DEMORS: A hybrid Multi-Objective Optimization Algorithm using Differential Evolution and Rough Set Theory for Constrained Problems

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

[1] CINVESTAV-IPN, Computer Science Department, Av. IPN No. 2508 Col. San Pedro Zacatenco, México D.F. 07360, México
lvspenny@hotmail.com, ccoello@cs.cinvestav.mx

[2] Pablo de Olavide University, Department of Economics, Quantitative Methods and Economic History, Ctra. de Utrera, km.1, 41013, Seville, Spain
agarher@upo.es

[3] University of Málaga, Department of Applied Economics (Mathematics), Campus El Ejido s./n. 29071, Spain
julian.molina@uma.es, rafael.caballero@uma.es

Summary. The aim of this paper is to show how the hybridization of a multi-objective evolutionary algorithm (MOEA) and a local search method based on the use of rough set theory, is a viable alternative to obtain a robust algorithm able to solve difficult constrained multi-objective optimization problems at a moderate computational cost. This paper extends a previously published MOEA [11], which was limited to unconstrained multi-objective optimization problems. Here, the main idea is to use this sort of hybrid approach to approximate the Pareto front of a constrained multi-objective optimization problem while performing a relatively low number of fitness function evaluations. Since in real-world problems the cost of evaluating the objective functions is the most significant, our underlying assumption is that, by aiming to minimize the number of such evaluations, our MOEA can be considered efficient. As in its previous version, our hybrid approach operates in two stages: in the first one, a multi-objective version of differential evolution is used to generate an initial approximation of the Pareto front. Then, in the second stage, rough set theory is used to improve the spread and quality of this initial approximation. To assess the performance of our proposed approach, we adopt, on the one hand, a set of standard bi-objective constrained test problems and, on the other hand, a large real-world problem with 8 objective functions and 160 decision variables. The first set of problems are solved performing 10,000 fitness function evaluations, which is a competitive value compared to the number of evaluations previously reported in the specialized literature for such problems. The real-world

* The fourth author is also associated to the UMI-LAFMIA 3175 CNRS.

problem is solved performing 250,000 fitness function evaluations, mainly because of its high dimensionality. Our results are compared with respect to those generated by NSGA-II, which is a MOEA representative of the state-of-the-art in the area.

Key words: Hybrid algorithms, multi-objective optimization, differential evolution, rough set theory

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 [9, 20].

In recent years, a wide variety of multi-objective evolutionary algorithms (MOEAs) have been proposed in the specialized literature [5, 6, 7]. However, the study of hybrids of MOEAs with other types of techniques is still relatively scarce. This paper presents a study of the combination of a MOEA and a local search method inspired by rough set theory as a viable way of obtaining a good approximation, both in quality and diversity, of the Pareto front of a constrained multi-objective optimization problem. Our main motivation for such a hybrid approach is to reduce the overall number of fitness function evaluations performed to approximate the Pareto front of a problem. For this aim, we consider one of the fastest MOEAs in the literature, differential evolution [26, 31], and improve its performance by using a local search method inspired by rough set theory. This kind of hybrid approach (Differential Evolution and rough set theory) has been already used and tested to solve box-constrained multi-objective optimization problems showing a high performance [11] when compared with highly competitive methods, and this is the main reason to approach its adaptation to constrained problems here. We opted to implement such adaptation also taking into account the increase in the demand of multi-objective solvers for real cases, where most of the problems are (hard) constrained. It is worth noting, however, that in spite of this demand, the number of multi-objective metaheuristics developed with a particular emphasis on reducing the number of objective function evaluations that they perform, is very scarce. Taking into account these last facts, we adapted and tested this hybrid method in order to validate it to be used for constrained multi-objective optimization problems.

The remainder of this paper is organized as follows: Section 2 provides some basic concepts required to understand the rest of the paper. In Section 3, we introduce differential evolution, which is the approach adopted as our search engine. An introduction to rough set theory is provided in Section 4. 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:

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

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

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the vector of decision variables, $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., k$ are the objective functions and $g_j, h_k : \mathbb{R}^n \to \mathbb{R}$, $j = 1, ..., m$, $k = 1, ..., p$ 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 objective vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$, we say that $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for all $i = 1, ..., k$, 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 a **nondominated** or **Pareto-optimal** solution if there does not exist another $\mathbf{x}' \in \mathcal{X}$ such that $\mathbf{f}(\mathbf{x}') \prec \mathbf{f}(\mathbf{x})$.

Definition 3. The **Pareto Optimal Set** $\mathcal{P}^*$ is defined by:

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

Definition 4. The **Pareto Front** $\mathcal{PF}^*$ is defined by:

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

3 Differential Evolution

Differential Evolution (DE) [26, 31] 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

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

been extended to solve multi-objective problems by several researchers (see for example [1, 2, 13, 15, 18, 19, 23, 28, 33]). 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 set theory as a local optimizer in order to improve the spread and convergence of the nondominated solutions obtained by our differential evolution implementation.

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, P, is randomly generated, and then evaluated. Then, the selection process takes place: three parents are chosen to generate a single offspring which competes with a fourth population member 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 the third one. In the context of single-objective optimization, if the resulting point yields a lower objective function value than the fourth element selected, the newly generated vector replaces this individual. In addition, the best individual $X_{best,g}$ is evaluated for every generation g in order to keep track of the progress that is made during the minimization process. Summarizing, for each individual in the population, an offspring is generated, using three randomly generated parents, and compared with a fourth individual in the population, in order to be replaced if this new solution performs better. More formally, the process is described as follows (for a given generation g and a population size $|P|$):

For each vector $x_{i,g}; i = 1, 2, \ldots, |P|$, a trial vector v is generated using:

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

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

The integers r_1, r_2 and r_3 are randomly chosen from the interval $[1, |P|]$ and are different from i. F is a real and constant factor which controls the amplification of the differential variation $x_{r2,g} - x_{r3,g}$. This trial solution v is compared with $x_{i,g}$, and will replace it if it is better.

