Boundary Constraint-Handling Methods in Differential Evolution for Mechanical Design Optimization

Sebastián-José de-la-Cruz-Martínez
National Laboratory for Advanced Informatics (LANIA A.C.)
Rébsamen 80, Centro
Xalapa, Veracruz, México
sdelacruz.mca17@lania.edu.mx

Efrén Mezura-Montes
Artificial Intelligence Research Center
University of Veracruz
Sebastián Camacho 5, Centro
Xalapa, Veracruz, México
emezura@uv.mx

***Abstract*—This paper presents an experimental comparison of nine boundary constraint-handling methods found in the specialized literature added to differential evolution when solving four mechanical design optimization problems. The experimental part considers a comparison of final results besides performance measures used in evolutionary constrained optimization as well as the number of vectors and variables repaired. The Kruskal-Wallis non-parametric test and the Bonferroni post-hoc test are computed to validate the findings. The final results suggest that the Projection method provides a better overall performance when compared with other approaches. However, Centroid 1+1 was the method which promotes less repairs.**

***Index Terms*—Boundary Constraints, Differential Evolution, Mechanical design**

I. Introduction

Evolutionary computing (EC) provides different optimization algorithms to solve complex search problems. The process involved refers to a population of individuals in a changing environment with limited resources where they reproduce, evolve and compete to survive. Among the different paradigms within EC, Differential Evolution (DE) is one of the most recent and competitive to solve, mainly, problems in continuous search spaces [1]. Its main feature is the differential mutation, which is a variation operator to extract information from a group of individuals (called vectors) to lead the search to promising regions of the search space. DE was originally proposed for solving unconstrained optimization problems [1]. However, it has shown a very competitive performance when dealing with constrained search spaces [2]. Precisely in engineering, it is quite common to solve problems in presence of constraints. Therefore, DE has been added with different constraint-handling techniques (CHT's) to guide the search to the best-known feasible solution [3]. A Constrained Numerical Optimization Problem (CNOP) is stated, without loss of generality, as to:

$$\textit{minimize } f(\vec{x})$$

subject to:

$$\begin{aligned} g_i(\vec{x}) &\leq 0,\ i = 1, 2, ..., m \\ h_j(\vec{x}) &= 0,\ j = 1, 2, ..., r \end{aligned} \tag{1}$$

where $\vec{x} = [x_1,\ x_2,\ ...,\ x_n]^\intercal$, $\vec{x} \in R^n$ is the design vector, whose elements are known as design variables which must satisfy boundary constraints, $L_i^l \leq x_i \leq L_i^u$, $i = 1 \ldots, n$, n represents the dimensions of the problem, $f(\vec{x})$, $g_i(\vec{x})$ and $h_j(\vec{x})$ are the objective function, the inequality constraints and the equality constraints, respectively.

An important amount of research has been devoted to design techniques to deal with the explicit equality and inequality constraints in EC [2], as those expressed in Eq. (1). In contrast, the satisfaction of boundary constraints seems to attract less attention when an Evolutionary Algorithm (EA) is adopted to solve a given optimization problem. As an example, in the case of DE, once the differential mutation operator is applied, the value of a given design variable may violate its upper or lower bound.

To tackle the presence of such invalid values there are boundary constraint-handling methods (BCHM's) [4]. However, particularly for CNOPs, the effect of a given BCHM is usually not analyzed and it is a common practice to just adopt one of them without further information. There are some works related with comparisons of different BCHM's in DE [5], [6] and also in Swarm Intelligence Algorithms (SIA's) such as Particle Swarm Optimization (PSO) [7]–[9]. However, they are focused on unconstrained optimization. Regarding CNOPs, the research is still scarce, only a few comparisons can be found in the literature [10], [11], but mainly using benchmark problems. In all the studies discussed, the authors concluded that there is an impact in the search algorithm regarding the BCHM adopted.

Motivated by the above mentioned, this paper presents an empirical assessment of different BCHM's in DE when solving four real-world mechanical design constrained optimization problems related to four-bar mechanisms. Besides final results, performance measures for constrained optimization and the

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

number of repaired variables and vectors are reported and discussed. The 95%-confidence Kruskal-Wallis and Bonferroni post-hoc tests are used to validate the findings in this paper.

The document is organized as follows: Section II briefly introduces DE. In Section III, the BCHM's used in this work are described. After that, Section IV presents the four real-world constrained mechanical design problems. Next, the settings performed to carry out the experiments are shown in Section V. The results are analyzed and discussed in Section VI. Finally, Section VII presents the conclusions and future work of this research.

II. Differential Evolution

DE was proposed by Storn and Price [1], where three user-defined parameters are required: the population size NP, a scale factor F and the crossover probability CR. The DE steps are the following:

1) Create an initial population of NP vectors. A population $P_{x,g} = x_{j,i,g}$, $i = 0, \ldots, NP-1$, $g = 0, \ldots, g_{max}$, $j = 0, 1, ..., D-1$ is defined randomly with uniform distribution, where g is the current generation, g_{max} is the maximum number of generations and D is the length (dimensionality) of the vector.
2) As long as the stop condition is not met, for each vector $x_{i,g}$ in the population (target vector), the mutation process is carried out. First, three integers are randomly generated, $r0 \neq r1 \neq r2 \neq i \in [1, NP]$, which represent the vectors indexes in the population. Vector subtraction is performed $(x_{r1,g} - x_{r2,g})$ and the resultant vector, called difference vector, is added to the third vector, known as base vector $(x_{r0,g})$, producing the mutant vector $v_{i,j}$. A complete representation of the aforementioned operator is in Eq. (2).

$$v_{i,g} = x_{r0,g} + F * (x_{r1,g} - x_{r2,g}), \; F > 0 \tag{2}$$

   Precisely at this point, the values of the mutant vector are verified so as they must be within the lower and upper limits defined by the optimization problem. If such values are outside the boundaries, the BCHM is then activated.
3) After mutation, the crossover between $x_{i,g}$ and $v_{i,g}$ is carried out to create the offspring $u_{i,g}$ called trial vector as detailed in Eq. (3).

