

Modified Differential Evolution for Constrained Optimization

Efrén Mezura-Montes, Jesús Velázquez-Reyes and Carlos A. Coello Coello

Abstract— In this paper, we present a Differential-Evolution based approach to solve constrained optimization problems. The aim of the approach is to increase the probability of each parent to generate a better offspring. This is done by allowing each solution to generate more than one offspring but using a different mutation operator which combines information of the best solution in the population and also information of the current parent to find new search directions. Three selection criteria based on feasibility are used to deal with the constraints of the problem and also a diversity mechanism is added to maintain infeasible solutions located in promising areas of the search space. The approach is tested in a set of test problems proposed for the special session on Constrained Real Parameter Optimization. The results obtained are discussed and some conclusions are established.

I. INTRODUCTION

Evolutionary Algorithms (EAs) have been successfully applied to solve optimization problems [1], [2]. However, in their original versions, EAs lack a mechanism to deal with the constraints of a problem. Therefore, several approaches have been proposed to incorporate information of feasibility into the EAs' fitness function. The most popular approach is the use of penalty functions [3]. The aim of penalty functions is to decrease the fitness value of those infeasible individuals (which do not satisfy the constraints of the problem). In this way, feasible solutions will have more probabilities to be selected and the EA will approach the feasible region of the search space. However, the main drawback of penalty functions is that they require the definition of penalty factors. These factors will determine the degree of penalization. If the penalty value is very high, the feasible region will be approached mostly at random and the feasible global optimum will be hard to get. On the other hand, if the penalty is too low, the probability of not reaching the feasible region will be high. Therefore, the penalty factors must be carefully tuned as they are problem-dependent [4].

Differential Evolution (DE) is a novel EA proposed by Storn & Price [5] to solve optimization problems mainly in continuous search spaces. DE shares similarities with traditional EAs. However it does not use binary encoding as a simple genetic algorithm [6] and it does not use a probability density function to self-adapt its parameters as an Evolution Strategy [7]. Instead, DE performs mutation based on the distribution of the solutions in the current population. In this

way, search directions and possible stepsizes depend on the location of the individuals selected to calculate the mutation values.

There is more than one DE model. Such models vary in the type of recombination operator used and also in the number and type of solutions used to calculate the mutation values. The most popular model is called “*DE/rand/1/bin*”, where “DE” means Differential Evolution, “rand” indicates that individuals selected to compute the mutation values are chosen at random, “1” is the number of pairs of solutions chosen and finally “bin” means that a binomial recombination is used. As any EA, DE lacks a mechanism to deal with constrained search spaces [5]. The model “*DE/rand/1/bin*” is detailed in Figure 1. The “CR” parameter controls the influence of the parent in the generation of the offspring. Higher values mean less influence of the parent. The “F” parameter scales the influence of the set of pairs of solutions selected to calculate the mutation value (one pair in the case of the algorithm in Figure 1).

In this paper, we propose a DE-based approach to solve constrained optimization problems. The original “*DE/rand/1/bin*” is modified in order to incorporate information of the best solution in the current population (but not as the “*DE/best/1/bin*” does) and also information of the current parent to define the new search directions by using the DE mutation operator. Besides, we allow each parent to generate more than one offspring in the same generation (in the “*DE/rand/1/bin*” model, only one offspring is generated by each parent). Finally the constraint-handling mechanism avoids the use of a penalty function. Instead, a set of feasibility rules coupled with a diversity mechanism is proposed. The goal is to maintain, besides competitive feasible solutions, solutions with a promising value of the objective function, regardless of feasibility, in order to explore new regions of the search space and to avoid premature convergence.

This paper is organized as follows: In Section II, we summarize the previous related work. A detailed description of our approach is given in Section III. After that, Section IV presents the obtained results in the suggested format for the Special Session on Constrained Optimization. These results are discussed in Section V and finally in Section VI we state our conclusions and some of the possible paths for future research.

II. RELATED WORK

Other authors have proposed different approaches to solve constrained optimization problems with DE-based approaches. A feasible region shrinking mechanism was proposed by Storn [8]. The aim was to relax all the constraints of the problems at the beginning of the process. As time

Efrén Mezura-Montes is with the Laboratorio Nacional de Informática Avanzada, Rébsamen 80, Centro, Xalapa, Veracruz 91090, MEXICO (email: emezura@lania.mx)

Jesús Velázquez-Reyes and Carlos A. Coello Coello are with the Evolutionary Computation Group (EVOGINV) at CINVESTAV-IPN, Computer Science Section, Electrical Engineering Department, Av. IPN No. 2508 Col. San Pedro Zacatenco México D.F. 07300, MEXICO (email: jesus.velazquez@gmail.com ccoello@cs.cinvestav.mx)

```

1 Begin
2   G=0
3   Create a random initial population  $\bar{x}_G^i \forall i, i = 1, \dots, NP$ 
4   Evaluate  $f(\bar{x}_G^i) \forall i, i = 1, \dots, NP$ 
5   For G=1 to MAX.GENERATIONS Do
6  $\Rightarrow$    For i=1 to NP Do
7     Select randomly  $r_1 \neq r_2 \neq r_3 :$ 
8      $j_{rand} = \text{randint}(1, D)$ 
9     For j=1 to D Do
10      If ( $\text{rand}_j[0, 1) < CR$  or  $j = j_{rand}$ ) Then
11         $u_{j,G+1}^i = x_{j,G}^{r_3} + F(x_{j,G}^{r_1} - x_{j,G}^{r_2})$ 
12      Else
13         $u_{j,G+1}^i = x_{j,G}^i$ 
14      End If
15  $\Rightarrow$    End For
16  $\mapsto$    If ( $f(\bar{u}_{G+1}^i) \leq f(\bar{x}_G^i)$ ) Then
17      $\bar{x}_{G+1}^i = \bar{u}_{G+1}^i$ 
18   Else
19      $\bar{x}_{G+1}^i = \bar{x}_G^i$ 
20  $\mapsto$    End If
21   End For
22   G = G + 1
23 End For
24 End