All details about our differential evolution implementation are provided in Section 6.

4 Rough Set Theory

Rough set 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 [34]. Rough set theory was proposed by Pawlak [24], and presents another attempt to address this problem. Rough set theory has been used by many researchers and practitioners all over the world and has been adopted in many interesting applications. The rough set 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 [25]), special issues of journals (see [17]), 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 (X is a finite set). We want to characterize the set X with respect to R. The way rough set theory expresses vagueness is employing a boundary region of the set X built once we know a finite number of 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).

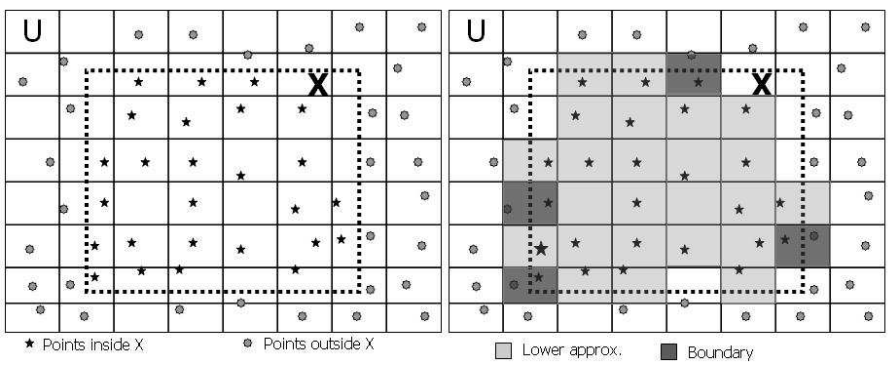

Fig. 1. Rough set approximation

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 implicit grid 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, 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.

4.1 Use of Rough Set Theory in Multi-Objective Optimization

For our MOPs we will try to approximate the Pareto front using a rough set 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. Conversely, 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, we need 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 *nondominated 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 several feasible and infeasible points divided in three sets: the nondominated points (ES), the dominated points (DS), and the infeasible solutions (IS). Using these three 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 can easily use this information to generate better solutions 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, where the black points are dominated solutions, the grey points are infeasible solutions, the white points are nondominated solutions, and the "x" solutions correspond to offspring randomly generated inside the atoms. Thus, it can be seen how the addition of both dominated solutions and infeasible solutions at the time at which the grid is generated avoids that the atoms generated around the nondominated solutions expand towards little promising or undesired regions (i.e., regions that have been already explored and that are occupied by dominated or infeasible solutions).

We must note the importance of the DS and IS sets as in a rough set method the information comes from the description of the boundary of the three sets. The goal is not to have an exhaustive knowledge of the boundary among these three sets (nondominated, dominated and infeasible), but to

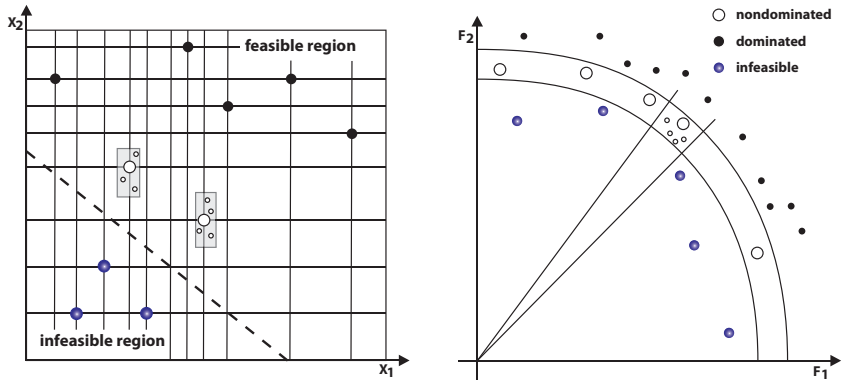

Fig. 2. Decision variable space (left) and objective function space (right)

narrow down the search towards the feasible nondominated solutions in such a way that as the quality of the sets DS and IS improves (in the sense that they only contain nondominated solutions which are well-distributed), the atoms generated for the nondominated solutions become more and more accurate (in other words, the boundary among the three sets becomes smaller). In this way, not considering the two sets DS and IS would cause an important increase in the size of the atoms, which would transform the offspring generation process into something very similar to a random search procedure. 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 of ES, DS, and IS. Moreover, 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, IS 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. For all of the above reasons, both for narrowing down the size of these sets and for improving their quality (quality is measured in terms of nondominance), we use a Pareto-adaptive ϵ-dominance grid, which is described in the next section. Algorithm 1 describes a rough set iteration.

5 Pareto-adaptive ϵ-dominance

One of the concepts that has raised more interest within evolutionary multi-objective 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 [16] is certainly

Algorithm 1 Rough Set Iteration

```
Choose NumEff non-explored points of ES.
Choose NumDom non-explored points of DS.
Choose NumInfea non-explored points of IS.
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 nondominated 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 and send to DS
    end if
    if new is infeasible then
      Include it in IS
    end if
  end for
end for
```

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 $pa\epsilon$-dominance was proposed in [12]. Briefly, the main difference is that in $pa\epsilon$-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 $\epsilon = (\epsilon_1, \epsilon_2, ..., \epsilon_k)$ associated to each $f = (f_1, f_2, ..., f_k) \in R^k$ 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 [12], 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 paper.

6 The Hybrid Method: DEMORS

Our proposed approach, called DEMORS (Differential Evolution for Multi-objective Optimization with local search based on Rough Set theory) [11], 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 performs the greatest effort for obtaining a good Pareto front approximation. For that reason, this phase consumes 65% of the total number of evaluations. During Phase II, a local search procedure based on rough set theory is applied for the remainder 35% of the total number of evaluations, in order to improve the solutions produced during the previous phase. These two values correspond to a balance of 65%-35% empirically derived after an exhaustive number of experiments. 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 [29]. Our approach keeps four populations: the main population (which is used to select the parents), a secondary (external) population, which is used to retain the nondominated solutions found, a third population that retains dominated solutions removed from the second population, and the fourth population that retains the infeasible solutions found so far. Both the size and the quality of these populations are controlled through the $pa\epsilon-$dominance grid.

