Improving Multi-Objective Evolutionary Algorithms by Using Rough Sets

Alfredo G. Hernández-Díaz[1], Luis V. Santana-Quintero[2]
Carlos A. Coello Coello[2], Rafael Caballero[3], and Julián Molina[3]

[1] Pablo de Olavide University, Department of Quantitative Methods, Seville, Spain
agarher@upo.es
[2] CINVESTAV-IPN, Computer Science Section, México
lvspenny@hotmail.com, ccoello@cs.cinvestav.mx
[3] University of Málaga, Department of Applied Economics (Mathematics), Spain
rafael.caballero@uma.es, julian.molina@uma.es

Summary. In this chapter, we propose the use of rough sets to improve the approximation provided by a multi-objective evolutionary algorithm. The main idea is to use this sort of hybrid approach to approximate the Pareto front of a multi-objective optimization problem with a low computational cost (only 3000 fitness function evaluations). The hybrid operates in two stages: in the first one, a multi-objective version of differential evolution is used as our search engine in order to generate a good approximation of the true Pareto front. Then, in the second stage, rough sets theory is adopted in order to improve the spread of the solutions found so far. To assess our proposed hybrid approach, we adopt a set of standard test functions and metrics taken from the specialized literature. Our results are compared with respect to the NSGA-II, which is an approach representative of the state-of-the-art in the area.

1 Introduction

Multi-Objective Programming (MOP) is a research field that has raised great interest over the last thirty years, mainly because of the many real-world problems which naturally have several (often conflicting) criteria to be simultaneously optimized [7, 17].

In recent years, a wide variety of multi-objective evolutionary algorithms (MOEAs) have been proposed in the specialized literature [4, 3]. However, the study of hybrids of MOEAs with other types of techniques is still relatively scarce. This chapter presents a study of the use of rough sets theory as a local search explorer able to improve the spread of the solutions produced by a MOEA. Our main motivation for such a hybrid approach is to reduce the overall number of fitness function evaluations performed to approximate the true Pareto front of a problem. Our proposed hybrid is able to produce

reasonably good approximations of the Pareto front of a variety of problems of different complexity with only 3000 fitness function evaluations.

The organization of the rest of the chapter is the following. Section 2 provides some basic concepts required to understand the rest of the chapter. An introduction to rough sets theory is provided in Section 3. In Section 4, we introduce differential evolution, which is the approach adopted as our search engine. Section 5 describes the relaxed form of Pareto dominance adopted for our secondary population (called Pareto-adaptive ϵ-dominance). Our proposed hybrid is described in Section 6. The experimental setup adopted to validate our approach and the corresponding discussion of results are provided in Section 7. Finally, our conclusions and some possible paths for future research are provided in Section 8.

2 Basic Concepts

We are interested in solving problems of the type[4]:

$$\text{Minimize } \mathbf{f}(\mathbf{x}) := [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \tag{1}$$

subject to:

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

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

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., k$ are the objective functions and $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., m$, $j = 1, ..., p$ are the constraint functions of the problem. To describe the concept of optimality in which we are interested, we will introduce next a few definitions.

Definition 1. Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$, we say that $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for $i = 1, ..., k$, and that $\mathbf{x}$ **dominates** $\mathbf{y}$ (denoted by $\mathbf{x} \prec \mathbf{y}$) if $\mathbf{x} \leq \mathbf{y}$ and $\mathbf{x} \neq \mathbf{y}$.

Definition 2. We say that a vector of decision variables $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ is **nondominated** with respect to $\mathcal{X}$, if there does not exist another $\mathbf{x}' \in \mathcal{X}$ such that $\mathbf{f}(\mathbf{x}') \prec \mathbf{f}(\mathbf{x})$.

Definition 3. We say that a vector of decision variables $\mathbf{x}^* \in \mathcal{F} \subset \mathbb{R}^n$ ($\mathcal{F}$ is the feasible region) is **Pareto-optimal** if it is nondominated with respect to $\mathcal{F}$.

Definition 4. The **Pareto Optimal Set** $\mathcal{P}^*$ is defined by:

$$\mathcal{P}^* = \{\mathbf{x} \in \mathcal{F} | \mathbf{x} \text{ is Pareto-optimal}\}$$

[4] Without loss of generality, we will assume only minimization problems.

Definition 5. The **Pareto Front** $\mathcal{PF}^*$ is defined by:

$$\mathcal{PF}^* = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^k | \mathbf{x} \in \mathcal{P}^*\}$$

We thus wish to determine the Pareto optimal set from the set $\mathcal{F}$ of all the decision variable vectors that satisfy (2) and (3).

3 Rough Sets Theory

Rough Sets theory is a new mathematical approach to imperfect knowledge. The problem of imperfect knowledge has been tackled for a long time by philosophers, logicians and mathematicians. Recently, it also became a crucial issue for computer scientists, particularly in the area of artificial intelligence (AI). There are many approaches to the problem of how to understand and manipulate imperfect knowledge. The most used one is the fuzzy set theory proposed by Lotfi Zadeh [26]. Rough sets theory was proposed by Pawlak [19], and presents another attempt to this problem. Rough sets theory has been used by many researchers and practitioners all over the world and has been adopted in many interesting applications. The rough sets approach seems to be of fundamental importance to AI and cognitive sciences, especially in the areas of machine learning, knowledge acquisition, decision analysis, knowledge discovery from databases, expert systems, inductive reasoning and pattern recognition. Basic ideas of rough set theory and its extensions, as well as many interesting applications, can be found in books (see [20]), special issues of journals (see [15]), proceedings of international conferences, and in the internet (see `www.roughsets.org`).

