkpoint.

1.choose random centroids
2.place each point in its corresponding cluster
3.d_{new} =sum of distances between each point and its centroid
4.$d_{old} = bignumber$
5.while ($d_{new} < d_{old}$) {
6. centroids=means, place points
7. (if needed) increase-eliminate-correct centroids, place points
8. $d_{old} = d_{new}$
9. update d_{new}}

Figure 4: Pseudocode of our clustering algorithm.

Clustering Analysis. We implemented a clustering algorithm based on the nature of the $k - means$ algorithm [8] (Figure 4). This algorithm begins with k random centroids and puts every point of the analyzed set on the cluster corresponding to the nearest centroid. After that, the *means* of each cluster are calculated and the process repeat until no further changes are done. The algorithm stops when the minimum of the sum of the distances between each point to its corresponding centroid is found. This algorithm has two disadvantages: (1) it depends on the initial centroids and (2) it requires the number of clusters needed. For this reason, we made two modifications to overcome these drawbacks.

Regarding the first disadvantage, we look for a point that could be a new centroid: Let x_i a point that belongs to a cluster with centroid c_i, and d_{min} the minimum distance between two centroids. If $c_i > d_{min}$, x_i will be a new centroid. To maintain the number of clusters constant, once we have selected a point to be a new centroid, we choose one of the closest centroids to be eliminated (line 7, Figure 4). With respect to the second disadvantage, we use the following mechanism [8]: Let x_i a point that belongs to cluster q (with centroid c_q) and K the current total number of clusters. The average distance between x_i and the K centroids is: $\bar{d}_i = (1/K)\sum_{k=1}^{K} d(x_i, c_k)$ We create a new cluster with centroid x_i when: $|d(x_i, c_q) - \bar{d}_i| \leq \bar{d}_i T$ where T is such that $0 < T < 1$. As big is the value of T, as big is the number of clusters created

(line 7, Figure 4). Since the previous mechanism creates new clusters, we use a simple mechanism to also eliminate clusters when the corresponding centroids are very close (line 7, Figure 4): If the distance between two centroids is less that T times the average distance between centroids, one of them is eliminated.

Parameters Required. Our proposed approach requires the following parameters (The parameter T of the clustering algorithm used was fixed to $T = 1$):

1. Crossover rate (p_c) and mutation rate (p_m).
2. Maximum number of generations ($Gmax$).
3. Size of the initial population ($popsize_{init}$) to be used during the first stage and minimum size of the secondary population ($popsize_{sec}$) to be used during the further stages.

6 Results

To validate our approach, we used the methodology normally adopted in the evolutionary multiobjective optimization literature [5]. We performed both quantitative comparisons (adopting four metrics) and qualitative comparisons (plotting the Pareto fronts produced) with respect to two MOEAs that are representative of the state-of-the-art in the are: the Nondominated Sorting Genetic Algorithm II (NSGAII) [6], and the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [18]. For our comparative study, we implemented the four following metrics:

Error Ratio (ER): This metric was proposed by Van Veldhuizen [15] to indicate the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set: $ER = \frac{\sum_{i=1}^{n} e_i}{n}$, 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 algorithm belong to the true Pareto optimal set of the problem.

Generational Distance (GD): The concept of generational distance was introduced by Van Veldhuizen & Lamont [16] as a way of estimating how far are the elements in the Pareto front produced by our algorithm from those in the true Pareto front of the problem. This metric is defined as: $GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$ where n is the number of nondominated vectors found by the algorithm being analyzed and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the true Pareto front. It should be clear that a value of $GD = 0$ indicates that all the elements generated are in the true Pareto front of the problem. Therefore, any other value will indicate how "far" we are from the global Pareto front of our problem.

Spacing (SP): This metric was proposed by Schott [14] as a way of measuring the range (distance) variance of neighboring vectors in the Pareto front known. This metric is defined as: $SP = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\bar{d} - d_i)^2}$ where $d_i = min_j(\sum_{k=1}^{m} |f_m^i - f_m^j|)$, $i, j = 1, ..., n$, m is the number of objectives, $\bar{d}$ is the mean of all d_i, and n is the number of vectors in the Pareto front found by the algorithm being evaluated. A value

of zero for this metric indicates all the nondominated solutions found are equidistantly spaced.