First, we randomly generate $|P|$ individuals, and use them to generate $|P|$ 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 evaluations is $MaxEval = 10,000$, this means that during the first 6,000 evaluations (60% of $MaxEval$), a random selection will be adopted, and during the last 4,000 evaluations an elitist selection will be adopted. Both selection mechanisms are described next:

1. **Random selection:** 3 parents are randomly selected from the main population.

Algorithm 2 Phase I pseudo-code

```
Initialize vectors of the population P
Evaluate the cost of each vector
eval ← |P|
repeat
  repeat
    Select three different vectors
    Perform crossover using DE scheme
    Perform mutation
    Evaluate objective values
    eval ← eval + 1
    if offspring is better than main parent then
      replace it in population
    end if
  until population is completed
  Identify nondominated solutions in population
  Add nondominated solutions into secondary population
  Add dominated solutions into third population
  Add infeasible solutions into fourth population
until 0.65 · MaxEval <eval
```

2. **Elitist selection:** 3 parents from the secondary population are randomly selected but in such a way that their mutual distance is less than f_{close}, where this parameter is computed as follows:

$$f_{close} = \frac{\sqrt{\sum_{i=0}^{k} \left(f_{i,max} - f_{i,min}\right)^2}}{2^k}$$

where: k = number of objective functions, $f_{i,max}$ = maximum value for the i-th objective function in the secondary population, $f_{i,min}$ = minimum value for the i-th objective function in the secondary population.

The recombination in the Phase I is done as follows: For each parent $p_i; i = 1, 2, \ldots, P$ (P = population), the offspring h is generated using:

$$\begin{cases} h_j = p_{r1,j} + F \cdot (p_{r2,j} - p_{r3,j}), & \text{with probability } p_c; \\ h_j = p_{ref,j}, & \text{otherwise;} \end{cases} \tag{4}$$

where $j = 1, 2, \ldots, n$ (n = number of variables), p_c = crossover probability, $p_{r1}, p_{r2}, p_{r3} \in [1, P]$ are different integer numbers, and $F > 0$. The integers r_1, r_2 and r_3 are the indices for the three randomly selected parents in [1, $|P|$] and ref is the index for the reference parent. F is a constant real number which controls the difference $p_{r2,j} - p_{r3,j}$.

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.

Next, we show the constraint-handling selected when comparing the parent and the offspring generated. It is important to mention that in our approach, we normalize the constraints so that their value ranges between 0 and 1. This normalization is transparent for the user (the algorithm operates without requiring any input from the user). This normalization mechanism is described next: For each constraint $constr[i]$, two different variables $UbC[i]$ and $LbC[i]$ keep track of the maximum and minimum values that each constraint reaches, respectively, throughout the evolutionary process. The values are updated only when the new value produced is either above or below the previously stored value. Therefore, whenever a constraint is required, the following expression is adopted to normalize it ($Norm_constr$ is the normalized constraint value):

$$Norm_constr[i] = \frac{constr[i] - LbC[i]}{UbC[i] - LbC[i]}$$

* *If the parent and the offspring are both infeasible*, the one closest to the feasible region is selected.
* *If the parent is feasible and the offspring is infeasible*, the offspring is selected if and only if the offspring is at a distance of 0.1 from the feasible region[5] and a $flip(0.5)$[6] returns TRUE. Otherwise, the parent is selected.
* *If the parent is infeasible but the offspring is feasible*, the parent is selected if and only if it is at a distance of 0.1 from the feasible region and a $flip(0.5)$ returns TRUE. Otherwise, the offspring is selected.
* *If both are feasible*, they are compared using the classical Pareto dominance relation:
  - *if the parent dominates its offspring*, the parent is selected.
  - *if the offspring dominates its parent*, the offspring is selected.
  - *if both are nondominated with respect to each other*, the one who gets TRUE out of a $flip(0.5)$ is selected.

Differential evolution does not use a 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 an additional mutation operator in order to allow a better exploration of the search space. We adopted uniform mutation for that sake [10].

As indicated before, our proposed approach uses several external archives (ES, DS, and IS). In order to include a solution into any of these archives, such a solution is compared with respect to each member already contained in the archive using the $pa\epsilon$-dominance grid [12]. Any member that is removed from the ES is included in the third population, DS. The $pa\epsilon$-dominance grid is created once we obtain 100 nondominated solutions and the same

[5] Let's keep in mind that the constraints space is normalized between 0 and 1.
[6] $flip(p)$ is a function that returns TRUE with a probability p.

grid is used for the three sets. If Phase I is not able to find at least 100 nondominated solutions, then the grid is not 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 considerably degrade.
- Once we create the grid, the number of points in this population considerably 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 for the benchmark problems considered in this paper and in our previous work [11].

6.2 Phase II : Local Search Based on Rough Set Theory

For the problems of our interest, we will try to approximate the Pareto front using a Rough Set theory grid. In order to do this, we will use an initial approximation of the Pareto front (provided by the first phase algorithm based on DE) and will implement a grid to get more information about the front that will let us improve this initial approximation. We aim to produce a grid that is not so expensive (computationally speaking) but that offers reasonably good knowledge about the Pareto front 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 *nondominated atoms*, we can easily intensify the search over these atoms as they are built in decision variable space. To create this grid, we will have as inputs N feasible points divided in three sets: the nondominated solutions (ES), the dominated solutions (DS) and the infeasible ones. Using these three sets we want to create a grid to describe the set ES in order to intensify the search on it.

We must note the importance of the DS and IS sets as in rough set theory, the information comes from the description of the boundary of the three sets. Then, the more nondominated points provided the better. However, it is also required to provide some dominated and infeasible points, since we need to estimate the boundary between being dominated, infeasible and being nondominated. Once this information is computed, we can simply generate more points in the nondominated side. 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, IS and ES coming from the initial approximation. To ensure these sets are really disjoint we mark each point as explored or non-explored (if it has been used or not to compute a grid) and we do not allow repetitions. Algorithm 3 describes a rough set iteration.