```

Fig. 1. ‘DE/rand/1/bin’ algorithm. $\text{randint}(\text{min}, \text{max})$ is a function that returns an integer number between min and max. $\text{rand}[0, 1)$ is a function that returns a real number between 0 and 1. Both are based on a uniform probability distribution. ‘NP’, ‘MAX GENERATIONS’, ‘CR’ and ‘F’ are user-defined parameters. The steps between arrows (6 to 15 and 16 to 20) are modified in our approach and detailed in Figures 2 and 3.

goes by, the pseudo-feasible region is shrunk at each generation until it matches the real feasible region. Storn also proposed the idea of each parent to generate more than one offspring. However, unlike the proposed approach in this paper, in Storn’s approach the process finishes when one offspring is better than its parent or when NT offspring have been generated. Moreover, his approach works over the ‘DE/best/1/bin’ and an aging mechanism is added to avoid a solution to remain in the population for too long. His approach performed well in problems with only inequality constraints, but showed some problems when dealing with equality constraints.

A static-penalty approach, coupled with DE to solve engineering design was proposed by Lampinen & Zelinka [9], [10], [11], [12]. The main drawback of the approach is the careful tuning required for the penalty factors.

Yung-Chien and Feng-Sheng [13] proposed an Augmented Lagrangian approach with an adaptive mechanism to update the penalty parameters. The approach performed well against typical EA-based techniques.

An extension of DE to solve constrained optimization problems was proposed by Lampinen [14]. The original DE replacement mechanism (based only on the objective function value of the parent and its corresponding offspring) was substituted by three selection criteria based on feasibility originally proposed by Deb [15]:

- Between 2 feasible solutions, the one with the highest fitness value wins.
- If one solution is feasible and the other one is infeasible, the feasible solution wins.
- If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred.

The difference between Deb’s and Lampinen’s approach is in the third rule. In Deb’s approach the solution with the lowest sum of constraint violation is selected. On the other hand, in Lampinen’s technique, the solution which Pareto-dominates the other in the constraints space will be selected.

Mezura et al. [16] proposed a DE-based approach where the newly generated offspring is added to the current generation (instead of including the solution into the next generation). The idea was to allow newly generated solutions to be selected to influence the selection of search directions of the offspring in the current generation and to speed up convergence. The approach used Deb’s rules [15] to handle the constraints of the problem.

III. OUR APPROACH

As it was noted in the previous Section, all approaches considered feasible solutions better than infeasible ones and they work using typical DE models as a search engine. The main motivation of this work is two-fold: (1) We argue that the incorporation of the information of the best solution and the current parent, coupled with a mechanism that allows each parent to generate more than one offspring, will increase the exploration and exploitation capabilities of the DE algorithm when solving optimization problems and that (2) the incorporation of a diversity mechanism will allow the DE approach to sample the feasible region in a better way as to reach the feasible global optimum. Hence, our approach is based on two modifications to the original DE:

```

1 For k=1 to  $n_o$  Do
2   Select randomly  $r_1 \neq r_2 \neq r_3 \neq i$ 
3   For j=1 to D Do
4     If ( $\text{rand}_j[0, 1) < CR$  or  $j = j_{rand}$ ) Then
5        $\text{child}_j = x_{j,G}^{r_3} + F_\alpha(x_{j,G}^{best} - x_{j,G}^{r_2}) + F_\beta(x_{j,G}^i - x_{j,G}^{r_1})$ 
6     Else
7        $\text{child}_j = x_{j,G}^i$ 
8     End If
9   End For
10  If  $k > 1$  Then
11    If (child is better than  $\bar{u}_{G+1}^i$ 
12      (based on the three selection criteria)) Then
13       $\bar{u}_{G+1}^i = \text{child}$ 
14    End If
15  Else
16     $\bar{u}_{G+1}^i = \text{child}$ 
17  End If
18 End For

```

Fig. 2. Multiple offspring pseudocode. \bar{u}_{G+1}^i keeps the best offspring from the n_o generated. \bar{x}_G^i is the current parent and \bar{x}_G^{best} is the best individual in the current population G . n_o is a user-defined parameter.

- 1) In order to increase the probability to generate better offspring, each parent is allowed to generate a number of offspring set by a user-defined parameter called n_o . New offspring are created in such a way that information of the best solution in the population and also the parent’s own information is combined in the mutation operator. The proposed expression for the

mutation operator is the following:

$$u_{j,G+1}^i = x_{j,G}^{r_3} + F_\alpha(x_{j,G}^{best} - x_{j,G}^{r_2}) + F_\beta(x_{j,G}^i - x_{j,G}^{r_1}) \quad (1)$$

where \bar{x}_G^{best} is the best solution in the current population, \bar{x}_G^i is the current parent and $\bar{x}_G^{r_1}$, $\bar{x}_G^{r_2}$ and $\bar{x}_G^{r_3}$ are randomly selected from the current population. The F_α and F_β factors indicate the influence of the best and parent solutions, respectively, in the search direction of the offspring. It is important to note that our mutation model is different from typical DE models [5] because of the way we combine the information of the best solution and the current parent with the three solutions chosen at random. In our proposal, we maintain the discrete recombination between the mutation vector (calculated by using equation 1) and the current parent. The best offspring, among the n_0 generated, is selected by using three selection criteria based on feasibility originally proposed by Deb [15]:

- Between 2 feasible solutions, the one with the highest fitness value wins.
- If one solution is feasible and the other one is infeasible, the feasible solution wins.
- If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred ($\sum_{i=1}^m \max(0, g_i(\bar{x}))$).

