Evolutionary Multiobjective Optimization in Non-Stationary Environments

Victoria Aragón, Susana Esquivel
Lab. de Investigación y Desarrollo en Inteligencia Computacional (LIDIC).
Dpto. de Informática - Universidad Nacional de San Luis
Ejército de los Andes 950 - 5700 - San Luis - Argentina
{*esquivel, vsaragon*}*@unsl.edu.ar*

Carlos A. Coello Coello
CINVESTAV-IPN (Evolutionary Computation Group)
Electrical Eng. Department, Computer Science Dept.
Av. IPN No. 2508, Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO
ccoello@cs.cinvestav.mx

Abstract

This paper proposes an approach, called Multi-objective Algorithm for Dynamic Environments (MADE), which extendes Fonseca and Fleming's MOGA (with an external archive) so that it can deal with dynamic environments. MADE includes two techniques to maintain diversity and also uses specialized functions that implements the dynamism required. In order to validate MADE, we defined a dynamic version of a static test problem (with 3 objectives) previously proposed in the specialized literature. The preliminary results obtained indicate that the proposed approach provides an acceptable response to the type of changes studied.

1 Introduction

In the context of single-objective optimization, many real-world problems are dynamic in nature. If there is a change over time in a certain problem, either because the objective function changes or because some of its constraint changes (or both), then evidently the global optimum changes as well. In order to deal with this type of problems, it is necessary to have heuristics that can adapt quickly enough to any changes. Given that adaptation in nature is a continuous process, the use of evolutionary algorithms to deal with non-stationary environments seems a natural choice. However, note that in practice evolutionary algorithms tend to converge to a stationary point (i.e., local optimum) over time, losing the diversity of the population necessary to explore the search space. Once this happens, an evolutionary algorithm loses its capability to adapt to any changes in the environment. Thus, the use of good mechanisms to maintain diversity is critical when dealing with dynamic functions.

The use of evolutionary algorithms for dealing with non-stationary (or dynamic) environments has received increasing attention from researchers [1, 2, 3]. However, the definion of dynamic multi-objective test functions and algorithms has been the subject of very little work in the specialized literature [4]. This paper provides a preliminary study regarding the use of a multi-objective evolutionary algorithm in dynamic environments. Unlike the work of Farina et al. [4], in our case, we don't focus our research on dynamic control problems nor on the design of dynamic test functions. Instead, the focus of our work is to study the behavior of relatively simple mechanisms to respond to dynamic changes. Such mechanisms are incorporated into a well-known multi-objective evolutionary algorithm (Fonseca and Fleming's MOGA [5]) aiming to provide some preliminary insights regarding the possible challenges that dynamic functions present for current MOEAs.

The remainder of this paper is organized as follows. In Section 2, we provide a brief introduction to optimization in non-stationary environments both in the single-objective and in the multi-objective cases. Section 3 describes our proposed approach. In Section 4, we provide the test function adopted for our study. Section 5 describes the performance measures adopted in our study. Section 6 describes our experiments and provides

a discussion of our findings. Finally, Section 8 provides our conclusions and some possible paths of future research.

2 Optimization in Non-stationary Environments

One possible approach to deal with dynamic functions is to treat each change as a new optimization problem that has to be solved from scratch [6]. It should be obvious that this sort of approach is impractical in many cases, because disregarding previous information from the problem will certainly increase the computational cost to solve it. Moreover, if the change is small, one would assume that the new solution will be similar to the previous one. Thus, it is desirable to have optimization algorithms capable of adapting solutions to a dynamic (i.e., non-stationary) environment, reusing information obtained in the past. In single-objective dynamic problems, as evolution progresses, different environments emerge, which must be optimized. Thus, the main goal is to find a set of points such that each of them satisfies each of the existing environments. The same situation arises when dealing with multiobjective optimization problems, only that in this case, the goal is not to find a single solution for each environment, but a set of them. Based on the previous discussion, we will denote as $P^*(t)$ and $PF^*(t)$ to the Pareto optimal set and the Pareto front, respectively, both defined at time t.

Two issues are of particular importance when dealing with dynamic environments: (1) the ability of an approach to detect that a change has occurred and (2) the proper reaction (i.e., the velocity of the response) to those changes. In the context of dynamic multiobjective optimization, we will call *environment* both to decision variable space and to objective function space. So, when we refer to changes in the environment, this could be in either of these spaces or in both. Farina et al. [4] proposed several types of changes that can be produced in dynamic multiobjective optimization:

- **Type I**: The Pareto optimal set P^* changes, while the Pareto front PF^* does not.
- **Type II**: Both P^* and PF^* change.
- **Type III**: P^* does not change but PF^* does.
- **Type IV**: The problem dynamically changes, but neither P^* nor PF^* change.