Let's assume that we are given a set of objects U called the *universe* and an indiscernibility relation $R \subseteq U \times U$, representing our lack of knowledge about elements of U (in our case, R is simply an equivalence relation based on a grid over the feasible set; this is, just a division of the feasible set in (hyper)-rectangles). Let X be a subset of U. We want to characterize the set X with respect to R. The way rough sets theory expresses vagueness is employing a boundary region of the set X built once we know points both inside X and outside X. If the boundary region of a set is empty it means that the set is *crisp*; otherwise, the set is *rough* (inexact). A nonempty boundary region of a set means that our knowledge about the set is not enough to define the set precisely (see Figure 1).

Then, each element in U is classified as *certainly* inside X if it belongs to the lower approximation or *partially* (*probably*) inside X if it belongs to the upper approximation (see Figure 1). The *boundary* is the difference of these two sets, and the bigger the boundary the worse the knowledge we have of set X. On the other hand, the more precise is the grid implicity used to define the indiscernibility relation R, the smaller the boundary regions are. But, the more precise is the grid, the bigger the number of elements in U, and then,

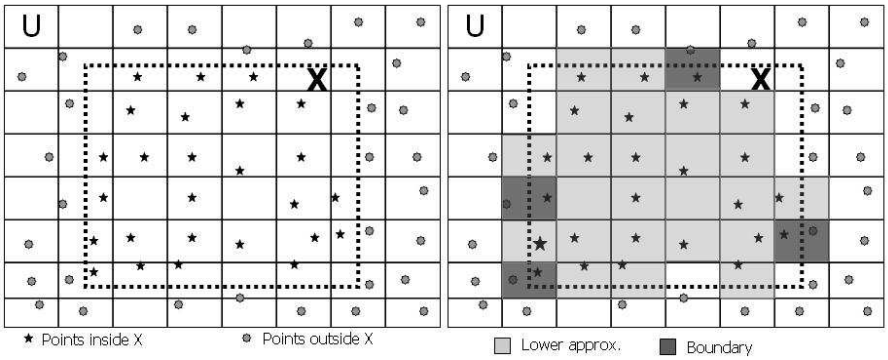

Fig. 1. Rough sets approximation

the more complex the problem becomes. Then, the less elements in U the better to manage the grid, but the more elements in U the better precision we obtain. Consequently, the goal is obtaining "small" grids with the maximum precision possible. These two aspects are called **Density** and **Quality** of the grid. If q is the number of criteria (in our case, the number of objectives), Q_i is the i-th criteria, b_j^i is the j-th value of the i-th criteria (we assume these value are ordered increasingly), then:

$$Density(G) = \sum_{i=1}^{q} \sum_{j=1}^{|Q_i|} x_j^i$$

$$Quality(G) = \frac{|Low(X)|}{|X|}$$

where x_j^i is 1 if b_j^i is active in the grid and $|Low(X)|$ is the cardinality of the lower approximation of X.

3.1 Use of Rough Sets in Multi-Objective Optimization

For our MOP problems we will try to approximate the Pareto front using a Rough Sets grid. To do this, we will use an initial approximation of the Pareto front (provided by any other method) and will implement a grid in order to get more information about the front that will let us improve this initial approximation. Then, at this point we have to face the following problem: the more precise the grid is, the higher the computational cost required to manage it, and the less precise the grid is, the less knowledge we get about the Pareto front. Thus, we need to design a grid that balances these two aspects. In other words, a grid that is not so expensive (computationally speaking) but that offers a reasonably good knowledge about the Pareto front to be used to improve the initial approximation. To this aim, we must design a grid and decide which elements of U (that we will call *atoms* and will be just

rectangular portions of decision variable space) are inside the Pareto optimal set and which are not. Once we have the *efficient atoms*, we could easily intensify the search over these atoms as they are built in decision variable space.

To create this grid, as an input we will have N feasible points divided in two sets: the nondominated points (ES) and the dominated ones (DS). Using these two sets we want to create a grid to describe the set ES in order to intensify the search on it. This is, we want to describe the Pareto front in decision variable space because then we could easily use this information to generate more efficient points and then improve this initial approximation. Figure 2 shows how information in objective function space can be translated into information in decision variable space through the use of a grid.