Two Set Coverage (SC): This metric was proposed in [17], and it can be termed *relative coverage comparison of two sets*. Consider X', X'' as two sets of phenotype decision vectors. SC is defined as the mapping of the order pair (X', X'') to the interval $[0, 1]$: $SC(X', X'') \triangleq (|\{a'' \epsilon X''; \exists a' \epsilon X' : a' \succeq a''\}|)/(|X''|)$. If all points in X' dominate or are equal to all points in X'', then by definition $SC = 1$. $SC = 0$ implies the opposite. In general, $SC(X', X'')$ and $SC(X'', X')$ both have to be considered due to set intersections not being empty. Of course, this metric can be used for both spaces (objective function or decision variable space), but in this case we applied it in objective function space.

For each of the test functions shown below, we perform 30 runs per algorithm and a total of 10,000 evaluations. The parameters for NSGAII were $popsize$=100 and 100 generations and for SPEA2 were $\alpha = \mu = \lambda = 100$ and 100 generations. All the algorithms used real representation, a bit mutation probability (p_m) equal to $1/codesize$ (Parameter Based Mutation) and a crossover probability (p_c) equal 0.8 (Simulated Binary Crossover). The Pareto fronts that we will show correspond to the median of the 30 runs with respect to the ER metric.

Regarding constraint-handling, we used the original scheme provided in the case of the NSGAII. However, since the other two algorithms (including our own) don't have such a mechanism, we implemented a simple penalty function over the value of objective functions of each unfeasible individual.

6.1 Test Function 1 (Deb)

$$\begin{aligned} \text{Minimize } f_1(x_1, x_2) &= x_1 \\ \text{Minimize } f_2(x_1, x_2) &= g(x_1, x_2) h(x_1, x_2) \end{aligned}$$

subject to:

$$g(x_1, x_2) = 11 + x_2^2 - 10cos(2\pi x_2)$$

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

$$0.0 \leq x_1 \leq 1.0$$

$$-30.0 \leq x_2 \leq 30.0$$

In this example, our approach used: $popsize_{init} = 100$, $popsize_{rec} = 30$ (38 gen). Table 1 shows the values of the metrics for each of the MOEAs compared.

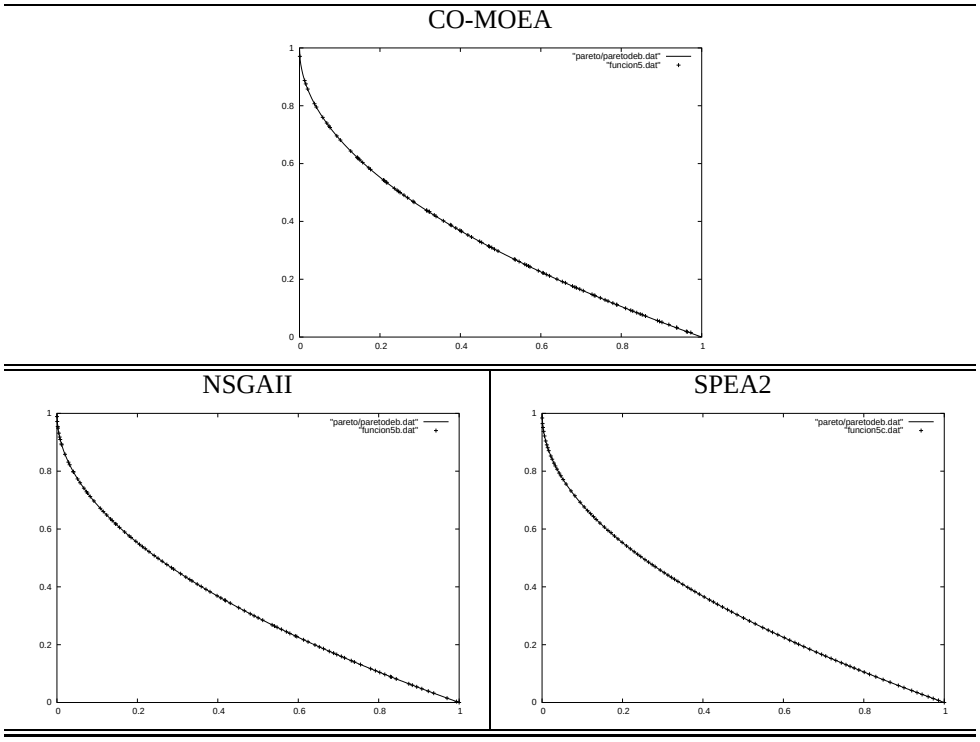

Figure 5: Pareto fronts obtained by our approach (CO-MOEA), the NSGAII [6] and the SPEA2 [18], for test function 1.

