International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 31, No. 11 (2017) 1759023 (49 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218001417590236

An Improved Centroid-Based Boundary Constraint-Handling Method in Differential Evolution for Constrained Optimization

Efrén Juárez-Castillo*, Nancy Pérez-Castro† and Efrén Mezura-Montes‡
Artificial Intelligence Research Center, University of Veracruz
Sebastián Camacho #5, Centro
Xalapa, Veracruz, 91000, MÉXICO
**juarezefren@hotmail.com*
†perez.castro.nancy@gmail.com
‡emezura@uv.mx

Received 6 November 2016
Accepted 6 March 2017
Published 15 May 2017

Differential Evolution (DE) is a population-based Evolutionary Algorithm (EA) for solving optimization problems over continuous spaces. Many optimization problems are constrained and have a bounded search space from which some vectors leave when the mutation operator of DE is applied. Therefore, it is necessary the use of a boundary constraint-handling method (BCHM) in order to repair the invalid mutant vectors. This paper presents a generalized and improved version of the *Centroid* BCHM in order to keep the search within the valid ranges of decision variables in constrained numerical optimization problems (CNOPs), which has been tested on a robust and comprehensive set of experiments that include a variant of DE specialized in dealing with CNOPs. This new version, named *Centroid* $K + 1$, relocates the mutant vector in the centroid formed by K random vectors and one vector taken from the population that is within or near the feasible region. The results show that this new version has a major impact on the algorithm's performance, and it is able to promote better final results through the improvement of both, the approach to the feasible region and the ability to generate better solutions.

Keywords: Boundary; constraint-handling; differential evolution; CNOP; centroid.

1. Introduction

In several areas such as engineering or industry, practitioners and scientists have been attracted to the use of Evolutionary Computing (EC) to tackle real-world problems that are difficult to solve with traditional computational strategies.[3] The constrained numerical optimization problems (CNOPs) represent a huge amount of real-world problems that can be treated with EC algorithms such as Evolutionary Algorithms (EAs), Swarm Intelligence Algorithms (SIAs) or hybrid evolutionary approaches.[28,38]

CNOPs are usually written as a nonlinear programming (NLP) problem[7] which can be defined as follows:

$$\text{Minimize}: \quad f(\mathbf{x}) \tag{1}$$

Subject to:

$$l_i \leq x_i \leq u_i, \tag{2}$$

$$g_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, J, \tag{3}$$

$$h_k(\mathbf{x}) = 0, \quad k = 1, \ldots, K, \tag{4}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$ is the solution vector, and $f(\mathbf{x})$ is the objective function.

Equations (2)–(4) represent the constraints and can be classified into two types.[17] Constraints type a, or *boundary constraints*, that limit the values of the decision variables, defined in Eq. (2), where each variable x_i in $\mathbf{x}$ is bounded by lower and upper limits which define the search space $\mathcal{S}$. Constraints type b, or *constraint functions*, represent a more complex type of constraints which are defined as functions. Equations (3) and (4) represent the constraints type b, where $g_i(\mathbf{x})$ is the ith inequality constraint, $h_k(\mathbf{x})$ is the kth equality constraint.

The equality constraints are usually transformed into inequality constraints of the form[32]: $|h_k(\mathbf{x})| - \epsilon \leq 0$, where ϵ is an allowed small tolerance value. The feasible region is denoted as $\mathcal{F}$ and represents the set of all solutions which satisfy all the constraints (type a and b) within $\mathcal{S}$.

It is very common, when EAs or SIAs are used to solve optimization problems, that the application of variation operators (such as the mutation operator), may lead to type a constraints violation, then boundary constraint-handling methods (BCHMs) are needed. In the specialized literature, the research on techniques for handling type b constraints[25] is more popular with respect to that on methods to deal with type a constraints.

Despite the importance of BCHM within the evolutionary process of algorithms such as Differential Evolution (DE),[36] the choice of a particular method and its impact in the search are usually overlooked.

Additionally, those studies about the performance of BCHM in DE have been focused in unconstrained numerical optimization problems i.e. without type b constraints, and by only considering the canonical DE.[2,12]

A BCHM for problems with type b constraints should take advantage of specific features of this kind of problems such as feasibility information, i.e. the presence or absence of feasible solutions during the evolutionary process.

In our previous work published in the 2015 IEEE Congress on Evolutionary Computation,[16] we proposed a BCHM specialized in problems with type b constraints called *Centroid*, which was tested in the canonical DE.

In this paper, we present a generalized and improved version called *Centroid* $K + 1$, which consists in relocating the mutant vector, which leaves the search space, in the centroid of an area that is formed by $K+1$ vectors. One vector, taken from the population, which bias the corrected vector position to a good fitness area, and K random vectors, which aim to guide the corrected vector position to new areas in the search space.

In order to compare the performance of *Centroid* $K+1$, it has been tested on a wide and robust set of experiments that includes a state-of-the-art DE algorithm for CNOPs solving a set of 36 well-known constrained problems.[24]

The rest of the paper is organized as follows: Section 2 describes DE algorithm and some used strategies. In Sec. 3, a literature review about BCHMs is detailed. In Sec. 4, the improved *Centroid* $K+1$ BCHM is detailed. The description of the experimental study and the respective methods used are described in Sec. 5. The obtained results are presented in Sec. 6, while the conclusions and future work are presented in Sec. 7.

2. Differential Evolution

DE is a population-based stochastic search algorithm for solving optimization problems over continuous spaces.[36] DE algorithm aims at evolving a population of NP n-dimensional vectors that represent candidate solutions at generation g (Eq. (5)):

$$\mathbf{x}_{i,g} = \{x_{i,g}^1, \ldots, x_{i,g}^n\}, i = 1, \ldots, \text{NP}. \tag{5}$$

The initial population is generated uniformly at random within the predefined lower l_i and upper u_i limits for each variable x_i. DE consists mainly in three operations that are repeated generation after generation until the termination criteria are satisfied. Those operations are described below.

- *Mutation operator.* A mutant vector $\mathbf{v}_{i,g}$ is created for each vector (*target vector*) in the current population. The mutant vector can be created by the strategy DE/rand/1/bin described in Eq. (6).

$$\mathbf{v}_{i,g} = \mathbf{x}_{r_1,g} + F \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}), \tag{6}$$

  where $r_1 \neq r_2 \neq r_3 \neq i$ are indexes generated within range [1, NP]. $F > 0$ is a real value that represents a mutation scale factor. Such mutation operator can generate vector values outside the boundaries of each variable. Therefore, a BCHM is necessary to keep the search within the space defined by the boundaries of each variable. In some experiments of this work DE/rand/1/bin, DE/best/1/bin and DE/target-to-best/1 variants are used. The strategies are outlined in Table 1.
- *Crossover operator.* The crossover operator generates a trial vector by the recombining the target vector and its corresponding mutant vector as shown in Eq. (7).

$$u_{i,g}^j = \begin{cases} v_{i,g}^j & \text{if } (\text{rand}_j[0,1] \leq \text{CR}) \quad \text{or} \quad j = j_{\text{rand}}, \\ x_{i,g}^j & \text{otherwise}, \end{cases} \tag{7}$$

Table 1. DE variants used in the experiments, rand_j is a random real number $\in [0,1]$, j_{rand} is a random integer number $\in [1, D]$. $\mathbf{x}_{\text{best},g}$ is the best vector at generation g.

Nomenclature	Variant
DE/rand/1/bin	$u^j_{i,g} = \begin{cases} x^j_{r_3,g} + F \cdot (x^j_{r_1,g} - x^j_{r_2,g}) & \text{if } \text{rand}_j[0,1] < \text{CR or } j = j_{\text{rand}} \\ x^j_{i,g} & \text{otherwise} \end{cases}$
DE/best/1/bin	$u^j_{i,g} = \begin{cases} x^j_{\text{best},g} + F \cdot (x^j_{r_1,g} - x^j_{r_2,g}) & \text{if } \text{rand}_j[0,1] < \text{CR or } j = j_{\text{rand}} \\ x^j_{i,g} & \text{otherwise} \end{cases}$
DE/target-to-best/1	$u^j_{i,g} = x^j_{i,g} + F \cdot (x^j_{\text{best},g} - x^j_{i,g}) + F \cdot (x^j_{r_1,g} - x^j_{r_2,g})$

where $j = 1, \ldots, n$, $\text{CR} \in [0, 1]$ is a user-defined value which indicates how similar the trial vector will be with respect to the mutant vector, rand_j is a randomly generated real value between 0 and 1 and j_{rand} is a randomly generated integer within the range $[1, n]$, its aim is to avoid duplicates between the target and the trial vectors.

- *Selection operator.* Finally, the selection operator is summarized in Eq. (8), where the best vector is selected according to the fitness values between the target vector and its trial vector. The vector with best fitness value will remain for the next generation.

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases} \tag{8}$$

Algorithm 1 presents the pseudocode of the DE/rand/1/bin algorithm.

Algorithm 1. DE/rand/1/bin

```
Require: j
Ensure: h
1:  $g = 0$
2:  Generate a random initial population $\mathbf{x}_{i,g} \forall_i, i = 1, \ldots, \text{NP}$
3:  Evaluate $f(\mathbf{x}_{i,g}) \forall_i, i, \ldots, \text{NP}$
4:  for g = 1 to MAX_GEN do
5:      for i=1 to NP do
6:          Select randomly $r_1 \neq r_2 \neq r_3 \neq i$
7:          $j_{\text{rand}}$ = randint[1,n]
8:          for j = 1 to $n$ do
9:              if $(\text{rand}_j[0,1]) < \text{CR}$ or $j = j_{\text{rand}}$ then
10:                 $u^j_{i,g} = x^j_{r_1,g} + F(x^j_{r_2} - x^j_{r_3,g})$
11:                 Apply Boundary Constraint-Handling Method
12:             else
13:                 $u^j_{i,g} = x^j_{i,g}$
14:             end if
15:         end for
16:         if $f(\mathbf{u}_{i,g}) \leq F(\mathbf{x}_{i,g})$ then
17:             $\mathbf{x}_{i,g+1} = \mathbf{u}_{i,g}$
18:         else
19:             $\mathbf{x}_{i,g+1} = \mathbf{x}_{i,g}$
20:         end if
21:     end for
22:     $g = g + 1$
23: end for
```

3. Related Work

There is scarce research in the literature about BCHM on DE. Originally, Price *et al.*[30] suggested that replacing an infeasible solution by a *randomly* generated one is the most unbiased approach. The authors also defined a *bounce back* strategy, where an infeasible solution y, generated by mutating a feasible solution x, is replaced by a new feasible solution located on a line between x and y.

In a similar way to *bounce back* works, the method proposed by Ronkkonen *et al.*[31] operates, where the infeasible solutions are *reflected* back from the bound by the amount of the violation. Another method was published by Brest *et al.*[4] where the component of a mutant vector that goes off the search space is set to the violated *bound* value.

Some of the aforementioned methods (*Random*, *Reflection* and *Boundary*) together with *Conservatism*, *Wrapping* and *Resampling* were compared in a study performed by Arabas *et al.*[2] The experimental comparison was based on final performance results obtained by using canonical DE to solve the CEC2005 problems. The authors showed that the *Resampling* method presented better results in the most of the functions and concluded that the choice of the BCHM may significantly influence the final result.

Gandomi and Yang,[12] proposed a new BCHM called *Evolutionary* which was tested by using DE to solve benchmark optimization problems with type a constraints. The *Evolutionary* method was compared against four state-of-the-art BCHM. The proposed method achieved good convergence property and produced better performance for optimization problems in comparison with classical boundary methods, although it was not compared against *Resampling*, one of the best methods for handling boundary constraints in DE.

BCHM have been addressed more broadly in relation to the Particle Swarm Optimization (PSO) algorithm because in the first generations of PSO it is very common that the particles leave the search space. One of the earliest works of this kind was published by Zhang *et al.*,[43] who proposed a BCHM called *Periodic* mode which works in a similar way with respect to the *Wrapping* method. *Periodic* performed better than *Boundary* and *Random* methods in PSO with type b constraints.

Huang and Sanagavarapu[14] proposed a new BCHM for PSO algorithms called *Damping*. The proposed method is a hybrid that combines the characteristics offered by the *Boundary* and *Reflecting* methods. The authors compared the performance of four BCH methods: *Boundary*, *Reflection*, *Invisible* and *Damping* in the PSO algorithm solving two standard test function with type a constraints.

A comparative study of six BCHM in PSO was presented by Xu and Rahmat.[41] The performance of the methods was tested on two mathematical benchmark functions with type a constraints and a real-world electromagnetic problem.

An experimental study on BCHM in PSO was presented by Shi *et al.*[33] The study was focused on the population diversity. The authors analyzed the effect of applying BCHM in PSO with different topologies to solve problems with type a constraints.

Chu *et al.*[5] performed several experiments to investigate the effects of BCHM on the application of PSO to solve high-dimensional complex problems. Three basic BCHMs were tested using standard and composition benchmark functions. They concluded that the role of boundary constraint-handling becomes critical when dealing with high-dimensional complex problems, because as the dimensionality increases, the majority of particles in the swarm tends to fly outside the search space.

Helwing *et al.*[13] compared a wide variety of BCHM for PSO by examining their performance on flat landscapes. In general, they concluded that most of the methods introduce a significant search bias.

Padhye *et al.*[27] presented a review of BCHM for the PSO algorithm. The authors proposed two new boundary methods which were found to be robust and consistent in terms of performance over several simulation scenarios. PSO with different boundary methods were evaluated on benchmark functions with type a and b constraints.

The study of BCHM has also motivated research based on other optimization algorithms like Evolution Strategies, Cuckoo Search, and Interior Search. In this sense, Wessing[39] proposed two BCHMs called Intersection-Projection (IP) and Intersection-Reflection (IR), and made a comparison against three classical BCHM: *Boundary*, *Reflection*, and *Wrapping* using the covariance matrix self-adaptation evolution strategy CMSA-ES.

The authors concluded that IR and IP seem to have a slight tendency to work better on the lower dimensional problems. On the other hand, if the global optimum lies on the edge or in a corner, *Boundary* would be the ideal choice. However, with its ability to complement the basin of attraction, *Reflection* is a good alternative with acceptable performance throughout all problems. In general, the boundary methods have a huge impact on the optimizer's performance.

Gandomi *et al.*,[11] presented a comparison between two BCHM (*Boundary* and *Evolutionary*) on Cuckoo Search to solve five benchmark functions, concluding that *Evolutionary* can make the optimization algorithm performance better without complex actions like hybridizing.

Recently, Trivedi *et al.*[37] compared the performance of five BCHM (*Reflect*, *Random*, *Wrapping*, *Boundary*, and *Evolutionary*) in the Interior Search Algorithm to solve sixteen numerical benchmark problems and concluded that the *Evolutionary* method is highly suitable for most optimization problems with improving convergence property.

BCHMs have also been used as a support in EAs to solve constrained multi-objective problems as in the work published by Liu *et al.*[20] where it was designed a boundary search method to improve the efficiency of an EA.

In summary, a BCHM has a huge impact on the algorithm's performance. Nevertheless, there is scarce research regarding BCHM in DE, and, to the best of the authors' knowledge, besides *Centroid*, none of those tackles the presence of type b constraints. In fact, among the PSO research works reported here, only two of them deal with type b constraints by considering only classical BCHM and canonical PSO versions as well. Therefore, research on BCHM with DE to deal with search spaces with type b constraints, by considering canonical DE variants and also state-of-the-art proposals is identified as a scarcely explored path of research.

4. Proposed Method

The previous version of *Centroid*, which we published in the 2015 IEEE Congress on Evolutionary Computation[16] focuses on repairing the vector that is outside the boundaries by calculating the centroid of an area that is formed by three vectors located within the boundaries of the search space. The first vector, taken from the population, bias the corrected vector position to a good fitness area. The other two vectors, generated at random, aim to guide the corrected vector position to new areas in the search space.

In this paper, we present a generalized version of *Centroid*, which involves calculating the centroid of an area that is formed by $K + 1$ vectors, one of them taken from the population and K generated at random. The main difference with respect to the previous version of centroid is that in this generalized version K random vectors ($K > 0$) are considered instead of just 2.

4.1. *Description of method*

The *Centroid* $K + 1$ method is applied to the *mutant vector* $\mathbf{v}$ (Produced by Eq. (6)), when it leaves the search space, so as to get it inside the search space defined by the lower l and upper u boundaries of each variable of the CNOP. The *Centroid* $K + 1$ method is summarized in Eq. (9).

$$\mathbf{v} = \begin{cases} \mathbf{v}^c & \text{if any element in } \mathbf{v} \text{ is outside its boundaries} \\ \mathbf{v} & \text{otherwise} \end{cases} \tag{9}$$

the *corrected vector* $\mathbf{v}^c$ is precisely the centroid of $K + 1$ vectors and it is calculated as in Eq. (10)

$$\mathbf{v}^c = (\mathbf{w}_p + \mathbf{w}_{r_1} + \cdots + \mathbf{w}_{r_K})/(K + 1), \tag{10}$$

where $\mathbf{w}_p$ is a *vector from the current population* that is selected according to the *amount of feasible solutions* (*AFS*) (see Eq. (11)).

$$\mathbf{w}_p = \begin{cases} \mathbf{x}_{\text{rand}} \in \text{SFS} & \text{If AFS} > 0 \text{ and rand}[0,1] > 0.5 \\ \mathbf{x}_{\text{best}} \in \text{SIS} & \text{otherwise}, \end{cases} \tag{11}$$

where SFS is the *set of feasible solutions* in the current population, SIS is the *set of infeasible solutions* in the current population, $\mathbf{x}_{\text{rand}}$ is a *feasible vector chosen at random* and $\mathbf{x}_{\text{best}}$ is the *best individual from* SIS i.e. an infeasible vector with the lowest sum of constraint violation.