This best offspring will compete against its parent based on the aforementioned criteria but with a diversity mechanism explained in the following point. The detailed pseudocode is presented in Figure 2.

- 2) The diversity mechanism will let infeasible individuals located in promising areas of the search space to remain in the population for the next generation. This is controlled by a user-defined parameter called S_r which represents the probability to select, between parent and best offspring, based only the objective function value (regardless of feasibility). In this way, $1 - S_r$ is the probability to select between these two solutions by using the selection criteria based on feasibility. The S_r requires an initial value, which will be decreasing during the first third of the process and the remaining two thirds, the value will remain fixed at its lowest value. This is a desired behavior, because, at the beginning of the process more exploration of the search space is required in order to find promising areas. On the other hand, in the last generations of the process, the search effort must be focused on keeping the feasible solutions found and to discard the infeasible ones. In fact, we are assuming that promising areas of the search space, and more importantly, promising areas of the feasible region have been reached. The details of the diversity mechanism is presented in Figure 3.

IV. RESULTS

The experimental design is presented in the format required for the Special Session on Constrained Real-Parameter Optimization. The 24 test problems used and their main

```

1  If flip( $S_r$ ) Then
2    If ( $f(\bar{u}_{G+1}^i) \leq f(\bar{x}_G^i)$ ) Then
3       $\bar{x}_{G+1}^i = \bar{u}_{G+1}^i$ 
4    Else
5       $\bar{x}_{G+1}^i = \bar{x}_G^i$ 
6    End If
7  Else
8    If ( $\bar{u}_{G+1}^i$  is better than  $\bar{x}_G^i$ 
9      (based on the three selection criteria)) Then
10      $\bar{x}_{G+1}^i = \bar{u}_{G+1}^i$ 
11   Else
12      $\bar{x}_{G+1}^i = \bar{x}_G^i$ 
13   End If
14 End If

```

Fig. 3. Diversity Mechanism incorporated in the selection of the best solution between parent (\bar{x}_G^i) and best offspring (\bar{u}_{G+1}^i). S_r is a user-defined parameter. The selection criteria are detailed in Section III.

features can be found in [17]. The details of the experiments are shown as follows:

A. PC Configure:

System: Debian GNU/Linux 2.6.14 i686
CPU: Mobile Intel Pentium 4 - M CPU 2.20GHz
RAM: 256 MB
Language: C (gcc 3.3.5 compiler)

B. Parameters Setting

1) *Parameter values used:* The following parameters were used for the 24 test problems.

- DE parameters:
 - Maximum number of generations: 3333.
 - Population size: 30
 - $CR = 0.9$
- Additional Parameters:
 - Number of children per solution (n_0): 5
 - $F_\alpha = 0.8$, $F_{beta} = 0.1$ (these values mean a higher influence of the best solution in the population and a moderated influence of the parent when computing the mutation vector).

Then, we have 3333 generations \times 30 solutions in the population \times 5 children per solution = 499,950 FES, which is slightly below the 500,000 FES required.

2) *Dynamic Ranges:* Based on previous experiments performed to set the values per each parameter of the approach, the following ranges are suggested.

- Population size (μ): [15, 60]
- Number or children per solution (λ): [2, 7]
- CR (discrete crossover of DE): $[0.0, 0.3] \cup [0.8, 1.0]$
- $F_\alpha = [0.7, 0.9]$
- $F_\beta = [0.1, 0.3]$

The parameter S_r is adapted by using a dynamic mechanism. We noted that, the probability of selecting infeasible solutions located in promising areas of the search space must be higher at the beginning of the process, because the algorithm is exploring new regions. On the other hand, late in the process, the approach must focus on the feasible region

found and must discard infeasible solutions. Therefore, we propose a dynamic adjustment of the S_r parameter which is implemented based on expression (2).

$$S_{r_{G+1}} = \begin{cases} S_{r_G} - \Delta_{S_r} & \text{if } G < GMAX/3 \\ 0.025 & \text{otherwise} \end{cases} \quad (2)$$

The initial and final value for the parameter are $S_{r_0} = 0.55$ and $S_{r_{GMAX}} = 0.025$, respectively, and the decrease value is calculated as follows:

$$\Delta_{S_r} = \frac{3(S_{r_0} - S_{r_{GMAX}})}{GMAX} \quad (3)$$

Then, the first third part of the process, selection will be made in such a way that infeasible solutions with a good value of the objective function will have a significant probability of being selected, regardless of feasibility (exploration). However, in the last two third parts of the process this probability will remain fixed with a very low value (exploitation of previously found promising regions).

3) *Parameter Tuning*: As can be noted in Section IV-B.1, our approach uses the same parameters for all problems. However, we provide the following suggestions based on the features of the problems to be solved: when the size of the feasible region is significant with respect to the size of the search space, combined with a high dimensionality (problems g02 and g19) the CR values can take values close to either 0 or 1 e.g. to use, mostly, the values from only one of the two solutions used in the recombination process (the mutation vector or the current parent, See Figure 2 rows 4–8). Furthermore, for this case of a wide feasible region, the population size can be slightly increased and the number of offspring can be slightly decreased. This change will improve the algorithm’s capabilities to sample a wider area (feasible region) in order to find the best feasible solution. In contrast, in most of the problems of the current benchmark, the approach will require more time to find the feasible region (due to several equality constraints in most cases) instead of sampling it in detail.