$$u_{j,i,g,} = \begin{cases} v_{j,i,g} & \text{if } rand(0,1) \leq CR \;||\; j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{3}$$

   where $j_{rand} \in [1...D]$ ensures that the trial vector is different from the target vector by including in it at least one of the $v_{i,g}$ elements.
4) For the survivor selection, check if the trial vector is better than the target vector, selecting the one with the best value of the objective function as in Eq. (4).

$$x_{i,g+1} = \begin{cases} u_{i,g} & \text{if } f(u_{i,g}) \leq f(x_{i,g}) \\ x_{i,g} & \text{otherwise} \end{cases} \tag{4}$$

Considering the fact that in this work CNOPs are solved, a constraint-handling technique for the explicit constraints must be added to DE. Based on [2], the set of feasibility rules proposed by Deb [12] is one of the most suitable to solve constrained problems. Therefore, they are adopted in this research and detailed below:

- Between two feasible vectors, the one with the best objective function value is selected.
- A feasible vector is preferred over an infeasible one.
- Between two infeasible vectors, the one with the smallest sum of constraint violation $\phi_{(x)}$, see Eq. (5), is selected.

$$\phi_{(x)} = \sum_{j=1}^{m} max(0, g_j(x)) \tag{5}$$

III. Boundary Constraint-Handling Methods

This section presents the BCHM's used in this work. Their details were taken from [6], [10]. L_j^u and L_j^l, represent the upper and lower limits of variable j, respectively. For sake of clarity in what follows in the paper, a vector $x_{i,g}$ or $v_{i,g}$ will be represented as x_i or v_i i.e., the generation g is omitted as it is not relevant for the BCHM's.

A. MidPoint Target

This method uses the information of the target vector x_i. If a variable j of the mutant vector v_i exceeds its limits, the corrected value is computed using the variable j of the target vector plus the violated limit divided by two, see Eq. (6):

$$v_{i,j} = \begin{cases} (L_j^l + x_{j,i})/2 & if \; v_{j,i} < L_j^l \\ (L_j^u + x_{j,i})/2 & if \; v_{j,i} > L_j^u \end{cases} \tag{6}$$

B. Reflection

This method replaces the invalid value by computing the scaled difference of the exceeded bound multiplied by two minus the invalid value, see Eq. (7):

$$v_{j,i} = \begin{cases} (L_j^l * 2) - v_{j,i} & if \, v_{j,i} < L_j^l \\ (L_j^u * 2) - v_{j,i} & if \, v_{j,i} > L_j^u \end{cases} \tag{7}$$

C. Projection

The invalid value is truncated to the nearest limit, see Eq. (8):

$$v_{j,i} = \begin{cases} L_j^l & if \, v_{j,i} < L_j^l \\ L_j^u & if \, v_{j,i} > L_j^u \end{cases} \tag{8}$$

D. Random Scheme

The invalid value is repaired by computing a random number between its established limits as in Eq. (9):

$$v_{j,i} = L_j^l + rand(0,1) * (L_j^u - L_j^l) \tag{9}$$

where rand(0,1) is a random real number between 0 and 1 generated with uniform distribution.

E. Reinitialize All

If at least one variable is outside its limits, a new vector is generated in a similar way as the Random Scheme in Eq. (9), but now for all variables.

F. Conservatism

If at least one variable of the mutant vector v_i exceeds its limits, the target vector x_i is preserved, see Eq. (10):

$$\begin{aligned} &\text{if } \exists\, v_{j,i} \in v_i : v_{j,i} < L_j^l \,|\, v_{j,i} > L_j^u \\ &\quad v_i = x_i \\ &\text{endif} \end{aligned} \tag{10}$$

G. Resampling

If one variable is outside of its limits the differential mutation is carried out again, as in Eq. (11):

$$\begin{aligned} &\text{while } \exists\, v_{j,i} \in v_i : v_{j,i} < L_j^l \,|\, v_{j,i} > L_j^u \\ &\quad v_i = \text{differential mutation as in Eq. 2} \\ &\text{end} \end{aligned} \tag{11}$$

H. Evolutionary

Using the best vector found so far x_{best}, the invalid value is corrected as in Eq. (12):

$$v_{j,i} = \begin{cases} \beta * L_j^l + (1-\beta) * x_{j,best} & if\ v_{j,i} < L_j^l \\ \alpha * L_j^u + (1-\alpha) * x_{j,best} & if\ v_{j,i} > L_j^u \end{cases} \tag{12}$$

where β and α are random numbers between 0 and 1 generated with uniform distribution.

I. Centroid K+1

If one value of the mutant vector violates its bounds, the centroid v_c is computed as follows:

1) $K+1$ vectors are required to compute the corrected vector. The first vector, called W_p, is taken from the current population as indicated in Eq. (13).

$$W_p = \begin{cases} x_{rand} \in \text{FS} & \text{If } FS \neq \emptyset \text{ and rand(0,1)} > 0.5 \\ x_{best} \in \text{IS} & \text{otherwise} \end{cases} \tag{13}$$

where FS and IS are the sets of feasible and infeasible vectors in the current population, respectively; rand(0,1) generates a real random number between 0 and 1 with uniform distribution, x_{rand} is a randomly chosen vector from the feasible solutions subset of the current population and x_{best} is the best vector in the infeasible set, i.e., the one with the lowest sum of constraint violation as in Eq. (5). If all solutions are feasible, W_p is taken from the FS set.

