A New Proposal for Multi-Objective Optimization using Differential Evolution and Rough Sets Theory

Alfredo G. Hernández-Díaz
Dept. of Quantitative Methods
Pablo de Olavide University
SPAIN
agarher@upo.es

Luis V. Santana-Quintero
CINVESTAV-IPN
Computer Science Section
MÉXICO
lvspenny@hotmail.com

Carlos Coello Coello
CINVESTAV-IPN
Computer Science Section
MÉXICO
ccoello@cs.cinvestav.mx

Rafael Caballero
Department of Applied Economics (Mathematics)
University of Málaga
SPAIN
rafael.caballero@uma.es

Julián Molina
Department of Applied Economics (Mathematics)
University of Málaga
SPAIN
julian.molina@uma.es

ABSTRACT

This paper presents a new multi-objective evolutionary algorithm (MOEA) based on differential evolution and rough sets theory. The proposed approach adopts an external archive in order to retain the nondominated solutions found during the evolutionary process. Additionally, the approach also incorporates the concept of $pa\epsilon$-dominance to get a good distribution of the solutions retained. The main idea of the approach is to use differential evolution (DE) as our main search engine, trying to translate its good convergence properties exhibited in single-objective optimization to the multi-objective case. Rough sets theory is adopted in a second stage of the search in order to improve the spread of the non-dominated solutions that have been found so far. Our hybrid approach is validated using standard test functions and metrics commonly adopted in the specialized literature. Our results are compared with respect to the NSGA-II, which is a MOEA representative of the state-of-the-art in the area.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization*

General Terms

Algorithms

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

Keywords

Multi-objective optimization, differential evolution, rough sets theory, hybrid algorithms

1. INTRODUCTION

Most real-world problems involve the simultaneous optimization of two or more (often conflicting) objectives. The solution of such problems (called "multi-objective") is different from that of a single-objective optimization problem. The main difference is that multi-objective optimization problems normally have not one but a set of solutions which are all equally good.

In the past, a wide variety of MOEAs have been reported in the specialized literature [3]. However, from the several types of MOEAs currently available, relatively few adopt DE [19] as their main search engine. The main motivation for using DE as a search engine is its proved success as a (single-objective) optimizer in continuous search problems within the last few years [16]. DE has 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. However, when extended to multi-objective problems, DE tends to be very good for coarse optimization, but not so efficient for fine-grain optimization. In other words, it can converge relatively fast to the vicinity of the true Pareto front of a problem, but may take a lot of computational effort to actually reach such front. On the other hand, rough sets theory can be useful at finding solutions within the neighborhood of a reference set. Thus, our motivation for combining it with DE relies on the potential of rough sets theory as a local optimizer that can improve the approximation of the Pareto front produced by a DE-based MOEA. Our main research goal is to use this hybrid approach to reduce the total number of fitness function evaluations. As we will see later on, our hybrid approach only performs 3000 fitness function evaluations while still producing reasonably good approximations of the true Pareto fronts of a wide variety of test functions (27 test functions

were adopted for the experimental study reported in this paper, although only 9 of them were included due to space limitations). As far as we know, this is the lowest number of fitness function evaluations ever reported for any DE-based MOEA that relies on Pareto ranking. There is only one MOEA that uses a lower number of fitness function evaluations [10], although it is not designed to solve a generic MOP but only some special cases where evaluations are very expensive (computationally speaking).

2. DIFFERENTIAL EVOLUTION

Differential Evolution [19] is a relatively recent heuristic designed to optimize problems over continuous domains. 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 $(\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$.

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 [21]. Rough sets theory was proposed by Pawlak [15], 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.

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. 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. 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. 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, the more complex the problem becomes. What we want to do know is to use a rough sets approach to approximate the Pareto optimal set of a multi-objective problem. 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 will intensify the search over these atoms. Note however, that at this point we will face the following problem:

- The more precise the grid is, the higher its computational cost to manage it.
- The less precise the grid is, the less knowledge we get about the Pareto optimal set.

In this paper, we will describe how to generate a grid representing a good balance for these two aspects: in other words, a grid that is not too expensive (computationally speaking) but that offers a reasonably good knowledge about the Pareto optimal set. Once this grid is available, it becomes relatively straightforward to generate more points on the *efficient atoms*, as these atoms are built in decision variable space.