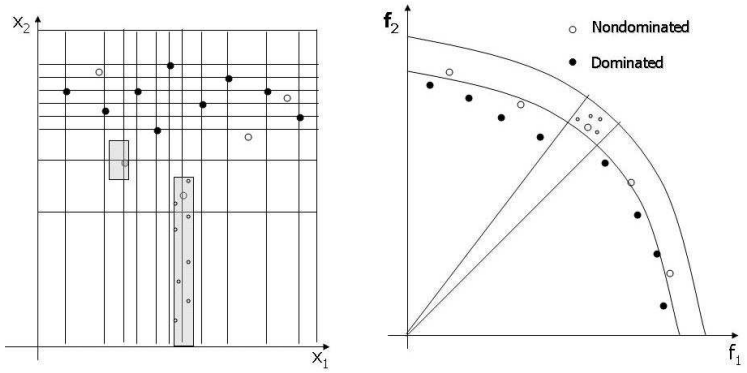

Fig. 2. Decision variable space (left) and objective function space (right)

We must note the importance of the DS set as in a rough sets method the information comes from the description of the boundary of the two sets. Then, the more efficient points provided the better. However, it is also required to provide dominated points, since we need to estimate the boundary between being dominated and being nondominated. Once this information is computed, we can simply generate more points in the "efficient side". The way in which these atoms are computed is described in Section 6.

Since the computational cost of managing the grid increases with the number of points used to create it, we will try to use just a few points. However, such points must be as far from each other as possible, because the better the distribution the points have in the initial approximation the less points we need to build a reliable grid. On the other hand, in order to diversify the search we build several grids using different (and disjoint) sets DS and ES coming from the initial approximation. To ensure these sets are really disjoint we will mark each point as explored or non-explored (if it has been used or not

to compute a grid) and we will not allow repetitions. Algorithm 1 describes a Rough Sets iteration.

Algorithm 1 Rough Sets Iteration

```
Choose NumEff non-explored points of ES.
Choose NumDom non-explored points of DS.
Generate NumEff efficient atoms.
for i = 0 to NumEff do
   for j = 0 to Offspring do
      Generate (randomly) a point new in atom i and send to ES
      if new is efficient then
         Include in ES
      end if
      if A point old in ES is dominated by new then
         Send old to DS
      end if
      if new is dominated by a point in ES then
         Remove new
      end if
   end for
end for
```

4 Differential Evolution

Differential Evolution (DE) [24, 21] is a relatively recent heuristic designed to optimize problems over continuous domains. DE has been shown to be not only very effective as a global optimizer, but also very robust producing in many cases a minimum variability of results from one run to another. DE has been extended to solve multi-objective problems by several researchers (see for example [1, 16, 2, 25, 18, 13, 11, 22]). However, in such extensions, DE has been found to be very good at converging close to the true Pareto front (i.e., for coarse-grained optimization), but not so efficient for actually reaching the front (i.e., for fine-grained optimization). Thus, we will show how these features can be exploited by our hybrid, which uses rough sets theory as a local optimizer in order to improve the spread of the nondominated solutions obtained by the MOEA adopted (which is based on differential evolution in our case).

In DE, each decision variable is represented in the chromosome by a real number. As in any other evolutionary algorithm, the initial population of DE is randomly generated, and then evaluated. After that, the selection process takes place. During the selection stage, three parents are chosen and they generate a single offspring which competes with a parent to determine who passes to the following generation. DE generates a single offspring (instead of

two as a genetic algorithm) by adding the weighted difference vector between two parents to a third parent. In the context of single-objective optimization, if the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector replaces the vector with respect to which it was compared. In addition, the best parameter vector $X_{best,G}$ is evaluated for every generation G in order to keep track of the progress that is made during the minimization process. More formally, the process is described as follows:

For each vector $\overrightarrow{x_{i,G}}; i = 0, 1, 2, \ldots, N-1.$, a trial vector $\overrightarrow{v}$ is generated using:

$$\overrightarrow{v} = \overrightarrow{x_{r1,G}} + F \cdot (\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$$

with $r_1, r_2, r_3 \in [0, N-1]$, integer and mutually different, and $F > 0$.

The integers r_1, r_2 and r_3 are randomly chosen from the interval $[0, N-1]$ and are different from i. F is a real and constant factor which controls the amplification of the differential variation $(\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$.

5 Pareto-adaptive ϵ-dominance

One of the concepts that has raised more interest within evolutionary multiobjective optimization in the last few years is, with no doubt, the use of relaxed forms of Pareto dominance that allow us to control the convergence of a MOEA. From such relaxed forms of dominance, ϵ-dominance [14] is certainly the most popular. ϵ-dominance has been mainly used as an archiving strategy in which one can regulate the resolution at which our approximation of the Pareto front will be generated. This allows us to accelerate convergence (if a very coarse resolution is sufficient) or to improve the quality of our approximation (if we can afford the extra computational cost). However, ϵ-dominance has certain drawbacks and limitations. For example: (1) we can lose a high number of nondominated solutions if the decision maker does not take into account (or does not know) the geometrical characteristics of the true Pareto front, (2) the extrema of the Pareto front are normally lost and (3) the upper bound for the number of points allowed by a grid is not easy to achieve in practice.

In order to overcome some of these limitations, the concept of *pa*ϵ-dominance was proposed in [10]. Briefly, the main difference is that in *pa*ϵ-dominance the hyper-grid generated adapts the sizes of the boxes to certain geometrical characteristics of the Pareto front (e.g., almost horizontal or vertical portions of the Pareto front) as to increase the number of solutions retained in the grid. This scheme maintains the good properties of ϵ-dominance but improves on its main weaknesses. In order to do this, it considers not only a different ϵ for each objective but also the vector $\epsilon = (\epsilon_1, \epsilon_2, ..., \epsilon_m)$ associated to each $f = (f_1, f_2, ..., f_m) \in R^m$ depending on the geometrical characteristics of the Pareto front. This is, the scheme considers different intensities of dominance for each objective according to the position of each point along the

Pareto front. Then, the size of the boxes is adapted depending on the portion of the Pareto front that is being covered. Namely, the boxes are, for example, smaller at the extrema of the Pareto front (since these regions are normally more difficult to cover), and they become larger towards the middle portions of the front.