On the other hand, $w_{r_1}, \ldots, w_{r_K}$ are K the *random vectors*. The random vectors initially take the same values as $\mathbf{v}$, and subsequently replace the invalid values by random values between the lower l and upper u bounds, in the same way that the vectors of the initial population are generated. The only difference with respect to a completely random vector is the fact that they will share the same values of the variables of the mutant vector $\mathbf{v}$ which have valid values.

It is important to mention that random vectors do not belong to the current population and they will not be evaluated in the objective function and the constraints of the CNOP.

4.2. *Pseudocode of centroid K+1 method*

Algorithm 2 presents the pseudocode of *Centroid* $K + 1$ method. In the first stage of the method, the vector $\mathbf{w}_p$ is selected according to the *amount of feasible solutions* AFS, i.e. if there are feasible solutions, the population vector $\mathbf{w}_p$ is randomly selected from the *set of feasible solutions* SFS. Otherwise, the vector with the lowest sum of constraint violation from the set of infeasible solutions SIS is selected as $\mathbf{w}_p$ (lines 9–13).

Subsequently, the K random vectors are generated, each of which is initialized with the same values of $\mathbf{v}$, then they are assigned new values in the invalid positions. These new values are chosen randomly between the lower l and upper u limits of each variable. As expressed in lines 17–21 of Algorithm 2.

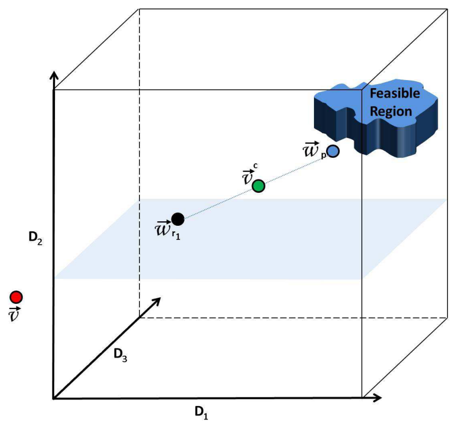
Finally, the centroid of the K random vectors and the vector $\mathbf{w}_p$ is calculated to know the position of the corrected vector $\mathbf{v}^c$.

4.3. *Graphical example of centroid K+1 method*

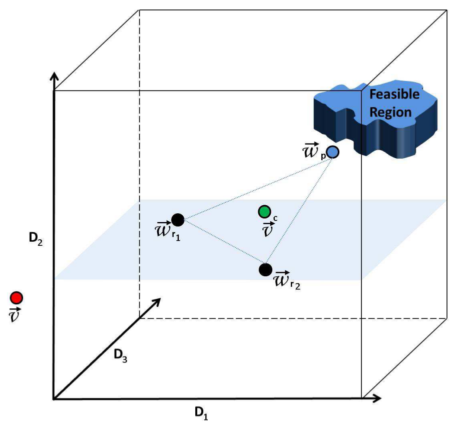
Figure 1 represents a hypothetical optimization problem in three dimensions (D_1, D_2 and D_3), where the mutant vector $\mathbf{v}$ has left the search space taking invalid values in dimensions D_1 and D_3.

Algorithm 2. Pseudocode of Centroid $K + 1$ method.

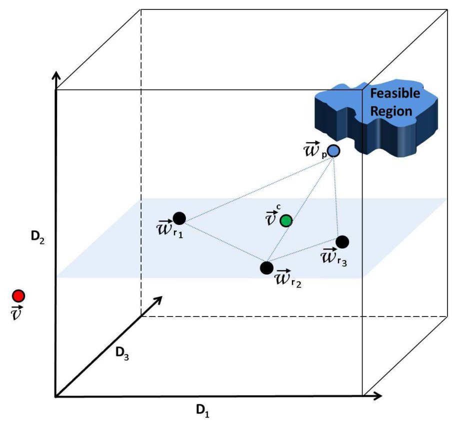
```
1: SFS : Set of Feasible Solutions
2: SIS : Set of Infeasible Solutions
3: AFS : Amount of Feasible Solutions
4: $K$ : Amount of Random Vectors
5: $D$ : Dimensionality of the problem
6: $l$ : Lower boundaries
7: $u$ : Upper boundaries
8: $\mathbf{v}$ : the mutant vector that is outside boundaries
9: if AFS $> 0$ and rand$[0,1] > 0.5$ then
10:     $\mathbf{w}_p = \mathbf{x}_{\text{rand}}$ {$\mathbf{x}_{\text{rand}}$ is randomly selected from SFS}
11: else
12:     $\mathbf{w}_p = \mathbf{x}_{\text{best}}$ {$\mathbf{x}_{\text{best}}$ is the best individual from SIS}
13: end if
14: {Creation of $K$ random vectors}
15: for i=1 to K do
16:     $\mathbf{w}_{r_i} = \mathbf{v}$
17:     for j=1 to D do
18:         if $w_{r_{i,j}} < l_j$ or $w_{r_{i,j}} > u_j$ then
19:             $w_{r_{i,j}} = l_j + \text{rand}[0,1] \times (u_j - l_j)$
20:         end if
21:     end for
22: end for
23: $\mathbf{v}^c = (\mathbf{w}_p + \mathbf{w}_{r_1} + \cdots + \mathbf{w}_{r_K})/(K+1)$ {Corrected vector}
```



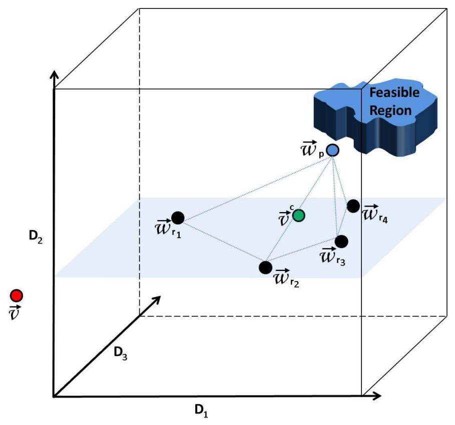
(a) Centroid 1+1.



(b) Centroid 2+1.



(c) Centroid 3+1.



(d) Centroid 4+1.

Fig. 1. Graphical explanation of the *Centroid* $K+1$ method for $K=1$ (a), $K=2$ (b), $K=3$ (c) and $K=4$ (d). In all four cases, the K random vectors share the same values as $\mathbf{v}$ in dimension D_2, which is the dimension where $\mathbf{v}$ contains valid values, and contain random values in dimensions D_1 and D_3. The corrected vector $\mathbf{v}^c$ is located in the centroid formed by the K random vectors and vector $\mathbf{w}_p$.

The corrected vector $\mathbf{v}^c$ is relocated in the centroid formed by the K random vectors and the vector $\mathbf{w}_p$. The K random vectors share the same value as $\mathbf{v}$ in dimension D_2, and take random values in dimensions D_1 and D_3, which are the dimensions violated by $\mathbf{v}$. $\mathbf{w}_p$ is a vector taken from the population, with the particularity that it is located in the feasible region or close to it, as expressed in Eq. (11).

In Fig. 1(a), where $K=1$, $\mathbf{v}^c$ is located in the centroid formed by $\mathbf{w}_p$ and $\mathbf{w}_{r_1}$; in Fig. 1(b), $K=2$, this situation in particular represents the previous version of *Centroid*,[16] where $\mathbf{v}^c$ is located in the centroid formed by $\mathbf{w}_p$, $\mathbf{w}_{r_1}$ and $\mathbf{w}_{r_2}$ (one vector taken from the population and two random vectors); in Figs. 1(c) and 1(d), subsections (c) and (d) the *Centroid* versions for $K=3$ and $K=4$ are shown, where one vector taken from the population is used, along with three and four random vectors, respectively.

In Fig. 1, it is clearly seen that the previous version of *Centroid* (1b) is only one of several possible variants of this new generalized version *Centroid* $K+1$. The main goal of the *Centroid* $K+1$ method is to locate the corrected vector near a promising region (either inside the feasible region or close to it), but biased by K random vectors so as to avoid a possible premature convergence.

5. Experimental Design

In order to evaluate the performance of the proposed *Centroid* $K+1$ method, an experimental study composed of six parts was designed. A detailed description of each experiment with their respective materials and methods is shown below. In all experiments, 18 benchmark problems (10D and 30D, 36 in total) presented in the special session on *Single Objective Constrained Real-Parameter Optimization* in the 2010 Congress on Evolutionary Computation (CEC'2010) were used.[24]

5.1. *Experimental conditions for all experiments*

25 independent runs were performed for the 36 test problems. The maximum number of evaluations was set to 200,000 for 10D and 600,000 for 30D in each run. All experiments were carried out under the same PC configuration which is presented in Table 2.

5.2. *Boundary constraint-handling methods*

In the present study, the performance of *Centroid* $K+1$ is compared against the following BCHM which are commonly used in DE.

5.2.1. *Reflection*

In this method, the variables that violate boundary constraints are reflected back from the bound by the amount of violations.[31] This method is formulated as in Eq. (12).

$$x_i^c = \begin{cases} x_i & \text{if } l_i \leq x_i \leq u_i, \\ 2 \times l_i - x_i & \text{if } x_i < l_i, \\ 2 \times u_i - x_i & \text{if } x_i > u_i, \end{cases} \tag{12}$$

where x_i^c is the valid value, x_i is the value which violates the bound constraint, and l_i and u_i are the lower and upper limits for variable i, respectively. This process is repeated until the value is within the boundaries.

Table 2. PC Configuration used in each experiment.

System	Windows 8.1 64 bits
CPU	Intel (R) Core(TM) i5-4210U (2.40 GHz)
RAM	8 GB

5.2.2. *Projection*

In this method, also known as *Boundary*, the value for a variable is reset to the violated bound.[4] This is stated in Eq. (13).

$$x_i^c = \begin{cases} x_i & \text{if } l_i \le x_i \le u_i, \\ l_i & \text{if } x_i < l_i, \\ u_i & \text{if } x_i > u_i. \end{cases} \tag{13}$$

5.2.3. *Reinitialize by position*

This method, also called *random scheme*,[43] replaces the variables outside the boundary with a random value within the boundaries through Eq. (14).

$$x_i^c = l_i + \text{rand}(0,1) \times (u_i - l_i), \tag{14}$$

where rand(0,1) returns a real value within 0 and 1 with uniform distribution.

5.2.4. *Reinitialize all*

In this approach, if at least one of the variables of the solution is outside its limits, a complete new individual $\mathbf{x}$ is generated within the allowed boundaries[30] by using Eq. (14).

5.2.5. *Conservatism*

This method was proposed to work particularly with DE. If the differential mutation resulted in an infeasible vector, it is rejected and the original parent vector remains in the population. Conservatism assumes that the infeasible vector will not be repaired but it is rather rejected at the replacement stage.[9]

5.2.6. *Resampling*

This method works by applying the variation operator (differential mutation for DE) until a complete valid vector is generated. If at least one variable violates the bounds, the operator is applied again to generate a whole new solution.[2]

5.2.7. *Evolutionary*

In this method, the components which are beyond the boundaries are replaced through a random component between the related bound and the similar component of the best solution so far.[12] This method is formulated as in Eq. (15).

$$x_i^c = \begin{cases} \alpha \times l_i + (1-\alpha)x_i^{\text{best}} & \text{if } x_i < l_i, \\ \beta \times u_i + (1-\beta)x_i^{\text{best}} & \text{if } x_i > u_i, \end{cases} \tag{15}$$

where x_i^{best} is the value of variable i of the best solution so far and α and β are real numbers between 0 and 1.

5.3. *Experiments description*

Experiment 1: Due to the fact that the *Centroid* $K+1$ method consists of forming an area by $K+1$ vectors (where K represents an integer value) located within the boundaries of the search space. Experiment 1 consisted on evaluating the *Centroid* $K+1$ method by varying the number of K vectors from 1 to 9. This experiment was divided into four parts: (1) The final results obtained by centroid with different number of vectors, (2) the ability of each centroid variant to generate better solutions, (3) the overall percentage of repaired vectors by each centroid variant and (4) the percentage of repaired vectors who were better than their corresponding target vectors.

- *Parameters settings*: DE/rand/1/bin strategy and Deb's rules[7] were used as the constraint-handling technique. The parameter values for DE were $\mathrm{CR} = 1.0$, a population size $\mathrm{NP} = 100$ and scale factor $F = 0.7$.

Experiment 2: In this experiment and the following ones, the best variant of *Centroid* $K+1$ was compared against the seven BCHMs described in Sec. 5.2. Five aspects were analyzed in this experiment: (1) the final results obtained by each method employing different CR values, (2) the capability to help on reaching the feasible region on the search space, (3) the ability of each method to generate a better solution, (4) the overall percentage of repaired vectors by each BCHM and (5) the percentage of repaired vectors that were better than their corresponding target vectors.

- *Parameter setting*: DE/rand/1/bin strategy with Deb' rules were considered. $\mathrm{NP} = 100$, $F = 0.7$ and $\mathrm{CR} = 1.0$, 0.8 and 0.4 were considered.

Experiment 3: In this experiment, two DE strategies were used to evaluate the performance of the BCHMs. Five situations were analyzed in this experiment: (1) the final results obtained by each BCHM with different strategies, (2) the capability to help on reaching the feasible region on the search space, (3) the ability of each method to generate a better solution, (4) the overall percentage of repaired vectors by each BCHM and (5) the percentage of repaired vectors who were better than their corresponding target vectors.

- *Parameter setting*:

– DE/best/1/bin, $\mathrm{NP} = 100$, $\mathrm{CR} = 1.0$, $F = 0.7$.
– DE/target-to-best/1, $\mathrm{NP} = 100$, $F = 0.7$.
– Deb' rules were used.

Experiment 4: While conducting Experiments 2 and 3, some weaknesses of Resampling were observed; such failings may cause the evolutionary process of the DE algorithm to fall into an infinite loop or dramatically increase its runtime. Since these weaknesses have not been documented in previous work and due to the importance of the Resampling method in DE,[2,16] this experiment was developed to study this behavior. In this experiment, the number of resampling and reboots were analyzed by using different DE strategies with three different values for CR parameter. Besides, a graphical comparison between Resampling and Centroid in terms of required runtime is presented.

- *Parameter setting*: DE/rand/1/bin, DE/best/1/bin and DE/target-to-best/1 with Deb's rules were used. For the three strategies, NP $= 100$, $F = 0.7$ and CR $=$ 1.0, 0.8 and 0.4.

Experiment 5: In this experiment a global analysis about the percentage of repaired individuals for each test problem was carried out. The analysis was made by considering four aspects: the type of test problem, the dimensionality of each test problem, the DE strategy used, and the assigned value to the CR parameter.

Experiment 6: This last experiment aimed to evaluate the performance of *Centroid* $K + 1$ in a state-of-the-art DE algorithm for CNOPs (i.e. with type b constraints).

For this experiment, the algorithm called *Improved* $(\mu + \lambda)$-*Constrained Differential Evolution* (ICDE) proposed by Wang[15] was adopted. The proposed BCHT was tested with ICDE and was compared against other BCHT in ICDE.

- *Parameter setting*: ICDE algorithm used the following configuration, $\mu = 70$, $\lambda = 210$, $F = 0.8$, CR $= 0.9$, $p_m = 0.05$, $\eta = 200$, $k = 0.6$, $\delta = 0.0001$.

Experiments 1, 2, 3 and 6 show tables with statistical results (M: Mean, SD: Standard Deviation) obtained by the comparison among BCHM. Best results are highlighted in boldface. To validate the results obtained, the 95%-confidence Kruskal–Wallis test was applied to the samples of runs. "*" means that there was a significant difference favoring the base method. "+" means that there was a significant difference favoring the compared method with respect to the base method. If no symbols are included, it means that no significant difference was observed between the compared methods. "-" means no feasible solution found. (w) means that in only w runs, out of 25, feasible solutions were found.

6. Results

6.1. *Experiment* 1

Our previous *Centroid* version presented in Ref. 16 represents the *Centroid* $2 + 1$ variant (two random vectors and one vector of the population) of the generalized *Centroid* $K + 1$ proposal presented in this paper. Thus, different Centroid variants such as 1+1, 2+1, 3+1, and so on, are now possible.

In this experiment, the performance of nine *Centroid* $K+1$ variants, $K=\{1,2,\ldots,9\}$ were compared. Tables 3 and 4 show the results for 10D problems, Tables 5 and 6 present the results for 30D. Tables 3 and 5 show the functions in which at least one Centroid variant found feasible values in all 25 runs. In the four tables, *Centroid* $2+1$ was used as the base method for the Kruskal–Wallis tests, in order to compare the performance of the previous version of *Centroid* against this new generalized proposal of *Centroid* $K+1$.

Table 3 indicates that the best performance was obtained by the Centroid variant with $K=1$, since it was the only one that achieved a statistically superior performance than $K=2$ in 3 functions. In remaining functions, there were no statistical significant differences. Additionally, with $K=1$, the best mean value was achieved in 6 problems. When $K=3$ was used, a similar performance to that with $K=2$ was obtained. For $K>3$ values, the performance was poor.

Table 4 shows that when a value of $K=\{1,2,6,8\}$ was used, the highest number of feasible runs in 2 functions was achieved and, when a $K=\{3,4,9\}$ was used, the highest number of feasible runs was obtained for 1 function. As for the best mean value, when a value of $K=1$ was used, the best result was achieved in 4 functions, followed by $K=2$ that obtained the best mean value in 2 problems and finally for $K=\{3,5,6,7\}$ the best mean value was achieved in 1 function. Considering both, the number of feasible runs and the best mean value, it was observed that $K=1$ performed better, followed by $K=2$ and $K=6$.

