Epsilon-Constraint with an Efficient Cultured Differential Evolution

Ricardo Landa Becerra
Evolutionary Computation Group at
CINVESTAV-IPN (EVOCINV)
Electrical Eng. Dept., Computer Science Section
Av. IPN No. 2508 Col. San Pedro Zacatenco,
México D.F. 07300, MEXICO
rlanda@computacion.cs.cinvestav.mx

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

ABSTRACT

In this paper we present the use of a previously developed single-objective optimization approach, together with the ε-constraint method, to provide an approximation of the Pareto front in a multiobjective optimization problem. This approximation is usually very near of the true Pareto front, but its cost grows with the desired number of points in the output set. As an alternative, it is possible to generate only a few points, and execute a second phase which will generate intermediate points, to increase the size of the output set. We use a rough sets-based approach for this second phase, which is a very robust approach. The results of this two-phase approach are very competitive in hard multiobjective problems, and is less expensive than the ε-constraint method alone. This approach is very effective on hard multiobjective problems, where is able to find good approximations of the Pareto front with less funtion evaluations than other approaches, as the NSGA-II (against which is compared).

Categories and Subject Descriptors

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

General Terms

Algorithms, design

Keywords

Multi-objective optimization, ε-constraint, cultural algorithms, differential evolution

1. INTRODUCTION

Evolutionary multi-objective optimization consists of using evolutionary algorithms to solve problems with two or more (often conflicting) objective functions. This research area has become very popular in the last few years [1]. Concurrently, more challenging problems have been integrated into the most recent benchmarks, some of which require a considerably high number of objective function evaluations in order to be solved, or can even make current algorithms to fail in their efforts to generate the true Pareto front [6].

The ε-constraint method [5] is a mathematical programming technique, which transforms a multi-objective optimization problem into several constrained single-objective problems. This method has not been used too often in the evolutionary multi-objective optimization literature, due to the fact that it does not generate a set of nondominated solutions in a single run, as most multi-objective evolutionary algorithms (MOEAs) do. Moreover, it has been found that this method is relatively expensive when solving "easy" multi-objective problems, because of the several single-objective optimizations that need to be performed in order to generate the Pareto front. Nevertheless, and despite its disadvantages, we argue that the ε-constraint approach can be a very effective choice under certain conditions. For example, in [9] is shown that a hybrid of the ε-constraint with a carefully designed evolutionary algorithm can be adopted as a viable MOEA when dealing with very difficult two-objective optimization problems, which state-of-the-art MOEAs such as the NSGA-II cannot solve even when performing a very high number of fitness function evaluations. In this paper, we extend the work of [9] by introducing a mechanism that allows to solve problems with three or more objectives, and a further hybridization with rough sets, which are used as a local search algorithm. The resulting approach has a much more affordable computational cost, while still solving very difficult multi-objective optimization problems. The single-objective optimizer adopted by our hybrid with the ε-constraint method consists of a differential evolution-based cultural algorithm, which improves the optimization process by means of domain information extracted during the evolutionary search. Since the ε-constraint method requires an execution of the single-objective optimizer to obtain each point of the Pareto front, we propose to reduce the high computational cost associated with this process by generating only a few points, which are then processed by another approach able to diversify them such that the entire Pareto front can be covered. Obviously, this diversification approach, which acts as a local search algorithm, should be computationally affordable, so that the

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

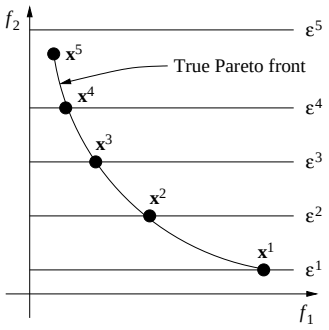


Figure 1: Generating different solutions with the ε-constraint method

total cost of this MOEA is reasonably low. As indicated before, in this paper, we propose the use of rough sets as our diversification method.

2. THE EPSILON-CONSTRAINT METHOD

The ε-constraint method is a multi-objective optimization technique, proposed by Haimes et al. [5], for generating Pareto optimal solutions. It makes use of a single-objective optimizer which handles constraints, to generate one point of the Pareto front at a time. For transforming the multi-objective problem into several single-objective problems with constraints it uses the following procedure (assuming minimization for all the objective functions):

$$\begin{array}{ll} \text{minimize} & f_l(\mathbf{x}) \\ \text{subject to} & f_j(\mathbf{x}) \leq \varepsilon_j \quad \text{for all } j = 1, 2, \ldots, m,\ j \neq l, \\ & \mathbf{x} \in S \end{array}$$

where $l \in \{1, 2, \ldots, m\}$ and S is the feasible region, which can be defined by any equality and/or inequality constraint. The vector of upper bounds, $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m)$, defines the maximum value that each objective can have. In order to obtain a subset of the Pareto optimal set (or even the entire set, in case this set is finite), one must vary the vector of upper bounds along the Pareto front for each objective, and perform a new optimization process for each new vector. The generation of different points of the Pareto front using different values of the upper bound is illustrated in Figure 1.