In [10], it is empirically shown that the advantages of $pa\epsilon$-dominance over ϵ-dominance make it a more suitable choice to be incorporated into a MOEA and therefore our decision of adopting this scheme for the work reported in this chapter.

6 The Hybrid Method: DEMORS

Our proposed approach, called DEMORS (Differential Evolution for Multi-objective Optimization with Rough Sets) [9], is divided in two different phases, and each of them consumes a fixed number of fitness function evaluations. During Phase I, our DE-based MOEA is applied for 2000 fitness function evaluations. During Phase II, a local search procedure based on rough sets theory is applied for 1000 fitness function evaluations, in order to improve the solutions produced at the previous phase. Each of these two phases is described next in more detail.

6.1 Phase I : Use of Differential Evolution

The pseudo-code of our proposed DE-based MOEA is shown in Algorithm 2 [23]. Our approach keeps three populations: the main population (which is used to select the parents), a secondary (external) population, which is used to retain the nondominated solutions found and a third population that retains dominated solutions removed from the second population.

First, we randomly generate 25 individuals, and use them to generate 25 offspring. Phase I has two selection mechanisms that are activated based on the total number of generations and a parameter called $sel_2 \in [0, 1]$, which regulates the selection pressure. For example, if $sel_2 = 0.6$ and the total number of generations is $G_{max} = 200$, this means that during the first 120 generations (60% of G_{max}), a random selection will be adopted, and during the last 80 generations an elitist selection will be adopted. In both selections (random and elitist), a single parent is selected as reference. This parent is used to compare the offspring generated by the three different parents. This mechanism guarantees that all the parents of the main population will be reference parents for only one time during the generating process. Both types of selection and recombination operators are described in [23].

Differential evolution does not use an specific mutation operator, since such operator is somehow embedded within its recombination operator. However, in multi-objective optimization problems, we found it necessary to provide

Algorithm 2 Phase I pseudo-code

```
1: Initialize vectors of the population P
2: Evaluate the cost of each vector
3: for i = 0 to G do
4:    repeat
5:       Select (randomly) three different vectors
6:       Perform crossover using DE scheme
7:       Perform mutation
8:       Evaluate objective values
9:       if offspring is better than main parent then
10:         replace it on population
11:      end if
12:   until population is completed
13:   Identify nondominated solutions in population
14:   Add nondominated solutions into secondary population
15:   Add dominated solutions into third population
16: end for
```

an additional mutation operator in order to allow a better exploration of the search space. We adopted uniform mutation for that sake [8].

As indicated before, our proposed approach uses an external archive (also called secondary population). In order to include a solution into this archive, it is compared with respect to each member already contained in the archive using the $pa\epsilon$-dominance grid [10]. Any member that is removed from the secondary population is included in the third population. The $pa\epsilon$-dominance grid is created once we obtain 100 nondominated solutions. If Phase I is not able to find at least 100 nondominated solutions, then the grid is created until Phase II (if during this second phase it is possible to find at least 100 nondominated solutions). The minimum number of nondominated solutions needed to create the grid is critical in several aspects:

- If we create the grid with just a few points, then the performance of the grid may significantly degrade.
- Once we create the grid, the number of points in this second population significantly decreases, and we have to ensure a minimum number of points that will be used by the Phase II.
- The behavior of the Phase II is a lot better if the grid was created during Phase I, since this ensures that the secondary population has a good distribution of solutions.

An exhaustive set of experiments undertaken by the authors indicated that 100 points was a good compromise to cover the three aspects indicated above.

The third population stores the dominated points needed for the Phase II. Every removed point from the secondary population is included in the third population. If this third population reaches a size of 100 points, a $pa\epsilon$-

dominance grid will be created in order to manage them and thus ensure a good distribution of points.

6.2 Phase II : Local Search using Rough Sets

Upon termination of Phase I, we start Phase II, which departs from the nondominated set generated in Phase I (ES). This set is contained within the secondary population. We also have the dominated set (DS), which is contained within the third population. It is worth remarking that ES can simply be a list of solutions or a $pa\epsilon$-dominance grid, depending on the moment at which the grid is created (if Phase I generated more than 100 nondominated solutions, then the grid will be built during that phase). This, however, does not imply any difference in the way in which the Phase II works.