2) The remaining K vectors share the variables of the mutant vector with valid values, while the violated ones are replaced by randomly generated valid values. The process is as follows.

```
for i = 1 to K do
  w_i = v_i
  for j = 1 to D do
    if w_{i,j} < L_j^l or w_{i,j} > L_j^u then
      w_{i,j} = L_j^l + rand(0,1) * (L_j^u - L_j^l)
    end if
  end for
end for
```

3) Finally, to get the corrected vector Eq. (14) is applied:

$$v_c = \{w_p + w_1 + w_2, ..., w_k\}/(K+1); \tag{14}$$

IV. Mechanical Design Optimization Problems

This section presents the four CNOPs (P01, P02, P03, P04) solved in the experiments. As all of them are based on the same type of mechanisms, the common features are described first. After that, each problem is detailed.

A. Four-bar mechanism description

Based in Fig. 1, where the four-bar mechanism solved in this work is presented, Eq. (15) includes the corresponding 9-dimension decision vector, where each variable is mapped to a feature of the mechanism.

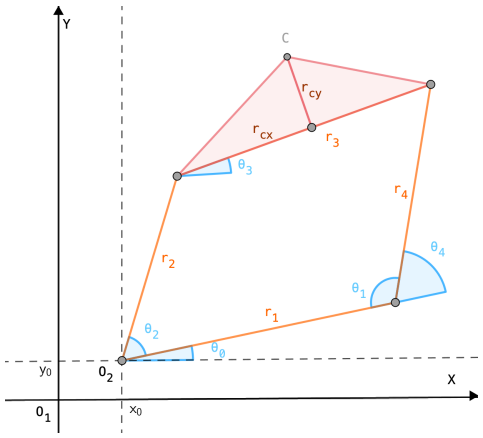

Fig. 1: Four-Bar mechanism.

$$\begin{aligned} \overrightarrow{p} &= [p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9]\,, \\ &= [r_1, r_2, r_3, r_4, r_{cx}, r_{cy}, \theta_0, x_0, y_0]\,, \end{aligned} \tag{15}$$

Variables r_1, r_2, r_3, r_4 correspond to the bar lengths, r_{cx}, r_{cy} correspond to the coupler position, θ_0 is the mechanism movement angle concerning the horizontal axis of the second system, and $O_2(x_0, y_0)$ is the starting point of the system.

The task developed by the four-bar mechanism is to pass through a set of precision points. Therefore the CNOP is defined as follows:

$$\text{Minimize } error = \sum_{i=1}^{n} \left[\left(C_{xd}^i - C_x^i \right)^2 + \left(C_{yd}^i - C_y^i \right)^2 \right] \tag{16}$$

Subject to :

$$\begin{aligned} g_1(\overrightarrow{p}) &= p_1 + p_2 - p_3 - p_4 \leq 0, \\ g_2(\overrightarrow{p}) &= p_2 - p_3 \leq 0, \\ g_3(\overrightarrow{p}) &= p_3 - p_4 \leq 0, \\ g_4(\overrightarrow{p}) &= p_4 - p_1 \leq 0, \end{aligned} \tag{17}$$

where $C_d^i = \left[C_{xd}^i, C_{yd}^i\right]^T$ is a precision point that defines the trajectory, a set of them as $\Omega = \{C_d^i | i \in N\}$ where N is the total number of points; and $C^i = \left[C_x^i, C_y^i\right]$, a generated point. The kinematics of the mechanisms can be found in [13], [14].

For mechanisms that must pass through pairs of precision points, the proposed function is as follows:

$$\begin{aligned} &\text{Minimize } error = Error_1 + Error_2, \\ &Error_1 = \sum_{i=1}^{n} \left[\left(C_{1xd}^i - C_x^i\right)^2 + \left(C_{1yd}^i - C_y^i\right)^2 \right], \\ &Error_2 = \sum_{i=1}^{n} \left[\left(C_{2xd}^i - C_x^i\right)^2 + \left(C_{2yd}^i - C_y^i\right)^2 \right]. \end{aligned} \tag{18}$$

subject to the same constraints in Eq. (17). It is important to remark that the calculations of the objective functions in Eq. (16) and Eq. (18) require the computation of the mechanism kinematics and such process is considered as nonlinear.

B. P01: mechanism that follows a vertical linear path

In this CNOP, taken from [15], the mechanism must follow a vertical linear path defined by six precision points:

$$\Omega = \{(20,20), (20,25), (20,30), (20,35), (20,40), (20,45)\} \tag{19}$$

The boundaries defined for each one of the nine design variables are:

$$\begin{aligned} &r_1, r_2, r_3, r_4 \in [0, 60] \\ &r_{cx}, r_{cy}, x_0, y_0 \in [-60, 60] \\ &\theta_0 \in [0, 2\pi] \end{aligned} \tag{20}$$

The objective function to this mechanical design problem is presented in Eq. (16), subject to constraints in Eq. (17).

C. P02: mechanism that follows a curve path defined by five precision points

In this CNOP the mechanism must follow five precision points that form a curve. The precision points are:

$$\begin{aligned} \Omega = \{&(3,3), (2.759, 3.363), (2.372, 3.663), \\ &(1.890, 3.862), (1.355, 3.943)\} \end{aligned} \tag{21}$$

The boundaries of the nine variables are:

$$\begin{aligned} &r_1, r_2, r_3, r_4 \in [0, 50] \\ &r_{cx}, r_{cy} \in [-50, 50] \\ &x_0, y_0, \theta_0 = 0 \end{aligned} \tag{22}$$

The objective function associated with this problem is found in Eq. (16), subject to the constraints in Eq. (17).

D. P03: mechanism that follows a path defined by ten pairs of precision points

In this case, the mechanism must pass through ten pairs of precision points that form a circle. The precision points pairs are presented in Table I.