In summary, it was found that for 10D problems the Centroid variant with $K=1$ reported the best performance, followed by $K=2$.

For 30D problems, Table 5 shows that for $K=4$ and $K=2$, a similar performance was obtained. It was because, with $K=4$, the method achieved a better performance in 3 functions and lost in 3. For $K=3$, a better performance was obtained in 1 function and a worse performance was observed in 2 functions. For $K=\{5,6,7,8,9\}$ the performance was similar because all of them were better in 2 functions and lost in 3. Finally, for $K=1$ although the best mean value was obtained for 2 problems, the differences were not significant and it also lost in 3 functions.

Table 6 shows that for $K=5$, the highest number of feasible runs was achieved for 3 functions, followed by $K=\{1,2,3,4,7\}$, which obtained the highest number of feasible runs in 1 function. Concerning to the mean value, the performance for Centroid with $K=1$ was markedly higher because it got the best results in 5 functions followed by $K=\{4,6\}$, which obtained the best performance in 1 function. It is also important to mention that there was a statistically significant difference favoring $K=1$ in function C16. Therefore, it is concluded that for problems in Table 6, 1+1 Centroid had the better performance, followed by $K=5$. In summary, from Tables 5 and 6, we concluded that for 30D problems the best variants were Centroid with $K=1$ or $K=2$.

As for the number of functions, all of Centroid variants solved the 18 problems in 10 dimensions and 14 of them for 30 dimensions. As for the number of feasible runs,

Table 3. Results obtained by Centroid $K + 1$ method on 10D problems where feasible solutions were found in each of the 25 runs by at least one method.

		Centroid 2+1[16]	Centroid 1+1	Centroid 3+1	Centroid 4+1	Centroid 5+1	Centroid 6+1	Centroid 7+1	Centroid 8+1	Centroid 9+1
C01	M	–7.46E-01	–7.46E-01	**–7.47E-01**	**–7.47E-01**	**–7.47E-01**	**–7.47E-01**	**–7.47E-01**	**–7.47E-01**	**–7.47E-01**
	SD	4.29E-03	4.29E-03	1.35E-03	1.35E-03	1.35E-03	2.44E-12	4.79E-13	1.68E-13	1.35E-03
C03	M	3.78E-24	**2.30E-24**	5.20E-24	6.73E-24*	8.62E-24*	7.70E-24*	5.50E-24*	6.72E-24*	5.41E-24*
	SD	5.65E-24	2.99E-24	5.82E-24	8.85E-24	1.14E-23	8.79E-24	6.15E-24	7.85E-24	5.00E-24
C04	M	**–1.00E-05**	**–1.00E-05**	**–1.00E-05**	**–1.00E-05**	**–1.00E-05***	**–1.00E-05**	**–1.00E-05***	**–1.00E-05**	**–1.00E-05***
	SD	1.73E-09	1.41E-09	2.08E-09	2.03E-09	2.44E-09	2.09E-09	2.05E-09	2.01E-09	3.17E-09
C07	M	7.79E-23	**8.73E-24+**	3.02E-23	3.99E-23	5.71E-23	6.75E-23*	8.29E-23*	1.30E-22*	6.42E-23
	SD	2.79E-22	1.92E-23	5.41E-23	7.32E-23	1.15E-22	8.95E-23	1.46E-22	4.04E-22	1.44E-22
C08	M	7.52E+00	7.92E+00	9.73E+00	8.47E+00	8.90E+00*	9.10E+00*	9.33E+00*	8.38E+00*	**7.36E+00**
	SD	4.68E+00	4.60E+00	2.93E+00	4.32E+00	3.96E+00	4.17E+00	3.52E+00	4.16E+00	4.89E+00
C11	M	**–1.52E-03**	**–1.52E-03+**	**–1.52E-03**	**–1.52E-03**	**–1.52E-03**	**–1.52E-03***	**–1.52E-03***	**–1.52E-03***	**–1.52E-03***
	SD	6.94E-14	6.14E-14	1.34E-13	8.70E-14	1.00E-13	1.45E-13	1.20E-13	1.48E-13	2.11E-13
C12	M	**–1.32E+01**	–6.69E+00	–6.56E+00	–5.63E+00	–7.69E+00	–2.56E+00	–1.18E+01	–5.69E+00	–1.12E+00*
	SD	6.06E+01	2.32E+01	2.28E+01	2.28E+01	2.32E+01	4.83E+00	3.15E+01	2.32E+01	3.23E+00
C13	M	–6.10E+01	–6.02E+01	**–6.20E+01**	–6.03E+01	–6.09E+01	–6.15E+01	–6.17E+01	–6.07E+01	–6.16E+01
	SD	2.53E+00	4.14E+00	2.39E+00	4.66E+00	4.55E+00	3.77E+00	3.04E+00	3.60E+00	4.55E+00
C14	M	2.70E+11	**1.34E+11+**	4.02E+11	4.09E+11	3.64E+11	6.22E+11*	4.46E+11	5.36E+11	5.32E+11*
	SD	2.28E+11	1.51E+11	3.08E+11	3.94E+11	2.23E+11	5.26E+11	4.60E+11	5.94E+11	4.30E+11
C15	M	4.91E+12	**4.77E+12**	7.30E+12	7.84E+12*	8.38E+12*	8.95E+12*	1.20E+13*	6.97E+12	1.02E+13*
	SD	3.52E+12	3.72E+12	6.96E+12	4.97E+12	8.40E+12	7.16E+12	1.18E+13	5.27E+12	7.59E+12

Table 4. Results obtained by Centroid $K + 1$ method on 10D problems where none of the methods found feasible solutions in all 25 runs.

		Centroid 2+1[16]	Centroid 1+1	Centroid 3+1	Centroid 4+1	Centroid 5+1	Centroid 6+1	Centroid 7+1	Centroid 8+1	Centroid 9+1
C02	M	3.10E+00(12)	2.76E+00**(16)**	3.50E+00(13)	3.26E+00(11)	2.80E+00(12)	**2.60E+00(16)**	3.45E+00(9)	2.98E+00(15)	3.15E+00(12)
	SD	1.04E+00	1.17E+00	6.80E-01	1.21E+00	9.97E-01	1.17E+00	8.92E-01	1.55E+00	1.37E+00
C05	M	3.23E+02(2)	**2.37E+02**(1)	3.50E+02(3)	3.95E+02(3)	2.80E+02(3)	2.62E+02(3)	2.44E+02(5)	3.29E+02**(7)**	2.70E+02(4)
	SD	1.42E+02	0.00E+00	8.62E+01	9.98E+01	1.77E+02	2.10E+02	1.83E+02	8.17E+01	7.93E+01
C06	M	4.45E+02(2)	**1.83E+02**(2)	3.03E+02(2)	3.33E+02**(4)**	3.90E+02(3)	4.60E+02(2)	4.03E+02(2)	3.64E+02(2)	4.14E+02(3)
	SD	6.41E+01	3.63E+01	2.85E+02	1.99E+02	7.98E+01	1.48E+02	2.13E+02	8.31E+01	7.11E+01
C09	M	4.36E+12**(9)**	2.84E+12(5)	2.96E+12(4)	5.56E+12(6)	4.01E+12(7)	6.33E+12(8)	**1.64E+12**(4)	2.85E+12(8)	5.08E+12**(9)**
	SD	3.60E+12	2.53E+12	1.33E+12	4.91E+12	4.08E+12	2.79E+12	2.21E+12	3.00E+12	3.28E+12
C10	M	2.16E+12(3)	4.91E+12(5)	**1.96E+12**(6)	4.17E+12(5)	4.90E+12(4)	4.79E+12**(7)**	2.90E+12(3)	3.95E+12(4)	4.47E+12(3)
	SD	1.86E+12	4.44E+12	2.09E+12	2.70E+12	2.50E+12	2.90E+12	2.03E+12	1.85E+12	2.90E+12
C16	M	**1.02E+00(15)**	**1.02E+00**(12)	1.04E+00(7)	1.03E+00(4)	**1.02E+00**(11)	1.04E+00(10)	1.03E+00(14)	1.03E+00(12)	1.04E+00(7)
	SD	7.69E-02	1.85E-02	4.06E-02	5.59E-03	2.41E-02	2.15E-02	3.94E-02	3.14E-02	1.17E-02
C17	M	**1.95E+02**(13)	3.87E+02(10)	3.11E+02**(18)***	2.40E+02(10)	2.86E+02(12)	2.39E+02(14)	3.34E+02(13)	2.23E+02**(18)**	2.82E+02(13)
	SD	1.30E+02	2.60E+02	1.71E+02	1.45E+02	1.90E+02	1.31E+02	2.10E+02	1.81E+02	2.27E+02
C18	M	6.26E+03(20)	**5.73E+03(24)**	7.89E+03(18)	5.97E+03(19)	7.32E+03(19)	5.83E+03(20)	7.51E+03(21)	6.14E+03(20)	7.13E+03(19)
	SD	3.45E+03	4.22E+03	3.78E+03	4.10E+03	4.93E+03	2.55E+03	5.97E+03	2.59E+03	4.46E+03

Table 5. Results obtained by Centroid $K+1$ method on 30D problems where feasible solutions were found in each of the 25 runs by at least one method.

		Centroid 2+1[16]	Centroid 1+1	Centroid 3+1	Centroid 4+1	Centroid 5+1	Centroid 6+1	Centroid 7+1	Centroid 8+1	Centroid 9+1
C01	M	–4.85E-01	–4.36E-01*	**–5.21E-01+**	–5.15E-01+	–5.00E-01	–5.14E-01	–5.11E-01	–4.88E-01	–4.77E-01
	SD	3.43E-02	2.36E-02	5.39E-02	5.21E-02	4.44E-02	6.46E-02	5.85E-02	4.86E-02	4.85E-02
C07	M	5.02E+06	3.17E+07*	2.42E+06	6.85E+05+	1.17E+06+	1.28E+06+	1.78E+06+	1.26E+06+	**6.19E+05+**
	SD	6.50E+06	3.69E+07	3.98E+06	7.58E+05	2.85E+06	1.91E+06	2.70E+06	2.02E+06	5.34E+05
C08	M	5.09E+06	1.93E+07*	2.68E+06	3.35E+06+	1.32E+06+	1.12E+06+	1.59E+06+	**1.08E+06+**	1.25E+06+
	SD	7.35E+06	2.05E+07	3.16E+06	7.20E+06	1.57E+06	1.29E+06	2.24E+06	1.36E+06	2.49E+06
C13	M	–4.99E+01	–5.25E+01	–5.17E+01	–5.13E+01	–4.93E+01	–5.06E+01	–5.20E+01	**–5.29E+01**	–5.00E+01
	SD	8.28E+00	7.49E+00	5.73E+00	6.02E+00	8.68E+00	7.62E+00	6.14E+00	3.22E+00	8.25E+00
C14	M	2.30E+12	**2.09E+12**	4.41E+12*	5.33E+12*	6.52E+12*	7.28E+12*	9.43E+12*	1.11E+13*	1.06E+13*
	SD	1.47E+12	1.45E+12	3.11E+12	3.39E+12	3.73E+12	4.47E+12	6.33E+12	6.36E+12	6.40E+12
C15	M	2.19E+13	**1.81E+13**	2.94E+13*	3.82E+13*	4.72E+13*	4.88E+13*	4.82E+13*	5.23E+13*	5.14E+13*
	SD	9.74E+12	6.89E+12	1.05E+13	1.56E+13	1.46E+13	1.84E+13	1.53E+13	1.18E+13	1.84E+13
C18	M	**1.33E+04**(24)	1.58E+04(22)	1.37E+04(24)	1.68E+04(24)*	1.88E+04(24)*	1.68E+04*	1.79E+04*	1.69E+04*	1.65E+04(24)*
	SD	3.99E+03	7.50E+03	4.26E+03	4.85E+03	9.19E+03	4.05E+03	6.01E+03	3.61E+03	3.32E+03

Table 6. Results obtained by Centroid $K + 1$ method on 30D problems where none of the methods found feasible solutions in all 25 runs.

		Centroid 2+1[16]	Centroid 1+1	Centroid 3+1	Centroid 4+1	Centroid 5+1	Centroid 6+1	Centroid 7+1	Centroid 8+1	Centroid 9+1
C02	M	4.20E+00**(20)**	4.02E+00(13)	4.14E+00(17)	**3.85E+00**(15)	3.93E+00(11)	4.03E+00 (13)	4.06E+00(17)	4.34E+00(5)	4.41E+00(12)
	SD	7.45E-01	6.75E-01	7.27E-01	5.93E-01	9.38E-01	8.96E-01	7.05E-01	7.39E-01	4.82E-01
C05	M	3.62E+02(11)	**3.16E+02**(9)	4.56E+02(8)	3.71E+02(9)	4.00E+02**(16)**	4.32E+02(7)	4.54E+02(5)	4.25E+02(9)	4.14E+02(7)
	SD	9.82E+01	9.94E+01	6.38E+01	7.69E+01	6.87E+01	5.58E+01	7.06E+01	6.88E+01	8.80E+01
C06	M	4.51E+02(7)	3.78E+02(7)	4.10E+02(6)	4.66E+02**(9)**	4.84E+02(5)	**3.74E+02**(6)	4.02E+02(5)	4.54E+02(6)	4.26E+02(3)
	SD	9.46E+01	7.88E+01	8.31E+01	7.31E+01	2.60E+01	1.16E+02	5.54E+01	6.95E+01	9.80E+01
C09	M	1.07E+13(13)	**8.00E+12**(12)	9.20E+12**(15)**	9.91E+12(14)	1.27E+13(9)	8.83E+12(9)	1.26E+13(8)	1.43E+13(8)	1.94E+13(7)*
	SD	6.28E+12	4.85E+12	6.34E+12	6.05E+12	4.83E+12	4.80E+12	6.40E+12	7.06E+12	4.75E+12
C10	M	9.23E+12(12)	**8.77E+12(14)**	1.02E+13(10)	1.46E+13(7)*	1.05E+13**(14)**	1.50E+13(4)*	1.76E+13(11)*	1.57E+13(7)	1.83E+13(7)
	SD	5.31E+12	6.75E+12	6.35E+12	4.88E+12	4.82E+12	2.99E+12	8.83E+12	7.84E+12	1.51E+13
C16	M	1.07E+00(15)	**1.05E+00**(15)+	1.07E+00(14)	1.09E+00(15)	1.08E+00(16)	1.09E+00(12)*	1.09E+00**(22)***	1.09E+00(13)	1.10E+00(12)*
	SD	2.50E-02	2.35E-02	2.54E-02	2.79E-02	3.77E-02	1.88E-02	1.89E-02	2.09E-02	3.53E-02
C17	M	8.08E+02(18)	**7.30E+02**(14)	7.74E+02(20)	8.20E+02(20)	9.08E+02**(21)**	8.14E+02(19)	9.37E+02(17)	9.44E+02(17)	1.07E+03(20)*
	SD	3.30E+02	2.71E+02	2.81E+02	2.94E+02	3.26E+02	2.32E+02	2.41E+02	3.39E+02	3.68E+02

as can be seen in Fig. 2(a), $K = 8$ obtained the most amount of feasible runs in 10D and $K = 2$ in 30D. Figure 2(b) shows that the variant of Centroid for $K = 1$ got a lower percentage of repairs, followed by $K = 2$ and $K = 3$; this parameter is important because it indicates the goodness of the method to keep the population within the search space. Figure 2(c) indicates that the highest percentage of individuals repaired that were better than its target vector was obtained using the Centroid variant for $K = 1$ followed by $K = 2$ and $K = 3$.

Conclusions of Experiment 1.

Considering the statistical tests, it is concluded that in 10D problems the variant with the best performance was $K = 1$ and for 30D problems was $K = 2$. Considering both 10D and 30D problems, the best 5 variants were $K = \{1, 2, 3, 4 \text{ and } 5\}$ in that order.

The variant with the largest number of feasible runs in 10D problems was $K = 8$ and for 30D it was $K = 2$. Together, by adding the number of feasible runs for 10D and 30D, the best 5 variants were $K = \{2, 5, 3, 1 \text{ and } 7\}$ in that order.

The variant $K = 1$ is the one that generates the largest number of individuals that, after being repaired, are better than their corresponding target vector. As the value of K increases, the percentage of repaired vectors that are better than their corresponding target vector is reduced.

As for the percentage of individuals repaired, the variant that performs least repairs is $K = 1$. This influences the runtime, since as the K value is reduced fewer random vectors are generated, less repairs are required and the runtime is reduced as well.

Considering the above, in subsequent experiments we will only use the variant *Centroid* 1+1 ($K = 1$), and we will refer to it just as Centroid.

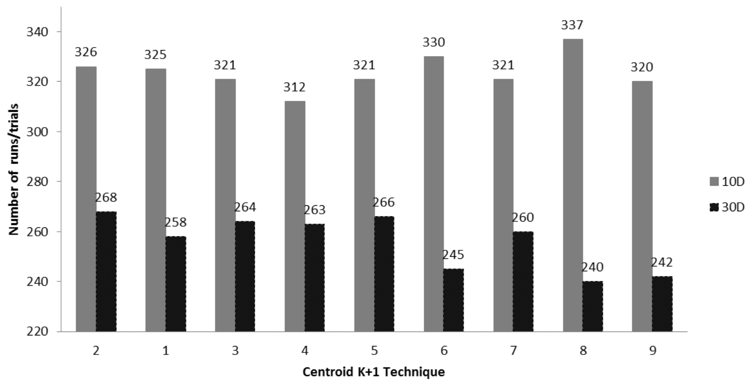
6.2. *Experiment* 2

In this experiment, the performance of the *Centroid* $K + 1$ ($K = 1$) method was compared against the seven BCHM described in Sec. 5.2, employing the DE/rand/1/bin variant with values for CR=$\{0.4, 0.8, 1.0\}$. The aim was to observe the stability of each method by varying the crossover rate.