For any nonlinear multi-objective optimization problem, the solution of an ε-constraint problem yields a weakly Pareto optimal solution [5]. A true Pareto optimal solution can be obtained either if the solution is unique, or if the optimizations are done for all the objectives before reporting the solution [10]. However, to improve the speed of the generation of solutions, and specially if the single-objective optimizer is a metaheuristic, only one optimization per point can be performed to obtain an approximation of the Pareto optimal set.

It is worth mentioning that the ε-constraint method only guarantees to obtain Pareto optimal points if the single-objective optimizer can find the global optimum of the ε-constraint problem, which is not possible for a general nonlinear problem in polynomial time. To the best of our knowledge, the only attempt to hybridize the ε-constraint method with an evolutionary algorithm is the approach called CMEA [15]. This approach performs the intermediate optimizations using a standard evolutionary algorithm. To reduce the computational cost of each independent optimization, the final population of one optimization process is used as the initial population for the next one; however, the authors noted the lack of diversity of the approach and proposed a high mutation rate at the beginning of each process. The authors provide no further details about the mechanism adopted to handle the constraints in the single-objective optimizer.

In [7], the authors proposed an extension of CMEA for three-objective problems. However, the algorithm seems to be unable to find the extreme points of the Pareto front itself, since they are provided *a priori* to the algorithm. Regarding the number of fitness function evaluations needed for CMEA to obtain good results, in [7], the authors mention that they perform 500,000 evaluations for solving three-objective knapsack problems.

3. CULTURED DIFFERENTIAL EVOLUTION

Instead of using an exact method to solve the constrained optimization problems derived from the ε-constraint method, our proposed approach is designed to use an efficient evolutionary technique, which is able to approximate the global minimum of constrained optimization problems requiring a relatively low computational cost.

The cultured differential evolution is a cultural algorithm [16] based on differential evolution [14], designed to solve nonlinear constrained optimization problems. In previous experiments [8], this algorithm exhibited a very good performance, obtaining competitive results when compared to other state-of-the-art evolutionary optimization techniques, but requiring only a fraction of their fitness function evaluations. This is because of the use of domain knowledge, extracted during the evolutionary process, to efficiently guide the search. Next, we will briefly describe this approach.

Cultural algorithms consist of two main components: (1) The *population space*, which consists of a set of possible solutions to the problem, and can be modeled using any population based technique, and (2) the *belief space*, which is the information repository in which the individuals can store their experiences for the other individuals to learn them indirectly; it may be composed by several *knowledge sources*. Our proposed approach uses differential evolution in the population space [14]. A pseudo-code of our approach is shown in Algorithm 1.

In the initial steps of the algorithm, a population of $popsize$ individuals, $\mathbf{x}^j, j = 1, \ldots, popsize$, is created; each individual contains the n parameters of the problem, $\mathbf{x}^j = (x_1^j, \ldots, x_n^j)$. An initial belief space is also created. For the offspring generation, the variation operator of differential evolution is modified by the $influence()$ function of a knowledge source, but the parameters CR and F of the standard differential evolution are also required. Since we want to solve constrained optimization problems, the objective function by itself does not provide enough information as to guide the search properly. To determine if a child is better than its parent, and, therefore, if it can replace it, we use the following rules: 1. A feasible individual is better than an infeasible one. 2. If both are feasible, the individual with the best objective function value is better. 3. Otherwise, the individual with less amount of constraint violation is considered better. The amount of constraint violation

Algorithm 1 Pseudo-code of the cultured differential evolution

```
Generate initial population of size popsize
Initialize the belief space
repeat
  for each individual j in the population do
    Randomly select a knowledge source k_s from the belief space
    Generate a random integer i_rand ∈ (1, n)
    for each parameter i do
      x_i^{j'} = { influence(k_s)   if rand(0,1) < CR or i = i_rand
                 { x_i^j            otherwise
    end for
    Replace x^j with the child x^{j'}, if x^{j'} is better
  end for
  Update the belief space
until the termination condition is achieved
```

is measured with normalized constraints, with the use of the following expression: $viol(\mathbf{x}) = \sum_{c=1}^{C} \frac{g_c(\mathbf{x})}{g_{maxc}}$ where $g_c(\mathbf{x})$ with $c = 1, \ldots, C$ are the constraints of the problem, and g_{maxc} is the largest violation found for the constraint $g_c(\mathbf{x})$ so far.