TABLE I: Precision points pairs for problem P03

Pair	C_{1d}	C_{2d}
1	$(1.768, 2.3311)$	$(1.9592, 2.44973)$
2	$(1.947, 2.6271)$	$(2.168, 2.675)$
3	$(1.595, 2.7951)$	$(1.821, 2.804)$
4	$(1.019, 2.7241)$	$(1.244, 2.720)$
5	$(0.479, 2.4281)$	$(0.705, 2.437)$
6	$(0.126, 2.0521)$	$(0.346, 2.104)$
7	$(-0.001, 1.720)$	$(0.195, 1.833)$
8	$(0.103, 1.514)$	$(0.356, 1.680)$
9	$(0.442, 1.549)$	$(0.558, 1.742)$
10	$(1.055, 1.905)$	$(1.186, 2.088)$

The suggested upper and lower limits for each one of the nine variables are the following:

$$\begin{aligned} &r_1, r_2, r_3, r_4 \in [0, 60], \\ &r_{cx}, r_{cy}, x_0, y_0 \in [-60, 60], \\ &\theta_0 \in [0, 2\pi]. \end{aligned} \tag{23}$$

The objective function considered in this problem is described in Eq. (18), subject to the constraints in Eq. (17).

E. P04: mechanism that follows an elliptical path defined by ten precision points

The mechanism in this CNOP must follow an elliptical path defined by the next ten precision points:

$$\begin{aligned} \Omega = \{&(20,10), (17.66, 15.142), (11.736, 17.878), (5, 16.928), \\ &(0.60307, 12.736), (0.60307, 7.2638), (5, 3.0718), \\ &(11.736, 2.1215), (17.66, 4.8577), (20,10)\} \end{aligned} \tag{24}$$

The limits for each one of the nine decision variables are:

$$\begin{aligned} &r_1, r_2, r_3, r_4 \in [5, 80], \\ &r_{cx}, r_{cy} \in [0, 80], \\ &x_0, y_0 \in [-80, 80], \\ &\theta_0 \in [0, 2\pi]. \end{aligned} \tag{25}$$

The objective function for this CNOP is described in Eq. (16), subject to the constraints in Eq. (17).

V. Experimental Design

In this work the nine BCHM's were added to the DE variant presented in Section II with the set of feasibility rules already presented. For all DE versions, 30 independent runs were performed using the following parameters: $NP = 100$, $CR \in [0.8, 1]$ generated at random at each generation, and $F \in [0.3, 0.9]$ generated also at random per individual. The stop criterion was set to 400,000 solution evaluations (MAX_NFE) for P01, MAX_NFE =15,000 for P02, MAX_NFE =200,000 for P03 and MAX_NFE =50,000 for P04. The 95%-confidence Kruskal-Wallis and Bonferroni post-hoc tests were computed

to validate the obtained results. The number of variables and individuals repaired were counted to get insights about the cost associated to each BCHM.

The following concepts are considered in the performance measures adopted [16]: a feasible solution is one that satisfies all the constraints of a CNOP. A feasible run is that with at least one feasible solution found. A successful run is that with at least one successful solution, where a successful solution is that close to the best known solution i.e., $|f(x_{gbest}) - f(x_{best})| \leq 1 \times 10^{-3}$, where $f(x_{gbest})$ corresponds to the best known feasible solution and $f(x_{best})$ is the feasible solution found by the algorithm under study. The value 10^{-3} was chosen because it is suitable for the type of task made by the mechanisms optimized in this work. The measures that help to complement the final statistical results shown later in this paper are the following:

- **Feasibility probability (FP):** number of feasible runs divided by the total number of independent runs. FP values $\in [0, 1]$ and a high value is desirable.
- **Convergence probability (P):** number of successful runs divided by the total number of independent runs. P values $\in [0, 1]$ and a high value is desirable.
- **Average number of function evaluations (AFES):** average number of evaluations required to find the first successful solution in a set of successful runs, see Eq. (26), where a low value is preferable.

$$AFES = \frac{\sum \text{evaluations to find the first successful solution}}{\text{number of successful runs}} \tag{26}$$

- **Successful performance (SP):** measures the capacity and reliability of an algorithm. It is computed as indicated in Eq. (27), where a low value is preferable.

$$SP = \frac{AFES}{P} \tag{27}$$

- **Progress ratio (PR):** estimates the ability of an algorithm to improve the solutions within the feasible region. It is calculated by the difference between the value of the objective function of the first ($f(x_{first})$) and the last ($f(x_{last})$) feasible solution found in a feasible run, a high value is preferred, see Eq. (28):

$$PR = |f(x_{first}) - f(x_{last})| \tag{28}$$

VI. Results and Discussion

Tables II, III, IV and V show the final statistical results, the number of feasible and successful runs and the number of repaired variables and vectors, obtained by all DE versions with different BCHM's in the four CNOPs presented in Section IV. Table VI presents the results obtained by the 95%-confidence Kruskal-Wallis test, which indicates significant differences in the last three problems, based on final results, and in two problems regarding the PR measure. Centroid K+1 was executed with two K values (K=1,2). Moreover, the results of the Bonferroni post-hoc test, based on the final results in Tables II, III, IV, V and on the 95%-confidence Kruskal-Wallis test results in Table VI, are summarized in Fig. 2.

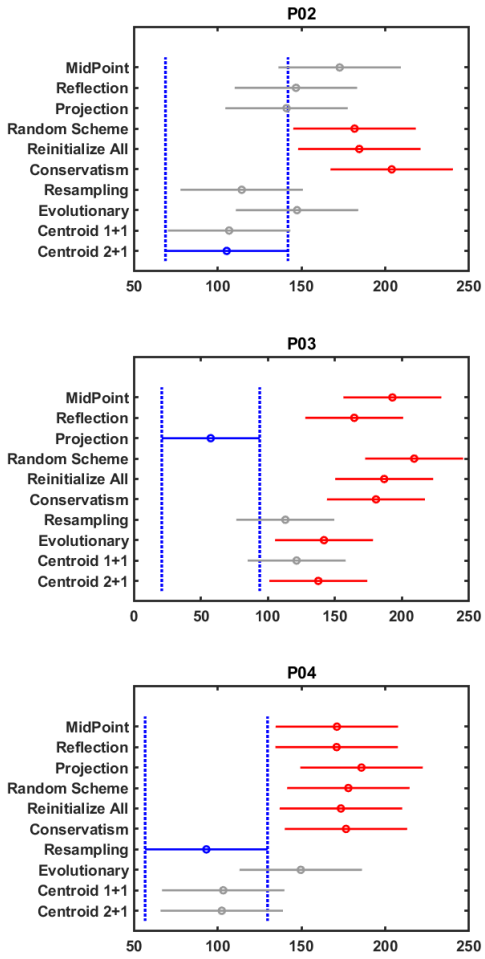

Fig. 2: Bonferroni post-hoc test results based on final results in test problems where significant differences were found.