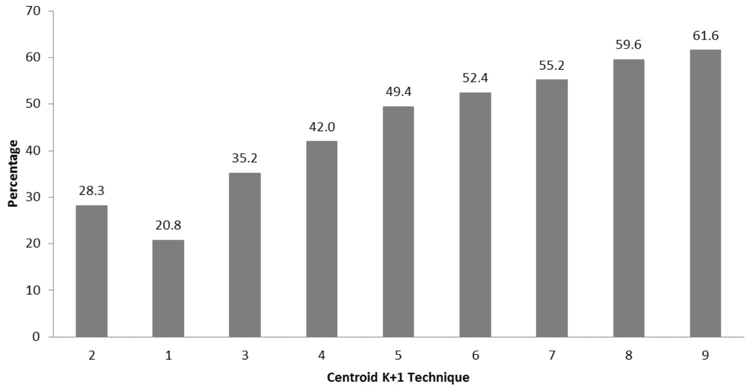
In Tables 7–10, the mean value of 25 runs for each problem and each CR value is reported. Tables 7 and 9 report functions in which at least one method found feasible solutions in all 25 runs with at least two different values of CR, the remaining functions are reported in Tables 8 and 10. Tables 7 and 8 report values for 10D problems and Tables 9 and 10 for 30D. In the four tables, *Centroid* $K + 1$ was used as the base method for the Kruskal–Wallis tests.

Table 7 suggests a clear dominance of *Centroid* $K + 1$ followed by Conservatism, which lost in 15 functions and won in 1 function, and Projection, which lost in 16 functions.

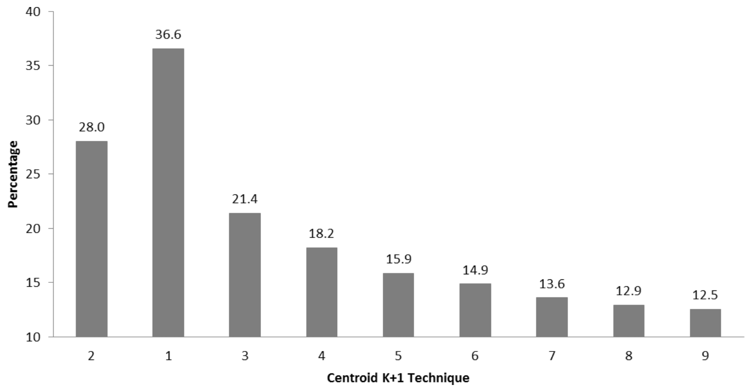
Table 8 shows that *Centroid* $K + 1$ got feasible results in a higher number of functions than any other method. It is also noted that *Centroid* $K + 1$ performed



(a) Number of runs/trials in which feasible solutions were found by each Centroid variant for 10D and 30D.



(b) Overall percentage of repaired vectors by each Centroid variant.



(c) Percentage of repaired vectors by each centroid variant who were better than their corresponding target vector.

Fig. 2. Graphical results of Experiment 1.

Table 7. Results obtained by each method on 10D problems where feasible solutions were found in each of the 25 runs by at least one method with at least 2 of the 3 tests with different CR values.

	CR	Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize all[30]
C01	1.0	–7.45E-01	**–7.47E-01**	–7.46E-01	–7.46E-01	**–7.47E-01**	–7.45E-01	–3.05E-01*	–7.47E-01
	0.8	–7.45E-01	**–7.47E-01***	–7.47E-01*	–7.47E-01*	–7.47E-01*	–7.47E-01*	–7.21E-01*	–7.47E-01*
	0.4	**–7.47E-01**	–7.47E-01*	–7.47E-01*	–7.47E-01*	–7.47E-01*	–7.47E-01*	–7.46E-01	–7.47E-01*
C04	1.0	**–1.00E-05**	–1.00E-05	–1.00E-05	–1.00E-05	–9.99E-06*	–1.00E-05	4.12E-02(24)*	–9.99E-06*
	0.8	**1.57E-03**	3.22E-03*	3.32E-03*	3.09E-03*	3.27E-03*	3.68E-03*	7.20E+00(22) *	3.50E-03*
	0.4	—	—	—	—	—	—	—	—
C07	1.0	**8.92E-24**	2.58E-22*	1.59E-01*	3.74E-22*	8.18E-22*	3.50E-21*	3.19E-01*	1.46E-21*
	0.8	**1.35E-10**	1.57E-09*	2.82E-09*	1.85E-09*	2.66E-09*	8.13E-09*	9.57E-01*	4.77E-09*
	0.4	**1.22E+00**	1.87E+00*	1.34E+00	1.86E+00	1.32E+00	1.42E+00	1.56E+00	1.75E+00
C08	1.0	9.06E+00	9.04E+00*	8.20E+00*	7.80E+00	8.34E+00*	9.49E+00*	4.08E+01*	**7.67E+00**
	0.8	9.82E+00	8.94E+00	9.00E+00	9.79E+00*	9.00E+00*	**8.16E+00**	1.45E+02*	9.82E+00*
	0.4	5.49E-01	3.14E-01	1.10E+00	**3.03E-01**	7.18E-01	1.99E+00	2.53E+01*	4.74E-01
C11	1.0	**–1.52E-03**	–1.52E-03*	–1.52E-03*	–1.52E-03*	–1.52E-03*	–1.52E-03*	–1.52E-03(22)*	–1.52E-03*
	0.8	**–1.52E-03**	–1.52E-03*	–1.52E-03*	–1.52E-03*	–1.52E-03*	–1.52E-03*	–1.52E-03(23)*	–1.52E-03*
	0.4	—	—	—	—	—	—	—	—
C12	1.0	–6.26E+00	–1.83E+01*	–6.21E-01*	–6.13E+00*	–2.66E+01*	–4.29E+01*	**–6.60E+01(24)**	–2.56E+00*
	0.8	–1.99E-01	–1.99E-01*	–1.05E+00*	–1.99E-01*	–3.11E+00*	–3.86E+00*	**–2.38E+02(24)+**	–6.30E-01*
	0.4	–1.99E-01(14)	–1.97E-01(6)*	–1.95E-01(3)*	–1.97E-01(3)*	–1.94E-01(4)*	–1.99E-01(1)	**–3.68E+01(19)**	–1.91E-01(4)*
C13	1.0	**–6.14E+01**	–5.94E+01	–6.07E+01	–6.13E+01	–6.05E+01	–6.09E+01	–6.10E+01	–6.10E+01
	0.8	–5.67E+01	–5.71E+01	–5.65E+01	**–5.72E+01**	–5.71E+01	–5.70E+01	–5.67E+01	–5.67E+01
	0.4	–6.84E+01	–6.84E+01	–6.84E+01	–6.84E+01	–6.84E+01	–6.84E+01	**–6.84E+01**	–6.84E+01
C14	1.0	**1.37E+11**	8.38E+12*	2.30E+12*	2.59E+12*	5.31E+12*	1.77E+13*	9.07E+13(16)*	6.82E+12*
	0.8	**9.48E+10**	1.13E+12*	2.46E+12*	1.68E+12*	3.41E+12*	1.25E+13*	4.22E+12	3.92E+12*
	0.4	**1.32E+10**	2.77E+11*	4.99E+11*	3.03E+11*	1.27E+12*	2.98E+12*	7.07E+10*	1.54E+12*
C15	1.0	**4.68E+12**	9.19E+13(23)*	3.91E+13*	3.95E+13*	4.86E+13*	1.87E+14*	—	4.63E+13*
	0.8	**2.65E+12**	1.86E+13*	2.94E+13*	2.70E+13*	5.12E+13*	1.65E+14*	8.91E+13(8)*	4.34E+13*
	0.4	**3.53E+12**	1.90E+13*	2.83E+13*	2.31E+13*	4.09E+13*	1.09E+14*	3.56E+13*	5.01E+13*

Table 8. Results obtained by each method on 10D problems where none of the methods found feasible solutions in all 25 runs with at least 2 of the 3 tests with different CR values.

	CR	Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C02	1.0	**3.30E+00**(15)	3.99E+00(11)*	3.60E+00**(16)**	3.53E+00(11)	3.59E+00(8)	3.98E+00(12)	4.68E+00(1)*	4.07E+00(8)*
	0.8	**2.73E+00(20)**	3.34E+00(16)	3.55E+00(16)	3.24E+00(19)	3.16E+00(16)	3.45E+00(13)	4.14E+00(7)*	3.67E+00(17*
	0.4	**2.02E+00(25)**	2.54E+00(23)	2.79E+00**(25)***	2.70E+00**(25)***	2.40E+00(21)	3.28E+00(24)*	2.44E+00**(25)**	3.40E+00(24)*
C03	1.0	2.38E-24**(25)**	**2.16E-25(25)**+	4.41E-25**(25)**+	6.93E-24**(25)***	5.02E-24**(25)***	1.36E-24**(25)**+	1.65E-22**(25)**+	2.24E-23**(25)***
	0.8	—	—	—	—	—	1.22E+15(2)	**3.24E+14(24)**	—
	0.4	—	—	—	—	—	—	—	—
C05	1.0	3.50E+02**(5)**	**2.62E+02**(3)	5.30E+02(3)	4.40E+02(4)	4.64E+02(1)	—	—	—
	0.8	—	—	—	—	—	—	**5.24E+02(4)**	—
	0.4	—	—	—	—	—	—	—	—
C06	1.0	**1.47E+02**(2)	3.89E+02(2)	—	4.17E+02(1)	—	4.62E+02**(8)**	—	—
	0.8	—	**3.63E+02**(2)	—	—	—	—	4.28E+02**(10)**	—
	0.4	—	—	—	—	—	—	—	—
C09	1.0	**1.47E+12**(7)	8.22E+12**(11)***	1.52E+13(1)*	4.93E+12(2)	1.67E+13(1)*	—	8.67E+12(1)*	9.25E+12(3)*
	0.8	**3.25E+11**(1)	8.22E+12**(2)**	—	—	—	3.43E+13(1)	1.55E+13**(2)**	—
	0.4	—	7.78E+12(2)	7.78E+12(3)	**7.61E+12(3)**	—	4.36E+13(1)	—	9.04E+12(1)
C10	1.0	**2.52E+12**(6)	5.93E+12**(7)**	1.85E+13(1)	7.73E+12(6)	9.36E+12(1)	—	—	—
	0.8	**2.59E+12(1)**	—	—	9.32E+12**(1)**	—	—	—	4.22E+12**(1)**
	0.4	**3.52E+12(4)**	—	6.87E+12(2)	—	—	—	6.55E+12(1)	4.43E+12(1)
C16	1.0	1.02E+00**(14)**	1.00E+00(12)	1.05E+00(3)	1.03E+00(6)	1.05E+00(3)	1.03E+00(5)	**9.59E-01**(2)	1.07E+00(3)
	0.8	1.03E+00 (4)	1.04E+00(3)	1.07E+00(5)*	1.04E+00(4)	1.04E+00(2)	1.04E+00(1)	**1.02E+00(11)**	1.07E+00(2)
	0.4	1.01E+00(14)	1.03E+00(15)*	1.03E+00(15)*	1.04E+00(11*	1.05E+00(13)*	1.06E+00(11)*	1.03E+00**(19)**	**9.98E-01**(12)*
C17	1.0	**2.28E+02(12)**	3.79E+02**(12)***	2.93E+02(10)	4.90E+02(9)*	6.96E+02(9)*	1.14E+03(6)*	3.56E+02(1)	5.45E+02(10)*
	0.8	**2.59E+02**(14)	4.13E+02**(15)***	4.24E+02(11)	4.45E+02(12)	4.97E+02(11)*	8.47E+02(14)*	6.01E+02(4)*	6.78E+02(10)*
	0.4	**1.43E+02**(10)	3.61E+02(14)*	4.68E+02**(17)***	5.35E+02(14)*	4.69E+02(12)*	7.17E+02(12)*	4.49E+02(16)*	5.38E+02(12)*
C18	1.0	**5.22E+03(23)**	1.01E+04(13)*	1.04E+04(17)*	1.08E+04(19)*	1.16E+04(18)*	2.48E+04(12)*	6.88E+03(2)*	1.12E+04(18)*
	0.8	**5.68E+03**(20)	1.11E+04(20)*	1.26E+04**(22)***	8.93E+03(21)*	1.47E+04**(22)***	1.97E+04(20)*	1.18E+04(5)*	9.91E+03(20)*
	0.4	**4.00E+03**(20)	8.74E+03(24)*	9.40E+03(23)*	1.03E+04(24*	1.29E+04(22)*	1.96E+04(24)*	1.09E+04**(25)***	1.03E+04(22)*

Table 9. Results obtained by each method on 30D problems where feasible solutions were found in each of the 25 runs by at least one method with at least 2 of the 3 tests with different CR values.

	CR	Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C01	1.0	–4.40E-01	–3.57E-01*	**–7.68E-01**+	–5.00E-01	–7.24E-01+	–5.46E-01+	–1.76E-01*	–2.51E-01*
	0.8	–7.82E-01	**–8.18E-01**+	–8.08E-01+	–4.65E-01*	–7.99E-01+	–5.40E-01*	–6.40E-01*	–2.62E-01*
	0.4	–7.94E-01	**–8.21E-01**+	**–8.21E-01**+	–8.20E-01+	**–8.21E-01**+	–8.18E-01+	–7.75E-01*	–4.29E-01*
C02	1.0	**3.98E+00(21)**	4.08E+00(3)	4.27E+00(16)	4.58E+00(19)	5.02E+00(2)	5.17E+00(3)	—	4.71E+00(5)*
	0.8	3.83E+00(24)	**3.51E+00**	4.13E+00(23)*	4.37E+00(20)	4.34E+00(23)	4.58E+00(13)	3.91E+00	4.46E+00(15)*
	0.4	3.19E+00	3.48E+00	3.64E+00	3.70E+00*	3.90E+00*	4.14E+00*	**3.13E+00**	4.05E+00*
C07	1.0	1.20E+07	2.12E+08*	**1.03E+06**+	2.36E+06+	2.47E+06+	8.60E+06	1.02E+09*	6.59E+06+
	0.8	**1.71E+01**	2.44E+01*	2.66E+01*	2.63E+01*	2.78E+01*	3.02E+01*	2.69E+08*	1.19E+02*
	0.4	**1.88E+01**	1.99E+01*	1.98E+01*	1.98E+01*	2.28E+01*	2.02E+01*	8.07E+01*	2.06E+01*
C08	1.0	1.78E+07	8.59E+07*	**7.06E+05**+	1.89E+06+	3.54E+06+	5.78E+06+	9.72E+08*	8.58E+06+
	0.8	**4.87E+01**	9.30E+04*	1.83E+05*	2.27E+05*	2.25E+05*	3.97E+05*	2.35E+03	6.28E+06*
	0.4	**2.13E+01**	2.30E+01*	2.31E+01*	2.30E+01*	2.33E+01*	2.43E+01*	2.04E+02*	2.51E+01*
C13	1.0	**–5.22E+01**	–5.18E+01	–5.05E+01	–4.98E+01	–5.09E+01	–5.19E+01	–5.06E+01	–5.08E+01
	0.8	–3.75E+01	–3.69E+01	–3.64E+01	–3.69E+01	–3.64E+01	–3.70E+01	**–4.94E+01**+	–3.66E+01
	0.4	–4.86E+01	–4.93E+01	–4.95E+01+	**–4.96E+01**+	–4.95E+01+	–4.94E+01+	–4.95E+01+	–4.92E+01
C14	1.0	**2.19E+12**	6.54E+13*	3.49E+13*	3.55E+13*	7.55E+13*	2.15E+14*	4.02E+14(13)*	1.25E+14*
	0.8	**3.51E+11**	1.85E+13*	3.70E+13*	3.32E+13*	6.67E+13*	2.63E+14*	2.04E+12	1.16E+14*
	0.4	8.18E+09	3.35E+12*	6.71E+12*	5.41E+12*	1.39E+13*	6.39E+13*	**2.20E+07**	5.00E+13*
C15	1.0	**1.74E+13**	1.86E+14(20)*	1.71E+14*	1.59E+14*	2.74E+14*	6.76E+14*	4.99E+14(2)*	2.46E+14*
	0.8	**1.47E+13**	9.44E+13*	1.90E+14*	1.50E+14*	2.40E+14*	7.17E+14*	4.78E+13*	2.34E+14*
	0.4	**1.62E+13**	1.29E+14*	1.95E+14*	1.60E+14*	2.54E+14*	6.14E+14*	2.20E+14*	2.27E+14*
C18	1.0	**1.47E+04**(24)	2.43E+04(11)*	3.03E+04*	2.72E+04*	4.02E+04(23)*	7.26E+04(18)*	3.69E+04(1)*	3.51E+04*
	0.8	**9.76E+03**	2.15E+04*	2.88E+04*	2.71E+04(24)*	3.28E+04	5.88E+04*	2.99E+04(23)*	3.28E+04*
	0.4	**9.88E+03**	2.40E+04	2.90E+04*	2.59E+04*	3.04E+04	5.21E+04*	3.21E+04*	3.26E+04*

Table 10. Results obtained by each method on 30D problems where none of the methods found feasible solutions in all 25 runs with at least 2 of the 3 tests with different CR values.