The change of Type IV is not of interest for us for obvious reasons. For the work reported in this paper, we considered only changes of Type III, because what changes is the location of the true Pareto front. Although the Pareto optimal set does not need to be changed for such problems to remain on the new PF^*, there may be an effect on the distribution of the solution on the new PF^* for the old solutions [4]. Thus, we implemented random changes as defined in [1] (i.e., each change does not depend from the previous change nor from time). Note that in this case, if the change is too large, the new problem to be optimized will be completely different to the previous one. It is worth noticing that we won't deal with the automatic detection of the changes, but only the reaction of the algorithm to such changes. Thus, we assume in this work that it is known that a change in the environment has occurred, since they are systematic (i.e., the changes are performed at certain intervals defined in terms of a number of generations).

3 Proposed Approach

The approach proposed in this paper is called Multiobjective Algorithm for Dynamic Environments (MADE), and it consists of an extension of Fonseca and Fleming's MOGA [5] with an external archive. The main focus of this work was to experiment with a relatively conventional multiobjective evolutionary algorithm extended with special diversity maintenance mechanisms that allow it to adapt to changes in the enviroment.

MADE keeps the basic characteristics from MOGA but adds specialized functions that implement the dynamism required. One of the few changes to MOGA is that we have eliminated its mating restrictions. This is mainly due to the fact that there is no clear consensus regarding the usefulness of mating restrictions [7]. Furthermore, MADE includes two techniques to maintain diversity and we considered unnecessary to introduce this additional mechanism. MADE uses real-numbers encoding, proportional selection, one-point crossover and uniform mutation.

We use two mechanisms to maintain diversity:

- **Recrudescence**: This approach was proposed in [8] and it consists of macromutations. The approach increases both recombination and mutation probabilities of a portion of the population. The operator is applied at each generation with a certain probability (p_{recru}) and produces a radical phenotypic reorganization of the individuals over

1. $t = 0$
2. Initialize $(P(0))$ and Empty_External_File()
3. Evaluate $(P(0), F(0))$
4. **while** $(t < Num\_max\_gen)$ **do**
5. $t = t + 1$
6. New_Generation $(P(t), P'(t))$
7. $P(t) = P'(t)$
8. Evaluate $(P(t), F(t))$
9. Elistism()
10. **if** (Change_Function (t))
11. Statistical_Report()
12. Elitist_Set_to_External_File()
13. Clean_Elitist_Set()
14. Function_go_to_Change $(F(t), F'(t))$
15. $F(t) = F'(t)$
16. Evaluate $(P(t), F(t))$
17. Elistim()
18. Insert_Random_Inmigrants()
19. Evaluate $(P(t), F(t))$
20. Elitism()
21. **end if**
22. **end while**

Figure 1: General outline of our MADE approach

which it operates. Individuals to which the operator is applied are randomly selected (adopting a uniform distribution).

- **Random Immigrants:** This idea was proposed in [9, 10]. The approach consists of replacing a percentage of the population by randomly generated individuals. The technique is applied only when there is a change in the environment.

3.1 Pseudocode of our MADE

Once we initialize the main and secondary populations (line 2), the main population is evaluated with the base function $F(0)$ (line 3). The algorithm enters a loop (line 4) that is executed during a certain number of generations. Such a number is determined based on the number of changes that the environment experiments, and the generational interval between them. In the procedure `New_Generation`, for each pair of individuals selected as parents, the recombination operator is invoked. Such a recombination operator includes the mutation operator, which is used with a low probability if the recrudescence operator is not applied. Otherwise, the recrudescence operator (macromutation) is invoked and the mutation and recombination probabilities are incremented. Once the next population $P^{'}(t)$ has been generated, it replace to the current $P(t)$ and is evaluated with the current $F(t)$ (line 8), and we apply elitism (line 9). The procedure `elitism` takes each nondominated individual from the population and verifies if it is not dominated with respect to the elistist set (which is the set of all the solutions that are nondominated with respect to all the solutions generated so far). Should that be the case, the individual is inserted in the elitist set. If during this checking, an individual in the elitist set is dominated by an individual from the current population, then the dominated individual is removed from the elitist set.