Regarding final results, in problem P01 no significant differences were observed. In problem P02 Centroid K+1 (K=1,2) achieved statistically significant better results (confidence intervals located at the left part without intersecting other intervals in the Bonferroni test graph) than Random Scheme, Reinitialize all and Conservatism (Fig 2 and Table III). Similarly, in problem P03 the Projection method provided statistically significant better results than MidPoint, Reflection, Random Scheme, Reinitialize all, Conservatism, Evolutionary and Centroid 2+1 (Fig 2 and Table IV). Finally, in problem P04, Resampling statistically outperformed MidPoint, Reflection, Projection, Random Scheme, Reinitialize all and Conservatism (Fig 2 and Table V).

It is interesting to note that, in all four test problems, Centroid K+1 was the BCHM that repaired less variables and vectors (see Fig. 4). Therefore, this BCHM promotes the usage of the differential mutation operator with less repairs.

TABLE II: Final results and repair numbers obtained by all DE versions in problem P01

BCHM	Best	Worst	Average	Median	Std.	Feasible runs	successful runs	Repaired variables	Repaired vectors
MidPoint	2.1730E-04	4.0357E-02	4.8594E-03	3.1762E-03	7.2149E-03	30	5	1.9922E+06	1.5358E+06
Reflection	1.9054E-04	1.8598E-02	4.8286E-03	3.3671E-03	5.2071E-03	30	12	2.3120E+06	1.7043E+06
Projection	4.0060E-04	1.2712E+00	4.9071E-02	4.0454E-03	2.3101E-01	30	2	3.8435E+06	2.8819E+06
Random Scheme	2.0947E-04	1.4689E-02	3.8981E-03	2.4718E-03	3.5338E-03	30	4	1.5101E+06	1.2284E+06
Reinitialize All	2.8960E-04	2.0282E-02	5.4137E-03	3.5144E-03	5.0372E-03	30	7	N/A	1.3040E+06
Conservatism	1.1992E-04	1.5888E-02	3.1455E-03	2.5744E-03	3.3134E-03	30	11	N/A	1.3492E+06
Resampling	1.6872E-04	2.1291E-02	4.1678E-03	2.5183E-03	4.7358E-03	30	11	N/A	1.1231E+06
Evolutionary	2.4626E-04	1.5888E-02	3.6447E-03	2.9931E-03	3.9000E-03	30	9	2.0095E+06	1.5846E+06
Centroid 1+1	**1.0564E-04**	1.5273E-02	5.2491E-03	4.8037E-03	3.7495E-03	30	4	**1.4621E+06**	1.2777E+06
Centroid 2+1	**1.6427E-04**	**9.8068E-03**	**2.9143E-03**	**2.2105E-03**	**2.6859E-03**	30	8	2.3168E+06	**9.9035E+05**

TABLE III: Final results and repair numbers obtained by all DE versions in problem P02

BCHM	Best	Worst	Average	Median	Std.	Feasible runs	successful runs	Repaired variables	Repaired vectors
MidPoint	9.1274E-04	1.0489E-01	1.8448E-02	1.1895E-03	3.5243E-02	30	23	1.4810E+05	1.2052E+05
Reflection	9.2561E-04	1.0473E-01	1.0322E-02	1.0532E-03	2.6387E-02	30	25	1.8066E+05	1.4398E+05
Projection	7.7605E-04	1.0440E-01	1.4226E-02	1.0725E-03	3.3662E-02	30	25	2.9512E+05	2.1527E+05
Random Scheme	9.0323E-04	6.6662E-02	4.9995E-03	1.3000E-03	1.4201E-02	30	28	1.2378E+05	1.0256E+05
Reinitialize All	8.6312E-04	1.0372E-01	1.2072E-02	1.4958E-03	2.8039E-02	30	24	N/A	1.0335E+05
Conservatism	9.0596E-04	1.0517E-01	1.4353E-02	1.5734E-03	3.1497E-02	30	22	N/A	1.0204E+05
Resampling	**5.5904E-04**	**6.4613E-02**	5.2591E-03	1.0072E-03	1.5054E-02	30	27	N/A	9.0725E+04
Evolutionary	7.9034E-04	1.0341E-01	1.3447E-02	1.0550E-03	3.1503E-02	30	24	1.6367E+05	1.3045E+05
Centroid 1+1	6.2198E-04	9.2475E-02	5.5269E-03	1.0169E-03	1.8295E-02	30	28	**7.5594E+04**	**6.4877E+04**
Centroid 2+1	7.8352E-04	6.9362E-02	**3.3124E-03**	**9.9181E-04**	**1.2476E-02**	30	**29**	1.6065E+05	6.9170E+04

TABLE IV: Final results and repair numbers obtained by all DE versions in problem P03