The remaining parameters did not require a fine tuning and their values were empirically derived.

4) *Parameter Tuning Cost in Terms of FES*: The only parameter that required further experiments to be calibrated was CR . Therefore, we defined 10 values between 0.0 and 1.0: [0.1, 0.2, 0.3, ..., 1.0]. In this way, The number of FES required to adjust the parameters for these experiments were approximately $30(\text{individuals}) \times 5(\text{offspring per individual}) \times 100(\text{generations}) \times 25(\text{runs}) \times 24(\text{functions}) \times 10(CR \text{ values}) = 9E+7$ FES.

C. Results Achieved

The error values achieved when $FES=5E+3$, $FES=5E+4$ and $FES=5E+5$ are presented in Table I for problems g01 to g06, in Table II for problems g07 to g12, in Table III for problems g13 to g18 and in Table IV for problems g19 to g24. In these Tables, the c values represent the number of violated constraints at the median solution by

more than 1.0, 0.01 and 0.0001, respectively. \bar{v} is the mean value of the violations of all constraints at the median solution. The number in parenthesis indicates the number of violated constraints at the corresponding statistical value (best, median and worst).

The statistics of the number of FES to achieve the fixed accuracy level (≤ 0.0001), the Feasible Rate (rate of runs where a feasible solution is found), the Success Rate (rate of runs where the best solution is found with the accuracy required) and Success Performance (the mean FES for successful runs multiplied by the number of total runs and divided by the number of successful runs) for all 24 test problems are summarized in Table V.

The convergence behavior for the median run (out of the 25 runs performed) for each test problem is presented in Figure 4 for problems g01 to g06, in Figure 5 for problems g07 to g12, in Figure 6 for problems g13 to g18 and finally in Figure 7 for problems g19 to g24.

D. Algorithm Complexity

The results of the experiments to detect the algorithm complexity are presented in Table VI, where $T1$ is the average time to compute 10000 FES for the 24 test problems and $T2$ is the average time for the proposed algorithm to perform 10000 FES.

TABLE VI
COMPUTATIONAL COMPLEXITY (TIME GIVEN IN SECONDS)

$T1$	$T2$	$(T2 - T1)/T1$
0.035229	0.105034	1.981464

V. DISCUSSION OF RESULTS

The results reported in Tables I, II, III and IV, show that our approach is able to find a very good feasible approximation of the best solution reported by using just 5×10^4 FES, except in problems g10, g14, g19, g20, g21, g22 and g23. Nonetheless, the only test problems that our approach is not able to solve in 5×10^5 FES were g20 and g22, but it is important to note that the final solution found by the approach in problem g20 is almost feasible. Despite the fact that the 20 constraints are violated in the best, mean and worst cases, only one of them is violated for more than 0.01 (See Table IV column 4). It is important to remark that, for all problems, except g20 and g22, the best known solution is found. This behavior is shown in the left handside graphs in Figures 4 to 7.

Another interesting behavior is how fast the approach finds the feasible region of the problem. In most of them, the feasible region was clearly found after 5×10^4 FES, except for problems g14, g20, g21, g22 and g23. Again, from these five problems, the feasible region is reached in all cases except in problems g20 and g22 after 5×10^5 FES, but for g20, the solutions are very close to the feasible region. This issue is confirmed in the convergence graphs for the constraints (right handside graphs in Figures 4, 5,

TABLE I

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 FOR PROBLEMS 1-6.

FES		g01	g02	g03	g04	g05	g06
5×10^3	Best	8.119230(2)	0.423383(0)	0.091672(1)	1.274382(0)	-84.598808(3)	0.000023(0)
	Median	10.795020(0)	0.501704(0)	0.750780(1)	6.863769(0)	-5.893107(3)	0.002904(0)
	Worst	15.384258(0)	0.549707(0)	0.999807(1)	44.187280(0)	182.032342(3)	3.258425(0)
	c	0.0,0	0.0,0	0.0,1	0.0,0	2.3,3	0.0,0
	\bar{v}	0.000000	0.000000	0.000386	0.000000	1.372402	0.000000
	Mean	10.947525	0.502400	0.681406	12.325465	-0.637820	0.200413
5×10^4	Std	1.557386	0.026944	0.251153	10.263079	52.933785	0.649617
	Best	0.001332(0)	0.005415(0)	0.000991(0)	0.000000(0)	0.000000(0)	0.000000(0)
	Median	0.004906(0)	0.048010(0)	0.000692(0)	0.000004(0)	0.000000(0)	0.000000(0)
	Worst	0.020865(0)	0.386253(0)	0.004481(0)	0.000200(0)	0.000000(0)	0.000000(0)
	c	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0
	\bar{v}	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5×10^5	Mean	0.005837	0.133469	0.001110	0.000038	0.000000	0.000000
	Std	0.004386	0.133130	0.001109	0.000061	0.000000	0.000000
	Best	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)	-0.000000(0)	0.000000(0)
	Median	0.000000(0)	0.017460(0)	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)
	Worst	0.000000(0)	0.050393(0)	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)
	c	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0
5×10^5	\bar{v}	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	Mean	0.000000	0.019353	0.000000	0.000000	-0.000000	0.000000
	Std	0.000000	0.012603	0.000000	0.000000	0.000000	0.000000

TABLE II

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 FOR PROBLEMS 7-12.