The function `Change_Function` (line 10) determines if it is necessary to produce a change in the environment in the current generation. This is done by checking if the current generation is a multiple of the number of generations between changes that was provided as an input to the algorithm. The changes in the environment are produced at constant intervals (defined in terms of a certain number of generations) during the evolutionary process. Each time the environment is about to change, the corresponding statistics are reported (line 11). Such statistics include the number of nondominated individuals, ESS, etc. The nondominated solutions found so far (and temporarily retained in memory) are dumped into an external archive (line 12). Then, the elitist set is emptied, since the objective functions stored within do not correspond to the new function any more (line 13). The function `Function_go_to_Change` is responsible for introducing changes in the environment. The changes implemented are both ascending and descending displacements in all the objective functions or some of them. This is determined by the user (line 14). The old objective function is replaced by the new one (line 15) and the population is evaluated using the new objective function (line 16). We then apply elitism (line 17) so that we can retain the nondominated vectors present in the population. Next, a percentage of the population is replaced by individuals randomly generated (line 18). The individuals selected to be replaced are those dominated by some other individual in the population. In case the number of individuals to be replaced is less than the number of individuals that are dominated, then we replace as many nondominated individuals as necessary until completing the (pre-defined) percentage. It is worth noticing that it is irrelevant to lose nondominated individuals from the population, since they have already been stored in the secondary population (line 20). We then evaluate again the population (with the new inserted individuals) and we apply elitism all over again.

4 Test Function

In order to validate our proposed approach, we introduce a dynamic version of a well-known test problem (DTLZ2 [11]) which, in its static version, has been used to validate multi-objective evolutionary algorithms. This function was chosen because it is scalable both in decision variable space and in objective function space. Such scalability facilitates to study the capability of an algorithm to react to changes in both spaces. Although other test functions have been adopted to validate our approach, we chose to include only one to allow a more detailed analysis of the behavior of the mechanisms proposed.

DTLZ2: Min $(f_1(x), f_2(x), f_3(x))$, where: $f_1(x) = (1 + g(x_3, x_4))cos(x_1\pi/2)cos(x_2\pi/2)$, $f_2(x) = (1+g(x_3, x_4))cos(x_1\pi/2)sin(x_2\pi/2)$, and $f_3(x) = (1 + g(x_3, x_4))sin(x_1\pi/2)$,
with $0 \leq x_i \leq 1; i = 4$ and $g(x) = \sum_{x_i \in x}(x_i - 0.5)^2$.

On this base function, all objective functions take non-negative values and the desired front is the first quadrant of a sphere of radius one. The dynamic environment is generated by translating the base function **DTLZ2** along a linear trajectory according to [12]:

$$\textbf{DTLZ2\_Dyn}(x,t) = \textbf{DTLZ2}(x) + \delta(t)$$

where $t \in N_0$ denotes time (generation number). The displacement of the function is determined by function $\delta(t) = (\delta_1(t), \delta_2(t), \delta_3(t))$ and it depends on the update frequency of the function (i.e., the number of generations between changes) and the severity s (a factor that determines the length of the function displacement).

For an ascending linear displacement, we have: $\delta_1(0) = \delta_2(0) = \delta_3(0) = 0$

$$\delta(t+1) = \begin{cases} \delta_i(t) + s_i & \text{if } (t+1) \bmod \text{interval} = 0 \\ \delta_i(t) & \text{otherwise} \end{cases} \quad (1)$$

where i is the objective function number to be changed, `interval` is the number of generations between changes and s_i is the severity degree of the displacement of f_i.

For a linear descending displacement, we have:

$\delta_1(0) = amount\_of\_change * s_1$
$\delta_2(0) = amount\_of\_change * s_2$
$\delta_3(0) = amount\_of\_change * s_3$

$$\delta(t+1) = \begin{cases} \delta_i(t) - s_i & \text{if } (t+1) \bmod \text{interval} = 0 \\ \delta_i(t) & \text{otherwise} \end{cases} \quad (2)$$

where i is index of the objective function to be changed, `interval` is the number of generations allowed between changes, s_i is the degree of severity of the displacement of f_i and `amount_of_change` is the number of changes that the environment will experiment.

If the severity is too high, then the sequence of problems to be optimized won't share anything in common. This would be similar to solving completely different problems by separate. On the contrary, if the severity is small, there could be no perceptible difference between two consecutive changes and this can be considered as a non-dynamic problem (i.e., it could be treated as a static problem and one could build robust solutions for such problem [1]). As a consequence, the severity of the changes produced in our experiments is such that the set of feasible solutions between changes gets partially overlapped.