Algorithm 3 Rough set Iteration

```
Input nondominated solutions from the first phase ES.
Input dominated solutions from the first phase DS.
Input Infeasible solutions from the first phase IS.
Output nondominated solutions found by the RS.
eval ← 0
repeat
   Choose NumEff unexplored points of ES.
   Choose NumDom unexplored points of DS.
   Choose NumInf unexplored points of IS.
   Generate NumEff nondominated atoms.
   for i = 0 to NumEff do
      for j = 0 to Offspring do
         Generate (randomly) a point new in atom_i
         eval ← eval + 1
         if new is infeasible then
            Send new to IS
         else
            if new is nondominated then
               Include it in ES
            end if
            if A point old in ES is dominated by new then
               Send old to DS
            end if
         end if
      end for
   end for
until 0.35 · MaxEval < eval
```

7 Computational Experiments

In order to validate our proposed approach, our results are compared with respect to those generated by NSGA-II [8], which is a MOEA representative of the state-of-the-art in the area.

The first phase of our approach uses four parameters: crossover probability (p_c), elitism (sel_2), population size (P), and the amplification value (F). On the other hand, the second phase uses four more parameters: number of points randomly generated inside each atom ($Offspring$), number of atoms per generations ($NumEff$), number of dominated points considered to generate the atoms ($NumDom$), and the number of infeasible points considered to generate the atoms ($NumInfea$). Moreover, the maximum number of fitness function evaluations must be prefixed to $MaxEval$. Finally, the minimum number of nondominated points needed to generate the $pa\epsilon$-dominance grid is set to 100 for all problems (except for the real case, in which the grid is not used).

Our approach is validated using 7 test constrained problems from the benchmark (in all cases, the decision variables have bound constraints): Binh2 (two decision variables and two inequality constraints) [3], Kita (two decision variables and three inequality constraints) [14], Osyczka1 (two decision variables and two inequality constraints) and Osyczka2 (six decision variables and six inequality constraints) [22], Srinivas (two decision variables and two inequality constraints) [30], Tanaka (two decision variables and two inequality constraints) [32] and the Welded Beam problem (four decision variables and four constraints) [27]. All of them, except for Kita, are minimization problems (Kita is a maximization problem). It is worth noting that all of these problems have their constraints active at the Pareto front (i.e., the Pareto front is located exactly in the boundary between the feasible and infeasible regions). The definition of each of these problems, except for the Welded Beam can be found at:

`http://www.cs.cinvestav.mx/~emoobook/apendices/appendix-a.pdf`

These problems have been selected trying to cover all different complexities: convex, non-convex and disconnected Pareto fronts, linear and nonlinear objective functions and constraints or the number of decision variables (from 2 to 6). Since the welded beam problem is not included in the above appendix, its description is provided next:

This problem was originally proposed in [27]. In this problem, the aim is to minimize the cost and the end deflection of the beam subject to constraints on the shear stress in the weld (τ), bending stress in the beam (ρ), buckling load on the bar (P_c, and side constraints). The problem has four decision variables: h, l, t and b:

$$\begin{aligned} \text{minimize} \quad & f_1(\mathbf{x}) = 1.10471h^2l + 0.04811tb(14.0 + l), \\ \text{minimize} \quad & f_2(\mathbf{x}) = \frac{4 \cdot F \cdot L^3}{Et^3b} \end{aligned}$$

subject to:

$$g_1(\mathbf{x}) \equiv \tau_{max} - \tau \geq 0,$$
$$g_2(\mathbf{x}) \equiv \rho_{max} - \rho \geq 0,$$
$$g_3(\mathbf{x}) \equiv b - h \geq 0,$$
$$g_4(\mathbf{x}) \equiv 0.125 - h \geq 0,$$
$$g_5(\mathbf{x}) \equiv P_c - F \geq 0$$

where:
$F = 6,000\ lb \qquad L = 14\ in$
$E = 30e6\ psi \qquad G = 12e6\ psi$
$\tau_{max} = 13,600\ psi \qquad \rho_{max} = 30,000\ psi$
$\alpha = \frac{1}{(3G \cdot t \cdot b^3)}$
$I = \frac{1}{(12 \cdot t \cdot b^3)}$
$P_c = (4013\sqrt{EI\alpha}/L^2) \cdot (1 - (t/2L) \cdot \sqrt{EI/\alpha})$
$\rho = (6FL)/(bl^2)$
$J = 2 \cdot (0.707hl \cdot (l^2/12 + ((h+b)/2)^2)$
$R = \sqrt{l^2/4 + ((h+l)/2)^2}$
$M = F(L + l/2)$
$cost = l/2R$
$\tau'' = MR/J$
$\tau' = F/(\sqrt{2} + hl)$
$\tau = \sqrt{(\tau'^2 + 2\tau'\tau'' cost + \tau''^2)}$

The bounds of the decision variables are the following:

$$0 \leq h, b \leq 2.0$$
$$0 \leq l, t \leq 10.0$$

In all cases, the parameters of our approach were set as follows: $p_c = 0.3$, $sel_2 = 0.5$, $P = 25$, $F = 0.5$, $Offspring = 1$, $NumEff = 2$, $NumDom = 5$, and $NumInfea = 5$. These parameter values are the same that were adopted in [11], except for sel_2, which has been changed from 0.1 to 0.5 due to the difficulty associated with the addition of constraints. This requires that the elitist selection is delayed in order to ensure first the existence of a reasonable number of feasible nondominated solutions before increasing the selection pressure (which speeds up convergence). Also, the maximum number of objective function evaluations is set here in $MaxEval = 10,000$, instead of the 3,000 evaluations adopted in some of our previous work [11]. Again, the presence of constraints requires a higher computational effort than before. NSGA-II was used with the following parameters: crossover rate = 0.9, mutation rate = $1/n$, $\eta_c = 15$, $\eta_m = 20$, population size = 100 and maximum number of generations = 100. The population size of 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 adopt real-numbers encoding and performed 10,000 fitness function evaluations per run.