6.2 Test Function 2 (Kursawe)

$$\text{Minimize } f_1(x) = \sum_{i=1}^{2}(-10e^{-0.2*\sqrt{x_i^2+x_{i+1}^2}})$$

$$\text{Minimize } f_2(x) = \sum_{i=1}^{3}(|x_i|^{0.8} + 5sin(x_i^3))$$

subject to:

$$-5.0 \leq x_1, x_2, x_3 \leq 5.0$$

In this case, our approach used: $popsize_{init} = 100$, $popsize_{rec} = 30$ (40 gen). Table 2 shows the values of the metrics for each of the MOEAs compared.

Test Function 1				
		CO-MOEA	NSGAII	SPEA2
ER	best	0.02	0.00	0.02
	median	0.10	0.07	0.05
	worst	0.44	0.47	0.39
	average	0.15	0.13	0.08
	std. dev.	0.1112	0.1289	0.0856
GD	best	0.0001	0.0047	0.0048
	median	0.0040	0.0056	0.0056
	worst	0.0910	0.0061	0.6116
	average	0.0159	0.0055	0.0829
	std. dev.	0.0249	0.0004	0.1303
SP	best	0.0045	0.0064	0.0027
	median	0.0090	0.0073	0.0041
	worst	0.9069	0.0084	1.9915
	average	0.1344	0.0073	0.4252
	std. dev.	0.2491	0.0006	0.6227

Two Set Coverage Metric SC				
	X	$SC(X,I)$	$SC(X,II)$	$SC(X,II)$
I	CO-MOEA	0.00	0.00	0.00
II	NSGAII	0.02	0.00	0.01
III	SPEA2	0.00	0.03	0.00
Average		1%	2%	0%

Table 1: Comparison of results between our approach (denoted by CO-MOEA), the NSGAII [6] and the SPEA2 [18] for test function 1.

Test Function 2				
		CO-MOEA	NSGAII	SPEA2
ER	best	0.12	0.16	0.19
	median	0.23	0.27	0.26
	worst	0.35	0.37	0.36
	average	0.24	0.28	0.25
	std. dev.	0.0578	0.0578	0.0412
GD	best	0.0028	0.0032	0.0028
	median	0.0032	0.0036	0.0033
	worst	0.0038	0.0044	0.0035
	average	0.0032	0.0037	0.0032
	std. dev.	0.0002	0.0004	0.0002
SP	best	0.0519	0.0450	0.0991
	median	0.1100	0.0553	0.0867
	worst	0.1534	0.1060	0.0801
	average	0.1069	0.0606	0.0866
	std. dev.	0.0306	0.0156	0.0042

Two Set Coverage Metric SC				
	X	$SC(X,I)$	$SC(X,II)$	$SC(X,II)$
I	CO-MOEA	0.00	0.07	0.17
II	NSGAII	0.07	0.00	0.14
III	SPEA2	0.07	0.09	0.00
Average		7%	8%	16%

Table 2: Comparison of results between our approach (denoted by CO-MOEA), the NSGAII [6] and the SPEA2 [18] for test function 2.