	CR	Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C05	1.0	4.14E+02(6)	4.39E+02**(10)**	5.43E+02(1)	—	—	—	**3.91E+02**(1)	—
	0.8	—	—	—	—	—	—	—	—
	0.4	—	—	—	—	—	—	—	—
C06	1.0	**4.15E+02**(8)	4.52E+02**(10)**	4.65E+02(2)	—	—	—	—	—
	0.8	—	—	—	—	—	—	**5.36E+02(4)**	—
	0.4	—	—	—	—	—	—	—	—
C09	1.0	**7.69E+12**(11)	2.14E+13**(14)**	4.28E+13(1)*	2.27E+13(1)	—	—	8.08E+12(1)	—
	0.8	**5.23E+12**(2)	2.43E+13(1)	2.61E+13(1)	3.26E+13(1)	—	—	—	4.11E+13**(3)**
	0.4	**8.61E+12** (1)	2.60E+13**(3)**	3.16E+13(2)	2.67E+13(2)	5.48E+13(1)	9.84E+13(2)	3.96E+13(1)	3.72E+13(2)
C10	1.0	**9.20E+12(11)**	2.12E+13(9)*	3.47E+13(1)	3.19E+13(1)	3.71E+13(1)	—	—	2.95E+13(1)
	0.8	**1.32E+13(3)**	2.06E+13(1)	4.03E+13(2)	—	3.45E+13(1)	7.71E+13(1)	4.25E+13(2)	—
	0.4	**9.17E+12 (3)**	3.09E+13(1)	6.54E+13(1)	—	3.56E+13(2)	6.23E+13(1)	—	4.15E+13(1)
C12	1.0	—	—	—	—	—	—	—	—
	0.8	–**6.42E-02(19)**	—	—	—	—	—	—	—
	0.4	–**1.98E-01(22)**	–1.96E-01**(22)***	–1.94E-01(21)*	–1.98E-01(21)*	–1.91E-01(20)*	–1.97E-01(18)*	–1.92E-01(6)*	–1.96E-01(18)*
C16	1.0	**1.06E+00**(13)	1.08E+00(16)*	1.12E+00**(20)***	1.13E+00(15)*	1.16E+00(7)*	1.15E+00(10)*	1.08E+00(6)	1.21E+00(2)*
	0.8	**1.07E+00**(5)	—	1.19E+00(1)	1.24E+00(2)	1.15E+00(1)	1.23E+00(1)	1.19E+00**(11)***	—
	0.4	**1.07E+00**(14)	1.14E+00(10)*	1.15E+00(7)*	1.13E+00(7)*	1.19E+00(6)*	1.27E+00(5)*	1.17E+00**(20)***	1.18E+00(5)*
C17	1.0	**7.63E+02(16)**	1.03E+03(10)	1.90E+03(14)*	1.52E+03(14)*	2.04E+03(5)*	2.98E+03(6)*	1.17E+03(4)*	1.99E+03(10)*
	0.8	**6.33E+02(18)**	1.34E+03(12)*	1.82E+03**(18)***	1.51E+03(10)*	1.92E+03(9)*	3.15E+03(16)*	1.64E+03(17)*	2.10E+03(13)*
	0.4	**6.23E+02(23)**	1.13E+03(19)*	1.69E+03(16)*	1.45E+03(18)*	1.83E+03(17)*	2.70E+03(21)*	1.61E+03(19)*	1.68E+03(21)*

better in most functions, followed by Evolutionary and Conservatism which provided a better performance than Centroid in 1 function but lost in 8. The rest of the methods had a worse performance.

Table 9 also shows a better performance for *Centroid* $K + 1$ in most of the functions, followed by Reflection which obtained a better performance than Centroid in 6 functions but lost in 12. Evolutionary had a better performance in 6 functions but lost in 14, while the rest of the methods performed worse.

Table 10 shows that *Centroid* $K + 1$ reached the feasible region in 16 functions, followed by Evolutionary which generated feasible values in 15 functions and Resampling in 14. It is also evident that Centroid obtained the best mean values in 15 functions followed by Conservatism which generated the best mean value in 2 functions. Regarding the amount of feasible runs, Centroid achieved the best results in 8 functions, followed by Resampling in 5 and Conservatism in 3. Finally, there was not any method that achieved a statistically significant better value for any function with respect to the results obtained by Centroid. Therefore, we can conclude that Centroid was the most competitive method based on the results reported in Table 10, followed by Resampling.

Data reported in Fig. 3 are the average values obtained by each method and the corresponding crossover rate.

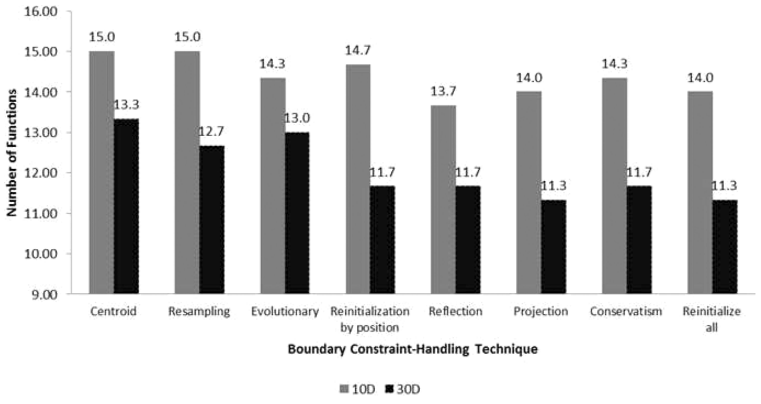
Figure 3(a) shows the mean value of the amount of functions in which feasible values were obtained by each CR value, CR=1.0, 0.8, 0.4, according to the BCHM and the dimensionality. For example, it is seen that for 10D, *Centroid* $K + 1$ got feasible values in 15 functions on average, of which 18 were for CR=1.0, 15 for CR=0.8 and 12 for CR=0.4. Resampling also obtained feasible values in 15 10D functions, followed by Reinitialize by position. In 30D, *Centroid* $K + 1$ also provided the highest number of feasible values in most of the functions, followed by Evolutionary and Resampling.

Figure 3(b) shows that for 10D, *Centroid* $K + 1$ obtained the highest number of feasible runs, followed by Resampling and Reinitialize by position, while in 30D, again, *Centroid* $K + 1$ computed the highest number of feasible runs, followed by Resampling and Evolutionary.

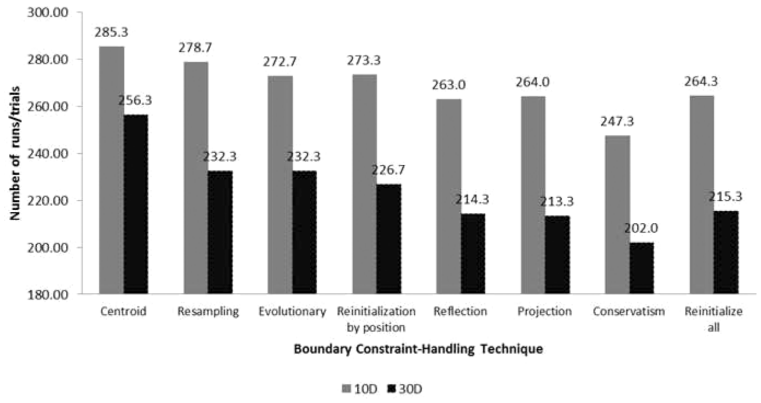
Figure 3(c) shows the percentage of repaired vectors. In this case, *Centroid* $K + 1$ obtained the lowest percentages of mutant vectors repaired for both 10D and 30D, followed by Resampling and Reflection. Figure 3(d) depicts the percentage of repaired mutant vectors who were better than their target vectors. Similarly, *Centroid* $K + 1$ got a better performance, followed by Evolutionary and Resampling.

Conclusions of Experiment 2.
From Experiment 2, we concluded that when the *DE/rand/1/bin* variant is used by varying the crossover factor, the *Centroid* $K + 1$ method provides the best stability and the best overall performance.

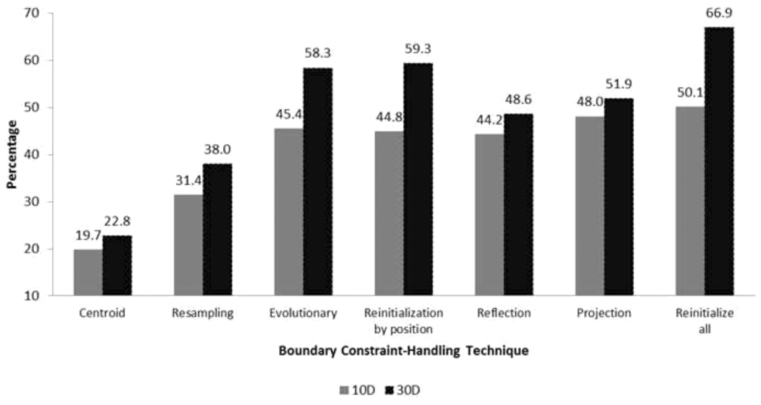
Centroid $K + 1$ showed the best performance in statistical tests in both 10D and 30D problems showing its capacity to generate better solutions.



(a) Number of functions in which feasible solutions were found by each method.

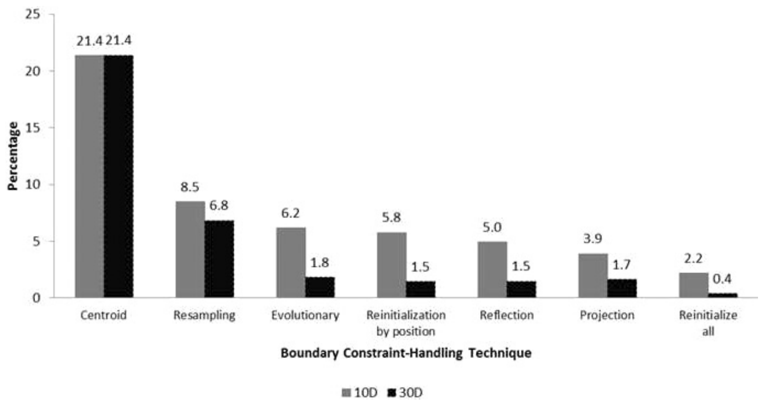


(b) Average number of runs/trials in which feasible solutions were found by each method.



(c) Overall percentage of repaired vectors by each method.

Fig. 3. Graphical results of Experiment 2.



(d) Percentage of repaired vectors by each method which were better than their corresponding target vectors.

Fig. 3. (*Continued*)

With the *Centroid* $K + 1$ method, feasible solutions were obtained in the larger number of independent runs and in the larger number of functions, showing its ability to reach the feasible region.

Centroid $K + 1$ was also the method that made fewer repairs, and of the repairs performed, it was the one that obtained the largest number of repaired vectors better than its corresponding target vector.

Evolutionary and Resampling methods also showed stability to the variation of the crossover factor, the first one due to its ability to generate good solutions and the second one due to its ability to reach the feasible region.

6.3. *Experiment* 3

Once the tests with the most popular DE variant were performed in Experiment 2, in this experiment another two variants commonly used in DE were tested: *DE/best/1/bin* and *DE/target-to-best/1* and the performance of *Centroid* $K + 1$ was compared against seven BCHM.

Tables 11 and 12 contain the 10D results, and those for 30D are included in Tables 13 and 14. In the four tables, rows labeled as *best* correspond to the *DE/best/1/bin* variant and the *target* label correspond to the *DE/target-to-best/1* variant. Tables 11 and 13 show the results of the functions in which at least one BCHM found feasible values in all 25 runs in at least one of the variants above mentioned.

Table 11 indicates that *Centroid* $K + 1$ had the best performance in the reported functions, followed by Reinitialize by position who had a better performance than Centroid in 2 functions but lost in 3. Reinitialize all had a better performance than Centroid in 3 functions but lost in 4.

Table 11. Results obtained by each method in 10D problems where feasible solutions were found in each of the 25 runs by at least one method with at least 1 of the 2 DE variants.

	Variant	Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C01	best	–5.08E-01	–5.83E-01+	–6.51E-01+	–6.74E-01+	**–6.85E-01**+	–5.96E-01+	–6.35E-01+	–5.84E-01+
	target	–6.15E-01	–6.81E-01+	–6.82E-01+	–6.96E-01+	–7.08E-01+	–6.91E-01+	**–7.33E-01**+	–6.86E-01+
C03	best	2.06E-04	2.11E-05	1.36E-03(24) *	3.45E-05	4.91E-05	1.06E+14(13)	6.26E-05	**1.50E-05**
	target	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
C04	best	1.44E+00(24)	8.62E-01	2.66E+00(19)	2.62E+00	8.14E-01(23)	**1.67E-01**(20)	3.45E+00(22)	1.94E+00
	target	**2.08E-01**(24)	1.43E+00(24)	6.82E-01	1.98E+00	1.86E+00	1.49E+00	7.76E-01	8.35E-01
C07	best	**3.42E+01**	1.19E+02	3.52E+01	1.00E+02	1.98E+02	9.14E+04	1.42E+02	4.38E+01
	target	2.67E+00	1.24E+01	3.68E+01	**2.56E+00**	2.72E+01	8.99E+00	5.83E+01	8.58E+00
C08	best	5.59E+01	2.32E+02	3.82E+01	**1.67E+01**	3.71E+02*	4.54E+02*	1.50E+02*	2.16E+02
	target	**1.77E+01**	6.16E+01	4.93E+01	5.58E+01	1.87E+01	6.66E+01	2.16E+01	2.73E+01
C13	best	–6.02E+01	–5.93E+01	**–6.10E+01**	–5.99E+01	–6.09E+01	–6.08E+01	–5.91E+01	–5.94E+01
	target	–6.14E+01	–6.17E+01	–6.21E+01	–6.22E+01	–6.24E+01	–6.18E+01	–6.20E+01	**–6.32E+01**+
C14	best	**3.04E+12**	2.83E+13*	1.52E+13*	1.14E+13*	2.16E+13*	2.78E+13*	8.88E+13*	1.79E+13*
	target	**1.18E+00**	5.76E+06*	2.57E+09*	3.35E+08*	5.38E+08*	3.04E+08*	6.00E+11*	4.52E+10*
C15	best	**6.43E+13**(17)	2.09E+14(5)*	2.34E+14(16)*	9.68E+13(17)	1.49E+14(11)*	2.91E+14(13)*	4.29E+15(13)*	1.45E+14**(18)***
	target	**3.09E+12**	6.33E+12*	2.30E+13*	9.88E+12*	2.13E+13*	5.07E+13*	6.34E+14*	2.83E+13*

Table 12. Results obtained by each method in 10D problems where none of the methods found feasible solutions in all 25 runs with none of the 2 DE variants.

	Variant	Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C02	best	3.43E+00 (4)	—	4.02E+00(**9**)	4.20E+00(1)	3.56E+00(2)	3.89E+00(2)	**3.07E+00**(3)	3.93E+00(1)
	target	2.93E+00(**22**)	**2.81E+00**(16)	3.33E+00(17)	3.45E+00(10)	3.27E+00(18)	3.31E+00(17)	3.40E+00(16)	2.91E+00(12)
C05	best	**3.45E+02(10)**	—	5.24E+02(4)	—	4.76E+02(2)	5.77E+02(1)	—	—
	target	—	—	—	—	—	—	—	—
C06	best	**3.13E+02(24)**	5.04E+02(5)*	4.66E+02(22)*	—	5.00E+02(4)*	5.81E+02(11)*	—	—
	target	—	—	—	—	—	—	—	—
C09	best	**1.08E+12**(1)	—	2.15E+13(**5**)	—	—	5.10E+13(1)	—	—
	target	7.95E+12(**2**)	**4.86E+12(2)**	—	—	9.63E+12(1)	1.57E+13(1)	2.49E+13(1)	—
C10	best	**8.49E+12**(2)	—	1.42E+13(**5**)	—	—	3.77E+13(1)	—	—
	target	**6.82E+12(1)**	—	—	—	1.98E+13(**1**)	—	—	—
C11	best	–**1.52E-03(1)**	–**1.52E-03(1)**	—	—	–**1.52E-03(1)**	–**1.52E-03(1)**	–**1.52E-03(1)**	–**1.52E-03(1)**
	target	–**1.52E-03**(3)	–**1.52E-03**(1)	–**1.52E-03(4)**	–**1.52E-03**(3)	–**1.52E-03**(3)	–**1.52E-03**(2)	–**1.52E-03**(2)	–**1.52E-03**(2)
C12	best	–1.68E+01(14)	–1.97E+00(7)	–1.27E+01(9)	–1.87E+01(**17**)	–4.50E+01(11)	–**7.60E+01**(11)	–4.33E+01(12)	–7.72E-01(15)
	target	–5.43E+00(19)	7.94E+00(18)	5.31E-01(17)	–8.39E+00(**22**)	1.14E+01(16)	1.55E+01(14)	1.86E+00(17)	–**3.74E+01**(20)
C16	best	**9.49E-01(23)**	1.02E+00(6)	1.04E+00(11)*	1.01E+00(5)	1.06E+00(7)*	1.05E+00(6)*	1.05E+00(6)*	—
	target	1.03E+00(**17**)	1.04E+00(13)*	1.03E+00(15)	**1.02E+00**(15)	1.03E+00(8)	1.04E+00(8)	1.03E+00(8)	1.05E+00(2)
C17	best	3.56E+02(8)	5.11E+02(4)	1.05E+03(**9**)*	5.20E+02(1)	4.32E+02(1)	9.52E+02(5)*	—	**1.57E+02**(2)
	target	**3.31E+02(18)**	5.17E+02(17)*	7.23E+02(16)*	3.88E+02(11)	5.81E+02(8)*	9.86E+02(13)*	1.36E+03(8)*	5.13E+02(9)
C18	best	6.75E+03(**12**)	**4.05E+03**(1)	2.01E+04(9)*	1.31E+04(4)	1.31E+04(4)	3.17E+04(2)*	—	1.33E+04(5)*
	target	**5.58E+03(24)**	7.56E+03(22)	1.34E+04(20)*	8.83E+03(17)*	1.24E+04(19)*	1.52E+04(22)*	4.46E+04(15)*	1.14E+04(20)*

Table 13. Results obtained by each method in 30D problems where feasible solutions were found in each of the 25 runs by at least one method with at least 1 of the 2 DE variants.