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

Size of the space covered (SSC): This performance measure was proposed by Zitzler and Thiele [35], 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. The reference points required for this performance measure, were obtained from the optimum values for each of the objectives (considered separately) of each problem. This information was retrieved of the true Pareto fronts generated in order to assess our results.

Unary additive epsilon indicator ($I^1_{\varepsilon+}$): The epsilon indicator family has been introduced by Zitzler et al. [36] and comprises a multiplicative and an 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 added 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. A value smaller than zero would imply that A would strictly dominate the real front R.

Spread (Δ): In order to measure both the spread of the approximation set A and the distances from the extreme points of A to the extremes of the real Pareto front R, we use *Spread* [7]:

$$\Delta = \frac{\sum_{m=1}^{2} d^e_m + \sum_{i=1}^{|A|} |d_i - \overline{d}|}{\sum_{m=1}^{m} d^e_2 + |A| \cdot \overline{d}}$$

where d_i has been taken to be the Euclidean distance of the $i-th$ point in A to the $i+1-th$ point in A (once these points are ranked in ascending order), $\overline{d}$ is the mean value of d_i, d^e_m is the Euclidean distances between the extreme solutions of both fronts corresponding to the $m-th$ objective function ($m = 1, 2$). So, $0 \leq \Delta \leq \infty$ and the lower the value of Δ, the better the distribution of vectors in A. A perfect distribution, that is $\Delta = 0$, means that $d_i = \overline{d}$ for all i and $d^e_m = 0$ for all m (so the extremes of the true Pareto front have been achieved).

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 produces the best mean values in 3 cases (Binh2, Kita and Srinivas) for all the performance measures, although NSGA-II obtains similar results. For the 4 other problems (Osyczka1, Osyczka2, Tanaka and Welded Beam), NSGA-II obtains the best mean values for SSC and $I^1_{\varepsilon+}$ performance measures but DEMORS improves all its Δ values, while results on the first two performance measures are very similar, too. This is, results in SSC and $I^1_{\varepsilon+}$ performance measures are very similar for all the problems, but DEMORS performs better according to the distribution performance measure Δ for all the problems.

	SSC				$I^1_{\varepsilon+}$				Δ			
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
Binh2	**0.808**	0.000	0.806	0.000	**0.007**	0.001	0.013	0.002	**0.452**	0.018	0.492	0.036
Kita	**0.879**	0.002	**0.879**	0.001	**0.013**	0.006	**0.013**	0.003	**0.964**	0.068	1.038	0.060
Osyczka1	0.821	0.084	**0.875**	0.021	0.107	0.099	**0.051**	0.027	**0.716**	0.164	0.816	0.062
Osyczka2	0.946	0.012	**0.963**	0.000	0.034	0.022	**0.009**	0.003	**0.771**	0.287	1.210	0.068
Srinivas	**0.557**	0.000	0.556	0.000	**0.012**	0.003	0.013	0.002	**0.165**	0.015	0.277	0.015
Tanaka	0.718	0.008	**0.749**	0.001	0.021	0.012	**0.012**	0.002	**0.918**	0.115	1.140	0.045
Welded Beam	0.953	0.020	**0.974**	0.003	0.030	0.021	**0.009**	0.003	**0.732**	0.188	0.898	0.133

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

The graphical results shown in Figures 3 and 4 serve to reinforce our argument about convergence. These plots correspond to the run in the median value with respect to the unary additive additive epsilon indicator and, therefore, are not really representative of the average diversity obtained in Table 1. In all the optimization problems, the true Pareto front[7] is shown with a continuous line together with the approximation obtained by each algorithm. In Figures 3 and 4, we can clearly see that both algorithms have already converged to the true Pareto front after only 10,000 fitness function evaluations.

Our results indicate that DEMORS is a competitive MOEA for constrained multi-objective optimization problems when performing only 10,000 fitness function evaluations, and it is also able to ensure a good spread and distribution of the solutions within this reduced number of evaluations.

7.2 Evaluating the Importance of Using Rough Set Theory

A natural question to ask regarding the use of rough set theory in this case is if they really provide an aggregated value to the MOEA adopted. It is

[7] The true Pareto front shown in each case was obtained through an exhaustive search process conducted over a discretized version of each problem.

