Using Clustering Techniques to Improve the Performance of a Multi-objective Particle Swarm Optimizer

Gregorio Toscano Pulido and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)
Depto. de Ing. Elect./Sección de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D.F. 07300, MEXICO
gtoscano@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

Abstract. In this paper, we present an extension of the heuristic called "particle swarm optimization" (PSO) that is able to deal with multiobjective optimization problems. Our approach uses the concept of Pareto dominance to determine the flight direction of a particle and is based on the idea of having a set of sub-swarms instead of single particles. In each sub-swarm, a PSO algorithm is executed and, at some point, the different sub-swarms exchange information. Our proposed approach is validated using several test functions taken from the evolutionary multiobjective optimization literature. Our results indicate that the approach is highly competitive with respect to algorithms representative of the state-of-the-art in evolutionary multiobjective optimization.

1 Introduction

Particle swarm optimization (PSO) is a relatively recent heuristic inspired by the choreography of a bird flock which has been found to be quite successful in a wide variety of optimization tasks [1]. Its high speed of convergence and its relative simplicity make PSO a highly viable candidate to be used for solving not only problems with a single objective function, but also problems with several objectives (called "multiobjective optimization problems") [2]. In this paper, we present a proposal, called "another multi-objective particle swarm optimization" (AMOPSO), which extends PSO to deal with several objectives. The main novelty of the approach consists on using a clustering technique in order to divide the population of particles into several swarms in order to have a better distribution of solutions in decision variable space. The introduction of this mechanism significantly improves the quality of the Pareto fronts obtained, when comparing our results with respect to other multiobjective PSO previously reported in the literature and with respect to algorithms representative of the state-of-the-art in evolutionary multiobjective optimization.

2 Related Work

There have been several recent proposals to extend PSO to handle multiple objectives. We will review next the most important of them:

- **The Swarm Metaphor of Ray & Liew [3]**: This algorithm uses Pareto dominance and combines concepts of evolutionary techniques with the particle swarm. It uses crowding to maintain diversity and a multilevel sieve to handle constraints.
- **The algorithm of Parsopoulos & Vrahatis [4]**: This algorithm adopts different types of aggregating functions to solve multiobjective optimization problems.
- **Dynamic Neighborhood PSO proposed by Hu and Eberhart [5]**: In this algorithm, only one objective is optimized at a time using a scheme similar to lexicographic ordering. A revised version of this approach that uses a secondary population is presented in [8].
- **The Multi-objective Particle Swarm Optimizer (MOPSO) by Coello & Lechuga [6]**: This 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 repository is used by the particles to identify a leader that will guide the search.
- **The approach of Fieldsend & Singh [7]**: This approach incorporates an unconstrained elite archive (in which a special data structure called "dominated tree" is adopted) to store the nondominated individuals found along the search process. The archive interacts with the primary population in order to define local guides. This approach also uses a "turbulence" operator which is basically a mutation operator that acts on the velocity value used by PSO.
- **The algorithm of Mostaghim & Teich [9]:** This approach uses a sigma method in which the best local guides for each particle are adopted to improve the convergence and diversity of a PSO algorithm used for multiobjective optimization. They also use a "turbulence" operator, but applied on decision variable space. The use of the sigma values increases the selection pressure of PSO (which was already high). This may cause premature convergence in some cases.
- **The Nondominated Sorting PSO of Li [10]**: This approach incorporates the main mechanisms of the NSGA-II [11] into a PSO algorithm. The proposed approach showed a very competitive performance with respect to the NSGA-II (even outperforming it in some cases).

Our approach is based on the use of Pareto ranking and a subdivision of decision variable space into several sub-swarms (this is done using clustering techniques). Since independent PSOs are run into each swarm, our approach can be seen as a meta-MOPSO algorithm. After a certain (pre-defined) number of iterations, the leaders of each swarm are migrated to a different swarm in order to variate the selection pressure. This sort of scheme is a novel proposal to solve multiobjective optimization problems using PSO. Also, note that AMOPSO does not use an external population, since elitism in this case is an emergent process derived from the migration of leaders.