BCHM	Best	Worst	Average	Median	Std.	Feasible runs	successful runs	Repaired variables	Repaired vectors
MidPoint	4.8616E-01	1.7845E+00	9.9770E-01	9.4558E-01	3.8521E-01	30	0	3.9284E+06	2.4737E+06
Reflection	4.0268E-01	1.7198E+00	8.3086E-01	7.7351E-01	3.0477E-01	30	0	6.1600E+06	3.2574E+06
Projection	4.3029E-01	**9.4113E-01**	**5.1605E-01**	**5.0680E-01**	**8.8929E-02**	30	0	5.5039E+06	2.6152E+06
Random Scheme	5.3791E-01	1.7658E+00	1.0621E+00	9.9627E-01	4.1314E-01	30	0	3.4898E+06	2.3580E+06
Reinitialize All	**3.7041E-01**	1.6888E+00	9.5691E-01	8.6221E-01	4.0538E-01	30	0	0.0000E+00	2.0478E+06
Conservatism	4.5717E-01	1.7956E+00	9.7326E-01	8.2090E-01	4.3638E-01	30	0	0.0000E+00	1.6520E+06
Resampling	4.3525E-01	1.6433E+00	6.7749E-01	5.6253E-01	2.9266E-01	30	0	0.0000E+00	1.4356E+06
Evolutionary	4.4428E-01	1.5859E+00	7.5892E-01	6.1503E-01	3.1117E-01	30	0	4.5470E+06	2.6830E+06
Centroid 1+1	4.4762E-01	1.5681E+00	7.0884E-01	5.7483E-01	3.2259E-01	30	0	**1.0986E+06**	**8.8311E+05**
Centroid 2+1	4.4851E-01	1.5316E+00	7.3719E-01	6.2618E-01	2.8712E-01	30	0	2.8502E+06	1.1062E+06

TABLE V: Final results and repair numbers obtained by all DE versions in problem P04

BCHM	Best	Worst	Average	Median	Std.	Feasible runs	successful runs	Repaired variables	Repaired vectors
MidPoint	3.8560E-02	2.3512E-01	7.6937E-02	6.6987E-02	4.3286E-02	30	0	7.4308E+05	4.8165E+05
Reflection	4.3135E-02	2.2850E+01	1.2433E+00	6.8099E-02	4.6689E+00	30	0	1.0182E+06	6.0597E+05
Projection	**2.2631E-02**	4.0253E-01	1.1089E-01	6.6208E-02	8.2078E-02	30	**1**	1.5105E+06	7.9320E+05
Random Scheme	3.8172E-02	1.7347E-01	7.6603E-02	6.6599E-02	3.1007E-02	30	0	6.7384E+05	4.5265E+05
Reinitialize All	4.1062E-02	2.6423E+01	1.0028E+00	6.5719E-02	4.8059E+00	30	0	N/A	4.8015E+05
Conservatism	4.9397E-02	1.2705E-01	7.0316E-02	6.9623E-02	1.7114E-02	30	0	N/A	3.9676E+05
Resampling	3.8304E-02	**7.2765E-02**	**5.4795E-02**	5.6195E-02	**8.8884E-03**	30	0	N/A	3.1403E+05
Evolutionary	3.9875E-02	5.2366E-01	8.2096E-02	6.1884E-02	8.7836E-02	30	0	7.2743E+05	4.7649E+05
Centroid 1+1	3.4263E-02	8.1309E-02	5.6076E-02	**5.5074E-02**	1.1234E-02	30	0	**2.8603E+05**	**2.2622E+05**
Centroid 2+1	4.4702E-02	1.0241E-01	5.7997E-02	5.6462E-02	1.2769E-02	30	0	6.8487E+05	2.6458E+05

Tables VII and VIII include the FP, AFES, P and SP measures results in two out of four mechanical design problems. Problem P03 was omitted because, even all runs were feasible for all DE versions (i.e., FP = 1), no successful runs were reported in all cases (i.e., P = 0), then AFES and SP could not be computed. In problem P04 only Projection reported values: FP = 1, AFES = 4.4642E+04, P = 0.03, and SP = 1.3393E+06. Moreover, Tables IX, X, XI and XII include the statistical results of the PR measure and Fig. 3 presents the results of the 95%-confidence Kruskal-Wallis and Bonferroni post-hoc tests based on the PR results.

The FP results indicate that all DE versions were able to consistently reach the feasible region in the four test problems, including P03 and P04. On the other hand, for the P measure there is not a clear pattern, because Reflection was better in problem P01, while Centroid 2+1 was better in problem P02 and Projection was the only method to provide successful runs in problem P04.

A similar diverse behavior was observed for the AFES measure because Resampling was better in problem P01, Centroid 1+1 outperformed their counterparts in problem P02, and Projection was the only method with results in problem P04.

The results of the SP measure provide more clarity in the performance presented by the compared BCHM's. This is because Resampling was better in problem P01 (similar to the AFES results), Centroid 2+1 was better in problem P02 (similar to the P results), and Projection was the only method with computed results in problem P04.

The PR results indicate that Projection was statistically better (confidence intervals now located at the right part without intersecting other intervals in the Bonferroni test graph) in problem P02 (Fig. 3 and Table X), with respect to almost all the compared methods, with the exception of Reflection and Conservatism. Moreover, in problem P03 Projection provided slightly better results, i.e., almost significantly better than Centroid 1+1 but with no significant differences with respect to the other methods.

From the above discussed overall results it was found that Projection provided better quality results, while Centroid K+1 performed the lowest number of repairs.

TABLE VI: Kruskal-Wallis Test Results

Problem	*p-value (Problems)*	*p-value (Progress Ratio)*
P01	9.6273E-02	7.0278E-02
P02	**2.0283E-06**	**6.0518E-08**
P03	**1.6303E-12**	**2.6316E-02**
P04	**3.4748E-07**	4.3929E-01

TABLE VII: FP, AFES, P and SP results in problem P01

BCHM	FP	AFES	P	SP
MidPoint	1	2.4647E+05	0.17	1.4788E+06
Reflection	1	1.8830E+05	**0.40**	4.7075E+05
Projection	1	2.2032E+05	0.07	3.3049E+06
Random Scheme	1	1.8697E+05	0.13	1.4023E+06
Reinitialize All	1	2.1451E+05	0.23	9.1931E+05
Conservatism	1	1.8762E+05	0.37	5.1170E+05
Resampling	1	**1.6541E+05**	0.37	**4.5112E+05**
Evolutionary	1	1.9300E+05	0.30	6.4334E+05
Centroid 1+1	1	1.9832E+05	0.13	1.4874E+06
Centroid 2+1	1	1.7339E+05	0.27	6.5021E+05