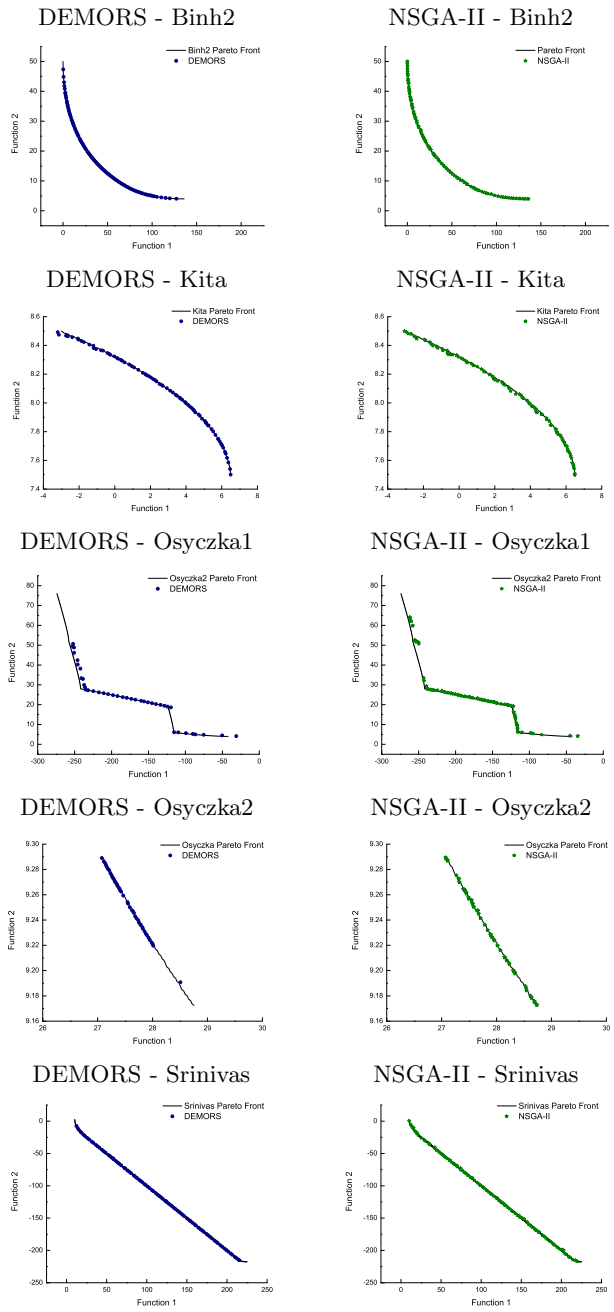

Fig. 3. Pareto fronts generated by DEMORS (left) and NSGA-II (right) for the first five test problems adopted.

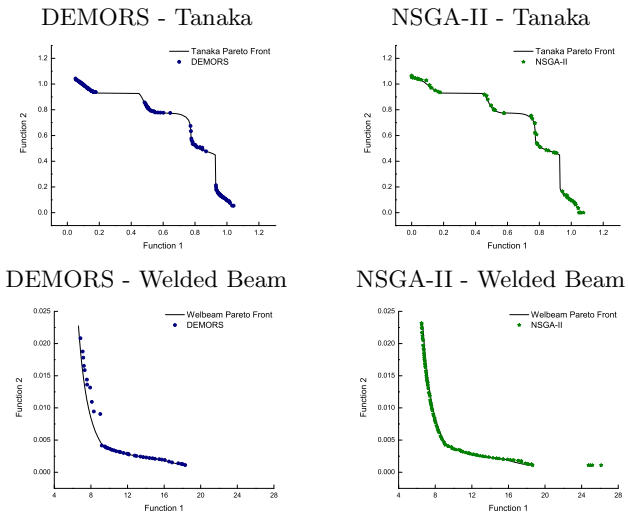

Fig. 4. Pareto fronts generated by DEMORS (left) and NSGA-II (right) for the problems Tanaka and Welded Beam.

reasonable to ask if the multi-objective extension of differential evolution use for the first stage of our approach is powerful enough to converge to the Pareto front without any further help. In order to answer this question we carried out some experiments: we evaluated the outcome produced when applying only the first stage of the algorithm, and then we compared such results with 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 set theory improved the performance of the first phase both in terms of convergence to the true Pareto front and in terms of distribution along it. However, it is worth noting that in the second phase, our approach deteriorates in two cases with respect to spread. This is due to the use of our $pa\epsilon-$dominance grid, which, because of the definition of ϵ-dominance, tends to lose certain points of the Pareto front each time the grid is recomputed (see [12] for further details about this).

7.3 Solving a real case

In order to complement our validation, we also considered a real multi-objective optimization problem related to the Mexican Economy [4]. In this problem, the effects of the public investment in the social and economical development in Mexico are studied. This investment is divided into five different sectors (industry, agriculture, education, health and infrastructure) for each of the 32 states in Mexico. Therefore, we want to obtain values for $5 \times 32 = 160$

	SSC				$I^1_{\varepsilon+}$				Δ			
Function	DEMORS		Phase I		DEMORS		Phase I		DEMORS		Phase I	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
Binh2	**0.797**	0.000	0.796	0.000	0.081	0.001	**0.080**	0.001	0.610	0.013	**0.603**	0.016
Kita	**0.879**	0.002	0.867	0.005	**0.013**	0.006	0.035	0.014	**0.964**	0.068	1.130	0.069
Osyczka1	**0.821**	0.084	0.747	0.123	**0.107**	0.099	0.170	0.131	**0.716**	0.164	1.072	0.170
Osyczka2	**0.946**	0.012	0.838	0.065	**0.034**	0.022	0.187	0.076	0.771	0.287	**0.427**	0.231
Srinivas	**0.557**	0.000	0.555	0.001	**0.012**	0.003	0.018	0.005	**0.165**	0.015	0.245	0.029
Tanaka	**0.718**	0.008	0.672	0.086	**0.021**	0.012	0.099	0.128	**0.918**	0.115	1.207	0.203
Welded Beam	**0.953**	0.020	0.916	0.028	**0.030**	0.021	0.075	0.032	**0.732**	0.188	0.940	0.163

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

decision variables representing the investment for the five different sectors for the 32 states in Mexico. To measure the effects of the investment designed, four criteria are formulated related to four different aspects of this development: sanitary facilities rate, education level rate, children mortality rate and the gross inner product. But one of the main problems currently faced in Mexico is the big gap found among the different states: some of them present very good levels in these four rates and some others fall into poverty and underdevelopment. To try to balance the situation among the 32 states in Mexico, four objectives are added: standard deviation of the sanitary facilities rate among the 32 states, standard deviation of education level rate among the 32 states, standard deviation of children mortality rate among the 32 states and standard deviation of the gross inner product among the 32 states.

As a result, we have a nonlinear multi-objective optimization problem with 8 objective functions, 160 continuous variables and several blocks of constraints, including some upper and lower bounds for investments, as well as some economical constraints related to investment conditions. We solved this problem with three different methods, NSGA-II, DEMORS and SSPMO [21] (this approach was included because it is the one that originally solved this problem). This last approach is a competitive multi-objective metaheuristic, based on scatter search, which has been found to be very competitive over several standard test function sets, as it is shown in [21]. We performed 10 independent runs with each method but, due to the fact that SSPMO has a self-adaptive stopping criterion and performed on average 250,000 fitness function evaluations, we had to use the settings described next. SSPMO was used with the same settings as described in [21]; NSGA-II was used with the same above parameters except for the population size and the maximum number of generations, which were defined in this case as 1000 and 250, respectively. Several of the parameters of DEMORS remained the same as before ($p_c = 0.3$, $sel_2 = 0.5$, $P = 25$, and $F = 0.5$), but others had to be increased due to the high dimensionality of the problem, which requires a grid of better precision in order to generate the atoms produced for Phase II of the algorithm: $Offspring = 10$, $NumEff = 10$, $NumDom = 10$, and $NumInfea = 10$.