3 Description of the Proposed Approach

The analogy of particle swarm optimization with evolutionary algorithms makes evident the notion that using a Pareto ranking scheme could be the straightforward way to extend the approach to handle multiobjective optimization problems. However, merging a Pareto ranking scheme with the PSO algorithm will produce not one but a set of nondominated leaders and the selection of an "appropriate" leader becomes difficult (by definition, all nondominated solutions are equally good). Additionally, it is known that several difficult multiobjective optimization problems have a disconnected decision variable space. This issue is particularly important when using PSO, because it could be the case that a particle tries to follow a leader that resides in a disconnected region away from it. In this case, a lot of search effort would be wasted and the algorithm might not be able to converge to the true Pareto front of the problem. The use of neighborhoods may be useful in this case. However, we argue that the use of a neighborhood may delay convergence, because the selection pressure is significantly lowered when they are used, since in this case, particles spend most of their search effort following leaders that reside far away from the true Pareto front. Our proposal is to use several swarms (each with a fixed size). Each swarm will over-fly a specific region of the Pareto optimal set (i.e., decision variable space), and will have its own niche of particles and particle guides. The algorithm used to associate leaders into a swarm is the hierarchical single-connected clustering algorithm [13]. The appropriate selection of leaders is essential for the good performance of PSO when applied to multiobjective optimization problems. If the particle chooses an inappropriate leader (i.e., a leader who is too far away in the search space) then most of the flight will be fruitless because the particle will not be traversing promisory regions of search space. In this paper, we propose to use not one but several swarms to avoid this type of problem. However, even if we adopt a multi-swarm scheme, a good strategy to select a leader is still necessary. Some possible strategies for this sake are the following: (1) Randomly (a leader is randomly selected—no constraints are imposed on what sort of leader can a particle choose—, (2) The closest (a particle picks as a leader to the geographically closest leader), and (3) one at a time (a single leader is selected by all the particles at a time). In this paper, we adopted the first scheme (random selection of a leader). The way in which our algorithm works is shown next:

```
**function** AMOPSO Algorithm
**Begin**
  **For** each swarm
      1. Initialize its particles
      2. Initialize $g_{leader}$ set (i.e., the set of global leaders)
  **EndFor**
  **Do**
      **For** each swarm
          **Do**
              **For** each particle
                  4. Select a leader
                  5. Perform the flight
                  6. Update values
                  **If** it is a leader **then** add to the $g_{leader}$ set
              **EndFor**
          **While** maximum number of iterations is not reached
          7. Store leaders in $g_{leader}$ set in $n_{swarms}$ groups
```

```
        **EndFor**
        8. Assign each leader group to a swarm
    **While** maximum number of iterations is not reached
**End.**
```

The proposed algorithm requires the following parameters:

- $GMax$: it refers to the total number of generations that the algorithm will be executed.
- $n_{particles}$: it refers to the total number of particles that will be over-flying the search space.
- n_{swarms}: it refers to the number of particle groups. The swarm size is fixed because the total number of particles is a fixed value.
- $sgmax$: is the number of internal generations that the particles of each swarm will run before sharing their leaders.

The complete execution process of our algorithm can be divided in three stages: initialization, flight and generation of results. At the first stage, every swarm is initialized. Each swarm creates and initializes its own particles and generates the leaders set among the particle swarm set by using Pareto ranking. In the second stage is where the algorithm performs its strongest effort. First, it performs the execution of the flight of every swarm; next, it applies a clustering algorithm to group the guide particles. This is performed until reaching a total of $GMax$ iterations. The execution of the flight of each swarm can be seen as an entire PSO process (with the difference that it will only optimize an specific region of the search space). First, each particle will select a leader to which it will follow. At the same time, each particle will try to outperform its leader and to update its position. If the updated particle is not dominated by any member of the leaders set, then it will become a new leader. The execution of the swarm will start again until a total of $sgmax$ iterations is reached. Constraints are handled in AMOPSO when checking Pareto dominance. When we compare two individuals, we first check their feasibility. If one is feasible and the other is infeasible, the feasible individual wins. If both are infeasible, then the individual with the lowest amount of (total) constraint violation wins. If they both have the same amount of constraint violation (or if they are both feasible), then the comparison is done using Pareto dominance. Once all the swarms have finished theirs flights, a clustering algorithm takes the control by grouping the closest particle guides into n_{swarms} swarms. These particle guides will try to outperform each swarm in the next iteration. This is mainly done by grouping the leaders of all the swarms into a single set, and then splitting this set among n_{swarms} groups (clustering is done with respect to closeness in decision variable space). Each resulting group will be assigned to a different swarm. The third and final stage will present the results, i.e. it will report all the nondominated solutions found.