TABLE VIII: FP, AFES, P and SP results in problem P02

BCHM	FP	AFES	P	SP
MidPoint	1	1.0719E+04	0.77	1.3981E+04
Reflection	1	1.0422E+04	0.83	1.2506E+04
Projection	1	1.0718E+04	0.83	1.2861E+04
Random Scheme	1	1.1716E+04	0.93	1.2553E+04
Reinitialize All	1	1.1956E+04	0.80	1.4945E+04
Conservatism	1	1.1884E+04	0.73	1.6206E+04
Resampling	1	9.8455E+03	0.90	1.0939E+04
Evolutionary	1	1.0114E+04	0.80	1.2642E+04
Centroid 1+1	1	**9.8376E+03**	0.93	1.0540E+04
Centroid 2+1	1	1.0048E+04	**0.97**	**1.0394E+04**

TABLE IX: Progress ratio results in problem P01

BCHM	Best	Worst	Mean	Std.
MidPoint	7.86E+04	1.17E+02	1.61E+04	2.01E+04
Reflection	1.02E+05	3.27E+02	2.89E+04	2.73E+04
Projection	**1.62E+05**	**9.60E+02**	**3.17E+04**	**3.42E+04**
Random Scheme	1.16E+05	9.72E+01	2.61E+04	3.19E+04
Reinitialize All	6.39E+04	2.53E+02	1.86E+04	1.92E+04
Conservatism	1.14E+05	1.51E+02	2.88E+04	2.87E+04
Resampling	8.87E+04	5.11E+02	2.30E+04	2.14E+04
Evolutionary	8.29E+04	1.44E+02	1.97E+04	2.37E+04
Centroid 1+1	5.86E+04	4.56E+02	1.64E+04	1.84E+04
Centroid 2+1	7.75E+04	2.83E+02	1.89E+04	1.96E+04

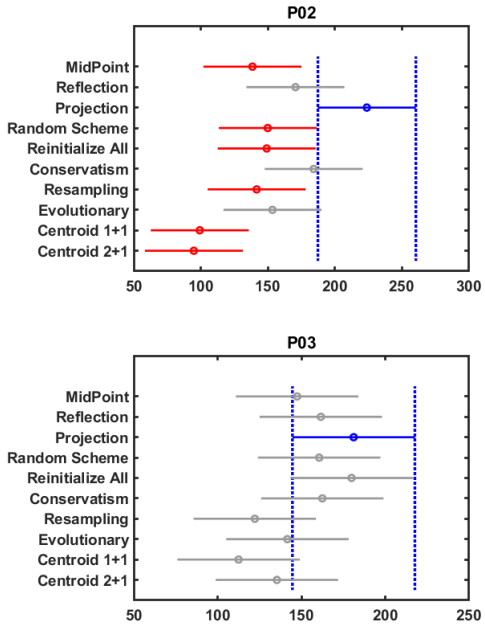

Fig. 3: Bonferroni post-hoc test results for the Progress ratio measure in test problems where significant differences were found.

TABLE X: Progress ratio results in problem P02

BCHM	Best	Worst	Mean	Std.
MidPoint	1.09E+04	2.51E+00	3.57E+03	3.05E+03
Reflection	1.55E+04	2.05E+01	5.61E+03	4.78E+03
Projection	2.33E+04	**2.50E+02**	**8.91E+03**	5.57E+03
Random Scheme	2.38E+04	3.84E+01	4.53E+03	4.90E+03
Reinitialize All	1.41E+04	2.59E+00	4.55E+03	4.32E+03
Conservatism	1.79E+04	3.76E+00	6.30E+03	5.09E+03
Resampling	**2.60E+04**	1.71E+00	4.80E+03	**5.83E+03**
Evolutionary	1.68E+04	2.03E+02	4.67E+03	4.85E+03
Centroid+1	1.12E+04	1.46E+00	2.19E+03	2.81E+03
Centroid+2	1.01E+04	7.19E-01	2.18E+03	2.88E+03

VII. Conclusions and Future Work

A set of Boundary Constraint-Handling Methods were added to the Differential Evolution algorithm to solve four real-world constrained mechanical design optimization problems. Based on the final results and also in the number of variables and vectors repaired, as well as in the results of five performance measures for constrained optimization, it can be concluded that the Projection method was the most consistent BCHM in this study, i.e., better overall final results, more successful runs and better improvement inside the feasible

TABLE XI: Progress ratio results in problem P03

BCHM	Best	Worst	Mean	Std.
MidPoint	1.88E+05	8.91E+01	4.54E+04	4.57E+04
Reflection	1.25E+05	**4.27E+02**	4.66E+04	3.54E+04
Projection	**5.14E+05**	2.24E+01	**1.00E+05**	**1.25E+05**
Random Scheme	2.31E+05	1.46E+01	6.12E+04	6.97E+04
Reinitialize All	2.25E+05	5.70E+01	6.96E+04	6.35E+04
Conservatism	2.42E+05	1.97E+01	6.66E+04	7.18E+04
Resampling	1.74E+05	3.34E+01	3.75E+04	4.76E+04
Evolutionary	3.06E+05	2.98E+02	5.56E+04	8.10E+04
Centroid+1	1.64E+05	1.14E+02	2.84E+04	3.71E+04
Centroid+2	1.12E+05	1.24E+02	3.58E+04	3.49E+04