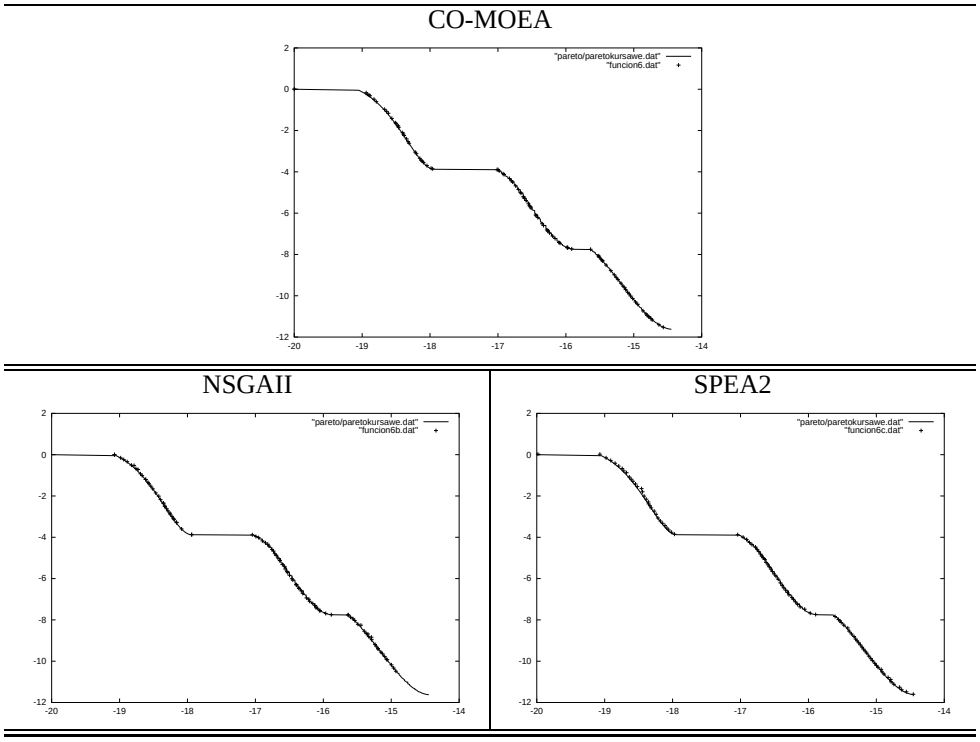

Figure 6: Pareto fronts obtained by our approach (CO-MOEA), the NSGAII [6] and the SPEA2 [18], for test function 2.

6.3 Test Function 3 (Kita)

$$\begin{aligned} \text{Minimize } f_1(x_1, x_2) &= -x_1^2 + x_2 \\ \text{Minimize } f_2(x_1, x_2) &= \frac{1}{2}x_1 + x_2 + 1 \end{aligned}$$

subject to:

$$\begin{aligned} g_1(x_1, x_2) = \frac{1}{6}x_1 + x_2 - \frac{13}{2} \leq 0 \\ g_2(x_1, x_2) = \frac{1}{2}x_1 + x_2 - \frac{15}{2} \leq 0 \\ g_3(x_1, x_2) = 5x_1 + x_2 - 30 \leq 0 \\ 0.0 \leq x_1, x_2 \leq 7.0 \end{aligned}$$

In this example, our approach used: $popsize_{init} = 100$, $popsize_{rec} = 30$ (40 gen). Table 4 shows the values of the metrics for each of the MOEAs compared.