In our approach, the belief space is divided into 4 knowledge sources:

Situational Knowledge: consists of the best exemplar found along the evolutionary process. Its infuence function modifies the direction of the variation operator to follow the leader.

Normative Knowledge: contains the intervals for the decision variables where good solutions have been found, to move new solutions towards them, through the use of its influence function.

Topographical Knowledge: creates a kind of map of the fitness landscape of the problem during the evolutionary process. It consists of a set of cells, and the best individual found on each cell. The topographical knowledge has an ordered list of the best cells, based on the fitness value of the best individual on each of them. Its influence function moves newly generated individuals towards the best cells.

History Knowledge: was originally proposed for dynamic objective functions [17]. It records in a list, the location of the best individual found before each environmental change. In our approach, instead of detecting changes of the environment, we use it to escape from local optima.

At the beginning, all the knowledge sources have the same probability to be applied, but during the evolutionary process, the probability of applying each knowledge source is updated according to its success rate.

4. ROUGH SETS

The rough sets are an approach designed to deal with imperfect (i.e., uncertain) knowledge. In this paper, we adopt rough sets to generate nondominated points in less populated areas, obtaining a denser approximation of the Pareto front. The Rough sets theory was originally proposed by Pawlak [12, 13]. The basics of this approach are presented next.

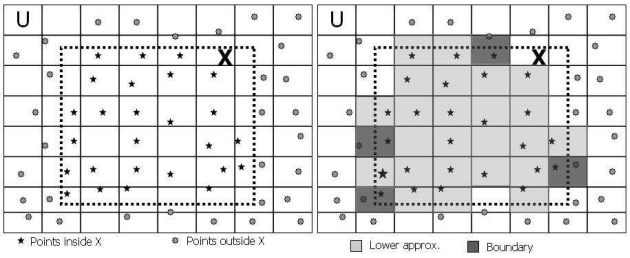


Figure 2: Rough sets approximation

Let us assume that we are given a set of objects U called the *universe* and an indiscernibility relation $R \subseteq U \times U$, representing our lack of knowledge about elements of U (in our case, R is simply an equivalence relation based on a grid over the feasible set; this is, just a division of the feasible set in (hyper)-rectangles). Let X be a subset of U. We want to characterize the set X with respect to R. The way rough sets theory expresses vagueness is employing a boundary region of the set X built once we know points both inside X and outside X. If the boundary region of a set is empty it means that the set is *crisp*; otherwise, the set is *rough* (inexact). A nonempty boundary region of a set means that our knowledge about the set is not enough to define the set precisely (see Figure 2).

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 2). 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. 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 to obtain "small" grids with the maximum precision possible. These two aspects are called **Density** and **Quality** of the grid. If q is the number of criteria (in our case, the number of objectives), Q_i is the i-th criteria, b_j^i is the j-th value of the i-th criteria (we assume these value are ordered increasingly), then:

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

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

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

4.1 Use of Rough Sets in Multi-Objective Optimization

Our main purpose for using rough sets in our work is that this technique can be utilized as a local search approach that takes as input a small set of points that lie either on

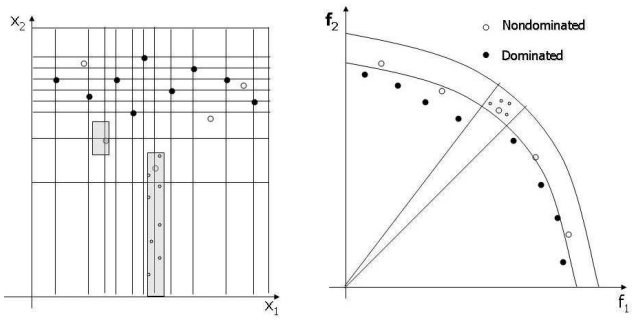


Figure 3: Decision variable space (left) and objective function space (right)

the Pareto front, or very close to it, and returns a larger set of points that approximate the entire Pareto front.