	Variant	Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C01	best	–3.06E-01	–2.04E-01*	**–4.22E-01**+	–3.52E-01+	–3.78E-01+	–3.65E-01+	–2.73E-01*	–2.37E-01*
	target	–3.33E-01	–3.38E-01	–4.99E-01+	–4.33E-01+	**–5.10E-01**+	–4.32E-01+	–4.45E-01+	–3.84E-01+
C02	best	**4.03E+00**(9)	4.38E+00(1)	4.71E+00**(19)***	4.55E+00(9)	5.24E+00(1)	4.47E+00(3)	4.83E+00(2)	4.81E+00(5)*
	target	**3.67E+00**	4.06E+00(19)*	4.40E+00(20)*	4.01E+00(19)	4.26E+00(13)*	4.58E+00(17)*	4.72E+00(13)*	4.59E+00(9)*
C06	best	**3.88E+02**	—	5.90E+02(18)*	—	—	6.07E+02(1)*	—	—
	target	—	—	—	—	—	—	—	—
C07	best	**3.03E+08**	2.49E+09*	1.45E+09*	4.32E+08	1.74E+09*	1.83E+09*	1.47E+09*	5.09E+08
	target	1.37E+08	1.37E+08	9.71E+07	4.10E+07+	2.09E+08	2.44E+08	3.33E+08*	**3.78E+07**+
C08	best	3.85E+08	1.45E+09*	1.00E+09*	3.59E+08	1.22E+09*	1.78E+09*	1.14E+09*	**2.54E+08**
	target	9.04E+07	1.83E+08*	2.12E+08*	8.32E+07	2.19E+08*	2.52E+08*	2.68E+08*	**6.37E+07**
C13	best	–4.53E+01(23)	**–4.87E+01**	–4.58E+01(20)	–4.71E+01	–4.49E+01(22)	–4.25E+01	–4.54E+01(23)	–4.76E+01
	target	–4.78E+01(23)	**–5.32E+01**+	–5.24E+01+	–5.28E+01+	–5.15E+01	–5.18E+01	–5.20E+01	–5.25E+01
C14	best	**2.16E+13**	1.80E+14(24)*	9.40E+13*	7.51E+13*	1.57E+14*	1.49E+14*	4.53E+14*	1.30E+14*
	target	5.50E+11	2.20E+11+	8.40E+11	1.81E+11+	1.10E+12*	9.66E+11*	1.24E+12*	**6.49E+10**+
C15	best	**1.55E+14**	2.72E+14(4)	5.70E+14(24)*	2.04E+14*	5.30E+14(20)*	8.91E+14(18)*	8.33E+15(14)*	2.80E+14*
	target	**1.40E+13**	1.52E+13	7.87E+13*	4.65E+13*	9.39E+13*	1.13E+14*	7.92E+14*	1.73E+14*
C18	best	**2.68E+04**(16)	—	7.26E+04(8)*	3.42E+04(11)	4.14E+04(4)	7.12E+04(3)*	4.04E+04(1)	3.85E+04**(20)***
	target	1.40E+04	**1.33E+04**	3.12E+04*	2.10E+04*	3.48E+04*	4.36E+04*	2.24E+05(24)*	3.40E+04*

In those functions in Table 12, *Centroid* $K+1$ found feasible values in 18 functions, Projection in 17 and Reflection in 16. Respect to the number of feasible runs, *Centroid* $K+1$ found the most of them in 11 functions, Evolutionary in 5 and finally Resampling, Reinitialize by position and Reflection in 2. Concerning the mean value, *Centroid* $K+1$ found the best values in 10 problems, Resampling in 5 problems and Reinitialize all found the best mean value in 4 problems. In summary, for the problems reported in Table 12, the best performance was obtained by *Centroid* $K+1$ followed by Resampling, Evolutionary and Projection.

Table 13 suggests that Reinitialize by position and *Centroid* $K+1$ obtained a similar performance considering that Centroid found feasible results in 17 functions against 16 of Reinitialize by position, though the latter outperformed Centroid in 5 functions and obtained a lower performance in 4 of them. Subsequently, Resampling and Reinitialize all also had a similar performance. Resampling found feasible values in 15 functions, outperformed Centroid in 2 of them and lost in 6, while Reinitialize all found feasible values in 16 functions, and in 3 provided a better performance than Centroid and in 8 got a lower performance.

In Table 14, *Centroid* $K+1$ and Evolutionary generated feasible values in 8 functions followed by Reinitialize by position, which did it in 6 functions. *Centroid* $K+1$ found the highest number of feasible runs in 5 functions, followed by Evolutionary in 3 and Resampling in 1. It was also observed that Centroid found the best mean value in 8 functions and Evolutionary in just 1. Therefore, for the functions reported in Table 14, *Centroid* $K+1$ was the best method followed by Evolutionary.

Figure 4(a) shows that, on average, *Centroid* $K+1$ found feasible values in most of the 10D functions, followed by Projection and Reflection. In 30D, *Centroid* $K+1$ and Evolutionary together generated feasible values in most of the functions, followed by Reinitialize by position. Figure 4(b) presents that, on average, *Centroid* $K+1$ provided the most feasible runs for both, 10D and 30D functions, and it was subsequently followed by Evolutionary and Reinitialize by position.

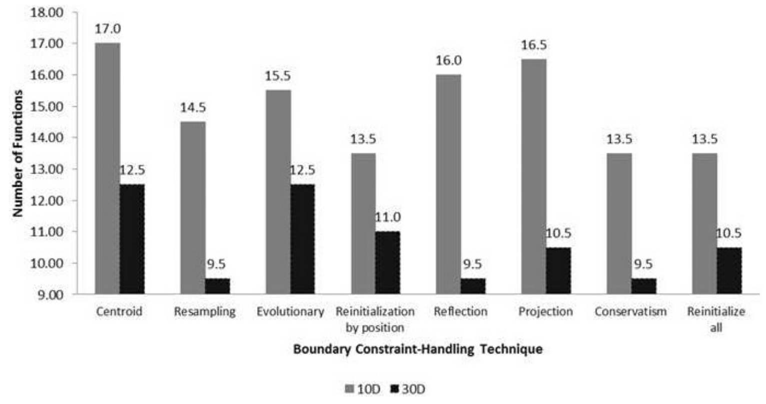
Regarding the percentage of mutant vectors repaired, Fig. 4(c) shows that *Centroid* $K+1$ is the method that made fewer repairs for both, 10D and 30D problems, followed by Resampling and Evolutionary. Figure 4(d) reports that *Centroid* $K+1$ was the method with the highest number of repaired individuals who were better than their target vectors for both, 10D and 30D, followed by Resampling and Evolutionary.

Conclusions of Experiment 3.
As a summary, from this experiment, it is concluded that *Centroid* $K+1$ was the method that showed a better stability and performance when using two different strategies: *DE/best/1/bin* and *DE/target-to-best/1*.

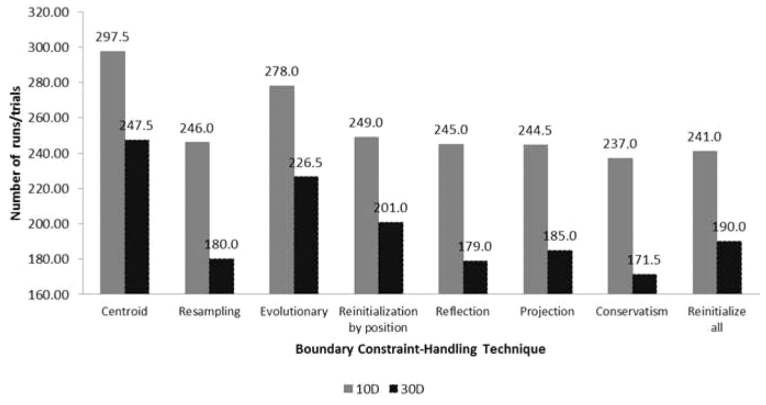
With *Centroid* $K+1$, the best results were obtained for both 10D and 30D problems and the largest number of feasible runs was obtained, no matter what strategy has been employed. In addition, *Centroid* $K+1$ was the method that performed the least repairs and the one that generated the highest percentage of

Table 14. Results obtained by each method in 30D problems where none of the methods found feasible solutions in all 25 runs with none of the 2 DE variants

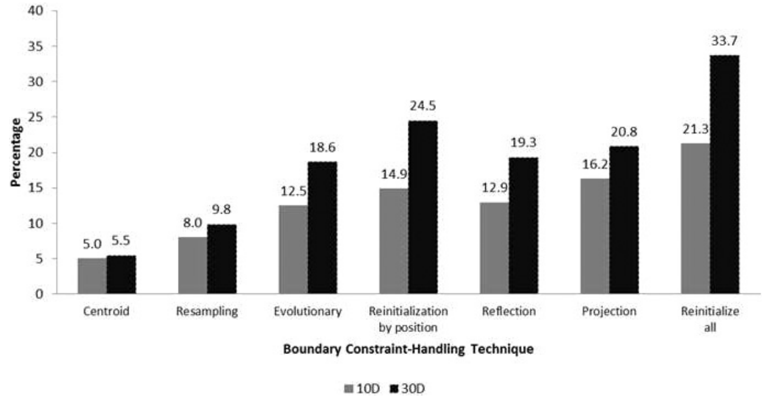
	Variant	Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C05	best	**3.53E+02(14)**	—	5.54E+02(7)*	—	—	—	—	—
	target	—	—	—	—	—	—	—	—
C09	best	—	—	**5.80E+13(6)**	—	—	—	—	—
	target	**1.53E+13(4)**	2.13E+13(3)	6.94E+13(1)	3.56E+13(1)	—	—	—	5.21E+13(1)
C10	best	**3.43E+12**(1)	—	5.63E+13**(9)**	3.71E+13(1)	—	—	—	3.50E+13(1)
	target	**1.20E+13**(3)	1.88E+13**(4)**	—	1.94E+13(1)	—	7.87E+13(2)	—	—
C16	best	**1.09E+00(24)**	—	1.15E+00(13)*	1.12E+00(18)*	1.15E+00(3)*	1.13E+00(4)	1.10E+00(1)	—
	target	**1.06E+00(22)**	1.07E+00(10)	1.09E+00(2)	**1.06E+00**(2)	1.09E+00(4)	1.07E+00(7)	1.09E+00(8)*	1.08E+00(1)
C17	best	**1.20E+03**(9)	—	2.61E+03**(13)***	—	—	—	—	2.25E+03(5)*
	target	**8.19E+02(22)**	8.53E+02(20)	2.39E+03(18)*	1.19E+03(15)*	1.86E+03(16)*	3.11E+03(15)*	1.25E+04(7)*	1.84E+03(13)*



(a) Number of functions in which feasible solutions were found by each method.

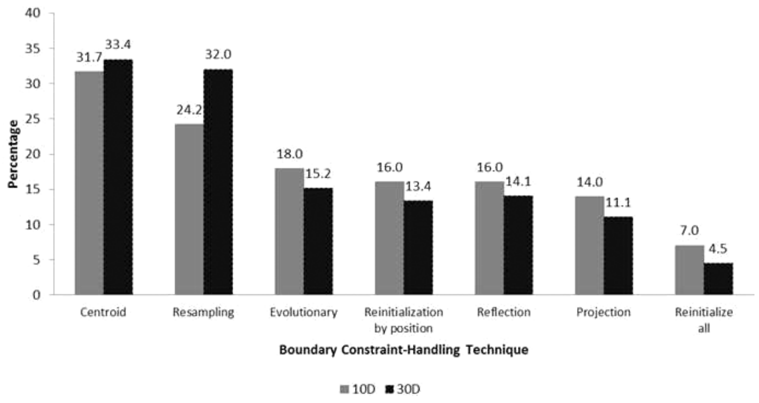


(b) Average number of runs/trials in which feasible solutions were found by each method.



(c) Overall percentage of repaired vectors by each method.

Fig. 4. Graphical results of Experiment 3.



(d) Percentage of repaired vectors by each method which were better than their corresponding target vectors.

Fig. 4. (*Continued*)

repaired individuals that were better than their corresponding target vector.

Another method that showed a good performance was Evolutionary, mainly for its ability to reach the feasible region and the quality of its solutions in 30D problems, followed by Resampling and Reinitialize by position.

6.4. *Experiment* 4

After analyzing the performance of the 8 BCHM in 3 DE variants with three different CR values, weaknesses were found with the *Resampling* method which will be reported in this experiment.

Resampling method takes a sample of k vectors at random and with them, the mutation operator is implemented in order to obtain a mutant vector. When the mutant vector goes out of the search space, a new sample is taken and this process is repeated until a valid mutant vector is obtained.

When *DE/rand/1/bin* is used, as shown in Table 1, three vectors are chosen randomly, i.e. $k = 3$ (r_1, r_2 and r_3). Considering that in Experiments 2 and 3, populations of size 100 were used and that the three vectors must be different from each other and different from the target vector, there are a total of 156 849 different ways to choose three vectors of a total of $n = 99$, this can be corroborated by using the binomial coefficient function[10]:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{16}$$

For *DE/best/1/bin* and *DE/target-to-best/1* variants, where two vectors are chosen randomly (see Table 1), there is a total of 4851 different ways to choose $k = 2$ vectors of a total of $n = 99$.

The above mentioned is very important since *Resampling* takes a new sample each time the mutant vector is invalid, and we must keep in mind that there is a finite number of different samples. Even more significant, this amount varies depending on the DE variant adopted and its population size.

6.4.1. *Reboot factor*

When all those possible combinations already discussed were used by *Resampling* and a valid vector is not obtained, the algorithm falls into an infinite loop and must be restarted with the hope that the same thing does not happen.

The Column labeled as *Reboots* in Table 15 shows the average Reboots required to obtain a total of 25 valid runs. As it can be seen, the worst instances are shown in the last four rows, i.e. when *DE/best/1/bin* and *DE/target-to-best/1* variants were employed in 30D functions. Such situation became worst when the CR value was decreased.

In such cases, the algorithm was cycled after having searched the 4851 different ways to choose the r_1 and r_2 vectors without producing a valid mutant vector. In such situations, it is not suitable to use the *Resampling* method.

6.4.2. *Resampling factor*

In addition to the reboot factor, the *Resampling* usage itself represents a problem, even when the algorithm does not reach cyclizing. In certain cases, it is necessary to perform a lot of samples, then increasing the execution time of the algorithm.

Table 15. Resampling and Reboots. The table shows the number of reboots necessary to get 25 valid runs and the amount of samples required to repair an invalid mutant vector required by *Resampling* method based on the problem dimensionality, the DE variant and the crossover rate.

Dim	Variant	CR	Resampling	Reboots
10D	best/1/bin	1.00	0.04	0.00
10D	rand/1/bin	1.00	0.39	0.00
10D	best/1/bin	0.80	0.47	0.00
30D	rand/1/bin	1.00	0.65	0.00
10D	target-to-best/1	1.00	0.84	0.00
10D	best/1/bin	0.40	1.95	0.00
10D	rand/1/bin	0.80	2.63	0.00
10D	rand/1/bin	0.40	3.01	0.00
30D	rand/1/bin	0.80	38.03	0.00
30D	rand/1/bin	0.40	117.64	0.00
30D	best/1/bin	1.00	–	9.17
30D	target-to-best/1	1.00	–	11.50
30D	best/1/bin	0.80	–	12.33
30D	best/1/bin	0.40	–	14.89

The Column labeled as Resampling in Table 15 shows the average samples needed to repair a single individual. Resampling values for the last four rows were omitted because in such cases the algorithm had to restart repeatedly.

Excluding the last four rows, it is noted that the number of samples increases as the dimensionality of the problem is increased as well and the CR value is reduced, e.g. on average, 117.64 samples were necessary to repair an individual in a 30D function using *DE/rand/1/bin* variant and CR = 0.4.

That behavior definitely affects the execution time of the algorithm, to such an extent that if the average time used to resolve a problem with *Resampling* method is compared against the average time used to resolve a problem with *Centroid* $K + 1$ the ratio is almost 2 to 1 (see Fig. 5).

Conclusions of Experiment 4.
From this experiment, it is concluded that the *Resampling* method can cycle the execution of DE algorithm when all possible re-samples were tested and it was not possible to generate a valid mutant vector, and even in cases where a valid mutant vector has been obtained the number of re-samples can be so high that it significantly increases the execution time of the algorithm.

The *Resampling* method is very sensitive to the dimensionality, the CR value, the population size, and the DE variant adopted.

6.5. *Experiment* 5

In this experiment, the percentage of repaired vectors by problem was analyzed as well as how this percentage was affected by the dimensionality, DE variant and crossover rate.

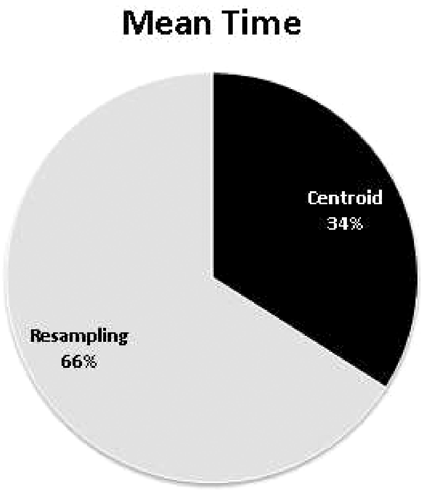


Fig. 5. Comparative of average runtime of the 36 test problems between Resampling and Centroid methods.