5 Performance Measures

It is obviously desirable that our multi-objective evolutionary algorithm (MOEA) is able to reach (either in static or dynamic environments) the true Pareto front of a problem with a good spread of points. In order to evaluate the performance of our approach, we adopted the following performance measures:

1. **Unsuccessful Counting (USCC)**: We define this measure based on the idea of the *Error Ratio* metric proposed in [13] which indicates the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set. In this case, we count the number of vectors (in the current set of nondominated vectors available) that are not members of the Pareto optimal set: $USCC = \sum_{i=1}^{n} u_i$, where n is the number of vectors in the current set of nondominated vectors available; $u_i = 1$ if vector i is not a member of the Pareto optimal set, and $u_i = 0$ otherwise. It should then be clear that $USCC = 0$ 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. For a fair comparison, when we use this measure, all the algorithms should limit their final number of non-dominated solutions to the same value.

2. **Inverted Generational Distance (IGD)**: The concept of generational distance was introduced by Van Veldhuizen [13] 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 measure 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. In our case, we implemented an "inverted" generational distance measure (IGD) in which we use as a reference the true Pareto front, and we compare each of its elements with respect to the front produced by an algorithm. In this way, we are calculating how far are the elements of the true Pareto front, from those in the Pareto front produced by our algorithm. Computing this "inverted" generational distance value reduces the bias that can arise when an algorithm didn't fully cover the true Pareto front.

3. **Efficiently Spaced Set** (ESS): 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 [14] proposed such a metric measuring the range (distance) variance of neighboring vectors in the nondominated vectors found so far. This metric is defined as:

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

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

4. **Number of Nondominated Individuals per Environment** (NNIE): This measure is self-explanatory.

Furthermore, we also considered the following performance measures:

1. **Average Unsuccessful Counting** (AUSCC): Average of the unsuccessful counting values of the population in the generation just before the change. It is defined as:

$$AUSCC = (1/k) \sum_{j=1}^{k} USCC_j$$

where k is the number of changes in the environment, $USCC_j$ is the unsuccessful counting in the environment j.

2. **Average Inverted Generational Distance** (AIGD): The average of the generational distance values of the population at the generation just before the change. It is defined by:

$$AIGD = (1/k) \sum_{j=1}^{k} IGD_j$$

where k is the number of changes in the environment, IGD_j is the generational distance in the environment j.

3. **Average Efficiently Spaced Set** (AESS): The average of the ESS values of the population at the generation just before the change. It is defined by:

$$AESS = (1/k) \sum_{j=1}^{k} ESS_j$$

where k is the number of changes in the environment, ESS_j is the Efficiently Spaced Set in the environment j.

4. **Average Number of Nondominated Individuals** (ANNIE): Average of the NNIE values of the population in the generation just before the change. It is defined by:

$$ANNIE = (1/k) \sum_{j=1}^{k} NNIE_j$$

where k is the number of changes in the environment, $NNIE_j$ is the number of individuals nondominated in the environment j.

6 Description of the Experiments

This section aims to describe the experiments performed to validate our proposed approach. Obviously, the aim is to evaluate the capability of our approach to track down the new location of the true Pareto front, once the algorithm has detected a change in the environment. The experiments took place on different scenarios. Each of these scenarios represents a different type of change:

1. Scenario 1: Positive Linear Displacement of $f_1(x)$ only,
2. Scenario 2: Positive Linear Displacement of $f_2(x)$ only,
3. Scenario 3: Positive Linear Displacement of $f_3(x)$ only,
4. Scenario 4: Positive Linear Displacement of both $f_1(x)$ and $f_2(x)$ only,
5. Scenario 5: Positive Linear Displacement of both $f_1(x)$ and $f_3(x)$ only,
6. Scenario 6: Positive Linear Displacement of both $f_2(x)$ and $f_3(x)$ only,
7. Scenario 7: Negative Linear Displacement of $f_1(x)$ only,
8. Scenario 8: Negative Linear Displacement of $f_2(x)$ only,
9. Scenario 9: Negative Linear Displacement of $f_3(x)$ only,
10. Scenario 10: Negative Linear Displacement of both $f_1(x)$ and $f_2(x)$ only,
11. Scenario 11: Negative Linear Displacement of both $f_1(x)$ and $f_3(x)$ only,
12. Scenario 12: Negative Linear Displacement of both $f_2(x)$ and $f_3(x)$ only,
13. Scenario 13: Positive Linear Displacement of $f_1(x)$, $f_2(x)$ and $f_3(x)$ simultaneously, and
14. Scenario 14: Negative Linear Displacement of $f_1(x)$, $f_2(x)$ and $f_3(x)$ simultaneously.