4. 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 [12] 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 [7].

In order to overcome some of these limitations, the concept of *pa*ϵ-dominance was proposed in [7]. 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 $f = (f_1, f_2, ..., f_m) \in \mathbb{R}^m$ 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.

Each Pareto front (that we will assume normalized : $0 \leq f_i \leq 1$ for any i) will be associated to one curve of the following family

$$\{x^p + y^p = 1 : 0 \leq x, y \leq 1, 0 < p < \infty\}.$$

for bi-objective optimization problems, or

$$\{x^p + y^p + z^p = 1 : 0 \leq x, y, z \leq 1, 0 < p < \infty\}$$

for three-dimensional problems. These families have the following property: For $p > 1$, the curve (or surface) is concave and the bigger the p value the longer the almost horizontal (and almost vertical) parts of the front; and, for $p < 1$, the curve (surface) is convex and the lower the p value the longer the almost horizontal (and almost vertical) stretches in the front. Finally, for $p = 1$ we get the linear front $x + y = 1$ and this approach exactly matches ϵ-dominance.

In order to decide the value of p, it is required to have an initial approximation of the Pareto front, denoted by F. This approximation will determine which value of p fits better to our front. This is, it uses F to be a model in which the p-curve should fit. Evidently, the number of nondominated points included in F is critical for the final performance of the approach, because if the value of p is not appropriate, then the grid will not work properly and a much lower number of solutions than expected will be retained. Obviously, the higher the number of nondominated points in F, the better performance of the grid generated. On the other hand, if we want to maintain the diversity properties of ϵ-dominance, we should generate the first grid as early as possible.

To compute the value of p, it is required to determine the area (hypervolume) under the polygonal line (surface) formed by points in F. Once this area is known, we estimate the value of $p \in (0, +\infty)$ by means of an interpolation process. We choose p when the area under $x^p + y^p = 1$ is as similar to the hypervolume as desired (this precision is set beforehand). Once the p value is estimated and the number T of desired points in the grid is known, we compute the sizes of the boxes for each objective $i \in \{1, 2, ..., m\}$, this is, the vector $\epsilon^i = \left(\epsilon_1^i, \epsilon_2^i, ..., \epsilon_T^i\right)$, using geometric sequences.

5. PREVIOUS RELATED WORK

There have been several recent proposals to extend Differential Evolution to multi-objective optimization. The most representative of them are briefly described next:

- **PDE** [1]: It handles only one (main) population. Reproduction is undertaken only among nondominated solutions, and offspring are placed into the population if they dominate the main parent. A distance metric relationship is used to maintain diversity.
- **PDEA** [13]: It combines DE with key elements from the NSGA-II [5] such as its nondominated sorting and ranking selection procedure.
- **MODE** [20]: It uses a variant of the original DE, in which the best individual is adopted to create the offspring. Also, the authors adopt $(\mu + \lambda)$ selection, Pareto ranking and crowding distance in order to produce and maintain well-distributed solutions.
- **DE for MO Optimization** [2]: This algorithm uses the single-objective DE strategy with an aggregating function to solve bi-objective problems. A single optimal solution is obtained after N iterations using both a *Penalty Function Method* and the *Weighing Factor Method* [4] to optimize a single value.
- **VEDE** [14]: It is a parallel, multipopulation DE approach, which is based on the Vector Evaluated Genetic Algorithm (VEGA) [18].
- **GDE** [11]: GDE extends the selection operator of the basic DE algorithm in order to be able to handle constrained multi-objective optimization problems. The authors report that the performance of GDE is similar to the NSGA-II, but they claim that their approach requires a lower CPU time.
- **NSDE** [8]: It is a simple modification to the NSGA-II [5] where the real-coded crossover and mutation operators of the NSGA-II are replaced with the DE scheme.
- **DEMO** [17]: It combines the advantages of DE with the mechanisms of Pareto-based ranking and crowding distances sorting. In DEMO, the newly created candidates immediately take part in the creation of the subsequent candidates.

None of the DE-based MOEAs reported so far in the specialized literature that have been tested with standard test functions and metrics perform less than 20000 fitness function evaluations. In fact, some of them perform over 50000 fitness function evaluations in test problems of high dimensionality such as the ZDT functions of 30 decision variables adopted in this paper.