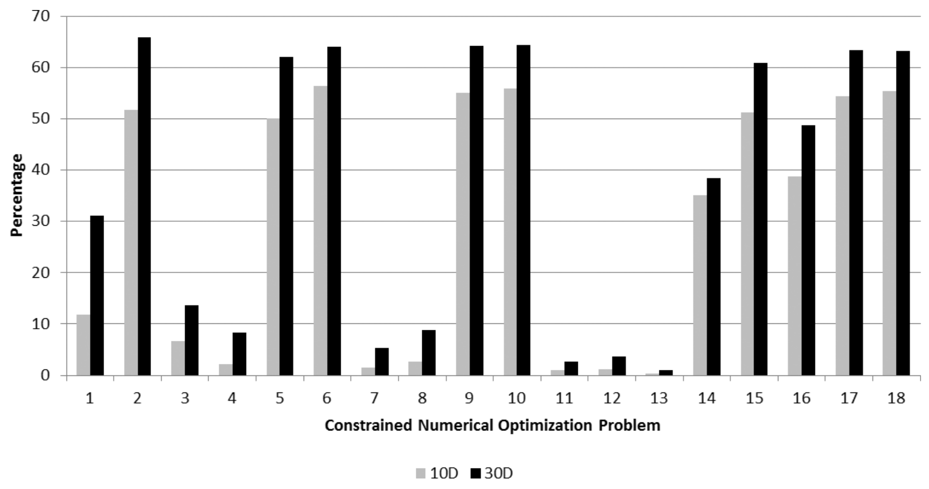


Fig. 6. Percentage of repaired individuals per each test problem in 10D and 30D.

All reparation percentages presented in this experiment were obtained as the average of the accumulation of mutant vectors repaired by all the 8 BCHM used in Experiments 2 and 3.

Figure 6 states that the main factor affecting the percentage of repaired vectors was the nature of the problem. There were problems that required few reparations, whereas in others cases, the majority of mutant vectors were repaired. Problems requiring a high percentage of repairs have the characteristic that they have separable restrictions of equality.

The second factor affecting the repair rate was the dimensionality, i.e. in high dimensionality problems, a high repair percentage was observed.

Table 16 shows the percentage of individuals repaired for each function in both, 10D and 30D, employing different values of CR = {0.4, 0.8, 1.0}; *DE/rand/1/bin* was used in this experiment.

As shown, the repair percentage is similar when values of CR = 0.4 or 0.8 were used. However, only when the crossover was not used, i.e. CR = 1, in problems that required a high percentage of reparations, a decrease was observed in the percentage of individuals repaired.

Table 17 summarizes the percentage of individuals repaired by function and dimension for different DE variants: *DE/best/1/bin*, *DE/target-to-best/1* and *DE/rand/1/bin*. In all cases, CR = 1 was used. In this table, it is observed that when DE/best/1/bin was used, fewer reparations were required, while between DE/target-to-best/1 and DE/rand/1/bin, no significant differences were found.

Conclusions of Experiment 5.
From this experiment, it is concluded that the percentage of repaired vectors in the functions solved in this paper can vary from a value less than 1% to a value greater than 90%.

Table 16. Percentage of individuals repaired by function and dimension, based on the crossover rate. CR={0.4, 0.8, 1.0}. Problems with the highest percentage of repaired individuals are highlighted in gray.

	10 Dimensions			30 Dimensions		
	0.4	0.8	1.0	0.4	0.8	1.0
C01	14.56	16.43	21.23	43.23	69.13	38.70
C02	81.27	79.36	49.26	97.83	96.02	50.26
C03	10.36	17.12	5.45	34.16	29.75	3.82
C04	5.40	2.46	1.94	20.20	15.72	4.91
C05	78.79	76.57	47.38	93.77	92.17	49.46
C06	85.59	83.25	54.10	95.40	93.16	50.75
C07	1.84	2.33	2.93	4.08	12.78	8.69
C08	3.73	4.02	4.40	9.99	23.34	9.87
C09	83.23	81.34	52.82	94.85	92.88	50.10
C10	82.86	81.45	55.66	95.00	92.87	51.03
C11	1.27	1.37	1.54	2.60	6.17	4.18
C12	1.62	1.82	2.06	3.41	8.26	5.93
C13	0.50	0.51	0.58	1.01	2.31	1.78
C14	59.62	63.41	41.70	66.56	78.68	35.45
C15	82.27	80.71	51.53	94.05	91.67	51.56
C16	68.66	66.77	38.01	90.91	88.93	34.63
C17	82.90	80.80	52.88	94.40	92.76	49.84
C18	84.41	82.21	54.72	94.63	92.88	51.02
Mean	46.05	45.66	29.90	57.56	59.97	30.67

Problems with separable equality constraints carry out a higher percentage of repairs. This percentage is also affected by a high dimensionality, the use of low CR values and employing variants such as *DE/rand/1/bin*.

When a high reparation percentage is required by a function, it should take special care in choosing the BCHM, which offers a good performance with a low computational cost.

6.6. *Experiment* 6

The goal of this experiment was to compare the performance of the 8 BCHM in a state-of-the-art DE-based algorithm for CNOPs named ICDE, proposed by Wang.[15]

Tables 18 and 19 present the mean and standard deviation results for 10D problems. The 30D results are included in Tables 20 and 21. Tables 18 and 20 have the results for functions where at least one method found feasible solutions in all 25 runs. The rest of functions are reported in Tables 19 and 21.

Table 18 indicates that *Centroid* $K + 1$ was the method with the best performance followed by Resampling and Evolutionary, which had a similar performance to Centroid in 7 functions and lost in 2 functions. Subsequently, Reinitialize by position and Projection only lost in 3 functions.

Table 17. Percentage of individuals repaired by function and dimension with different DE variants. The column labeled as best means that DE/best/1/bin variant was used, target means DE/target-to-best/1, and finally, rand means DE/rand/1/bin. In all cases, CR=1 was used.

	10 Dimensions			30 Dimensions		
	best	target	rand	best	target	rand
C01	2.65	3.77	21.23	1.20	3.63	38.70
C02	3.43	45.37	49.26	14.55	70.39	50.26
C03	0.38	0.34	5.45	0.17	0.18	3.82
C04	0.38	0.31	1.94	0.27	0.26	4.91
C05	3.68	43.54	47.38	5.52	69.50	49.46
C06	4.76	53.93	54.10	7.72	72.79	50.75
C07	0.42	0.39	2.93	0.39	0.44	8.69
C08	0.53	0.47	4.40	0.46	0.47	9.87
C09	5.04	53.22	52.82	8.35	74.66	50.10
C10	5.30	53.91	55.66	8.14	74.88	51.03
C11	0.22	0.19	1.54	0.17	0.15	4.18
C12	0.25	0.22	2.06	0.17	0.15	5.93
C13	0.11	0.11	0.58	0.13	0.13	1.78
C14	2.72	8.35	41.70	2.37	8.98	35.45
C15	4.18	37.43	51.53	10.42	56.65	51.56
C16	2.78	17.85	38.01	1.48	28.02	34.63
C17	3.90	51.63	52.88	7.20	72.65	49.84
C18	4.77	50.65	54.72	9.24	68.16	51.02
Mean	2.53	23.43	29.90	4.33	33.45	30.67

Table 19 shows that Centroid, Resampling, Reinitialize by position and Reflection found feasible values in 9 functions. Resampling found the highest number of feasible runs in 8 problems, followed by Centroid, Projection and Evolutionary, which found the highest number of feasible runs in 1 problem each. Regarding the best mean value, Centroid found the best values in 6 functions, followed by Resampling, Reflection and Evolutionary, which found the best values in 2 functions. Additionally, it is noted that none of the methods provided better statistically significant results than Centroid in any function. In summary, for the functions reported in Table 19, Resampling is the method that found the highest number of feasible runs, while Centroid found the best results in most of the functions.

Table 20 shows that Centroid is the method with the best performance in most of the reported problems, followed by Resampling, which had a better performance in 1 function but lost in 6. Conservatism had a better performance than Centroid in 1 function, lost in 6 functions and did not find feasible values in 2 functions.

Table 21 shows that Centroid was the only method that found feasible values in the 5 functions, followed by Resampling, Projection and Conservatism, which found feasible values in 4 functions. Resampling found the highest number of feasible runs

Table 18. ICDE results obtained by each method in 10D problems where feasible solutions were found in each of the 25 runs by at least one method.

		Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C01	M	**–7.47E-01**	**–7.47E-01**	**–7.47E-01**	–7.46E-01	**–7.47E-01**	–7.46E-01	–7.14E-01*	**–7.47E-01**
	SD	1.35E-03	3.12E-16	2.87E-16	2.24E-03	1.35E-03	2.24E-03	4.16E-02	3.28E-16
C03	M	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.42E-24	**0.00E+00**
	SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.11E-24	0.00E+00
C04	M	**–1.00E-05**	**–1.00E-05**	**–1.00E-05**	**–1.00E-05**	**–1.00E-05***	**–1.00E-05**	**–1.00E-05***	**–1.00E-05***
	SD	5.74E-13	1.16E-12	6.41E-13	7.91E-13	1.19E-12	6.54E-13	2.17E-12	1.55E-12
C07	M	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.59E-01	**0.00E+00**
	SD	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.97E-01	0.00E+00
C08	M	9.92E+00	9.49E+00	9.04E+00	9.49E+00	9.30E+00	**8.65E+00**	3.37E+01	1.06E+01
	SD	2.50E+00	3.15E+00	3.68E+00	3.15E+00	3.51E+00	4.07E+00	6.39E+01	1.22E-01
C12	M	–3.12E+00	–6.19E+00	–6.33E+00	–6.76E+00*	**–9.33E+00***	–1.70E+00*	–2.19E+00	–6.98E-01*
	SD	5.32E+00	2.32E+01	2.32E+01	2.32E+01	3.16E+01	4.14E+00	4.67E+00	2.49E+00
C13	M	–6.46E+01	–6.35E+01	**–6.55E+01**	–6.41E+01	–6.54E+01	–6.17E+01	–6.39E+01	–6.38E+01
	SD	2.62E+00	4.37E+00	3.66E+00	3.98E+00	2.88E+00	6.61E+00	4.28E+00	4.23E+00
C14	M	**1.78E+08**	5.28E+10*	8.84E+10*	9.95E+10*	1.88E+11*	2.43E+11*	6.33E+13*	2.48E+11*
	SD	3.51E+08	6.53E+10	9.37E+10	1.08E+11	1.33E+11	2.35E+11	5.95E+13	2.41E+11
C15	M	**1.94E+11**	5.05E+12*	7.41E+12*	1.48E+13*	1.33E+13*	7.69E+13*	5.74E+18(24)*	2.16E+13*
	SD	3.13E+11	5.44E+12	9.17E+12	1.88E+13	1.84E+13	1.07E+14	1.27E+19	2.12E+13

Table 19. ICDE results obtained by each method in 10D problems where none of the methods found feasible solutions in all 25 runs.

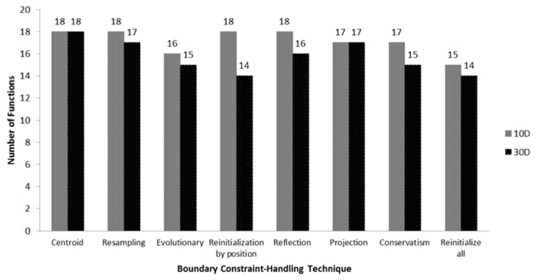
		Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C02	M	**2.04E+00**(19)	2.71E+00**(24)**	3.25E+00(14)*	3.27E+00(17)*	3.39E+00(12)*	3.24E+00(10)	3.95E+00(6)*	3.74E+00(17)*
	SD	1.61E+00	1.29E+00	1.04E+00	1.38E+00	1.48E+00	1.31E+00	1.74E+00	1.21E+00
C05	M	**2.49E+02**(2)	3.49E+02**(17)**	—	4.55E+02(1)	4.54E+02(2)	—	—	—
	SD	1.04E+02	1.15E+02	—	0.00E+00	5.25E+00	—	—	—
C06	M	4.24E+02(3)	3.80E+02**(14)**	—	4.53E+02(1)	**3.69E+02**(2)	4.47E+02**(14)**	3.02E+03(3)	—
	SD	1.64E+02	1.05E+02	—	0.00E+00	2.93E+02	2.02E+02	1.21E+03	—
C09	M	3.21E+12(6)	6.47E+12**(24)***	**2.35E+12**(2)	1.51E+13(1)	1.64E+13(2)	3.05E+13(9)*	1.82E+17(4)*	1.11E+13(5)*
	SD	5.27E+12	3.41E+12	4.59E+11	0.00E+00	1.14E+13	1.56E+13	2.21E+17	3.40E+12
C10	M	**9.47E+11**(3)	6.03E+12**(20)***	1.17E+13(1)	1.35E+13(1)	5.95E+12(1)	2.48E+13(3)	1.67E+16(2)	—
	SD	1.05E+12	4.66E+12	0.00E+00	0.00E+00	0.00E+00	3.41E+12	1.81E+16	—
C11	M	–**1.52E-03** (10)	–**1.52E-03**(12)	–**1.52E-03(15)**	–**1.52E-03**(10)	–**1.52E-03**(10)	–**1.52E-03**(13)	–**1.52E-03**(9)	–**1.52E-03**(10)
	SD	1.22E-17	1.44E-17	9.93E-18	7.11E-18	1.27E-17	6.77E-18	5.83E-17	9.37E-18
C16	M	9.60E-01 (13)	**9.37E-01(16)**	1.06E+00(1)	1.03E+00(2)	1.05E+00(3)	9.65E-01(4)	9.77E-01(4)	9.59E-01(1)
	SD	2.03E-01	1.68E-01	0.00E+00	1.88E-02	1.59E-02	1.83E-01	1.15E-01	0.00E+00
C17	M	**2.60E+02**(11)	3.42E+02**(20)**	5.59E+02(14)*	4.64E+02(8)*	5.43E+02(13)*	8.24E+02(11)*	9.45E+03(4)*	6.76E+02(6)*
	SD	1.36E+02	2.65E+02	2.34E+02	1.43E+02	3.18E+02	4.02E+02	7.30E+03	1.17E+02
C18	M	**3.95E+03(24)**	7.68E+03**(24)***	9.49E+03(20)*	9.28E+03(21)*	1.09E+04(16)*	1.99E+04(16)*	1.81E+06(7)*	1.24E+04(13)*
	SD	3.83E+03	5.10E+03	5.29E+03	4.23E+03	4.94E+03	1.31E+04	1.54E+06	5.64E+03

Table 20. ICDE results obtained by each method in 30D problems where feasible solutions were found in each of the 25 runs by at least one method.

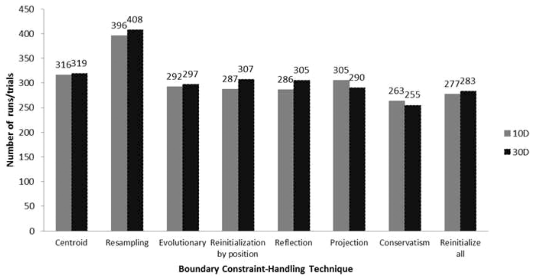
		Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C01	M	−7.45E-01	−7.94E-01$^+$	−8.14E-01$^+$	−7.43E-01	**−8.17E-01**$^+$	−7.59E-01$^+$	−7.76E-01$^+$	−5.25E-01*
	SD	2.94E-02	1.92E-02	8.09E-03	6.45E-02	5.19E-03	3.11E-02	5.50E-02	5.98E-02
C02	M	3.37E+00(20)	**2.99E+00**	4.23E+00(15)*	4.24E+00(18)*	4.32E+00(19)*	4.53E+00(5)*	5.36E+00(1)	4.78E+00(9)*
	SD	9.25E-01	1.09E+00	6.09E-01	5.43E-01	9.07E-01	7.56E-01	0.00E+00	4.54E-01
C03	M	1.09E+13	**4.49E+12**	4.26E+14*	1.13E+13	1.16E+14	1.88E+14*	7.10E+12	8.35E+12
	SD	1.56E+13	6.71E+12	8.64E+14	1.58E+13	3.53E+14	4.68E+14	1.36E+13	9.43E+12
C07	M	**1.58E-11**	5.48E-09*	1.19E-06*	3.61E-05*	1.26E-06*	2.06E-05*	1.61E-01*	6.53E-02*
	SD	6.90E-11	1.56E-08	2.77E-06	1.78E-04	2.77E-06	7.77E-05	7.97E-01	1.54E-01
C08	M	1.21E+01	4.51E+00*	5.37E+00*	6.94E+00*	**1.12E+00***	1.44E+01*	1.43E+02*	2.96E+01*
	SD	3.36E+01	1.82E+01	2.10E+01	2.70E+01	1.08E+00	6.40E+01	2.48E+02	6.50E+01
C09	M	**1.10E+13**(1)	2.48E+13	—	3.92E+13(3)	4.28E+13(4)	1.27E+14(2)	—	—
	SD	0.00E+00	9.28E+12	—	1.26E+13	2.46E+13	1.75E+13	—	—
C11	M	**−3.91E-04**	−3.84E-04(24)*	−3.63E-04(24)*	−3.84E-04(24)*	−3.84E-04(24)*	−3.64E-04*	−3.71E-04(24)*	−3.57E-04(24)*
	SD	1.82E-06	1.11E-05	4.95E-05	1.34E-05	1.57E-05	4.13E-05	3.05E-05	9.61E-05
C12	M	−1.80E-01(23)	−1.30E-01(23)	−1.73E-01(23)	−1.60E-01(23)	−1.25E-01(24)	−1.58E-01	**−1.99E-01**(22)	−1.93E-01(23)
	SD	7.12E-02	2.80E-01	1.27E-01	1.89E-01	3.60E-01	2.07E-01	4.01E-07	2.23E-02
C13	M	−6.10E+01	−6.12E+01	−6.20E+01	**−6.21E+01**	−6.12E+01	−6.11E+01	−6.17E+01	−6.13E+01
	SD	2.15E+00	1.86E+00	1.87E+00	2.11E+00	1.64E+00	1.71E+00	1.75E+00	1.58E+00
C14	M	**3.00E+05**	1.21E+11*	2.25E+12*	2.90E+12*	2.61E+12*	1.47E+12*	3.79E+14*	1.09E+13*
	SD	1.03E+06	3.24E+11	2.36E+12	2.95E+12	2.41E+12	1.52E+12	2.27E+14	1.08E+13
C15	M	**7.30E+11**	9.17E+12*	3.35E+13*	4.53E+13*	5.11E+13*	1.36E+14*	7.41E+17*	1.68E+14*
	SD	8.31E+11	7.21E+12	2.08E+13	2.83E+13	4.65E+13	9.18E+13	1.52E+18	9.04E+13
C17	M	**5.94E+02**(21)	8.69E+02	1.72E+03(13)*	1.51E+03(17)*	1.80E+03(14)*	2.90E+03(13)*	—	1.75E+03(13)*
	SD	3.16E+02	4.91E+02	5.57E+02	4.98E+02	5.77E+02	8.69E+02	—	4.24E+02
C18	M	**5.19E+03**	1.22E+04*	2.61E+04*	2.32E+04*	2.56E+04*	3.92E+04*	1.42E+07*	3.05E+04*
	SD	1.98E+03	4.69E+03	8.19E+03	6.51E+03	7.71E+03	1.82E+04	2.02E+07	8.85E+03

Table 21. ICDE results obtained by each method in 30D problems where none of the methods found feasible solutions in all 25 runs.