Algorithm 3 Phase II pseudo-code

```
1: ES ← nondominated set generated by Phase I
2: DS ← dominated set generated by Phase I
3: eval ← 0
4: repeat
5:    Items ← NumEff points ∈ ES &NumDom points ∈ DS
6:    Range Initialization
7:    Compute Atoms
8:    for i ← 0, Offspring do
9:       eval ← eval + 1
10:      ES ← Offspring generated
11:      Add Offspring into ES set
12:   end for
13: until 1000 < eval
```

From the set ES we choose $NumEff$ points previously unselected. If we do not have enough unselected points, we choose the rest randomly from the set ES. Next, we choose from the set DS $NumDom$ points previously unselected (and in the same way if we do not have enough unselected points, we complete them in a random fashion). These points will be used to approximate the boundary between the Pareto front and the rest of the feasible set in decision variable space. What we want to do now is to intensify the search in the area where the nondominated points reside, and refuse finding more points in the area where the dominated points reside. For this purpose, we store these points in the set $Items$ and perform a rough sets iteration:

1. **Range Initialization:** For each decision variable i, we compute and sort (from the smallest to the highest) the different values it takes in the set $Items$. Then, for each decision variable i, we have a set of $Range_i$ values, and combining all these sets we have a (non-uniform) grid in decision variable space.

2. **Compute Atoms:** We compute $NumEff$ rectangular atoms centered in the $NumEff$ efficient points selected. To build a rectangular atom associated to a nondominated point $x^e \in Items$ we compute the following upper and lower bounds for each decision variable i:
   - Lower Bound i: Middle point between x_i^e and the previous value in the set $Range_i$.
   - Upper Bound i: Middle point between x_i^e and the following value in the set $Range_i$.

   In both cases, if there are no previous or subsequent values in $Range_i$, we consider the absolute lower or upper bound of variable i. This setting lets the method to explore close to the feasible set boundaries.
3. **Generate Offspring:** Inside each atom we randomly generate $Offspring$ new points. Each of these points is sent to the set ES (that, as mentioned, can be a $pa\epsilon$-dominance grid) to check if it must be included as a new nondominated point. If any point in ES is dominated by this new point, it is sent to the set DS.

7 Computational Experiments

In order to validate our proposed approach, our results are compared with respect to those generated by the NSGA-II [5], which is a MOEA representative of the state-of-the-art in the area.

The first phase of our approach uses three parameters: crossover probability (Pc), elitism (sel_2) and population size (Pop). On the other hand, the second phase uses three more parameters: number of points randomly generated inside each atom ($Offspring$), number of atoms per generations ($NumEff$) and the number of dominated points considered to generate the atoms ($NumDom$). Finally, the minimum number of nondominated points needed to generate the $pa\epsilon$-dominance grid is set to 100 for all problems.

Our approach was validated using 27 test problems, but due to space constraints, only 9 were included in this chapter: 5 from the **ZDT** set [27] and 4 from the **DTLZ** set [6]. In all cases, the parameters of our approach were set as follows: $Pc = 0.3$, $sel_2 = 0.1$, $Pop = 25$, $Offspring = 1$, $NumEff = 2$ and $NumDom = 10$. The NSGA-II was used with the following parameters: crossover rate = 0.9, mutation rate = 1/num_var (num_var = number of decision variables), $\eta_c = 15$, $\eta_m = 20$, population size = 100 and maximum number of generations = 30. The population size of the NSGA-II is the same as the size of the grid of our approach, in order to allow a fair comparison of results, and both approaches adopted real-numbers encoding and performed 3000 fitness function evaluations per run.

In order to allow a quantitative comparison of results, we adopted the three following performance measures:

Size of the space covered (SSC): This metric was proposed by Zitzler and Thiele [28], and it measures the hypervolume of the portion of the

objective space that is dominated by the set, which is to be maximized. In other words, SSC measures the volume of the dominated points. Hence, the larger the SSC value, the better.

Unary additive epsilon indicator ($I^1_{\varepsilon+}$): The epsilon indicator family has been introduced by Zitzler et al. [29] and comprises a multiplicative and additive version. Due to the fact that the additive version of ϵ-dominance has been implemented in the hybrid algorithm, we decided to use the unary additive epsilon indicator ($I^1_{\varepsilon+}$) as well. The unary additive epsilon indicator of an approximation set A ($I^1_{\varepsilon+}(A)$) gives the minimum factor ϵ by which each point in the real front R can be add such that the resulting transformed approximation set is dominated by A:

$$I^1_{\varepsilon+}(A) = inf_{\epsilon \in \mathbf{R}}\{\forall z^2 \in R \backslash \exists z^1 \in A : z^2_i \leq z^1_i + \epsilon \ \forall i\}.$$

$I^1_{\varepsilon+}(A)$ is to be minimized and a value smaller than 0 implies that A strictly dominates the real front R.

Standard Deviation of Crowding Distances (SDC): In order to measure the spread of the approximation set A, we compute the standard deviation of the crowding distance of each point in A:

$$SDC = \sqrt{\frac{1}{|A|}\sum_{i=1}^{|A|}(d_i - \overline{d_i})^2}$$

where d_i is the crowding distance of the $i-th$ point in A (see [4] for more details of this distance) and $\overline{d_i}$ is the mean value of all d_i. Nevertheless, other types of measures could be use for d_i. Now, $0 \leq SDC \leq \infty$ and the lower the value of SDC, the better the distribution of vectors in A. A perfect distribution, that is $SDC = 0$, means that d_i is constant for all i.

7.1 Discussion of Results

Table 1 shows a summary of our results. For each test problem, we performed 30 independent runs per algorithm. The results reported in Table 1 are the mean values for each of the three performance measures and the standard deviation of the 30 runs performed. The best mean values in each case are shown in **boldface** in Table 1.