6. OUR PROPOSED APPROACH

Our proposed approach, called DEMORS (Differential Evolution for Multiobjective Optimization with Random Sets), 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. These two values (2000 and 1000 fitness function evaluations, which correspond to a balance of 65%-35%) were 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 1. 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.

Algorithm 1 Algorithm of the Phase I

```
Initialize vectors of the population P
Evaluate the cost of each vector
for i = 0 to G do
  repeat
    Select (randomly) three different vectors
    Perform crossover using DE scheme
    Perform mutation
    Evaluate objective values
    if offspring is better than main parent then
      replace it on population
    end if
  until population is completed
  Identify nondominated solutions in population
  Add nondominated solutions into secondary population
  Add dominated solutions into third population
end for
```

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 are described next:

1. **Random Selection:** 3 different parents are randomly selected from the main population.
2. **Elitist Selection:** 3 different parents are selected from the secondary population. It is required that these 3 parents are close from each other. If no set of 3 individuals exists that fulfills this requirement, then a set of 3 individuals is randomly selected from the secondary population. The expression adopted to determine closeness is the following:

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

where:
FUN = number of objective functions
$X_{i,max}$ = upper bound of the i-th objective function in the secondary population
$X_{i,min}$ = lower bound of the i-th objective function in the secondary population

Recombination in Phase I is performed using the following procedure: For each parent vector $\overrightarrow{p_i}$; $i = 0, 1, 2, \ldots, P-1$ (P = population), the offspring vector $\overrightarrow{h}$ is generated as:

$$(x \in U(0,1)) \begin{cases} h_j = p_{r1,j} + F \cdot (p_{r2,j} - p_{r3,j}), & \text{if } x < p_c; \\ h_j = p_{ref,j}, & \text{otherwise.} \end{cases} \quad (1)$$

where: $j = 0, 1, 2, \ldots, var - 1$ (var = number of decision variables for each solution vector). p_c = crossover probability, $p_{r1}, p_{r2}, p_{r3} \in [0, P-1]$, are integers and mutually different, and $F > 0$. Integers r_1, r_2 and r_3 are the indexes of the selected parents randomly chosen from the interval $[0, N-1]$ and ref is the index of the reference parent. F is a real and constant factor which controls the amplification of the differential variation $p_{r2,j} - p_{r3,j}$.

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

Once a child has been generated, it is compared with respect to the reference parent, against which it competes in order to determine who passes to the following generation. The rules of comparison between a child and its parent are the following:

- *if parent dominates child*, the parent is chosen
- *if child dominates parent*, the child is chosen
- *if both are nondominated with respect to each other*, perform a *flip* (0.5) to determine who passes to the following generation.

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 [7]. 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 decreases a lot, 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 a candidate to be included in the third population. This is, this third population is also managed as a

Pareto front, and a point is finally included in this population if it is not dominated in this set and any point dominated by this new candidate is removed from the set. As it is done with the second 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 ensure a good distribution of points.

6.2 Phase II : Local Search using Rough Sets

Upon termination of Phase I (2000 fitness function evaluations), 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. The pseudo-code of phase II is shown in Algorithm 1.