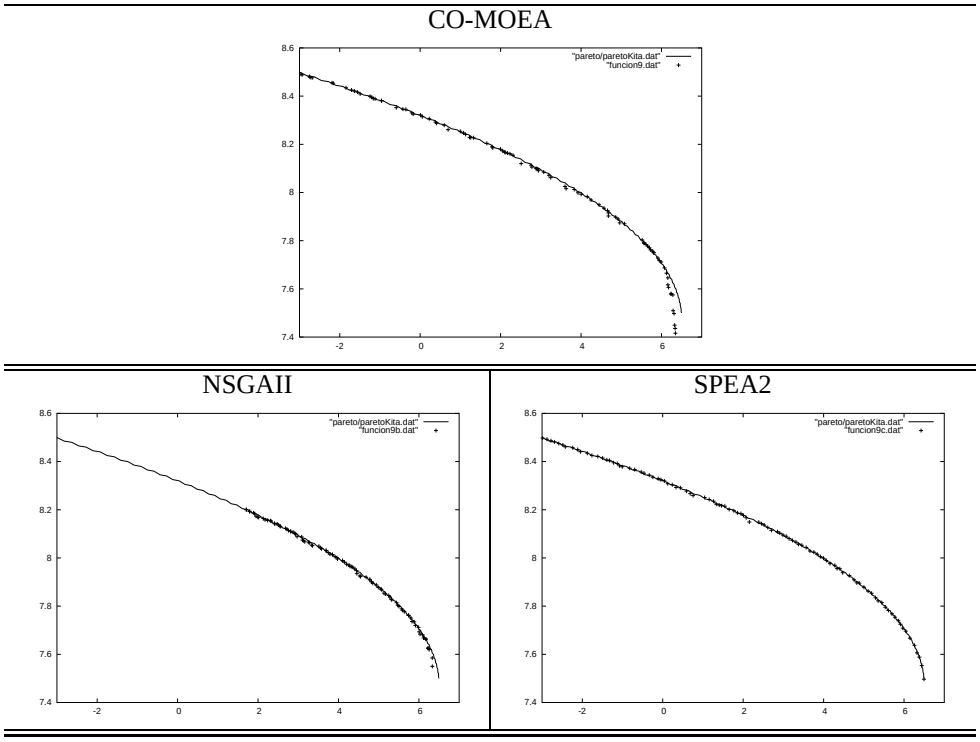

Figure 7: Pareto fronts obtained by our approach (CO-MOEA), the NSGAII [6] and the SPEA2 [18], for test function 3.

6.4 Test Function 4 (Tanaka)

$$\begin{aligned} \text{Minimize } f_1(x_1, x_2) &= x_1 \\ \text{Minimize } f_2(x_1, x_2) &= x_2 \end{aligned}$$

subject to:

$$g_1(x_1, x_2) = -x_1^2 - x_2^2 + 1 + 0.1cos(16arctan\frac{x_1}{x_2}) \leq 0$$

$$g_2(x_1, x_2) = (x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2 - \frac{1}{2} \leq 0$$

$$0.0 \leq x_1, x_2 \leq \pi$$

In this example, our approach used: $popsize_{init} = 100$, $popsize_{rec} = 30$ (40 gen). Table 4 shows the values of the metrics for each of the MOEAs compared.

Test Function 3				
		CO-MOEA	NSGAII	SPEA2
ER	best	0.37	0.30	0.27
	median	0.52	0.44	0.33
	worst	0.67	0.58	0.43
	average	0.51	0.46	0.35
	std. dev.	0.0793	0.0624	0.0455
GD	best	0.0025	0.0019	0.0026
	median	0.0056	0.0025	0.0038
	worst	0.1980	0.5224	0.1569
	average	0.0239	0.0627	0.0222
	std. dev.	0.0411	0.1412	0.0395
SP	best	0.0351	0.0192	0.0190
	median	0.0539	0.0248	0.0284
	worst	0.1868	2.8625	1.3698
	average	0.0636	0.1513	0.1504
	std. dev.	0.0295	0.5286	0.3025

Two Set Coverage Metric SC				
	X	$SC(X,I)$	$SC(X,II)$	$SC(X,II)$
I	CO-MOEA	0.00	0.15	0.14
II	NSGAII	0.28	0.00	0.07
III	SPEA2	0.39	0.29	0.00
Average		34%	22%	11%

Table 3: Comparison of results between our approach (denoted by CO-MOEA), the NSGAII [6] and the SPEA2 [18] for test function 3.

Test Function 4				
		CO-MOEA	NSGAII	SPEA2
ER	best	0.05	0.01	0.02
	median	0.16	0.08	0.06
	worst	0.29	0.17	0.10
	average	0.15	0.08	0.06
	std. dev.	0.0461	0.0339	0.0216
GD	best	0.0009	0.0008	0.0010
	median	0.0012	0.0013	0.0012
	worst	0.0015	0.0016	0.0014
	average	0.0012	0.0012	0.0012
	std. dev.	0.0001	0.0002	0.0001
SP	best	0.0047	0.0065	0.0037
	median	0.0085	0.0099	0.0052
	worst	0.0185	0.0155	0.0079
	average	0.0092	0.0101	0.0053
	std. dev.	0.0027	0.0022	0.0009

Two Set Coverage Metric SC				
	X	$SC(X,I)$	$SC(X,II)$	$SC(X,II)$
I	CO-MOEA	0.00	0.14	0.18
II	NSGAII	0.17	0.00	0.17
III	SPEA2	0.34	0.20	0.00
Average		26%	17%	18%

Table 4: Comparison of results between our approach (denoted by CO-MOEA), the NSGAII [6] and the SPEA2 [18] for test function 4.