It can be clearly seen in Table 1 that our DEMORS produced the best mean values in all cases. The graphical results shown in Figures 3 and 4 serve to reinforce our argument of the superiority of the results obtained by our DEMORS. These plots correspond to the run in the mean value with respect to the unary additive epsilon indicator. In all the bi-objective optimization problems, the true Pareto front (obtained by enumeration) is shown with a continuous line and the approximation obtained by each algorithm is shown with black circles. In Figures 3 and 4, we can clearly see that in the ZDT problems, the NSGA-II is very far from the true Pareto front, whereas our

DEMORS has already converged to the true Pareto front after only 3000 fitness function evaluations. The spread of solutions of our DEMORS is evidently not the best possible, but we argue that this is a good trade-off (and the performance measures back up this statement) if we consider the low computational cost achieved. Evidently, the quality of the spread of solutions is sacrificed at the expense of reducing the computational cost required to obtain a good approximation of the Pareto front.

Our results indicate that the NSGA-II, despite being a highly competitive MOEA is not able to converge to the true Pareto front in most of the test problems adopted when performing only 3000 fitness function evaluations. If allowed a higher number of evaluations, the NSGA-II would certainly produce a very good (and well-distributed) approximation of the Pareto front.

	SSC		$I^1_{\varepsilon+}$		**SDC**	
Function	**D**EMORS	**N**SGA-II	**D**EMORS	**N**SGA-II	**D**EMORS	**N**SGA-II
	Mean σ	**M**ean σ	**M**ean σ	**M**ean σ	**M**ean σ	**M**ean σ
ZDT1	**0.852** 0.001	0.635 0.021	**0.006** 0.001	0.193 0.022	**0.008** 0.004	0.051 0.010
ZDT2	**0.794** 0.014	0.555 0.032	**0.031** 0.036	0.342 0.053	**0.033** 0.026	0.159 0.041
ZDT3	**0.788** 0.002	0.647 0.025	**0.017** 0.006	0.154 0.020	0.091 0.016	**0.073** 0.005
ZDT4	**0.993** 0.002	0.866 0.029	**0.002** 0.001	0.137 0.030	**0.011** 0.012	0.128 0.070
ZDT6	**0.899** 0.002	0.333 0.042	**0.004** 0.002	0.572 0.054	**0.016** 0.030	0.211 0.089
DTLZ1	**0.997** 0.0007	0.996 0.002	**0.023** 0.007	0.046 0.009	0.096 0.013	**0.040** 0.018
DTLZ2	**0.941** 0.0017	0.930 0.004	**0.067** 0.008	0.079 0.015	0.026 0.011	**0.007** 0.007
DTLZ3	**0.996** 0.0006	0.996 0.004	**0.042** 0.018	0.060 0.014	0.110 0.036	**0.043** 0.016
DTLZ4	0.821 0.115	**0.890** 0.032	0.352 0.078	**0.245** 0.038	0.136 0.050	**0.039** 0.010

Table 1. Comparison of results between our DEMORS and the NSGA-II for the ZDT and DTLZ problems adopted. σ refers to the standard deviation over the 30 runs performed.

7.2 Evaluating the Importance of Using Rough Sets

A natural question to ask regarding the use of rough sets in this case is if they really provide an aggregated value to the MOEA adopted. Some may think that the multi-objective extension of differential evolution that we adopted for the first stage of our approach is powerful enough as to converge to the Pareto front of the problems that we studied without any further help. We have argued that this is not the case, but some numerical results may be a more convincing argument. For that sake, we conducted a small experimental study in which we evaluated the outcome produced when applying only the first stage of the algorithm, and then we compared such results with respect to those generated upon applying the second stage. Table 2 shows this comparison of results. The values in **boldface** are the best mean results. By looking at Table 2, one can clearly appreciate that in most cases, and with respect to the three performance measures adopted, the use of rough sets improved (on

average) the performance of the algorithm (mainly with respect to the unary additive epsilon indicator metric).

	SSC		$I^1_{\varepsilon+}$		SDC	
Function	**D**EMORS	**P**hase-I	**D**EMORS	**P**hase-I	**D**EMORS	**P**hase-I
	Mean σ	**M**ean σ	**M**ean σ	**M**ean σ	**M**ean σ	**M**ean σ
ZDT1	**0.852** 0.001	0.849 0.002	**0.006** 0.001	0.020 0.008	**0.008** 0.004	0.031 0.023
ZDT2	0.794 0.014	**0.796** 0.010	0.031 0.036	**0.016** 0.012	**0.033** 0.026	0.035 0.026
ZDT3	**0.788** 0.002	**0.788** 0.002	**0.017** 0.006	0.023 0.007	0.091 0.016	**0.073** 0.009
ZDT4	**0.993** 0.002	0.992 0.002	**0.002** 0.001	0.004 0.003	**0.011** 0.012	0.024 0.020
ZDT6	**0.899** 0.002	0.896 0.002	**0.004** 0.002	0.017 0.005	**0.016** 0.030	0.062 0.036
DTLZ1	**0.997** 0.0007	0.996 0.007	**0.023** 0.007	**0.023** 0.008	0.096 0.013	**0.019** 0.014
DTLZ2	**0.941** 0.0017	0.933 0.002	**0.067** 0.008	0.077 0.015	**0.026** 0.011	0.029 0.015
DTLZ3	**0.996** 0.0006	0.995 0.008	**0.042** 0.018	**0.042** 0.014	0.110 0.036	**0.024** 0.016
DTLZ4	**0.821** 0.115	0.7570 0.108	**0.352** 0.078	0.394 0.038	**0.136** 0.050	0.185 0.056

Table 2. Comparison of results between our DEMORS and the Phase 1 of our algorithm for the ZDT and DTLZ problems adopted. σ refers to the standard deviation over the 30 runs performed.