Algorithm 2 Algorithm of the Phase II

```
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 following the procedure shown in [7]. If any point in ES is dominated by this new point, it is sent to the set DS.

Offspring are generated randomly inside each efficient atom as the key point of the process is the fact that these atoms are located in the most promising areas and then there's no need to design an strategy to decide where to generate this offspring inside the atom. On the other hand, we ensure some diversity by using this random feature. This process is applied during 1000 fitness function evaluations, i.e., until 1000 new individuals are generated.

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 paper: 5 problems from the **ZDT** set [22] 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 [23], 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. [24] 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 can be added such that the resulting transformed approximation set is dominated by A:

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

$I^1_{\varepsilon+}(A)$ 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 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.

8. 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 most cases. Regarding SSC, there was only one case in which the NSGA-II outperformed our approach. Regarding the unary additive epsilon indicator, our DEMORS outperformed the NSGA-II in all cases.

Finally, with respect to SDC, the NSGA-II outperformed our approach only in two cases. This is certainly remarkable if we consider the fact that the SDC metric relies on the behavior of the crowded comparison operator of the NSGA-II. Thus, it was expected that the NSGA-II would be favored by this performance measure. The graphical results shown in Figures 1 and 2 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 1 and 2, we can clearly see that in problems ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6, 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. This behavior is less evident in the plots corresponding to DTLZ1, DTLZ2, DTLZ3 and DTLZ4, mainly due to the difficulties of displaying the results in 3 dimensions. However, our DEMORS also outperforms the NSGA-II in the DTLZ test problems. 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, 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. However, our aim was precisely to provide an alternative approach that could require a lower number of evaluations than a state-of-the-art MOEA while still providing a highly competitive performance. Such an approach could be useful in real-world applications with objective functions requiring a very high evaluation cost (computationally speaking).

9. CONCLUSIONS AND FUTURE WORK

We have presented a new hybrid between a MOEA based on differential evolution and a local search mechanism based 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. This is remarkable if we consider that some of the test problems adopted have up to 30 decision variables and that no other MOEA had previously reported results for such a low number of fitness function evaluations as the one used in this paper.