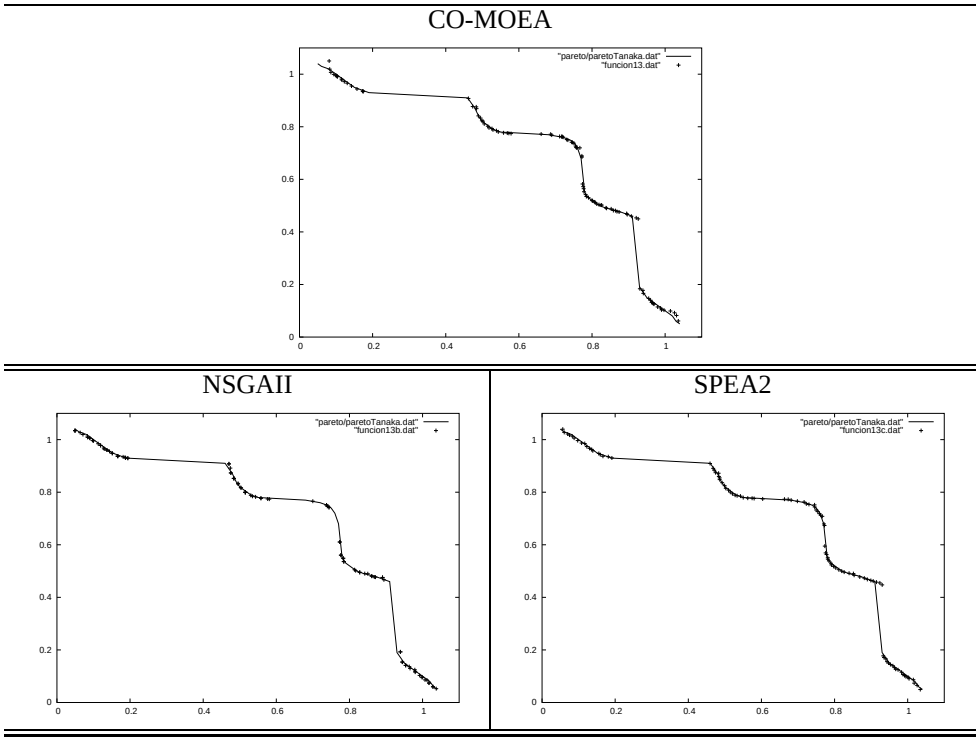

Figure 8: Pareto fronts obtained by our approach (CO-MOEA), the NSGAII [6] and the SPEA2 [18], for test function 4.

7 Discussion of Results

In the first function, the three algorithms have similar results with respect to the ER metric, however the SPEA2 algorithm has the best results on average. In the case of the GD and SP metrics, although our algorithm and the SPEA2 algorithm respectively have better best and median results than the other algorithms, the best results on average are from the NSGAII algorithm in both cases given its low standard deviation. Finally, regarding the SC metric all the algorithms have very similar results (between 0 and 2%).

In the case of the second function, with respect to the ER and GD metrics, the three algorithms have very similar results, being our algorithm marginally the best. Regarding the SP metric, the NSGAII algorithm has the best results. Finally, in this case with respect to the SC metric, our algorithm has the best results with an average of 7% of its points dominated by other algorithms. In this case, the percentages of NSGAII and SPEA2 are: 8% and 16%, respectively.

In the third function, there are clear differences between the three algorithms in the values of the ER metric: the best is SPEA2 followed by NSGAII and our algorithm.

With respect to the GD metric, although our algorithm has the best value, on average the SPEA2 has the best results, followed by our algorithm and the NSGAII. In the case of the SP metric, although the better best and median results are from SPEA2 and NSGAII, the best results on average are from our algorithm given its low standard deviation. Regarding the SC metric, SPEA2 has the best results with an average of 11% of its points dominated by other algorithms. In this case, the percentages of NSGAII and our algorithm are: 22% and 34%, respectively.