8 Conclusions and Future Work

We have presented a new technique to improve the results of a MOEA based on a local search mechanism inspired on Rough sets theory. The proposed approach was found to provide very competitive results in a variety of test problems, despite the fact that it performed only 3000 fitness function evaluations. Within this number of evaluations, NSGA-II, a highly competitive MOEA, is not able to converge to the true Pareto front in most of the test problems adopted. This led us to conclude that Rough Sets is a suitable tool to be hybridized with a MOEA in order to improve the local exploration around the nondominated solutions found so far. If the search engine adopted to produce a coarse-grained approximation of the Pareto front is efficient (as in our case), then a good approximation of the true Pareto front can be achieved with a low computational cost.

As part of our future work, we are interested in coupling the local search mechanisms described in this chapter to different search engines. Particularly, we are interested in exploring a hybridization with particle swarm optimization [12], which has also been found to be a very effective search engine in multiobjective optimization.

Acknowledgments

The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the

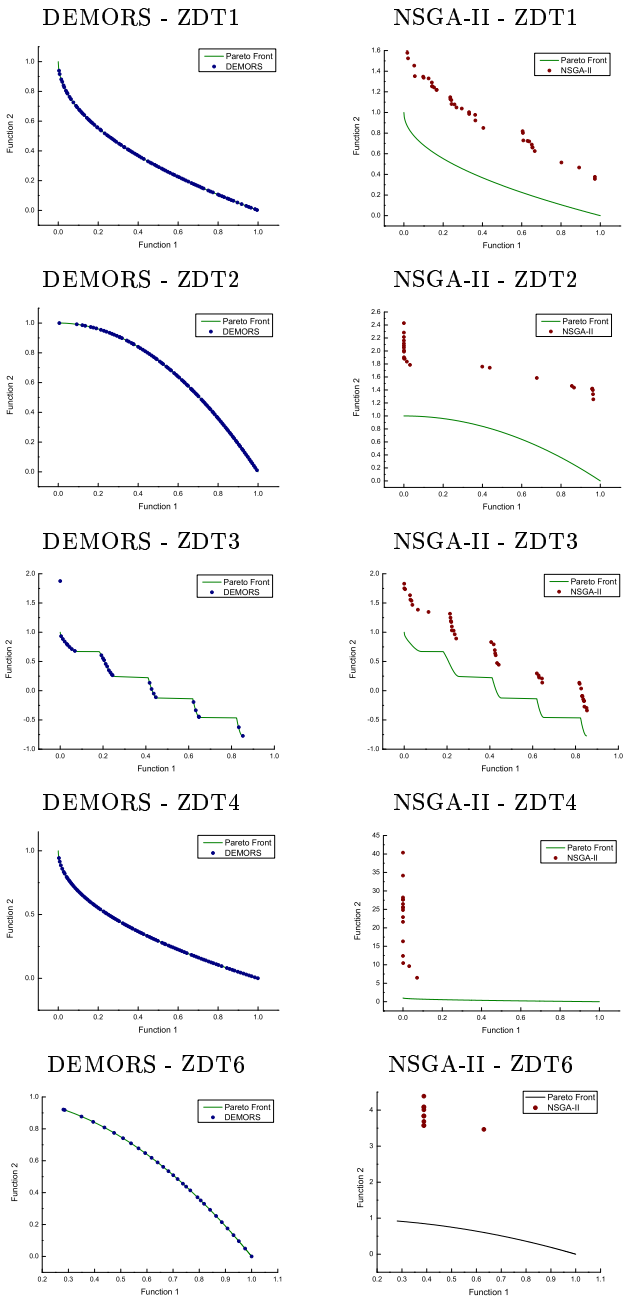

Fig. 3. Pareto fronts generated by DEMORS (left) and NSGA-II (right) for ZDT1, ZDT2, ZDT3 , ZDT4 and ZDT6