FES		g07	g08	g09	g10	g11	g12
5×10^3	Best	5.373277(0)	0.000000(0)	0.061042(0)	2412.461369(0)	0.000000(0)	0.000000(0)
	Median	11.157435(0)	0.000000(0)	0.213357(0)	5760.374012(0)	0.001276(1)	0.000000(0)
	Worst	27.862314(0)	0.000000(0)	0.596713(0)	18053.004122(1)	0.249784(1)	0.000000(0)
	c	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0
	\bar{v}	0.000000	0.000000	0.000000	0.000000	0.000012	0.000000
	Mean	12.154470	0.000000	0.260198	6407.827022	0.041754	0.000000
5×10^4	Std	5.977032	0.000000	0.136158	3592.548347	0.080467	0.000000
	Best	0.009890(0)	0.000000(0)	0.000000(0)	65.428563(0)	0.000000(0)	0.000000(0)
	Median	0.026109(0)	0.000000(0)	0.000000(0)	165.928900(0)	0.000000(0)	0.000000(0)
	Worst	0.051554(0)	0.000000(0)	0.000001(0)	294.886078(0)	0.000000(0)	0.000000(0)
	c	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0
	\bar{v}	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5×10^5	Mean	0.026007	0.000000	0.000000	154.006365	0.000000	0.000000
	Std	0.009174	0.000000	0.000000	59.383433	0.000000	0.000000
	Best	0.000000(0)	0.000000(0)	0.000000(0)	-0.000000(0)	0.000000(0)	0.000000(0)
	Median	0.000000(0)	0.000000(0)	0.000000(0)	-0.000000(0)	0.000000(0)	0.000000(0)
	Worst	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)
	c	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0
5×10^5	\bar{v}	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	Mean	0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000
	Std	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

TABLE III

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 FOR PROBLEMS 13-18.

FES		g13	g14	g15	g16	g17	g18
5×10^3	Best	0.010652(3)	-358.235364(3)	-0.084729(2)	0.001178(0)	2.956297(4)	-1.715758(6)
	Median	0.166531(3)	-278.662424(3)	0.024493(2)	0.005429(0)	37.981555(4)	1.045926(3)
	Worst	2.089679(3)	-218.841524(3)	2.649105(2)	0.064526(0)	111.600531(4)	2.399959(10)
	c	0.3,3	3.3,3	0.2,0	0.0,0	3.4,4	0.3,3
	\bar{v}	0.059504	8.138720	0.003206	0.000000	2.309186	0.031967
	Mean	0.376192	-272.995203	0.295674	0.010946	51.986707	0.681670
5×10^4	Std	0.458401	65.939829	0.584447	0.013484	37.640140	1.033089
	Best	0.000000(0)	-160.100521(3)	0.000000(0)	0.000000(0)	0.000000(0)	0.001064(0)
	Median	0.000000(0)	-133.726584(3)	0.000000(0)	0.000000(0)	0.000000(0)	0.004179(0)
	Worst	0.384628(3)	-79.394052(3)	0.000000(0)	0.000000(0)	0.000275(0)	0.191904(0)
	c	0.0,0	3.3,3	0.0,0	0.0,0	0.0,0	0.0,0
	\bar{v}	0.000000	3.897770	0.000000	0.000000	0.000000	0.000000
5×10^5	Mean	0.015755	-125.339481	0.000000	0.000000	0.000012	0.026794
	Std	0.075304	34.877157	0.000000	0.000000	0.000054	0.060866
	Best	-0.000000(0)	0.000002(0)	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)
	Median	-0.000000(0)	0.000010(0)	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)
	Worst	-0.000000(0)	0.000067(0)	0.000000(0)	0.000000(0)	0.000000(0)	0.000000(0)
	c	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0	0.0,0
5×10^5	\bar{v}	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	Mean	-0.000000	0.000015	0.000000	0.000000	0.000000	0.000000
	Std	0.000000	0.000014	0.000000	0.000000	0.000000	0.000000

6 and 7). Except for problems g20 and g22, the feasibility of solutions, in the worst case, is reached in 1.5×10^5 FES and, in the best case, before 0.5×10^5 . It is worth noting that for problem g20 the sum of constraint violation continues to decrease. It is expected, with more FES, to reach the feasible region.

This high-rate to reach the feasible region is also confirmed by the results provided in Table V. Except for problems g20 and g22, the feasible rate is 100% in all cases. Regarding the

success rate, except for functions g02, g19, g20 and g22, the approach reached a value of 100% in the remaining cases.

Furthermore, the success performance shows that the approach requires less than 2×10^5 FES to find the accuracy required for the session (only problems g14 and g23 require 2.9×10^5 and 3.6×10^5 FES respectively, and problems g20 and g22 could not be solved). In fact, for some problems like g01, g02, g03, g04, g05, g06, g08, g09, g11, g12, g13, g15, g16, g17 and g24, less than 1×10^5 FES are required

TABLE IV
 ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 FOR PROBLEMS 19-24.