TABLE XII: Progress ratio results in problem P04

BCHM	Best	Worst	Mean	Std.
MidPoint	1.63E+05	**1.14E+03**	3.97E+04	4.28E+04
Reflection	**2.07E+05**	4.21E+02	**5.49E+04**	4.88E+04
Projection	2.02E+05	4.22E+02	5.09E+04	5.22E+04
Random Scheme	1.75E+05	2.38E+02	4.33E+04	4.76E+04
Reinitialize All	1.37E+05	4.12E+02	4.25E+04	3.96E+04
Conservatism	1.44E+05	2.78E+02	4.96E+04	4.72E+04
Resampling	1.18E+05	4.52E+02	4.87E+04	3.50E+04
Evolutionary	1.87E+05	1.86E+02	5.39E+04	**5.31E+04**
Centroid+1	9.80E+04	5.81E+02	2.93E+04	2.72E+04

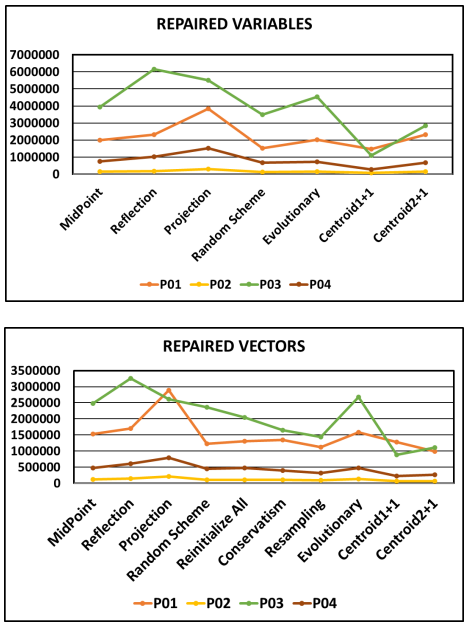

Fig. 4: Line graphs of variables and individuals repaired in all optimization problems.

region. However, the method which promoted less repairs was Centroid K+1 (particularly Centroid 1+1). Such findings are important because, even in this particular set of test problems, different behaviors were observed depending the BCHM adopted and such effect may be more evident in a wide range of optimization problems.

Part of the future work includes the combination of different BCHM's in the same algorithm with a suitable mechanism to mix them, as well as using constrained optimization problems of a different domain.

ACKNOWLEDGMENTS

The first author acknowledges support from the Mexican National Council of Science and Technology (CONACyT) through a scholarship to pursue graduate studies at LANIA. The second author acknowledges support from the University of Veracruz to carry out this research.

REFERENCES

[1] R. M. Storn and K. V. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, December 1997.

[2] E. Mezura-Montes and C. A. Coello Coello, "Constraint-Handling in Nature-Inspired Numerical Optimization: Past, Present and Future," *Swarm and Evolutionry Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[3] H. S. Bernardino, H. J. C. Barbosa, and J. S. Angelo, "Differential evolution with adaptive penalty and tournament selection for optimization including linear equality constraints," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–8.

[4] E. Juárez-Castillo, H.-G. Acosta-Mesa, and E. Mezura-Montes, "Adaptive boundary constraint-handling scheme for constrained optimization," *Soft Computing*, vol. 23, no. 17, pp. 8247–8280, 2019.

[5] J. Arabas, A. Szczepankiewicz, and T. Wroniak, "Experimental comparison of methods to handle boundary constraints in differential evolution," in *Parallel Problem Solving from Nature, PPSN XI*, R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 411–420.

[6] V. Kreischer, T. Tavares Magalhães, H. Barbosa, and E. Krempser, "Evaluation of bound constraints handling methods in differential evolution using the CEC2017 benchmark," 10 2017.

[7] A. H. Gandomi and A. R. Kashani, "Evolutionary bound constraint handling for particle swarm optimization," in *2016 4th International Symposium on Computational and Business Intelligence (ISCBI)*, Sep. 2016, pp. 148–152.

[8] E. T. Oldewage, A. P. Engelbrecht, and C. W. Cleghorn, "Boundary constraint handling techniques for particle swarm optimization in high dimensional problem spaces," in *Swarm Intelligence*, M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. Reina, and V. Trianni, Eds. Cham: Springer International Publishing, 2018, pp. 333–341.

[9] N. Padhye, K. Deb, and P. Mittal, "Boundary handling approaches in particle swarm optimization," in *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)*, J. C. Bansal, P. K. Singh, K. Deep, M. Pant, and A. K. Nagar, Eds. India: Springer India, 2013, pp. 287–298.

[10] E. Juárez-Castillo, N. Pérez-Castro, and E. Mezura-Montes, "An improved centroid-based boundary constraint-handling method in differential evolution for constrained optimization," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 31, no. 11, p. 1759023, 2017. [Online]. Available: https://doi.org/10.1142/S0218001417590236

[11] S. Sapre and S. Mini, "Opposition-based moth flame optimization with cauchy mutation and evolutionary boundary constraint handling for global optimization," *Soft Computing*, 10 2018.

[12] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311–338, June 2000.

[13] M.-F. Zapata-Zapata, E. Mezura-Montes, and E.-A. Portilla-Flores, "Differential Evolution with parameter-memory to optimize four-bar mechanisms," *Research in Computing Science*, vol. 134, pp. 9–22, 2017, (in Spanish).

[14] B. Hernández, M. del P. Pozos, E. Mezura, E. A. Portilla, E. Vega, and M. B. Calva, "Two-Swim Operators in the Modified Bacterial Foraging Algorithm for the Optimal Synthesis of Four-Bar Mechanisms," *Hindawi Publishing Corporation Computational Intelligence and Neuroscience*, p. 18, January 2016.

[15] S. Sleesongsom and S. Bureerat, "Four-bar linkage path generation through self-adaptive population size teaching-learning based optimization," *Knowledge-Based Systems*, August 2017.

[16] H. Cervantes-Culebro, C. A. Cruz-Villar, M.-G. Martínez-Peñaloza, and E. Mezura-Montes, "Constraint-Handling Techniques for the Concurrent Design of a Five-Bar Parallel Robot," *IEEE Access*, vol. 5, pp. 23 010–23 021, 2017.