Our comparison of results indicates that our approach clearly outperforms the NSGA-II, which is one of the most competitive MOEAs known to date. These results, although preliminary, seem to indicate that our approach could be a viable alternative for real-world applications in which each evaluation of the fitness function is very expensive (computationally speaking). In such applications, we can afford sacrificing a good distribution of solutions for the sake of obtaining a reasonably good approximation of the Pareto front with a low number of evaluations.

As part of our future work, we intend to improve the performance of the differential evolution algorithm adopted, by exploring alternative differential evolution models and operators. Additionally, we aim to test our algorithm in some real-world problem to see if the good performance that it has shown with the benchmark adopted in this paper can be extrapolated to a practical application. Finally, we are also interested in coupling the local search mechanisms described in this paper to different search engines. Particularly, we are interested in exploring a hybridization with particle swarm optimization [9], which has also been found to be a very effective search engine in multiobjective optimization.

	SSC				$I_{\epsilon+}^1$				SDC			
Function	**D**EMORS		**N**SGA-II		**D**EMORS		**N**SGA-II		**D**EMORS		**N**SGA-II	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
ZDT1	**0.8544**	0.0057	0.6524	0.0198	**0.0248**	0.0117	0.1850	0.0208	**0.0363**	0.0228	0.0506	0.0103
ZDT2	**0.7398**	0.0262	0.4524	0.0392	**0.0519**	0.0694	0.4210	0.0656	**0.0508**	0.0381	0.1594	0.0414
ZDT3	**0.81**	0.0019	0.6827	0.0220	**0.0197**	0.0123	0.1409	0.0189	**0.0684**	0.0096	0.0726	0.0050
ZDT4	**0.9915**	0.0027	0.8658	0.0286	**0.0033**	0.0027	0.1370	0.0297	**0.0146**	0.0227	0.1285	0.0710
ZDT6	**0.92**	0.0051	0.4541	0.0324	**0.0066**	0.0060	0.4379	0.0411	**0.0458**	0.0566	0.2117	0.0918
DTLZ1	**0.9959**	0.0017	0.9938	0.0007	**0.0501**	0.0115	0.0897	0.0113	**0.0202**	0.0130	0.0498	0.0189
DTLZ2	0.8727	0.0037	**0.8949**	0.0054	**0.0792**	0.0091	0.0831	0.0177	0.0316	0.0162	**0.0086**	0.0007
DTLZ3	**0.9943**	0.0026	0.9908	0.0017	**0.0770**	0.0208	0.1329	0.0163	**0.0345**	0.0523	0.0558	0.0142
DTLZ4	**0.9276**	0.1152	0.8760	0.0354	**0.1165**	0.1625	0.2501	0.0461	0.0511	0.0690	**0.0417**	0.0205

Table 1: Comparison of results between our approach (called DEMORS) and the NSGA-II for the nine test problems adopted. The best values are in boldface. σ refers to the standard deviation over the 30 runs performed.

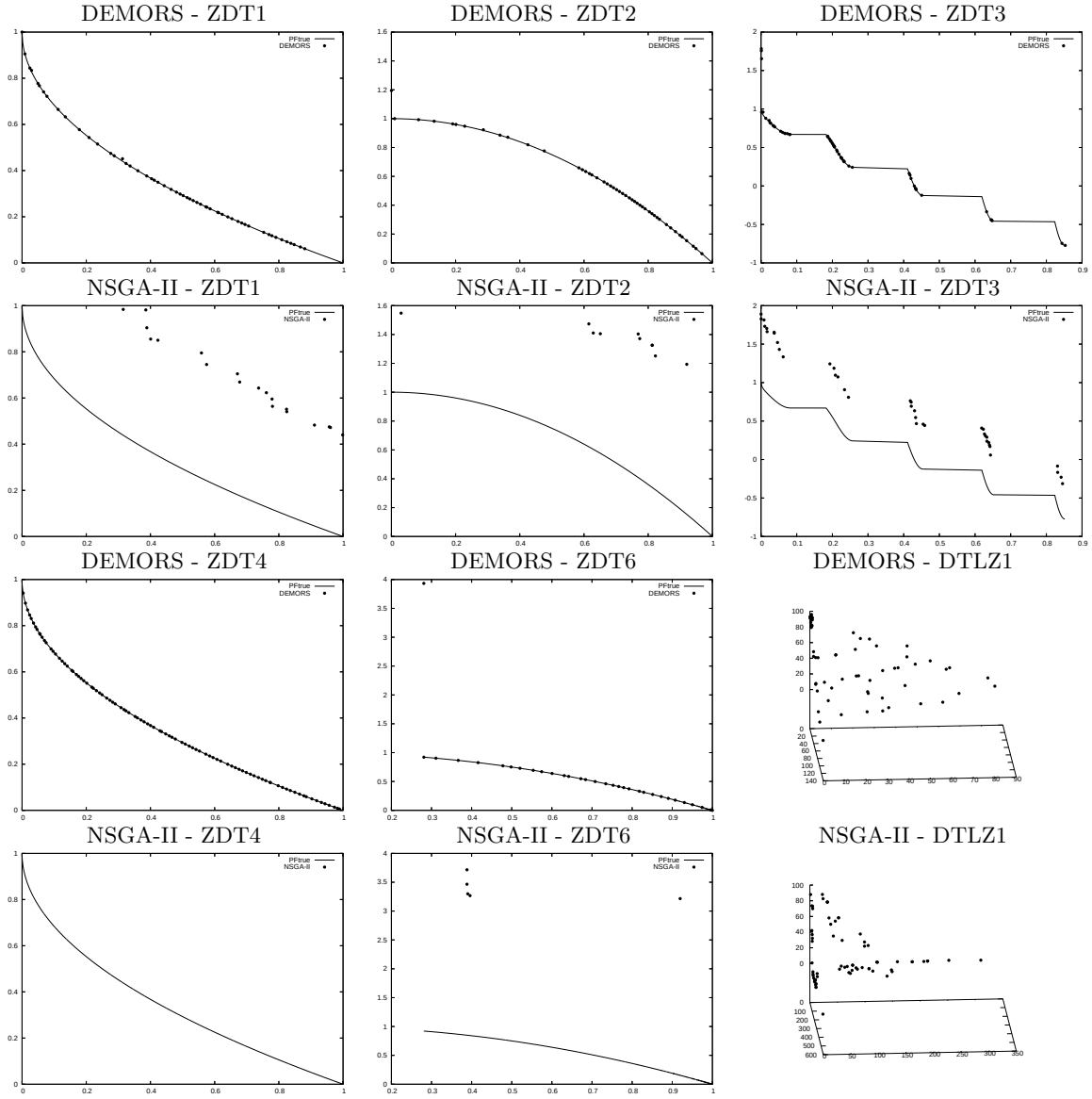

Figure 1: Pareto fronts generated by DEMORS and NSGA-II for ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 and DTLZ1

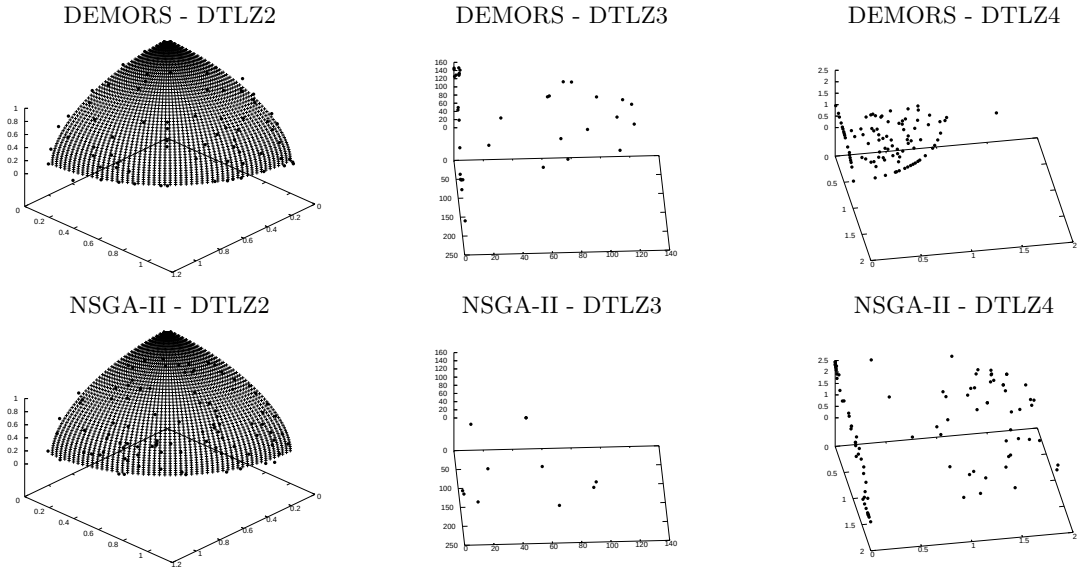