The parameters required by our approach are the following:

1. Probabilities for the operators: recombination, mutation, recrudescence, increase for the recombination, increase for the mutation.
2. Percentage of Random Immigrants,
3. Number of Generations between Changes of the Environment,
4. Number of Changes of the Function,
5. Objective Functions that we wish to modify,
6. Degree of severity for f_1 ,f_2 and f_3,
7. Type of Displacement (positive or negative).

Table 1: Values adopted for the parameters used by our approach

Population size	200
Size of the secondary population	250
Initialization of the Secondary Population	Random
$P_{crossover}$	0.65
$P_{mutation}$	0.5
$P_{recombination-increase}$	0.8
$P_{mutation-increase}$	0.8
$P_{recrudescence}$	0.2
Percentage of Random Immigrants	30 individuals
Number of Changes of the Function	9
s_1 for DTLZ2_Dyn	0.5
s_2 for DTLZ2_Dyn	0.5
s_3 for DTLZ2_Dyn	0.5

Each experiment was repeated 10 times (with 10 different random seeds), and we collected statistics from the corresponding runs. For each scenario, the changes in the environment were produced at every 5 and 10 generations. The values adopted for the parameters of our approach are shown in Table 1.

7 Results

The performance of MADE using the benchmark function previously described (with changes at every 5 and 10 generations, and with both ascending descending displacements) varies from good to relatively good on the different scenarios. In the following Tables, the labels of the columns 2 to 4 and 2 to 5 are interpreted as follows: the A or D means Ascending or Descending displacement, respectively, and the number following the underscore indicates the scenario number. First, we analyze the results for the scenarios 13 and 14 where we apply positive and negative displacements over the three objectives, simultaneously. In this case, regardless of the size of the interval between changes and regardless of the displacement types, the Pareto fronts produced by MADE are very close to the true Pareto fronts with a good distribution of points in both cases. This is shown in Figures 2 to 5 for the case in which the changes are produced at every 5 generations.

Analogously, when analyzing the results from Table 2, we can see that we obtain little variability of results for all metrics, except for ANNIE. In the case of this metric (average number of non-dominated individuals), we obtain a lower value for descending displacements. Between two consecutive changes, regardless of the displacement, the old and new feasible zone share a set of solutions. But, when the displacements are 1) de-

scending: the old PF^* is contained in the new feasible zone and 2) ascending: the new PF^* is contained in the old feasible zone. The reason of the MADE's poor performance, on descending displacements could be because of the new PF^* is not contained in the old feasible zone, so any nondominated individuals from the old feasible zone belong to the new PF^*. The new PF^* has to be found from scratch but the old PF^* could be used like a base to find the new PF^*. However, on ascending displacements, the new PF^* is contained in the old feasible zone, so the search of the new PF^* could not start from scratch if some nondominated individuals belong to the new PF^* or they are close from it.

For scenarios 4 to 6 and 10 to 12, where the displacements are applied simultaneously to pairs of objectives (see Tables 3 to 6), we can see in Figures 6 to 11 that our approach can obtain reasonably good approximations and a good spread of solutions.

Finally, for the scenarios 1 to 3 and 7 to 9, where only one of the objective functions is displaced, MADE produces good approximations of the true Pareto front, with a good spread of solutions. However, in all cases, a few solutions are produced away from the true Pareto front (see Figures 12 to 17).

When looking at the numerical results in Tables 7, 8, 9 and 10, we observe that when only f_1 is displaced, regardless of the interval between changes, ascending displacements seem to be the most difficult to handle by MADE.

It is worth emphasizing that, despite the high values obtained for the AUSCC metric in all the scenarios studied, our approach systematically converged very close to the true Pareto front (this can be appreciated by looking at the values of the IGD metric). However, since the exact front was not reached, the AUSCC metric provided poor results. So, the reachability problem seems to be more an accurancy problem which may be related to the small number of generations between changes (i.e., the algorithm doesn't have enough time to produce a finer approximation of the Pareto front).