3.1 Clustering Algorithm

We use Johnson's algorithm to cluster the leaders in groups [13]. The pseudocode of this algorithm is shown next:

```
function Single-link clustering
Begin
    1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence
    number $m = 0$.
    Do
        2. Find the least dissimilar pair of clusters in the current clustering,
        say pair $(r), (s)$, according to $d[(r), (s)] = mind[(i), (j)]$ where the
        minimum is over all pairs of clusters in the current clustering
        3. Increment the sequence number: $m = m + 1$. Merge clusters $(r)$ and
        $(s)$ into a single cluster to form the next clustering $m$. Set the level of
        this clustering to $L(m) = d[(r), (s)]$
        4. Update the proximity matrix, $D$, by deleting the rows and columns
        corresponding to clusters $(r)$ and $(s)$ and adding a row and a column
        corresponding to the newly formed cluster. The proximity between
        the new cluster, denoted $(r, s)$ and the old cluster $(k)$ is defined as:
        $d[(k), (r, s)] = mind[(k), (r)], d[(k), (s)]$
    while objects are not in $N$ clusters.
End.
```

The algorithm requires a proximity matrix as a parameter (in our case, we use a dissimilarity matrix, and the Euclidean distance in variable space to represent the dissimilarity).

4 Comparison of Results

Several test functions were taken from the specialized literature to compare our approach, but due to space limitations, only 3 are included in this paper. In order to allow a quantitative assessment of the performance of a multiobjective optimization algorithm, we adopted the following metrics:

1. **Error Ratio** (ER): This metric was proposed by Van Veldhuizen [14] to indicate the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set:

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

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 Pareto optimal set of the problem.

2. **Generational Distance** (GD): This metric was proposed by Van Veldhuizen [14] as a way of estimating how far are the elements in the set of nondominated vectors found so far from those in the Pareto optimal set and is defined as:

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

where n is the number of vectors in the set of nondominated solutions found so far and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the Pareto optimal set. It should be clear that a value of $GD = 0$ indicates that all the elements generated are in the Pareto optimal set.

3. **Spacing** (SP): Here, one desires to measure the spread (distribution) of vectors throughout the nondominated vectors found so far. Since the "beginning" and "end" of the current Pareto front found are known, a suitably defined metric judges how well the solutions in such front are distributed. Schott [15] proposed such a metric measuring the range (distance) variance of neighboring vectors in the nondominated vectors found so far. This metric is defined as:

$$S \triangleq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d} - d_i)^2}\ , \tag{3}$$

where $d_i = \min_j(\mid f_1^i(\boldsymbol{x}) - f_1^j(\boldsymbol{x}) \mid + \mid f_2^i(\boldsymbol{x}) - f_2^j(\boldsymbol{x}) \mid), i, j = 1, \ldots, n$, $\overline{d}$ is the mean of all d_i, and n is the number of nondominated vectors found so far. A value of zero for this metric indicates all members of the Pareto front currently available are equidistantly spaced.

In order to know how competitive was our approach, we decided to compare it against two multiobjective evolutionary algorithms that represent the state-of-the-art and with respect to a multi-objective particle optimizer (MOPSO) that is publicly available:

1. **Nondominated Sorting Genetic Algorithm II**: Proposed by Deb et al. [11], this algorithm is based on several layers of classifications of the individuals. It incorporates elitism (through the use of $(\mu + \lambda)$-selection), a crowded comparison operator and it keeps diversity without specifying any additional parameters. It remains as one of the most competitive multi-objective evolutionary algorithms known to date.
2. **Pareto Archived Evolution Strategy**: This algorithm was introduced by Knowles and Corne [16]. PAES consists of a (1+1) evolution strategy (i.e., a single parent that generates a single offspring) in combination with a historical archive that records some of the nondominated solutions previously found. This archive is used as a reference set against which each mutated individual is being compared. The archive is not only the elitist mechanism of PAES, but also incorporates an approach to maintain diversity (a crowding procedure that divides objective space in a recursive manner).
3. **Multiobjective Particle Swarm Optimization (MOPSO)**: This approach was introduced in [6]. It 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. It also uses a mutation operator that acts both on the particles of the swarm, and on the range of each design variable of the problem to be solved.