Rough sets use a grid to distribute the points that the algorithm receives as input, and the precision of such grid is critical to its performance: a high precision provides high-quality results, but at a high computational cost, and vice-versa. Thus, reaching a good balance in the design of the rough sets grid is an important issue. To create this grid, as an input we will have N feasible points divided in two sets: the nondominated points (ES) and the dominated ones (DS). Using these two sets we want to create a grid to describe the set ES in order to intensify the search on it. This is, we want to describe the Pareto front in decision variable space because then we could easily use this information to generate more efficient points and then improve this initial approximation. Figure 3 shows how information in objective function space can be translated into information in decision variable space through the use of a grid.

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

Since the computational cost of managing the grid increases with the number of points used to create it, we will try to use just a few points. However, such points must be as far from each other as possible, because the better the distribution the points have in the initial approximation the less points we need to build a reliable grid. On the other hand, in order to diversify the search, we build several grids using different (and disjoint) sets DS and ES coming from the initial approximation. To ensure these sets are really disjoint we will mark each point as explored or non-explored (if it has been used or not to compute a grid) and we will not allow repetitions. Algorithm 2 describes a Rough Sets iteration.

Algorithm 2 Rough Sets Iteration

```
1: Choose NumEff non-explored points of ES.
2: Choose NumDom non-explored points of DS.
3: Generate NumEff efficient atoms.
4: for i = 0 to NumEff do
5:   for j = 0 to Offspring do
6:     Generate (randomly) a point new in atom i and
       send to ES
7:     if new is efficient then
8:       Include in ES
9:     end if
10:    if A point old in ES is dominated by new then
11:      Send old to DS
12:    end if
13:    if new is dominated by a point in ES then
14:      Remove new
15:    end if
16:  end for
17: end for
```

5. OUR PROPOSED APPROACH

First, we will describe the hybridization of the ε-constraint method with the cultured differential evolution, which represents the first stage of our approach.

As our proposed approach is designed to deal with real-valued problems which are likely to have a continuous Pareto front, the ε_j must vary from the best to the worst value for the objective j, *i.e.* the search must move from the ideal to the nadir objective vector. The estimation of the ideal objective vector involves individual optimizations of one objective at a time. On the other hand, the estimation of the nadir objective vector is a more difficult task [10]. Only for the two-objective case, there exists a simple method that can provide a good estimation, which is called the *payoff table*. As we also want to solve problems with three or more objectives, a more sophisitcated mechanism is required. In this work we use a technique based on the approach in [2], which is a modification of the crowding mechanism of the NSGA-II. The modification consists of emphasizing the generation of nondominated solutions near to the edges of the Pareto front, and not only the extreme values. In this work, we perform the estimation using a standard differential evolution approach, incorporating the new ranking rules before the selection procedure. The decision of using a differential evolution algorithm is due to the speed required, because this is only the first step of the process, and the estimation obtained will be relaxed in the next steps.

In the following, let's assume that the procedure $nadir\_st(\mathbf{f}, g)$ performs the modified differential evolution previously described for g generations. It will return two arrays, **lb** and **ub** with the estimated ideal and nadir objective vectors (assuming minimization).

The single-objective optimizer, in which our method is based, is the cultured differential evolution previously described. Let's now assume that it is available as the procedure
$cde(f_l, \varepsilon, g)$, which performs the optimization process of the ε-constraint method during g generations and returns the best point found. The pseudo-code of the ε-constraint with CDE (εCCDE) is shown in the Algorithm 3.

In Algorithm 3, the lower and upper bounds, **lb** and **ub**, are increased by a tolerance **t**; this is done since the results of the *nadir_st* procedure are only approximations, and it is possible to find a better point outside of them. We use $t_j = 0.1(ub_j - lb_j)$. The ε values are updated with a **delta**, which is dependent of the number of points in the Pareto front desired by the user or the decision maker. It is obtained as follows: $\delta_j = \frac{ub_j - lb_j}{p}$. This way, we aim that the