Also, DEMORS performed $MaxEval = 250,000$ fitness function evaluations per run in order to obtain a fair comparison of results among the algorithms.

In order to compare results, we used the Size of the Space Covered (SSC), the Coverage of Two Sets performance measures (C), and the standard deviation of crowding distance (SDC) to compare results. The Coverage of Two Sets, $C(A, B)$, [35] measures the proportion of the solutions on the estimated frontier B that are dominated by the solutions on the estimated frontier A, while SDC measures the spread of the approximation set A. To this end, 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 [7] for more details on this distance) and $\overline{d_i}$ is the mean value of all d_i. Nevertheless, other types of measures could be used 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. It is important to mention that the true Pareto front of this problem is unknown, and we decided to execute the DEMORS for a very long period of time until it reached the 2 million of function evaluations. The Pareto front obtained from this experiment, was used as the true Pareto front and as a reference to calculate the SSC metric.

Table 3 summarizes the results for SSC, SDC, and C for this problem. With respect to the C metric, we show the mean and standard deviation of comparing the 10 runs in pairs (e.g. demors_run-1.dat with nsga2_run-1.dat). Since they are independent from each other, it is unnecessary to compare all the runs with respect to each other in order to obtain a single value. This results show how both NSGA-II and DEMORS outperform SSPMO for all the performance measures considered. Comparing DEMORS and NSGA-II only, it is worth mentioning that DEMORS presents the best value for SSC, which measures the convergence to the true Pareto front. On the other hand, NSGA-II obtains the best SDC value (this is because NSGA-II uses crowding distance to preserve diversity while the paϵ-dominance grid is inactive in DEMORS due to the high number of objective functions of this problem). And finally, DEMORS outperforms NSGA-II for the coverage measure: DEMORS dominates on average to 17.88% of the results produced by NSGA-II while NSGA-II only dominates, on average, to 0.13% of the results generated by DEMORS. This is, performance of DEMORS for this real world problem is also competitive with respect to NSGA-II and SSPMO.

With respect to the computational time that is required by the algorithms to achieve 250,000 fitness function evaluations, we found the following:[8] 1) DE-

[8] In the benchmark problems, the CPU times were too small to allow a reasonable comparison.

	SSC		SDC		C (σ)		
Algorithm	Mean	σ	Mean	σ	**DEMORS**	**NSGA-II**	**SSPMO**
DEMORS	**0.446**	0.0360	0.0041	0.0011	-	**0.1788** (0.0481)	0.9705 (0.0180)
NSGA-II	0.271	0.0517	**0.0035**	0.0002	0.0013 (0.0020)	-	**0.9775** (0.0139)
SSPMO	0.0	0.0	0.0176	0.0047	0 (0)	0 (0)	-

Table 3. Comparison of results between our DEMORS, NSGA-II and SSPMO for the real Mexican problem. σ refers to the standard deviation over the 10 runs performed.

MORS takes 990 seconds 2) NSGA-II takes 550 seconds and 3) SSPMO takes 225 seconds. However, it is worth noting that SSPMO has a self-adaptive stopping criterion, which makes it difficult to compare (in terms of CPU time or total number of evaluations performed) with respect to the other two MOEAs in a fair way. Additionally, SSPMO has a very poor performance, when compared to the other two approaches. Thus, we will focus our analysis only on DEMORS and NSGA-II.

Clearly, when performing the same number of evaluations (and considering a negligible cost for evaluating the objective functions), DEMORS, being a hybrid approach, is more expensive (computationally speaking) than NSGA-II. However, the question remains of which MOEA can achieve a better performance given a certain number of evaluations. This was precisely the focus of our study. For that sake, we adopted the SSC performance measure, which was applied for both MOEAs at intervals of 10,000 objective function evaluations, until reaching 250,000 evaluations. Figure 5 shows the corresponding values in a graphical form. There, it can be clearly seen that DEMORS obtained better results most of the time, since the beginning of the search.

In a second experiment, we stopped both MOEAs after a fixed amount of time: 1,000 seconds. After that time, DEMORS performed about 250,000 objective function evaluations, and NSGA-II performed about 500,000 evaluations. However, as can be seen in Figure 5, even with this number of evaluations (which is twice the number of evaluations performed by our DEMORS), NSGA-II went from an SSC value of 0.174 (achieved with 250,000 evaluations) up to a value of 0.271. This is still considerably lower than the SSC value of 0.446 achieved by our DEMORS with 250,000 evaluations. This led us to conclude that the quality of the search performed by our DEMORS is higher than that of NSGA-II. Thus, if dealing with problems that have objective functions that are very expensive to evaluate (e.g., in aeronautical engineering), this better search quality may translate into important savings in terms of CPU time.

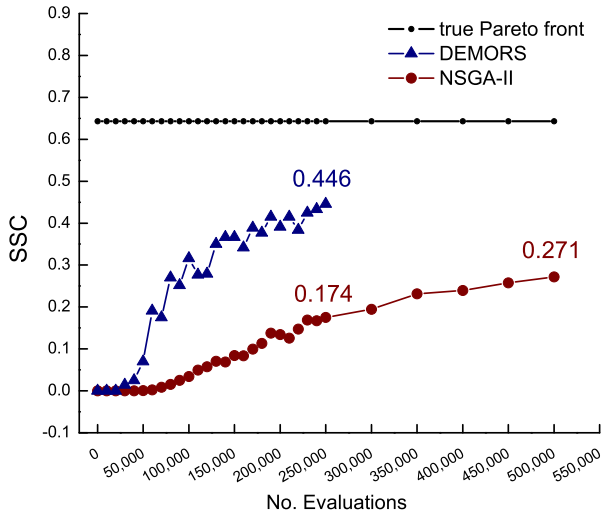

Fig. 5. Graphical representation of the values of the SSC performance measure at certain numbers of iterations performed by two MOEAs: DEMORS and NSGA-II.