In the case of the fourth function, the best results on average with respect to the ER metric are from SPEA2, followed by NSGAII and our algorithm. With respect to the GD metric, the results of the three algorithms are almost equal. In the case of the SP metric, the best results on average are from SPEA2, followed by our algorithm and NSGAII. Regarding the SC metric, NSGAII has the best results with an average of 17% of its points dominated by other algorithms. In this case, the percentages of SPEA2 and our algorithm are: 18% and 26%, respectively.

In general, in the first two functions our algorithm obtained very good results. The values of the metrics on these two functions indicate that our algorithm was able to approximate the true Pareto front in each case as good as the other algorithms. This is reflected in the values of the SC metric too. In the case of the third and fourth functions, our algorithm obtained results somehow poor against the other algorithms with respect to the ER metric. This means that our algorithm couldn't find as many points of the true Pareto front as the other algorithms, and this is the reason for the relatively high values of our algorithm in the SC metric. However, the values obtained in these cases on the GD metric indicate that our algorithm was as close of the true Pareto front as the other algorithms. Regarding the distribution (SP metric), except in the fourth function, the values of our algorithm are not as good as the other algorithms. This is an improvement that we have to do as part of our future work.

8 Conclusions and Future Work

We have presented a new version of a coevolutionary multi-objective evolutionary algorithm whose main idea is to detect the most "promising" sub-regions of the search space and focus the search on them. With this aim, the proposed algorithm applies a clustering algorithm on the set of decision variables of the known Pareto front. The proposed approach was validated using several test functions taken from the specialized literature. Our comparative study showed that the proposed approach is competitive with respect two other algorithms that are representative of the state-of-the-art in the area. Since we are proposing a coevolutive scheme that uses MOGA as search engine, we consider it is an interesting idea in to use some more efficient multiobjective algorithm in order to improve the obtained results.

Acknowledgments

The first author acknowledges support from the Mexican Consejo Nacional de Ciencia y Tecnolog´ıa (CONACyT) through a scholarship to pursue graduate studies at CIN-

VESTAV-IPN's. The second author acknowledges support from CONACyT through project number 42435-Y.

References

[1] Helio J. C. Barbosa. A coevolutionary genetic algorithm for a game approach to structural optimization. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 545–552, San Mateo, California, July 1997. Michigan State University, Morgan Kaufmann Publishers.

[2] Helio J.C. Barbosa and Andr´e M.S. Barreto. An interactive genetic algorithm with coevolution of weights for multiobjective problems. 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 203–210, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[3] Carlos A. Coello Coello and Margarita Reyes Sierra. A Coevolutionary Multi-Objective Evolutionary Algorithm. In *Proceedings of 2003 Congress on Evolutionary Computation*, volume 1, pages 482–489, Canberra, Australia, December 2003. IEEE Press.

[4] 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.

[5] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.

[6] 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.

[7] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[8] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

[9] Nattavut Keerativuttitumrong, Nachol Chaiyaratana, and Vara Varavithya. Multi-objective Co-operative Co-evolutionary Genetic Algorithm. In J.J. Merelo Guervs et al., editor, *Proceedings of PPSN VII*, pages 288–297. Springer-Verlag, September 2002.

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

[11] J. Paredis. Coevolutionary algorithms. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *The Handbook of Evolutionary Computation, 1st supplement*, pages 225–238. Institute of Physics Publishing and Oxford University Press, 1998.

[12] Ian C. Parmee and Andrew H. Watson. Preliminary Airframe Design Using Co-Evolutionary Multiobjective Genetic Algorithms. In W. Banzhaf et al., editor, *Proceedings of GECCO'99*, volume 2, pages 1657–1665, San Francisco, California, July 1999. Morgan Kaufmann.

[13] M. Potter and K. De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings from the Fifth Parallel Problem Solving from Nature*, pages 530–539, Jerusalem, Israel, 1994. Springer-Verlag.

[14] 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.

[15] 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.

[16] David A. Van Veldhuizen and Gary B. Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation*, volume 1, pages 204–211, Piscataway, New Jersey, July 2000. IEEE Service Center.

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

[18] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K.C. Giannakoglou et al., editor, *Proceedings of the EUROGEN2001 Conference*, pages 95–100, Barcelona, Spain, 2002. CIMNE.