Algorithm 3 ε-Constraint with CDE.

```
P = ∅
(lb, ub) = nadir_st(f, g)
ub = ub + t, lb = lb − t
∀j ∈ (2, ..., m), ε_j = lb_j + δ_j
while ε_m ≤ ub_m do
  x = cde(f_1, ε, g)
  if x is nondominated with respect to P then
    P = P − {y ∈ P | x ≻ y}
    P = P ∪ {x}
  end if
  ε_2 = ε_2 + δ_2
  for j = 2 to m − 1 do
    if ε_j > ub_m then
      ε_j = lb_j
      ε_{j+1} = ε_{j+1} + δ_{j+1}
    end if
  end for
end while
```

final points are equally spaced in their projection over the f_2 to f_m axis. g is an input parameter of the algorithm, but it is very important, because together with p and the population size of the *cde* procedure, *popsize*, define the total number of fitness function evaluations required for the approach. The number of fitness function evaluations is $p^{m-1} \cdot g \cdot popsize$. Algorithm 3 shows f_1 as the objective to be optimized, and f_2 to f_m as the constraints. However, one can interchange the roles of the objectives if the problem looks harder to solve in the original setting. In the experiments shown in this paper, the original setting was always preserved, and f_1 was always taken as the objective to optimize, to allow a fair comparison. In order to improve the performance of each optimization process, the algorithm shares a percentage of the population, in the initial population of the next process. This helps because the problems to be solved are very similar, and the only change is the upper bound of the objective functions which is treated as a constraint. When all the population is shared, the loss of diversity leads to premature convergence. In practice, we found that a small percentage (around 10%) of the population to be shared is enough to improve convergence without losing diversity. According to our previous experiments [9], the εCCDE approach provides good results by itself, but it can result computationally expensive when many points of the Pareto front are required, because it needs an individual optimization process for each point. At this point, we propose to use rough sets theory to spread the few points obtained earlier, an then cover a larger area of the Pareto Front. To achieve this objective, in addition to the final nondominated points provided by the ε-constraint approach, we need some dominated points obtained from the intermediate optimizations (these points are randomly chosen). These sets of points provide enough information to the rough sets to enhance the approximation of the Pareto front.

6. COMPARISON OF RESULTS

In order to validate the performance of the proposed approach, some test functions have been taken from the specialized literature. One may think that the several single-objective optimizations required may give rise to a prohibitively high computational cost, which is unnecessary considering that a modern MOEA may produce a similar approximation of the Pareto front at a much more affordable computational cost. There are problems, however, where this is not the case, and in which a modern MOEA cannot converge to the true Pareto front even if we do not restrict the number of evaluations performed. It is precisely in those cases for which we believe that our approach can be a viable alternative. In order to validate our hypothesis we looked specifically for hard multi-objective problems within the existing benchmarks. Our search led us to the use of two recent benchmarks, proposed by Huband et al. [6] and Okabe et al. [11], where we found very hard problems (WFG1, WFG2, WFG9, OKA1 and OKA2). In the case of WFG problems, each of them has 24 variables. WFG1 is strongly biased toward small values of the first 4 variables, WFG2 is non-separable and has also a disconnected Pareto front, and WFG9 is a deceptive problem. In the case of OKA problems, they only have 2 and 3 variables and 2 objectives, but the geometry of their optimal sets is nonlinear, and they are also strongly biased to the opposite side of the Pareto front. We also use four ZDT problems [18], because they have been used frequently in the specialized literature, and they constitute a reference point for many researchers in the field. We decided to compare results with respect to the NSGA-II [3], since this is an approach representative of the state-of-the-art in the area. It is also known, that the NSGA-II performs very well in the ZDT problem set.

6.1 Experimental Setup

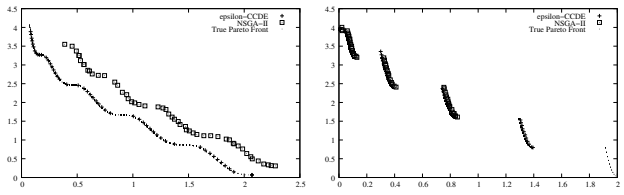
6.1.1 εCCDE alone

In this case, and due to space constraints, we only use the WGF problems, with two objectives. We ran both algorithms during 250,000, 25,000 and 50,000 fitness function evaluations each (for WFG1, WFG2 and WFG3, respectively). We aimed to obtain a set of 50 points as a result of each run, so we adapted the parameters according to that. For the εCCDE, the parameters adopted were: $p = 120$, 50 and 50 for each problem, $g = 48$, with 10% of the population shared between optimizations (this 10% is chosen at random). For the *cde* procedure we used $popsize = 40$, 20 and 10 for each problem, $F = 0.7$, $CR = 0.5$. The population size of NSGA-II was set to 52, and the number of generations to 4808, 962 and 481 for each problem. The rest of the parameteres were set as recommended by its authors: probability of crossover = 0.9, probability of mutation = $1/n$, the value of the distribution index for crossover = 15, and the value of the distribution index for mutation = 20. It is worth mentioning that, in the case of WFG1, even with this large number of iterations, the NSGA-II was not able to reach the true Pareto front. In Figure 4, we show the results of a single run for each test problem.