		Centroid $K+1$	Resampling[2]	Evolutionary[12]	Reinitialize by Position[43]	Reflection[31]	Projection[4]	Conservatism[9]	Reinitialize All[30]
C04	M	**2.75E-02**(18)	7.98E-02(20)	5.33E-02(18)	3.76E-02**(22)**	1.45E-01(17)	1.79E-01(16)	1.02E-01(12)*	1.23E-01(13)
	SD	3.10E-02	1.99E-01	1.54E-01	3.08E-02	2.48E-01	2.85E-01	1.75E-01	2.38E-01
C05	M	**2.21E+02**(1)	4.00E+02**(24)**	—	—	—	—	3.78E+03(1)	—
	SD	0.00E+00	7.00E+01	—	—	—	—	0.00E+00	—
C06	M	3.75E+02(1)	3.94E+02**(19)**	—	—	—	**6.00E+02**(1)	—	—
	SD	0.00E+00	8.45E+01	—	—	—	0.00E+00	—	—
C10	M	**3.24E+12(3)**	—	4.47E+13(2)	—	2.44E+13(1)	8.52E+13(1)	4.88E+17(2)	—
	SD	1.56E+12	—	2.26E+13	—	0.00E+00	0.00E+00	1.27E+17	—
C16	M	1.07E+00(6)	1.07E+00**(23)**	1.18E+00(2)	—	1.18E+00(2)	1.10E+00(2)	**1.06E+00**(6)	1.12E+00(1)
	SD	2.37E-02	4.00E-02	3.52E-02	—	9.00E-02	7.20E-02	3.12E-02	0.00E+00



(a) Number of functions in which feasible solutions were found by each method in 10D and 30D.



(b) Number of runs/trials in which feasible solutions were found by each method in 10D and 30D.

Fig. 7. Graphical results of Experiment 6.

in 3 functions, while Centroid and Reinitialize by position found the highest number of feasible runs in 1 function. As for the best mean values, Centroid found the best results in 3 functions while Projection and Conservatism found the best results in 1 function. In summary, for the functions reported in Table 21, Centroid found the best results closest to the global optimum, while Resampling found the highest number of feasible runs.

Figure 7(a) shows that Centroid, Resampling, Reflection and Reinitialize by position found feasible values in 10D in all 18 functions. In 30D, Centroid found feasible values in all 18 functions, followed by Projection and Resampling, which found feasible values in 17 functions.

Figure 7(b) suggests that Resampling is the method that found the highest number of runs in both, 10D and 30D functions, followed by Centroid.

Conclusions of Experiment 6.
As result of Experiment 6, it is concluded that by employing the state-of-the-art DE-based algorithm ICDE, the BCHMs with the best performance were Centroid and Resampling.

Resampling showed good ability to reach the feasible region in 10D and 30D problems, getting the highest number of feasible runs, followed by Centroid.

Based on the statistical tests, it was observed that Centroid showed good capacity to generate the best solutions in 10D and 30D problems, followed by Resampling.

7. Conclusions and Future Work

An improved and generalized BCHM called *Centroid* $K + 1$, for DE in constrained numerical search spaces was proposed in this paper. This method consists in relocating the mutant vectors, which leaves the search space, in the centroid of an area that is formed by $K + 1$ vectors. One vector taken from the population, which bias the corrected vector position to a good fitness area, and K random vectors, which aim to guide the corrected vector position to new areas in the search space.

In the first experiment of this work, the performance of nine *Centroid* $K + 1$ variants was compared, including the *Centroid* 2+1 variant corresponding to the original version of Centroid. The results showed that there are very stable variants, like *Centroid* 1+1, which allowed to obtain a better performance than the original version of Centroid.

The *Centroid* 1+1 variant was analyzed in-depth through a comprehensive set of experiments, including a state-of-the-art DE algorithm for CNOPs.

The results of the experiments performed in this work indicate that the percentage of mutant vectors repaired in DE for constrained optimization may be too high, depending on the nature of the problem, the dimensionality and the parameters used. Therefore, it becomes important to choose a proper BCHM, as this can make a significant difference in the performance of the DE algorithm.

In most of the experiments carried out in this research, it has been shown that *Centroid* $K + 1$ is one of the best methods for handling boundary constraints in CNOPs.

The *Centroid* $K + 1$ method showed to be robust on crossover ratio variations with different DE variants, even one state-of-the-art DE-based algorithm. In the same way, it was found that *Centroid* $K + 1$ is the method that performs fewer repairs, which is an indication that the method is able to keep the population within the search space without compromising the performance of the algorithm.

It has been also found that *Centroid* $K + 1$ was capable to generate the highest percentage of mutant vectors, which after being repaired, were better than their corresponding target vectors.

Additionally, methods like Resampling showed good performance. Nevertheless, Resampling presented a high computational cost, and, in the worst case, it caused the algorithm to fall into an infinite cycle.

Because DE can be significantly influenced by the selected BCHM, part of the future work is mainly focused on four paths research:

(1) Based on granular computing paradigms[6,42] applied to evolutionary computing techniques,[1,23,26,35] a DE granular system can be development, where

unbounded solution vectors are presented, these can be repaired by a different BCHM.

(2) Rule-based granular[19,22,29] EAs frameworks can be proposed for suitable selection of BCHM to address repaired solutions toward feasible regions.

(3) Multi-levels granular computing system[8,18,34] where different EAs (defined as granule) and BCHM may work together to solve CNOPs.

(4) Finally, the usage of BCHM as a mean to improve the efficiency of DE, as well as carrying out tests with repair methods in other nature-inspired algorithms as PSO, employing modern topologies like the Dynamic Small World Network,[21] and other type of test problems (e.g. engineering design problems) or real-life applications.[40]

Acknowledgments

The first and second authors acknowledge support from the Mexican National Council for Science and Technology (CONACyT) through a scholarship to pursue graduate studies at the University of Veracruz. The third author acknowledges support from CONACyT through project No. 220522.

References

1. M. Antonelli, P. Ducange, B. Lazzerini and F. Marcelloni, Multi-objective evolutionary design of granular rule-based classifiers, *Granular Comput.* **1**(1) (2016) 37–58.
2. J. Arabas, A. Szczepankiewicz and T. Wroniak, Experimental comparison of methods to handle boundary constraints in differential evolution, in *Int. Conf. Parallel Problem Solving from Nature* (Springer, 2010), pp. 411–420.
3. T. Bäck, D. B. Fogel and Z. Michalewicz, *Handbook of Evolutionary Computation* (IOP Publishing, 1997).
4. J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Trans. Evolutionary Computation* **10**(6) (2006) 646–657.
5. W. Chu, X. Gao and S. Sorooshian, Handling boundary constraints for particle swarm optimization in high-dimensional search space, *Inf. Sci.* **181**(20) (2011) 4569–4581.
6. D. Ciucci, Orthopairs and granular computing, *Granular Comput.* **1**(3) (2016) 159–170.
7. K. Deb, An efficient constraint handling method for genetic algorithms, *Comput. Methods Appl. Mech. Eng.* **186**(2) (2000) 311–338.
8. D. Dubois and H. Prade, Bridging gaps between several forms of granular computing, *Granular Comput.* **1**(2) (2016) 115–126.
9. V. Feoktistov, *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)* (Springer-Verlag, New York, 2006).
10. D. Fowler, The binomial coefficient function, *American Math. Monthly* **103**(1) (1996) 1–17.
11. A. H. Gandomi, A. R. Kashani and M. Mousavi, Boundary constraint handling affection on slope stability analysis, in *Engineering and Applied Sciences Optimization* (Springer, 2015), pp. 341–358.
12. A. H. Gandomi and X.-S. Yang, Evolutionary boundary constraint handling scheme, *Neural Comput. Appl.* **21**(6) (2012) 1449–1462.

13. S. Helwig, J. Branke and S. Mostaghim, Experimental analysis of bound handling techniques in particle swarm optimization, *IEEE Trans. Evol. Comput.* **17**(2) (2013) 259–271.
14. T. Huang and A. Mohan, A hybrid boundary condition for robust particle swarm optimization, *IEEE Antennas Wirel. Propag. Lett.* **4** (2005) 112–117.
15. G. Jia, Y. Wang, Z. Cai and Y. Jin, An improved $(\mu + \lambda)$-constrained differential evolution for constrained optimization, *Inf. Sci.* **222** (2013) 302–322.
16. E. Juárez-Castillo, N. Pérez-Castro and E. Mezura-Montes, A novel boundary constraint-handling technique for constrained numerical optimization problems, in *2015 IEEE Congress on Evolutionary Computation (CEC)* (IEEE, 2015), pp. 2034–2041.
17. S. Kukkonen and J. Lampinen, Gde3: The third evolution step of generalized differential evolution, in *2005 IEEE Congress on Evolutionary Computation*, Vol. 1 (IEEE, 2005), pp. 443–450.
18. P. Lingras, F. Haider and M. Triff, Granular meta-clustering based on hierarchical, network, and temporal connections, *Granular Comput.* **1**(1) (2016) 71–92.
19. H. Liu, A. Gegov and M. Cocea, Rule-based systems: A granular computing perspective, *Granular Comput.* **1**(4) (2016) 259–274.
20. H.-L. Liu, C. Peng, F. Gu and J. Wen, A constrained multi-objective evolutionary algorithm based on boundary search and archive, *Int. J. Pattern Recognit. Artif. Intell.* **30**(1) (2016) 1659002.
21. Q. Liu, B. J. van Wyk, S. Du and Y. Sun, Dynamic small world network topology for particle swarm optimization, *Int. J. Pattern Recognit. Artif. Intell.* **30**(09) (2016) 1660009.
22. L. Livi and A. Sadeghian, Granular computing, computational intelligence, and the analysis of non-geometric input spaces, *Granular Comput.* **1**(1) (2016) 13–20.
23. V. Loia, G. D'Aniello, A. Gaeta and F. Orciuoli, Enforcing situation awareness with granular computing: A systematic overview and new perspectives, *Granular Comput.* **1**(2) (2016) 127–143.
24. R. Mallipeddi and P. N. Suganthan, Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization, Nanyang Technological University, Singapore (2010).
25. E. Mezura-Montes and C. A. C. Coello, Constraint-handling in nature-inspired numerical optimization: Past, present and future, *Swarm Evol. Comput.* **1**(4) (2011) 173–194.
26. F. Min and J. Xu, Semi-greedy heuristics for feature selection with test cost constraints, *Granular Comput.* **1**(3) (2016) 199–211.
27. N. Padhye, K. Deb and P. Mittal, Boundary handling approaches in particle swarm optimization, in *Proc. Seventh Int. Conf. Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)* (Springer, 2013), pp. 287–298.
28. S. Pei, A. Ouyang and L. Tong, A hybrid algorithm based on bat-inspired algorithm and differential evolution for constrained optimization problems, *Int. J. Pattern Recognit. Artif. Intell.* **29**(04) (2015) 1559007.
29. G. Peters and R. Weber, Dcc: A framework for dynamic granular clustering, *Granular Comput.* **1**(1) (2016) 1–11.
30. K. Price, R. M. Storn and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series (Springer, Berlin Heidelberg, 2005).
31. J. Ronkkonen, S. Kukkonen and K. V. Price, Real-parameter optimization with differential evolution, in *Proc. IEEE CEC*, Vol. 1 (2005), pp. 506–513.
32. T. P. Runarsson and X. Yao, Stochastic ranking for constrained evolutionary optimization, *IEEE Trans. Evol. Comput.* **4**(3) (2000) 284–294.

33. Y. Shi, S. Cheng and Q. Qin, Experimental study on boundary constraints handling in particle swarm optimization: From population diversity perspective, *Int. J. Swarm Intell. Res.* **2**(3) (2011) 43–69.
34. A. Skowron, A. Jankowski and S. Dutta, Interactive granular computing, *Granular Comput.* **1**(2) (2016) 95–113.
35. M. Song and Y. Wang, A study of granular computing in the agenda of growth of artificial neural networks, *Granular Comput.* **1**(4) (2016) 247–257.
36. R. Storn and K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim.* **11**(4) (1997) 341–359.
37. I. N. Trivedi, A. H. Gandomi, P. Jangir and N. Jangir, Study of different boundary constraint handling schemes in interior search algorithm, *International Conference on Artificial Intelligence and Evolutionary Computations in Engineering Systems, ICAIE-CES-2016* (2015).
38. P. Wang, C. Zhang, B. Zhang, T. Liu and J. Wu, A dimensional diversity based hybrid multiobjective evolutionary algorithm for optimization problem, *Int. J. Pattern Recognit. Artif. Intell.* **30**(7) (2016) 1659020.
39. S. Wessing, Repair methods for box constraints revisited, in *European Conf. Applications of Evolutionary Computation* (Springer, 2013), pp. 469–478.
40. G. Wilke and E. Portmann, Granular computing as a basis of human–data interaction: A cognitive cities use case, *Granular Comput.* **1**(3) (2016) 181–197.
41. S. Xu and Y. Rahmat-Samii, Boundary conditions in particle swarm optimization revisited, *IEEE Trans. Antennas Propag.* **55**(3) (2007) 760–765.
42. Y. Yao, A triarchic theory of granular computing, *Granular Comput.* **1**(2) (2016) 145–157.
43. W.-J. Zhang, X.-F. Xie and D.-C. Bi, Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space, in *Congress on Evolutionary Computation, 2004, (CEC2004)*, Vol. 2 (IEEE, 2004), pp. 2307–2311.

Efren Juarez-Castillo was born in Tlanchinol, Hidalgo, Mexico, in 1975. He received his diploma degree in Computer Systems Engineering from the Pachuca Institute of Technology in 2000 and his M.Sc. degree in Artificial Intelligence from the University of Veracruz in 2012. He is currently pursuing his Ph.D. in Artificial Intelligence at the Artificial Intelligence Research Center at the University of Veracruz. His interests are in the areas of artificial intelligence and bio-inspired computing.

Nancy Perez-Castro was born in Xalapa, Veracruz, Mexico, in 1987. She received her diploma degree in Computer Systems from the University of Veracruz in 2009 and her M.Sc. degree in Applied Computing from the National Laboratory of Advanced Computer Science (LANIA) in 2013. She is currently pursuing her Ph.D. in Artificial Intelligence at the Artificial Intelligence Research Center at the University of Veracruz. Her research interests are temporal data mining and the design, study and application of nature-inspired algorithms for suitable model selection in data.

Efrén Mezura-Montes was born in Xalapa, Mexico, in 1973. He received his diploma degree in Computer Systems Engineering from the University of the Americas, Puebla, in 1997, his M.Sc. degree in Artificial Intelligence from the University of Veracruz in 2001, and his Ph.D. in Computer Science from the Center for Research and Advanced Studies of the National Polytechnic Institute (CINVESTAV-IPN) in 2004. He has published more than 110 papers in peer-reviewed journals and conferences. His research interests are in the design, study and application of nature-inspired metaheuristic algorithms to solve complex optimization problems. Dr. Mezura-Montes is currently the head of the Ph.D. graduate program in Artificial Intelligence of the Artificial Intelligence Research Center at the University of Veracruz. He is also a member of the IEEE Computational Intelligence Society Evolutionary Computation Technical Committee and the IEEE Systems, Man and Cybernetics Society Soft Computing Technical Committee. Dr. Mezura Montes is also a member of the Mexican National Researchers System Level 2 and a regular member of the Mexican Computing Academy.