FES		g19	g20	g21	g22	g23	g24
5×10^3	Best	29.147347(0)	3.659076(20)	-23.659401(6)	-0.396115(20)	-1431.193635(6)	0.000000(0)
	Median	62.667703(0)	5.013679(20)	195.769337(5)	2967.897433(19)	-1129.818674(5)	0.000000(0)
	Worst	120.992475(0)	6.674437(20)	707.402016(5)	19538.021295(19)	195.598037(5)	0.000031(0)
	c	0.0,0	2.17,20	2.4,5	19.19,19	3.4,5	0.0,0
	\bar{v}	0.000000	3.044829	1.816631	7162988.231154	1.037286	0.000000
	Mean	65.083141	5.110993	239.529680	5878.925317	-1008.981698	0.000001
5×10^4	Std	19.969016	0.756969	217.064782	6294.465923	408.074802	0.000006
	Best	0.142245(0)	0.009796(20)	-147.126988(5)	122.346252(20)	-1699.735511(4)	0.000000(0)
	Median	1.165419(0)	0.096863(20)	25.865051(5)	2649.915454(20)	-1695.322919(6)	0.000000(0)
	Worst	5.341293(0)	0.289732(20)	241.183053(4)	19592.602330(19)	-1347.015384(5)	0.000000(0)
	c	0.0,0	0.18,20	0.3,5	19.20,20	1.3,6	0.0,0
	\bar{v}	0.000000	0.069730	0.112701	524973.984597	0.582652	0.000000
5×10^5	Mean	1.527043	0.104528	17.507394	6041.750812	-1606.900798	0.000000
	Std	1.250269	0.080428	113.348189	6131.150874	335.626515	0.000000
	c	0.007323(0)	0.077444(20)	-0.000000(0)	2501.983748(11)	0.000000(0)	0.000000(0)
	Median	0.387033(0)	0.103314(20)	-0.000000(0)	9210.082460(18)	0.000000(0)	0.000000(0)
	Worst	3.527083(0)	0.173136(20)	0.000000(0)	17922.361766(10)	0.000000(0)	0.000000(0)
	c	0.0,0	0.1,3	0.0,0	8.9,16	0.0,0	0.0,0
5×10^5	\bar{v}	0.000000	0.008086	0.000000	500003.133718	0.000000	0.000000
	Mean	0.685658	0.105819	-0.000000	10132.618585	0.000000	0.000000
	Std	0.847517	0.021688	0.000000	4808.800969	0.000000	0.000000

TABLE V
 NUMBER OF FES TO ACHIEVE THE FIXED ACCURACY LEVEL ($(f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001$), SUCCESS RATE, FEASIBLE RATE AND SUCCESS PERFORMANCE.

Prob.	Best	Median	Worst	Mean	Std	Feasible Rate	Success Rate	Success Performance
g01	63300	75000	90900	75373	6708.9211	100%	100%	75373
g02	53250	71100	245550	96222	49862.6724	100%	16%	96222
g03	36000	45300	61350	44988	7145.6599	100%	100%	44988
g04	33900	39300	61950	41562	8129.8251	100%	100%	41562
g05	19350	20550	24000	21306	2021.5994	100%	100%	21306
g06	4650	5250	5250	5202	162.7759	100%	100%	5202
g07	124650	176400	380400	194202	65471.8878	100%	100%	194202
g08	900	900	1350	918	88.1816	100%	100%	918
g09	14850	15000	19200	16152	1753.2530	100%	100%	16152
g10	152400	163500	179850	164160	5916.4685	100%	100%	164160
g11	1200	1200	4950	3000	1873.4994	100%	100%	3000
g12	1200	1200	1650	1308	192.1874	100%	100%	1308
g13	19500	19950	24450	21732	2127.8101	100%	100%	21732
g14	208236	298072	408036	291642	49861.3046	100%	100%	291642
g15	9750	9750	11850	10458	959.5499	100%	100%	10458
g16	7950	8700	9450	8730	718.7489	100%	100%	8730
g17	20400	26550	34950	26364	3824.0952	100%	100%	26364
g18	54000	118050	133800	103482	17633.3314	100%	100%	103482
g19	NA	NA	NA	NA	NA	100%	NA	NA
g20	NA	NA	NA	NA	NA	NA	NA	NA
g21	82350	120450	201150	112566	23401.8876	100%	100%	112566
g22	NA	NA	NA	NA	NA	NA	NA	NA
g23	247500	374400	476250	360420	81403.4508	100%	100%	360420
g24	1650	1650	1950	1794	149.8800	100%	100%	1794

to find the accuracy requested in the best and mean cases.

The convergence behavior is detailed in the left handside graphs in Figures 4, 5, 6 and 7. It is important to remark that some plots are cut on its left edge. This cut means that until the corresponding FES, the first feasible solution was found. For problems g01 g04 g05 and g06 (left handside graph in Figure 4) a good convergence (less than 2×10^5 FES) was obtained for the six test problems. For Problems g02 and g03, the approach converged after 4×10^5 FES.

A similar fast convergence was found for problems g07 to g12 (left handside graph in Figure 5), but in this case it is clear that for problems g07 and g10, convergence is reached after 3.5×10^5 FES. For problems g13 to g18 (left handside graph in Figure 6) a fast convergence is also obtained, except for problems g14 and g18 where the approach converges after 3.5×10^5 and 2.5×10^5 FES respectively. In fact, for problem g14, the first feasible solution is found after 1.5×10^5 FES. These results are confirmed with the results provided in Table III, column 4, where it can be noted that the first feasible solution for problem g14 is found after 5×10^4 FES. Finally, in the left handside graph in Figure 7 we can observe that for problem g19 convergence

is clearly reached after 3.5×10^5 FES. For problems g20 and g22, the approach has an irregular behavior because the sum of constraint violation was never decreased (see right handside graph in Figure 7). For problem g21 the first feasible solution was reached after 0.5×10^5 FES and convergence is obtained after 2×10^5 FES and for problem g23, the first feasible solution was found after 1×10^5 FES and the best known solution is reached after 3.5×10^5 FES. Problem g24 is solved very fast; thus, the plot is a vertical line in the graph.