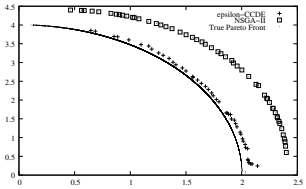
6.1.2 εCCDE+MORS

We ran both algorithms during 15,000 fitness function evaluations each[1] (except for OKA2 and WFG1). We aimed to obtain a set of 100 points as a result of each run, so we adapted the parameters according to that. For the εCCDE, the parameters adopted were, for the two objective problems: $p = 5$, $g = 100$, with 10% of the population shared

[1]Note that this number of evaluations includes those required to estimate the ideal vector.



(a) WFG1 (b) WFG2



(c) WFG9

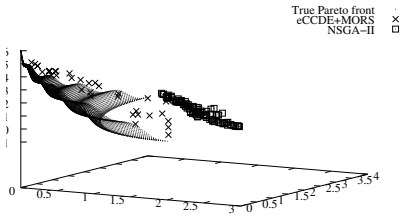
Figure 4: Results produced by our εCCDE and the NSGA-II on the two-objective WFG test problems.

between optimizations (this 10% is chosen at random), for the *cde* procedure we used $popsize = 20$, $F = 0.7$, $CR = 0.5$; and for the three objective problems: $p = 3$, $g = 70$ and $popsize = 16$ (the parameters must be different because the number of evaluations depends of the number of objectives m). The maximum number of evaluations performed by the rough sets algorithm was set to 5,000. The population size of the NSGA-II was set to 100, and the number of generations to 150. The rest of the parameteres were set as recommended by the NSGA-II's authors: probability of crossover = 0.9, probability of mutation = $1/n$, the value of the distribution index for crossover = 15, and the value of the distribution index for mutation = 20.

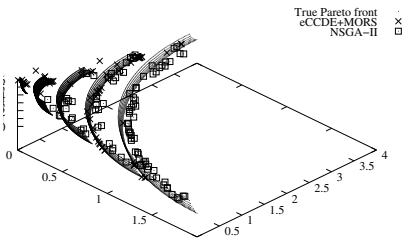
Only for OKA2 and WFG1, the total number of fitness function evaluations was increased to 25,000, because these are really difficult problems. In this case, we adopted, for the two objective problem (OKA2): $g = 150$, and for the three objective problem (WFG1): $g = 104$. In both cases the maximum number of evaluations of the rough sets algorithm was set to 10,000. The number of generations of the NSGA-II was changed in this case to 250, to allow a fair comparison. However, even with this large number of iterations, the NSGA-II was not able to reach the true Pareto front of WFG1, and in the case of OKA2, a very small portion of the Pareto front was covered. In Figures 5, 6 and 7, we show the results of the run on the median with respect to the two set coverage metric (CS) for each test problem adopted. Since a visual comparison of the results may be inaccurate, we also used some performance measures to allow a quantitative comparison of results.

6.2 Performance Measures

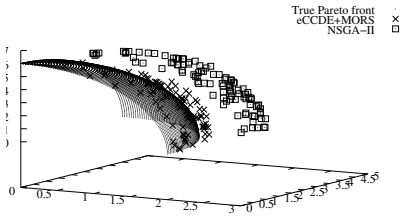
To assess the performance of the proposed approach, we adopted the two set coverage (CS) metric [18], which is an indicator of how much a set covers (or dominates) another one. A value of $CS(X, Y) = 1$ means that all points in X dominate or are equal to Y. If $CS(X, Y) = 0$, there are no points in X that dominate some point in Y.



(a) WFG1



(b) WFG2



(c) WFG9

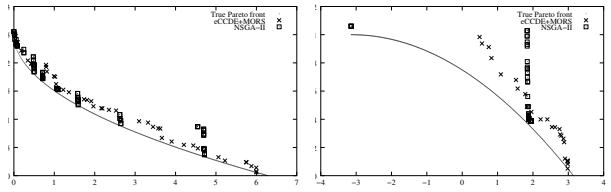
Figure 5: Results produced by our εCCDE+MORS and the NSGA-II on the WFG test problems.

Our second performance measure was the binary coverage (Q_c) [4], which is an indicator of the ability of an algorithm to obtain solutions near the extrema of the Pareto front, measuring the largest possible angle between two vectors of the output of an algorithm. This is a second criterion when proper convergence has been achieved. A value of $Q_c(X, Y) > 0$ means that X obtained points nearer to the extrema of the Pareto front Note that $Q_c(Y, X) = -Q_c(X, Y)$.