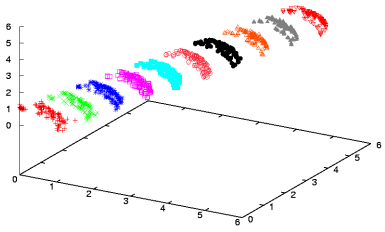

Figure 2: Front generated by MADE when f_1, f_2 and f_3 are displaced in an ascending way

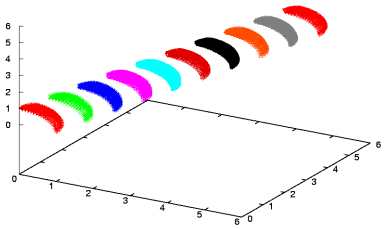

Figure 3: PF_{true} corresponding to the change indicated in Figure 2

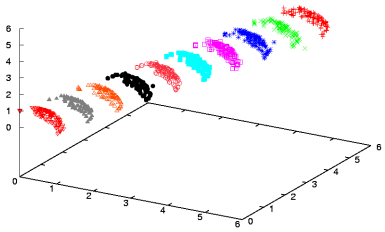

Figure 4: Front generated by MADE when f_1, f_2 and f_3 are displaced in a descending way

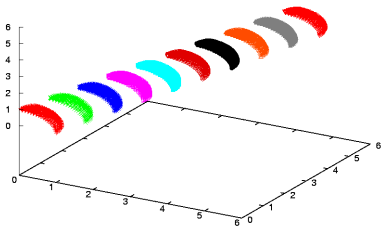

Figure 5: PF_{true} corresponding to the change indicated in Figure 4

Table 2: DTLZ2_Dyn, modifications in f_1, f_2 and f_3, values taken from the run in the median respect of IGD

	Interval = 5		Interval = 10	
Displacement	A_13	D_14	A_13	D_14
AESS	0.04	0.05	0.04	0.05
IGD	0.000022	0.000025	0.000023	0.000024
ANNIE	114	97	113	98
AUSCC	249.9	246.1	250.0	249.5

Table 3: DTLZ2_Dyn, modifications in f_1f_2; f_1f_3; f_2f_3; values of the run in the median respect of IGD

	Interval=5		
Displacement	A_4	A_5	A_6
AESS	0.05	0.05	0.05
AIGD	0.000023	0.000024	0.000024
ANNIE	100	104	97
AUSCC	248.5	249.8	247.6

Table 4: DTLZ2_Dyn, modifications in f_1f_2; f_1f_3; f_2f_3; values of the run in the median respect of IGD

	Interval=5		
Displacement	D_10	D_11	D_12
AESS	0.04	0.05	0.04
AIGD	0.000023	0.000023	0.000023
ANNIE	113	105	108
AUSCC	249.6	249.6	248.8

Table 5: DTLZ2_Dyn, modifications in f_1f_2; f_1f_3; f_2f_3; values of the run in the median respect of IGD

	Interval = 10		
Displacement	A_4	A_5	A_6
AESS	0.04	0.05	0.04
AIGD	0.000023	0.000024	0.000024
ANNIE	113	95	99
AUSCC	249.9	249.1	247.1

Table 6: DTLZ2_Dyn, modifications in f_1f_2; f_1f_3; f_2f_3; values of the run in the median respect of IGD

	Interval = 10		
Displacement	D_10	D_11	D_12
AESS	0.04	0.05	0.05
AIGD	0.000022	0.000023	0.000023
ANNIE	111	108	110
AUSCC	249.9	249.6	250.0

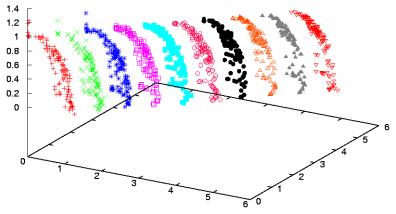

Figure 6: Front generated by MADE when f_1, and f_2 are displaced in an ascending way

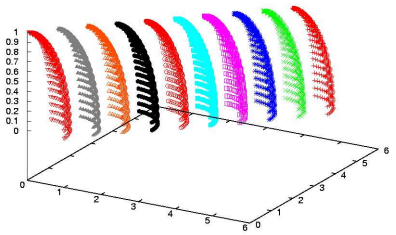

Figure 7: PF_{true} corresponding to the change indicated in Figure 6

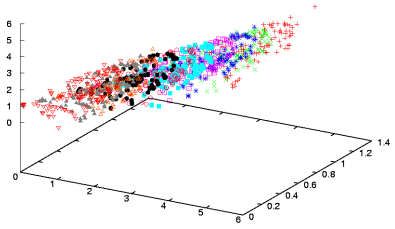

Figure 8: Front generated by MADE when f_1 and f_3 are displaced in a descending way