The results obtained for the computational complexity in Table VI show that the proposed approach is fast and that it does not add a significant computational cost. This is mainly because DE is a very simple algorithm whose main operators only perform simple arithmetic operations. Moreover, DE does not require any sorting nor encoding mechanism.

All these results suggest a very competitive overall performance of the proposed approach. However, some sources of difficulty were found. Our approach seems to have difficulties with the combination of a high dimensionality (≥ 22) as well as a considerable number of nonlinear inequality constraints (≥ 11). These features are found precisely in problems g20

and g22. However, in problem g20, solutions very close to be feasible region were found. As mentioned in Section IV-B.3, the inconsistent performance shown in functions g02 and g19 were because of the CR parameter. Values of $CR = 0.2$ for problem g02 and $CR = 0.99$ for g19 clearly improve the quality and robustness of the results. These two problems share the features previously detected as a reason to modify the CR values (high dimensionality: 20 and 15 variables respectively and a considerable large feasible region with respect to the whole search space: 99% and 33% respectively).

VI. CONCLUSIONS AND FUTURE WORK

A novel DE-based approach was proposed to solve constrained optimization problems. The main features of our approach are the multiple offspring mechanism added to DE in order to increase the probability of each parent to generate a better solution. This is due to a new mutation operator which uses information of the best solution in the current population and also information of the current parent to find new search directions. To deal with the constraints of the problem, three criteria based on feasibility are used. Furthermore, a diversity mechanism to maintain infeasible solutions with a good value of the objective function is added in the comparison between parent and best offspring. The results obtained showed a very competitive performance of the approach, based on quality of results (even some best known solutions were improved), number of FES required to find the feasible region and number of FES to reach (or even improve) the best known solution. Furthermore, the experiments suggested that, when a problem has a combination of a dimensionality higher than 22 with more than 11 nonlinear equality constraints, the approach is not able to solve it in the number of FES used in the experiments. Finally, the average computational cost of the approach (measured in seconds) seems to be very affordable. As part of our future work, we intend to analyze more in-depth the behavior of the CR parameter because the approach was sensitive to its value in problems with a large feasible region. Finally, we plan to extend the approach to solve multiobjective optimization problems.

ACKNOWLEDGMENT

The authors acknowledge support from the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through project number 42435-Y.

REFERENCES

- [1] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition, 1996.
- [2] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, June 2002. ISBN 0-3064-6762-3.
- [3] Alice E. Smith and David W. Coit. Constraint Handling Techniques—Penalty Functions. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.

- [4] K. Miettinen, M.M. Makela, and J. Toivanen. Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. *Journal of Global Optimization*, 27(4):427–446, December 2003.
- [5] Kenneth V. Price. An introduction to differential evolution. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 79–108. Mc Graw-Hill, UK, 1999.
- [6] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1989.
- [7] Hans-Paul Schwefel, editor. *Evolution and Optimization Seeking*. John Wiley & Sons, New York, 1995.
- [8] Rainer Storn. System Design by Constraint Adaptation and Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 3(1):22–34, April 1999.
- [9] Jouni Lampinen and Ivan Zelinka. Mechanical Engineering Design Optimization by Differential Evolution. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 127–146. Mc Graw-Hill, UK, 1999.
- [10] Jouni Lampinen and Ivan Zelinka. Mixed Variable Non-Linear Optimization by Differential Evolution. In *Proceedings of Nostradamus'99, 2nd International Prediction Conference*, pages 45–55. Zlin, Czech Republic. Technical University of Brno, Faculty of Technology Zlin, Department of Automatic Control, October 1999. ISBN 80-214-1424-3.
- [11] Jouni Lampinen and Ivan Zelinka. Mixed Integer-Discrete-Continuous Optimization by Differential Evolution, Part 1: the optimization method. In Pavel Ošmera, editor, *Proceedings of MENDEL'99, 5th International Mendel Conference on Soft Computing*, pages 71–76. Brno, Czech Republic. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, June 1999. ISBN 80-214-1131-7.
- [12] Jouni Lampinen and Ivan Zelinka. Mixed Integer-Discrete-Continuous Optimization by Differential Evolution, Part 2: a practical example. In Pavel Ošmera, editor, *Proceedings of MENDEL'99, 5th International Mendel Conference on Soft Computing*, pages 77–81. Brno, Czech Republic. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, June 1999. ISBN 80-214-1131-7.
- [13] Yung-Chien Lin, Kao-Shing Hwang, and Feng-Sheng Wang. Hybrid Differential Evolution with Multiplier Updating Method for Nonlinear Constrained Optimization Problems. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 872–877. Piscataway, New Jersey, May 2002. IEEE Service Center.
- [14] Jouni Lampinen. A Constraint Handling Approach for the Differential Evolution Algorithm. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 2, pages 1468–1473. Piscataway, New Jersey, May 2002. IEEE Service Center.
- [15] Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.
- [16] Efrén Mezura-Montes and Carlos A. Coello Coello. An Improved Diversity Mechanism for Solving Constrained Optimization Problems Using a Multimembered Evolution Strategy. In Kalyanmoy Deb et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2004)*, pages 700–712. Heidelberg, Germany, June 2004. Seattle, WA, Springer Verlag. Lecture Notes in Computer Science Vol. 3102.
- [17] J.J. Liang, Thomas Philip Runarsson, Efrén Mezura-Montes, Maurice Clerc, P.N. Suganthan, Carlos A. Coello Coello, and K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, School of EEE, Nanyang Technological University, Singapore, 2006.