6.2.1 εCCDE alone

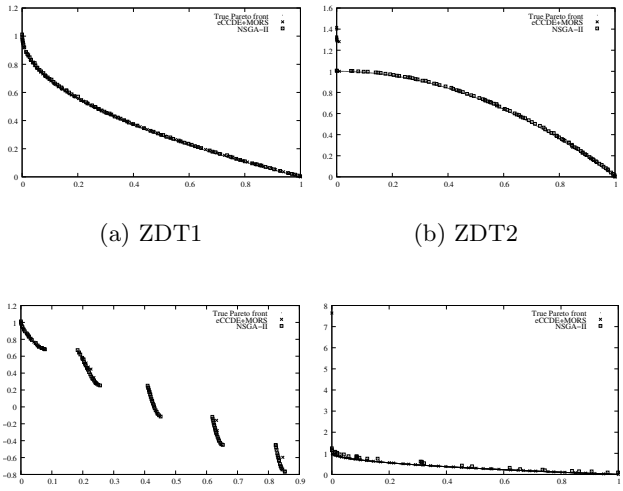
We executed our εCCDE 30 times per problem, and then executed the NSGA-II 30 times with the same random seeds, and we performed 30 one-to-one comparisons. The results are summarized in Table 1 and 2.

In all the problems in Table 1, the εCCDE obtained better average values. However, in WFG1, all the points of εCCDE always dominate the points produced by the NSGA-II, because the latter cannot properly converge. From Table 1, it can be seen that our approached obtained the largest value for WFG1. For WFG9, this metric indicates that the NSGA-II can cover a larger portion of the Pareto front. However, it is important to keep in mind that this is a secondary cri-



(a) OKA1 (b) OKA2

Figure 6: Results produced by our εCCDE+MORS and the NSGA-II on the OKA test problems.



(a) ZDT1 (b) ZDT2

(c) ZDT3 (d) ZDT4

Figure 7: Results produced by our εCCDE+MORS and the NSGA-II on the ZDT test problems.

Table 1: Mean and standard deviation of the CS measure (a larger value is better for the first algorithm)

Test Problem	CS(εCCDE, NSGA-II) mean (std. dev.)	CS(NSGA-II, εCCDE) mean (std. dev.)
WFG1	1.0000 (0.0000)	0.0000 (0.0000)
WFG2	0.8509 (0.1771)	0.0362 (0.0614)
WFG9	0.6415 (0.3669)	0.0995 (0.2114)

Table 2: Mean and standard deviation of the Q_c measure (a larger value is better for the first algorithm)

Test Problem	Q_c(εCCDE, NSGA-II) mean (std. dev.)
WFG1	0.2112 (0.0634)
WFG2	0.0677 (0.2172)
WFG9	-0.0913 (0.1154)

Table 4: Mean and standard deviation of the Q_c measure (a larger value is better for the first algorithm)

Test Problem	Q_c(εCCDE+MORS, NSGA-II) mean (std. dev.)
WFG1	0.6494 (0.1237)
WFG2	-0.1686 (0.1019)
WFG9	0.0085 (0.0796)
OKA1	0.0626 (0.0648)
OKA2	-0.1496 (0.1674)
ZDT1	0.0030 (0.0032)
ZDT2	0.0139 (0.0034)
ZDT3	-0.0980 (0.0249)
ZDT4	-0.3449 (0.4441)

terion, which becomes relevant only when convergence has been achieved. In this case, and based on the two set coverage measure, our εCCDE obtained a better convergence.

6.2.2 εCCDE+MORS

Again, we executed the proposed εCCDE+MORS 30 times per problem, and then executed the NSGA-II 30 times with the same random seeds, and we performed 30 one-to-one comparisons. The results are summarized in Table 3.