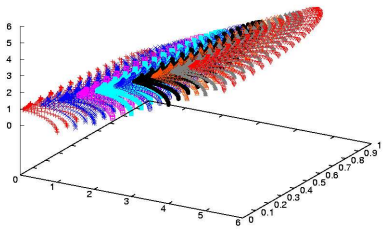

Figure 9: PF_{true} corresponding to the change indicated in Figure 8

Table 7: DTLZ2_Dyn, modifications in f_1; f_2; f_3; values of the run in the median respect of IGD

	Interval=5		
Displacement	A_1	A_2	A_3
AESS	0.05	0.04	0.05
AIGD	0.000024	0.000024	0.000024
ANNIE	94	101	97
AUSCC	247.0	248.3	249.3

Table 8: DTLZ2_Dyn, modifications in f_1; f_2; f_3; values of the run in the median respect of IGD

	Interval=5		
Displacement	D_7	D_8	D_9
AESS	0.05	0.04	0.04
AIGD	0.000023	0.000023	0.000023
ANNIE	104	111	110
AUSCC	249.3	250.0	250.0

Table 9: DTLZ2_Dyn, modifications in f_1; f_2;f_3; values of the run in the median respect of GDM

	Interval = 10		
Displacement	A_1	A_2	A_3
AESS	0.054	0.05	0.05
AIGD	0.000025	0.000024	0.000024
ANNIE	98	99	101
AUSCC	246.1	248.1	249.8

Table 10: DTLZ2_Dyn, modifications in f_1; f_2;f_3; values of the run in the median respect of GDM

	Interval = 10		
Displacement	D_7	D_8	D_9
AESS	0.04	0.04	0.04
AIGD	0.000023	0.000022	0.000023
ANNIE	113	111	108
AUSCC	250.0	249.8	249.8

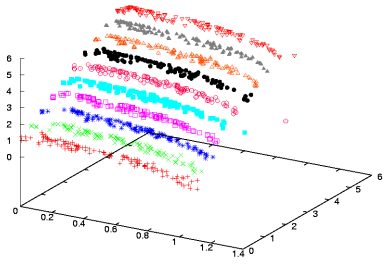

Figure 10: Front generated by MADE when f_2, and f_3 are displaced in an ascending way