8 Conclusions

We have presented a new hybrid multi-objective optimization algorithm for constrained MOPs based on the use of a fast differential evolution algorithm and a local search engine based on rough set theory. The proposed approach was found to provide very competitive results with respect to other previous approaches, in a variety of bi-objective test problems, in spite of the fact that it performed only 10,000 fitness function evaluations. Within this number of evaluations, both DEMORS and NSGA-II (which is a highly competitive MOEA), were able to converge to the true Pareto front in most of the test problems adopted. Both algorithms were also compared (performing 250,000 evaluations) in a real application with 8 objective functions and 160 decision variables. In this case, DEMORS found better results with respect to most of the performance measures considered.

This led us to conclude that the hybridization of a fast MOEA with a properly designed local search engine can be a suitable tool. If the search engine adopted to produce a coarse-grained approximation of the Pareto front is nondominated (as in our case), then a good approximation of the true Pareto front can be finally achieved with an extra small additional cost (35% of the total number of evaluations) by using our local optimizer based on rough set theory.

As part of our future work, we are interested in validating our proposed scheme with other MOPs (including more real-world applications). We also want to refine the hybridization scheme, such that a larger reduction in the total number of evaluations can be achieved. Additionally, we aim to reduce the number of parameters that need to be set by the user. We would like to have an approach that uses self-adaptation mechanisms in order to fine-tune by itself its parameters during its execution.

Acknowledgments

The authors thank the two anonymous reviewers for the comments which greatly helped them to improve the contents of this paper.

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Department at CINVESTAV-IPN. The fourth author acknowledges support from CONACyT project number 45683-Y.

References

1. Hussein A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)*, vol. 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. T. T. Binh and U. Korn, MOBES: A multiobjective evolution strategy for constrained optimization problems, In *The Third International Conference on Genetic Algorithms (Mendel 97)*, 176–182, Brno, Czech Republic, 1997.
4. B. Cobacho. *Planificación de la inversión pública federal en México mediante técnicas de análisis multicriterio* PhD. Dissertation, University of Cartagena, Spain,2007.
5. C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.
6. C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems, Second Edition*. Springer, 2007. ISBN: 978-0-387-33254-3.
7. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001. ISBN 0-471-87339-X.
8. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.
9. M. Ehrgott. *Multicriteria Optimization*. Springer, Berlin, second edition, 2005. ISBN 3-540-21398-8.

10. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts, USA, 1989.
11. A. G. Hernández-Díaz, L. V. Santana-Quintero, C. Coello Coello, R. Caballero, and J. Molina. A new proposal for multi-objective optimization using differential evolution and rough set theory. In *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, Seattle, Washington, USA, July 2006. ACM Press.
12. A. G. Hernández-Díaz, L. V. Santana-Quintero, C. A. Coello Coello, and J. Molina. Pareto adaptive - ϵ-dominance. *Evolutionary Computation*, 15(4):493–517, Winter 2007.
13. A. W. Iorio and X. 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.
14. H. Kita, Y. Yabumoto, N. Mori, and Y. Nishikawa. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In *H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, Parallel Problem Solving from Nature–PPSN IV*, 504–512. Springer-Verlag. Lecture Notes in Computer Science No. 1141, Berlin, Germany, September 1996.
15. S. Kukkonen and J. Lampinen. An Extension of Generalized Differential Evolution for Multi-objective Optimization with Constraints. In *Parallel Problem Solving from Nature - PPSN VIII*, 752–761, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.
16. M. Laumanns, L. Thiele, K. Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.
17. T.Y. Lin. Special issue on rough sets. *Journal of the Intelligent Automation and Soft Computing*, 2(2):*, Fall 1996.
18. N. 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.
19. E. Mezura-Montes, M. Reyes-Sierra, and C. A. Coello Coello. Multi-Objective Optimization using Differential Evolution: A Survey of the State-of-the-Art. In Uday K. Chakraborty, editor, *Advances in Differential Evolution*, pages 173–196. Springer, Berlin, 2008. ISBN 978-3-540-68827-3.
20. K. M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1999.
21. J. Molina, M. Laguna, R. Martí and R. Caballero. SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization. *Informs Journal on Computing*, Vol. 19, No. 1, pp. 91–100, January 2007.
22. A. Osyczka and S. Kundu, A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm, *Structural Optimization 10*, 94–99, 1995.
23. 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.
24. Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(1):341–356, Summer 1982.

25. Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991. ISBN 0-471-87339-X.
26. K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution. A Practical Approach to Global Optimization*. Springer, Berlin, Germany, 2005. ISBN 3-540-29859-6.
27. K. M. Ragsdell and D. T. Phillips, Optimal Design of a Class of Welded Structures Using Geometric Programming, *Journal of Engineering for Industry Series B*, **98**, 1021–1025, 1975.
28. T. Robič and B. 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.
29. L. Vicente Santana-Quintero and C. A. Coello Coello. An algorithm based on differential evolution for multi-objective problems. *International Journal of Computational Intelligence Research*, 1(2):151–169, 2005.
30. N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation, 2(3):221–248, fall 1994.
31. R. Storn and K. Price. Differential Evolution - A Fast and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.
32. M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino,GA-Based Decision Support System for Multicriteria Optimization, *Proceedings of the International Conference on Systems, Man, and Cybernetics 2*, 1556–1561, IEEE Piscataway, NJ, 1995.
33. F. Xue, A. C. Sanderson, and R. 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.
34. L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(1):338–353, Fall 1965.
35. E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.
36. E. Zitzler, L. Thiele, M. Laumanns, C. 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, April 2003.