In almost all the problems in Table 3, the approach presented here obtained better average values. WFG1 is a very hard problem for the NSGA-II, which again cannot properly converge, and almost all its points were dominated by the εCCDE+MORS outcome. On the other hand, in OKA1, both algorithms show difficulties to dominate the results of each other. Regarding the ZDT problems, the NSGA-II presented a better convergence in two problems (ZDT2 and ZDT3), while in the other two our approach obtained a better convergence. For WFG2 and OKA2, as well as for ZDT3 and ZDT4, the second measure (Table 4) indicates that the NSGA-II can cover a slighly larger portion of the Pareto front. In these cases, and based on the CS measure, our εCCDE+MORS obtained a better convergence than the NSGA-II, except in ZDT3, where the NSGA-II obtained both, a better convergence and a larger length over the Pareto front.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we explored the use of the ε-constraint method hybridized with an efficient evolutionary single-objective optimizer, in order to obtain a first approximation to the Pareto front, and then apply an approach based on rough sets theory to spread the results. We apply this approach to solve hard two- and three-objective optimization problems. Our results show that the proposed approach can solve problems that a highly competitive MOEA (the NSGA-II) cannot. This approach may be recommended when other algorithms cannot achieve a proper convergence, or when it is known that the problem is deceptive or strongly biased (as is the case of several of the test problems adopted).

8. REFERENCES

[1] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving*

Table 3: Mean and standard deviation of the CS measure (a larger value is better for the first algorithm)

Test Problem	CS(εCCDE+MORS, NSGA-II) mean (std. dev.)	CS(NSGA-II, εCCDE+MORS) mean (std. dev.)
WFG1	0.8927 (0.0240)	0.0000 (0.0000)
WFG2	0.4569 (0.3869)	0.3006 (0.3430)
WFG9	0.7323 (0.2126)	0.0890 (0.1079)
OKA1	0.2278 (0.2694)	0.1589 (0.2577)
OKA2	0.3774 (0.1892)	0.1602 (0.2793)
ZDT1	0.3795 (0.1645)	0.1645 (0.3213)
ZDT2	0.0528 (0.0163)	0.3576 (0.2119)
ZDT3	0.0244 (0.0118)	0.4246 (0.1244)
ZDT4	0.4994 (0.2512)	0.1661 (0.3929)

Multi-Objective Problems. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[2] K. Deb, S. Chaudhuri, and K. Miettinen. Towards Estimating Nadir Objective Vector Using Evolutionary Approaches. In M. K. et al., editor, *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, volume 1, pages 643–650, Seattle, Washington, USA, July 2006. ACM Press. ISBN 1-59593-186-4.

[3] 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.

[4] A. Farhang-Mehr and S. Azarm. Minimal Sets of Quality Metrics. In C. M. Fonseca et al., editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 405–417, Faro, Portugal, April 2003. Springer. LNCS. Volume 2632.

[5] Y. Y. Haimes, L. S. Lasdon, and D. A. Wismer. On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297, July 1971.

[6] S. Huband, P. Hingston, L. Barone, and L. While. A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, October 2006.

[7] S. V. Kumar and S. R. Ranjithan. Evaluation of the Constraint Method-Based Evolutionary Algorithm (CMEA) for a Tree-Objective Optimization Problem. In W. Langdon et al., editors, *Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 431–438, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[8] R. Landa Becerra and C. A. Coello Coello. Optimization with Constraints using a Cultured Differential Evolution Approach. In H.-G. Beyer et al., editors, *Genetic and Evolutionary Computation Conference (GECCO'2005)*, volume 1, pages 27–34, Washington, D.C., U.S.A., June 2005. ACM Press.

[9] R. Landa Becerra and C. A. Coello Coello. Solving hard multiobjective optimization problems using ε-constraint with cultured differential evolution. In T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, pages 543–552. Springer. Lecture Notes in Computer Science Vol. 4193, Reykjavik, Iceland, September 2006.

[10] K. M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1999.

[11] T. Okabe. *Evolutionary Multi-Objective Optimization - On the Distribution of Offspring in Parameter and Fitness Space* -. PhD thesis, Bielefeld University, Germany, 2004.

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

[13] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991. ISBN 0-471-87339-X.

[14] K. V. Price. An introduction to differential evolution. In D. Corne et al., editors, *New Ideas in Optimization*, pages 79–108. McGraw-Hill, London, UK, 1999.

[15] S. R. Ranjithan, S. K. Chetan, and H. K. Dakshima. Constraint Method-Based Evolutionary Algorithm (CMEA) for Multiobjective Optimization. In E. Zitzler et al., editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 299–313. Springer-Verlag. LNCS No. 1993, 2001.

[16] R. G. Reynolds. An Introduction to Cultural Algorithms. In A. V. Sebald and L. J. Fogel, editors, *Third Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, River Edge, New Jersey, 1994.

[17] S. M. Saleem. *Knowledge-Based Solution to Dynamic Optimization Problems using Cultural Algorithms*. PhD thesis, Wayne State University, Detroit, Michigan, 2001.

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