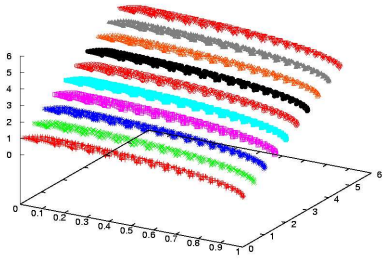

Figure 11: PF_{true} corresponding to the change indicated in Figure 10

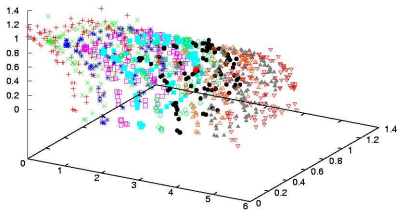

Figure 12: Front generated by MADE when f_1 is displaced in an ascending way

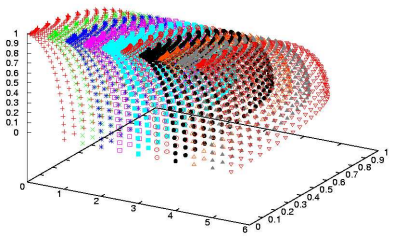

Figure 13: PF_{true} corresponding to the change indicated in Figure 12

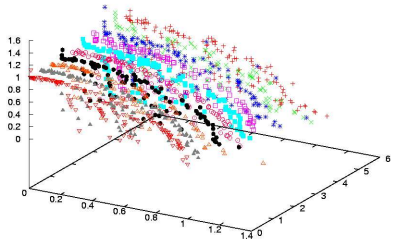

Figure 14: Front generated by MADE when f_2 is displaced in a descending way

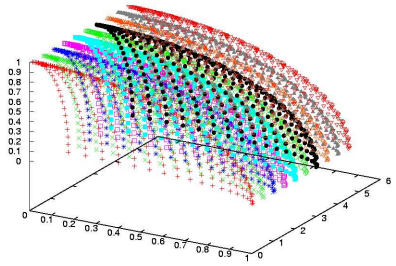

Figure 15: PF_{true} corresponding to the change indicated in Figure 14

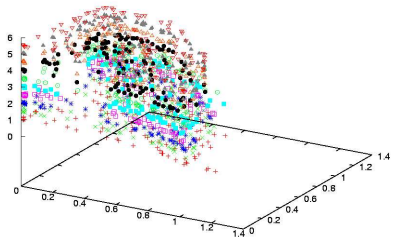

Figure 16: Front generated by MADE when f_3 is displaced in an ascending way

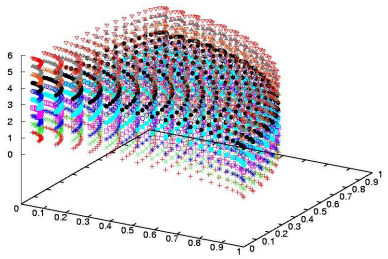

Figure 17: PF_{true} corresponding to the change indicated in Figure 16

8 Conclusions and Future Work

The performance of the proposed approach in the test case adopted turned out to be satisfactory in the sense that the aim was not to generate the complete true Pareto fronts, but to determine if the proposed approach was able to adapt fast enough to the changes in the location of the true Pareto front.

As part of our future work, we intend to explore the use of other (more sophisticated) operators to handle dynamic environments [15]. We will also validate the performance of our MADE when scaling the test function both on the number of decision variables and objective functions. Also, we want to explore the impact of ϵ-dominance [16] in the performance of our approach.

Another aspect that deserves more attention is the choice of performance measures adopted to validate the performance of our approach. Finally, we also aim to evaluate other search engines different from MOGA (e.g., we intend to use the NSGA-II [17] which is a highly competitive multi-objective evolutionary algorithm).

9 Acknowledgements

The first two authors acknowledge support from the Universidad Nacional de San Luis and the ANPCYT. The third author acknowledges support from the Consejo Nacional de Ciencia y Tecnología (CONACyT) through project number 42435-Y.

References

[1] Branke, J.: Evolutionary optimization in dynamic environments. Kluwer Academic Publishers (2002)

[2] Yamasaki, K.: Dynamic pareto optimum GA against the changing environments. In Branke, J., Bäck, T., eds.: Evolutionary Algorithms for Dynamic Optimization Problems, San Francisco, California, USA (2001) 47–50

[3] Jin, Y., Sendhoff, B.: Constructing dynamic optimization test problems using multi-objective optimization concept (2004) In G. R. Raidl, editor, Aplications of Evolutionary Computing, volume 3005 of LNCS, pages 525-536. Springer.

[4] Farina, M., Deb, K., Amato, P.: Dynamic Multiobjective Optimization Problems: Test Cases, Approximation, and Applications. In Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L., eds.: Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, Faro, Portugal, Springer. Lecture Notes in Computer Science. Volume 2632 (2003) 311–326

[5] Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Forrest, S., ed.: Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers (1993) 416–423

[6] Raman, N., Talbot, F.B.: The job shop tardiness problem: A descomposition approach. European Journal of Operational Research **69** (1993) 187–199

[7] Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York (2002) ISBN 0-3064-6762-3.

[8] Kwasnicka, H.: Redundancy of genotypes as the way for some advanced operators in evolutionary algorithms - Simulation Study. VIVEK, A Quarterly in Artificial Intelligence **10** (1997) 2–11

[9] Cobb, H., Grefenstette, J.: Genetic algorithms for tracking changing environments. In: Proceeding of the 5th IEEE International Conference on Genetic Algorithms, Morgan Kauffman (1993) 523–530

[10] Grefenstette, J.: Optimization of control parameters for genetic algorithms. IEEE Transaction on Systems, Man and Cybernetic **16** (1986) 122–128

[11] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: Congress on Evolutionary Computation (CEC'2002). Volume 1., Piscataway, New Jersey, IEEE Service Center (2002) 825–830

[12] Back, T.: On the behavior of evolutionary algorithms in dynamic environnments. In: Proceedings of International Conference on Evolutionary Computation, Piscataway, NJ, IEEE Press (1998) 446–451

[13] 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)

[14] Schott, J.R.: Fault tolerant design using single and multicriteria genetic algorith optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachussetts (1995)

[15] Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. Springer-Verlag, Berlin (2004)

[16] Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. Evolutionary Computation **10** (2002) 263–282

[17] 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

Received: Feb. 2005. Accepted: May 2005.