The source code of the NSGA-II, PAES and MOPSO is available from the EMOO repository located at: `http://delta.cs.cinvestav.mx/~ccoello/EMOO`. The source code of AMOPSO is available upon request by email to the first author. In the following examples, the NSGA-II was run using a population size of 100, a crossover rate of 0.8 (uniform crossover was adopted), tournament selection, and a mutation rate of $1/L$, where L = chromosome length (binary representation was adopted). PAES was run using an adaptive grid with a depth of five, a size of the archive of 100, and a mutation rate of $1/L$, where L refers to the length of the chromosomic string that encodes the decision variables. MOPSO (with real-numbers representation) used a (main) population of 100 particles, a repository size of 100 particles, a mutation rate of 0.05, and 30 divisions for the adaptive grid. AMOPSO used 40 particles, a maximum number of generations of 20, a maximum number of generations per swarm of 5, and a total of 5 swarms. The values of all these parameters were empirically derived. The total number of fitness function evaluations was set to 4000 for all the algorithms compared in all the test functions shown next.

4.1 Test Function 1

For our first example, we used the following problem proposed in [17]: Maximize $F = (f_1(x, y), f_2(x, y))$, where

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

subject to: $0 \geq \frac{1}{6}x + y - \frac{13}{2}, \;\; 0 \geq \frac{1}{2}x + y - \frac{15}{2}, \;\; 0 \geq 5x + y - 30$ and: $x, y \geq 0$.

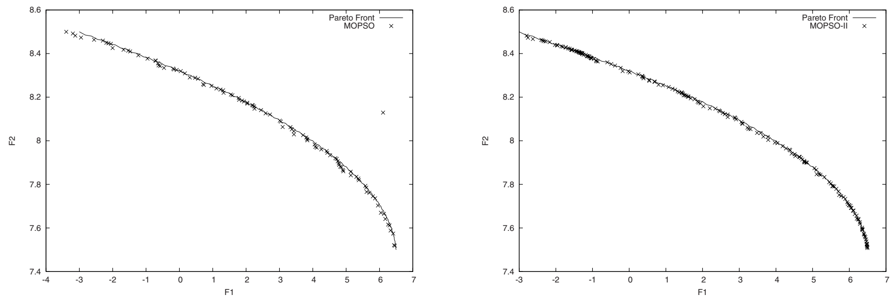
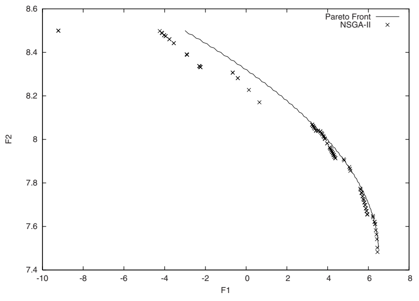


Fig. 1. Pareto fronts produced by MOPSO (left) and the AMOPSO (right) for the first test function.

Figures 1 and 2 show the graphical results produced by the PAES, the NSGA-II, MOPSO and our AMOPSO in the first test function chosen. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared. The true Pareto front of the problem is shown as a continuous line. Tables 1 and 2 show the comparison of results among the four algorithms considering the metrics previously described. It can be seen that the average performance of AMOPSO is the best with respect to the error ratio (by far), and with respect to generational distance. With respect



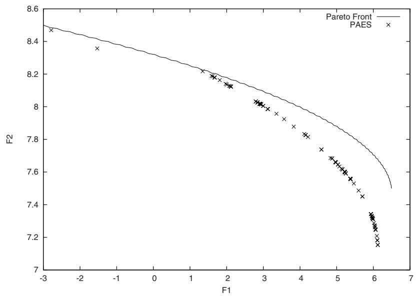


Fig. 2. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the first test function.

Table 1. Results of the Error Ratio (ER) and the Generational Distance (GD) metrics for the first test function.

	Error Ratio				**Generational Distance**			
	MOPSO	**AMOPSO**	**NSGA-II**	**PAES**	**MOPSO**	**AMOPSO**	**NSGA-II**	**PAES**
Average	0.6542	**0.4926**	0.8955	1.0023	0.0150	**0.0038**	0.0518	0.1041
Best	0.5300	0.3763	0.7200	0.9600	0.0024	0.0017	0.0032	0.0114
Worst	0.8261	0.5814	1.0000	1.0156	0.0859	0.02109	0.3170	0.6380
Median	0.6500	0.4932	0.9100	1.0100	0.0078	0.0022	0.0092	0.0264
St. Dev.	0.06263	0.05164	0.07564	0.01630	0.01960	0.00447	0.08641	0.16033

Table 2. Results of the Spacing (SP) metric for the first test function.