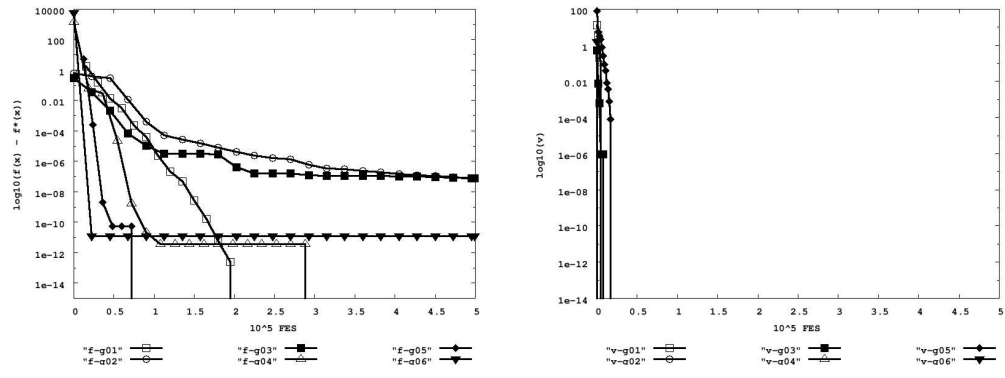


Fig. 4. Convergence Graph for Problems 1-6

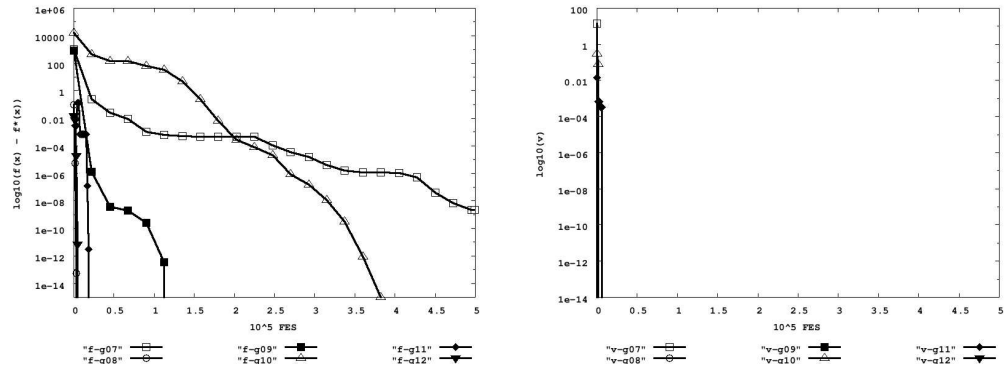


Fig. 5. Convergence Graph for Problems 7-12

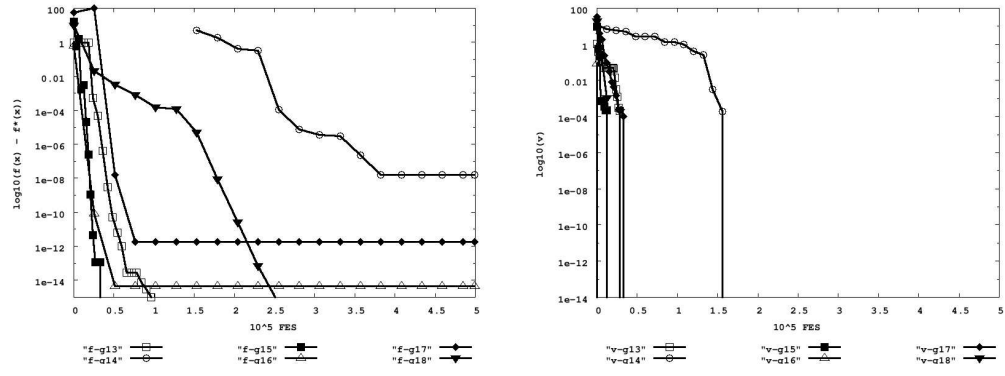


Fig. 6. Convergence Graph for Problems 13-18

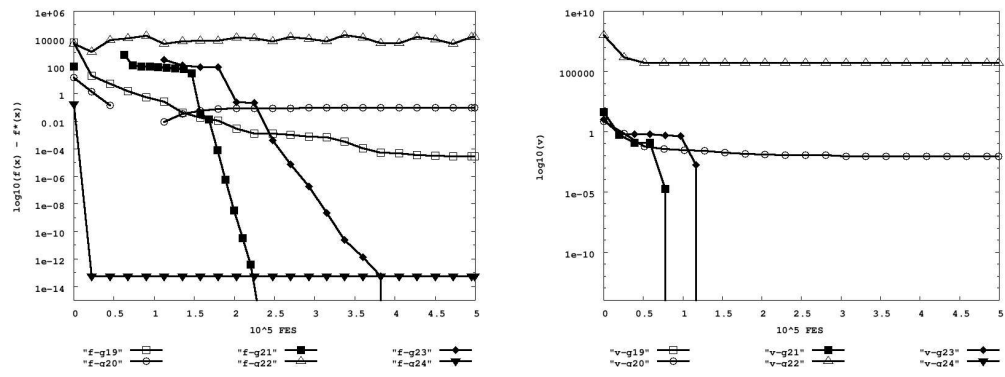


Fig. 7. Convergence Graph for Problems 19-24