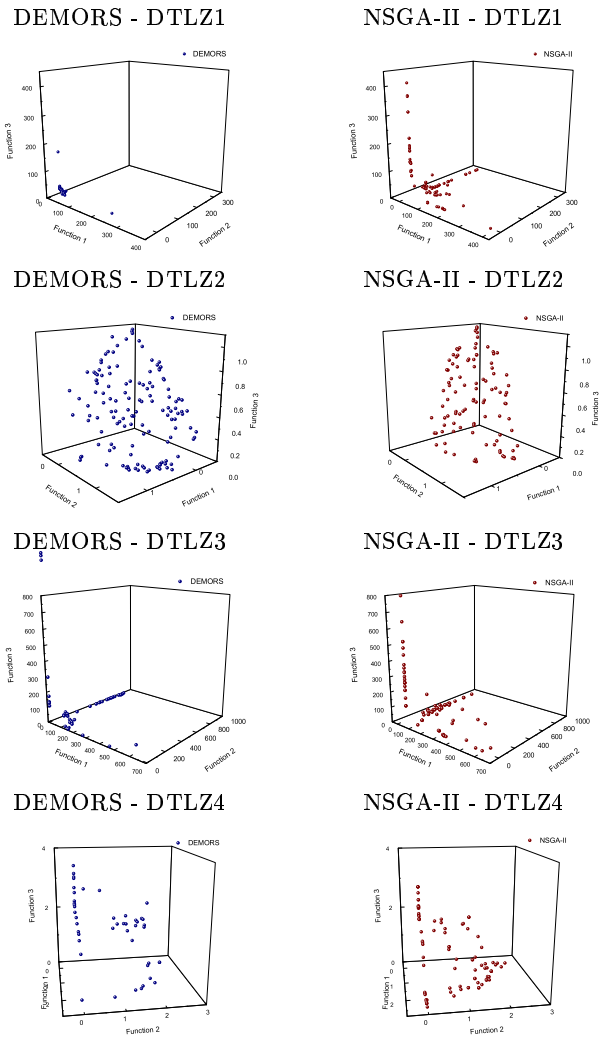

Fig. 4. Pareto fronts generated by DEMORS (left) and NSGA-II (right) for DTLZ1, DTLZ2, DTLZ3 and DTLZ4

Electrical Engineering Department at CINVESTAV-IPN. The third author acknowledges support from CONACyT project number 42435-Y.

References

1. Hussein A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 831–

836, Piscataway, New Jersey, May 2002. IEEE Service Center.
2. B.V. Babu and M. Mathew Leenus Jehan. Differential Evolution for Multi-Objective Optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 4, pages 2696–2703, Canberra, Australia, December 2003. IEEE Press.
3. 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. ISBN 0-3064-6762-3.
4. Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, 2001. ISBN 0-471-87339-X.
5. 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.
6. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In A. Abraham, L. Jain, and R. Goldberg, eds.: *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications.* pp. 105–145, Springer, USA, 2005.
7. Matthias Ehrgott. *Multicriteria Optimization.* Springer, Berlin, second edition, 2005. ISBN 3-540-21398-8.
8. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Co., Reading, Massachusetts, USA, 1989.
9. Alfredo G. Hernández-Díaz, Luis V. Santana-Quintero, Carlos Coello Coello, Rafael Caballero, and Julián Molina. A new proposal for multi-objective optimization using differential evolution and rough sets theory. In *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, Seattle, Washington, USA, July 2006. ACM Press. (accepted).
10. Alfredo G. Hernández-Díaz, Luis V. Santana-Quintero, Carlos A. Coello Coello, and Julián Molina. Pareto adaptive - ϵ-dominance. Technical Report EVOCINV-02-2006, Evolutionary Computation Group at CINVESTAV, México, March 2006.
11. Antony W. Iorio and Xiaodong Li. Solving rotated multi-objective optimization problems using differential evolution. In *AI 2004: Advances in Artificial Intelligence, Proceedings*, pages 861–872. Springer-Verlag, Lecture Notes in Artificial Intelligence Vol. 3339, 2004.
12. James Kennedy and Russell C. Eberhart. *Swarm Intelligence.* Morgan Kaufmann Publishers, California, USA, 2001.
13. Saku Kukkonen and Jouni Lampinen. An Extension of Generalized Differential Evolution for Multi-objective Optimization with Constraints. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 752–761, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.
14. Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.
15. T.Y. Lin. Special issue on rough sets. *Journal of the Intelligent Automation and Soft Computing*, 2(2):*, Fall 1996.
16. Nateri K. Madavan. Multiobjective Optimization Using a Pareto Differential Evolution Approach. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1145–1150, Piscataway, New Jersey, May 2002. IEEE Service Center.

17. Kaisa M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1999.
18. K.E. Parsopoulos, D.K. Taoulis, N.G. Pavlidis, V.P. Plagianakos, and M.N. Vrahatis. Vector Evaluated Differential Evolution for Multiobjective Optimization. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 1, pages 204–211, Portland, Oregon, USA, June 2004. IEEE Service Center.
19. Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(1):341–356, Summer 1982.
20. Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991. ISBN 0-471-87339-X.
21. Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution. A Practical Approach to Global Optimization*. Springer, Berlin, Germany, 2005. ISBN 3-540-29859-6.
22. Tea Robič and Bodgan Filipič. DEMO: Differential Evolution for Multiobjective Optimization. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 520–533, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.
23. Luis Vicente Santana-Quintero and Carlos A. Coello Coello. An algorithm based on differential evolution for multi-objective problems. *International Journal of Computational Intelligence Research*, 1(2):151–169, 2005.
24. Rainer Storn and Kenneth Price. Differential Evolution - A Fast and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.
25. Feng Xue, Arthur C. Sanderson, and Robert J. Graves. Pareto-based Multi-Objective Differential Evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 2, pages 862–869, Canberra, Australia, December 2003. IEEE Press.
26. L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(1):338–353, Fall 1965.
27. Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.
28. Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.
29. Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, Summer 2003.