SP	**MOPSO**	**AMOPSO**	**NSGA-II**	**PAES**
Average	0.109146	0.043361	**0.028961**	0.079803
Best	0.046508	0.028168	0.008999	0.021405
Worst	0.681124	0.125776	0.080856	0.230506
Median	0.059248	0.035124	0.025807	0.051504
St. Dev.	0.141827	0.023900	0.017047	0.060692

to spacing it places slightly below the NSGA-II, but with a lower standard deviation. By looking at the Pareto fronts of this test function, it can be easily seen that, except for MOPSO and our AMOPSO, none of the algorithms was able to cover the full Pareto front. It can also be seen that AMOPSO produced the best front. This is then an example in which a metric may be misleading, since the fact that the spacing metric provides a good value becomes meaningless if the nondominated vectors produced by the algorithm are not part of the true Pareto front of the problem.

5 Test Function 2

Our second test function was proposed in [18]:

$$\text{Min} f_1(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right); \text{Min} f_2(\boldsymbol{x}) = \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5\sin(x_i)^3\right) \quad (4)$$

where: $-5 \leq x_1, x_2, x_3 \leq 5$

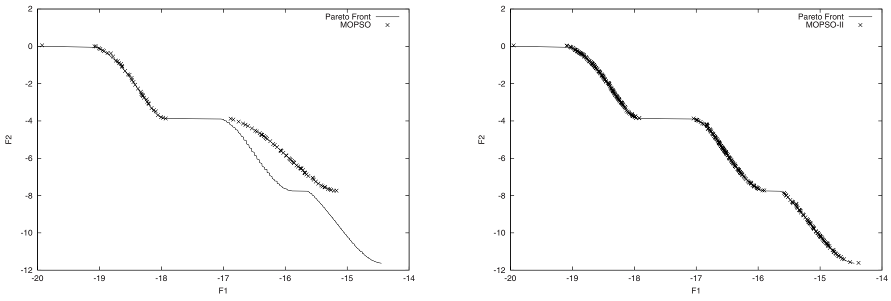


Fig. 3. Pareto fronts produced by MOPSO (left) and the AMOPSO (right) for the second test function.

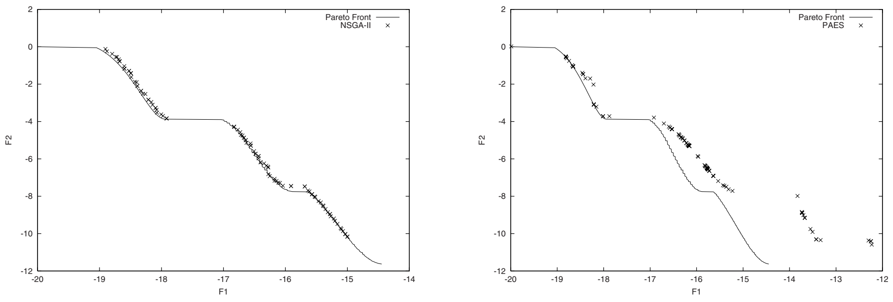


Fig. 4. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the second test function.

Figures 3 and 4 show the graphical results produced by the PAES, the NSGA-II, MOPSO and our AMOPSO in the second test function chosen. Tables 3 and 4 show the comparison of results among the four algorithms considering the metrics previously described. It can be seen that the average performance of AMOPSO is the best with respect to all the metrics adopted. This result can be corroborated by looking at Figures 3 and 4. Except for our AMOPSO, all the algorithms missed important segments of the true Pareto front.

Table 3. Results of the Error Ratio (ER) and the Generational Distance (GD) metrics for the second test function.

	Error Ratio				**Generational Distance**			
	MOPSO	**AMOPSO**	**NSGA-II**	**PAES**	**MOPSO**	**AMOPSO**	**NSGA-II**	**PAES**
Average	0.7810	**0.3490**	0.7365	0.9320	0.0335	**0.0020**	0.0242	0.1730
Best	0.7300	0.2423	0.2300	0.3700	0.0318	0.0015	0.0030	0.0240
Worst	0.8500	0.4565	1.010	1.2500	0.0346	0.0035	0.1058	1.5743
Median	0.7800	0.3614	0.7950	1.0000	0.0336	0.0019	0.0072	0.0909
St. Dev.	0.03194	0.05107	0.24594	0.17914	0.00069	0.00050	0.02935	0.33300

Table 4. Results of the Spacing (SP) metric for the second test function.