Figure 2: Pareto fronts generated by DEMORS and NSGA-II for DTLZ2, DTLZ3 and DTLZ4

Acknowledgements

The second author acknowledges support from CONACyT for granting him a scholarship to pursue graduate studies at CINVESTAV-IPN. The third author acknowledges support from CONACyT project number 45683-Y.

10. REFERENCES

[1] H. 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. Babu and M. M. L. Jehan. Differential Evolution for Multi-Objective Optimization. In *CEC'2003*, volume 4, pages 2696–2703, December 2003. IEEE Press.

[3] 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.

[4] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001. ISBN 0-471-87339-X.

[5] 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.

[6] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In A. Abraham et al. editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer, USA, 2005.

[7] A. G. Hernández-Díaz, L. V. Santana-Quintero, C. A. Coello Coello, and J. Molina. Pareto adaptive - ϵ-dominance. Technical Report EVOCINV-02-2006, Evolutionary Computation Group at CINVESTAV, México, March 2006.

[8] 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, LNAI Vol. 3339, 2004.

[9] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, California, USA, 2001.

[10] J. Knowles and E. J. Hughes. Multiobjective optimization on a budget of 250 evaluations. In *Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 176–190, March 2005. Springer-Verlag. LNCS Vol. 3410.

[11] 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*, pages 752–761, September 2004. Springer-Verlag. LNCS Vol. 3242.

[12] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.

[13] N. K. Madavan. Multiobjective Optimization Using a Pareto Differential Evolution Approach. In *CEC'2002*, volume 2, pages 1145–1150, May 2002. IEEE Service Center.

[14] K. Parsopoulos, D. Taoulis, N. Pavlidis, V. Plagianakos, and M. Vrahatis. Vector Evaluated Differential Evolution for Multiobjective Optimization. In *CEC'2004*, volume 1, pages 204–211, June 2004. IEEE Service Center.

[15] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(1):341–356, Summer 1982.

[16] K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution : A Practical Approach to Global Optimization*. Springer-Verlag, 2006. ISBN 3-540-20950-6.

[17] T. Robič and B. Filipič. DEMO: Differential Evolution for Multiobjective Optimization. In C. A. Coello Coello, et al., editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pp. 520–533, March 2005. Springer. LNCS Vol. 3410.

[18] J. D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[19] 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.

[20] 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)*, Vol. 2, pp. 862–869, December 2003. IEEE Press.

[21] L. Zadeh. Fuzzy sets. *Information and Control*, 8(1):338–353, Fall 1965.

[22] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[23] 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.

[24] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, Summer 2003.