SP	**MOPSO**	**AMOPSO**	**NSGA-II**	**PAES**
Average	0.086108	**0.037190**	0.038156	0.449358
Best	0.044509	0.019109	0.002839	0.094711
Worst	0.119586	0.055227	0.102906	5.124390
Median	0.092252	0.041811	0.035956	0.198565
St. Dev.	0.022214	0.014061	0.019729	1.102054

6 Test Function 3

Our third test function is to optimize a four-bar plane truss. The problem is the following [19]: Min $f_1(\mathbf{x}) = L(2x_1 + \sqrt{2x_2} + \sqrt{x_3} + x_4)$, $f_2(\mathbf{x}) = \frac{FL}{E}\left(\frac{2}{x_2} + \frac{2\sqrt{2}}{x_2} - \frac{2\sqrt{(2)}}{x_3} + \frac{2}{x_4}\right)$ such that $(F/\sigma) \leq x_1 \leq 3 \times (F/\sigma)$, $\sqrt{2}(F/\sigma) \leq x_2 \leq 3 \times (F/\sigma)$, $\sqrt{2}(F/\sigma) \leq x_3 \leq 3 \times (F/\sigma)$, $(F/\sigma) \leq x4 \leq 3 \times (F/\sigma)$ where: $F = 10kN$, $E = (2)10^5 kN/cm^2$, $L = 200cm$, $\sigma = 10kN/cm^3$

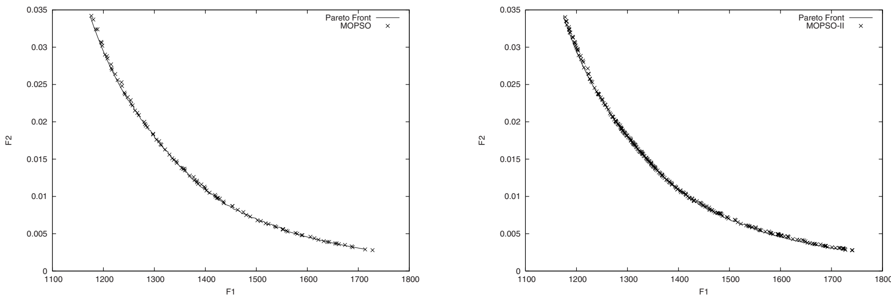


Fig. 5. Pareto fronts produced by MOPSO (left) and the AMOPSO (right) for the third test function.

Figures 5 and 6 show the graphical results produced by PAES, the NSGA-II, MOPSO and our AMOPSO in the third test function chosen. Tables 5 and 6 show the comparison of results among the four algorithms considering the metrics previously described. In this case, AMOPSO was the best with respect to the generational distance and spacing metrics, and it placed second (marginally) with respect to the error ratio metric (PAES was the best average performer with respect to this metric). Graphically, we can see that MOPSO and our AMOPSO were the only algorithms able to cover the entire Pareto front of this problem. Clearly, our AMOPSO produced the best front in this case (see Figures 5 and 6).

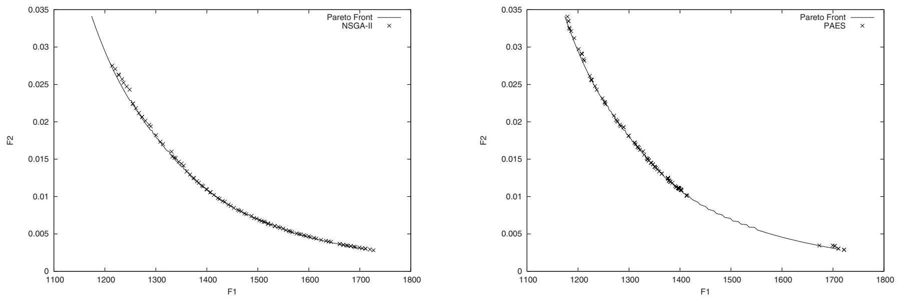


Fig. 6. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the third test function.

Table 5. Results of the Error Ratio (ER) and Generational Distance (GD) metrics for the third test function.

	Error Ratio				**Generational Distance**			
	MOPSO	**AMOPSO**	**NSGA-II**	**PAES**	**MOPSO**	**AMOPSO**	**NSGA-II**	**PAES**
Average	0.447869	0.436210	0.447500	**0.386888**	0.452556	**0.215814**	0.364229	1.079089
Best	0.260000	0.293548	0.210000	0.190000	0.312648	0.152786	0.249701	0.168401
Worst	0.640000	0.558824	0.960000	0.640000	0.933802	0.348571	0.585769	14.222200
Median	0.425000	0.434563	0.355000	0.360000	0.380676	0.212191	0.354397	0.238549
St. Dev.	0.096677	0.061168	0.189150	0.115806	0.178124	0.048167	0.065211	3.126444

7 Conclusions and Future Work

We have presented a new proposal to extend particle swarm optimization to handle multiobjective problems using sub-swarms, Pareto ranking and clustering techniques. The proposed approach was validated using the standard methodology currently adopted in the evolutionary multiobjective optimization community. The results indicate that our approach is a viable alternative since it outperformed some of the best multiobjective evolutionary algorithms known to date. One aspect that we would like to explore in the future is the study of alternative mechanisms to handle constraints through the use of infeasible solutions that can act as leaders in a special swarm. We believe that this sort of mechanism could improve the performance of our AMOPSO, particularly when dealing with problems in which the Pareto front lies on the boundaries between the feasible and infeasible regions. We also want to perform an analysis of the impact of the mechanism adopted to select leaders in the performance of the approach. Finally, we aim to devise a way to eliminate the n_{swarm} parameter through the use of self-adaptation.

Table 6. Results of the Spacing (SP) metric for the third test function.

SP	MOPSO	AMOPSO	NSGA-II	PAES
Average	2.665576	**1.209140**	2.172087	4.180568
Best	1.932520	0.934140	1.115380	1.250790
Worst	3.925990	1.527800	2.557980	22.745200
Median	2.643000	1.257940	2.248595	2.314210
St. Dev.	0.495079	0.177295	0.375880	5.440785

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

References

1. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco, California (2001)
2. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Boston (2002)
3. Ray, T., Liew, K.: A Swarm Metaphor for Multiobjective Design Optimization. Engineering Optimization **34** (2002) 141–153
4. Parsopoulos, K., Vrahatis, M.: Particle Swarm Optimization Method in Multiobjective Problems. In: Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002), Madrid, Spain, ACM Press (2002) 603–607
5. Hu, X., Eberhart, R.: Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization. In: Congress on Evolutionary Computation (CEC'2002). Volume 2., Piscataway, New Jersey, IEEE Service Center (2002) 1677–1681
6. Coello Coello, C.A., Salazar Lechuga, M.: MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In: Congress on Evolutionary Computation (CEC'2002). Volume 1., Piscataway, New Jersey, IEEE Service Center (2002) 1051–1056

7. Fieldsend, J.E., Singh, S.: A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. In: Proceedings of the 2002 U.K. Workshop on Computational Intelligence, Birmingham, UK (2002) 37–44
8. Hui, X., Eberhart, R.C., Shi, Y.: Particle Swarm with Extended Memory for Multiobjective Optimization. In: 2003 IEEE Swarm Intelligence Symposium Proceedings, Indianapolis, Indiana, USA, IEEE Service Center (2003) 193–197
9. Mostaghim, S., Teich, J.: Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In: 2003 IEEE Swarm Intelligence Symposium Proceedings, Indianapolis, Indiana, USA, IEEE Service Center (2003) 26–33
10. Li, X.: A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. In et al., E.C.P., ed.: Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I, Springer. Lecture Notes in Computer Science Vol. 2723 (2003) 37–48
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
12. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading, Massachusetts (1989)
13. Johnson, S.: Hierarchical Clustering Schemes. Psychometrika **32** (1967) 241–254
14. Veldhuizen, D.A.V.: 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 (1999)
15. Schott, J.R.: Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts (1995)
16. Knowles, J.D., Corne, D.W.: Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation **8** (2000) 149–172
17. Kita, H., Yabumoto, Y., Mori, N., Nishikawa, Y.: Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In et al., H.M.V., ed.: Parallel Problem Solving from Nature—PPSN IV. Springer-Verlag, Berlin (1996) 504–512
18. Kursawe, F.: A Variant of Evolution Strategies for Vector Optimization. In Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature. 1st Workshop, PPSN I. Volume 496 of Lecture Notes in Computer Science., Berlin, Germany, Springer-Verlag (1991) 193–197
19. Cheng, F., Li, X.: Generalized Center Method for Multiobjective Engineering Optimization. Engineering Optimization **31** (1999) 641–661