Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100220

Improved Modified Bacterial Foraging Optimization Algorithm to Solve Constrained Numerical Optimization Problems

Betania Hernández-Ocaña[1,*], *Ma. Del Pilar Pozos-Parra*[1] *and Efrén Mezura-Montes*[2]

[1] Juárez Autonomous University of Tabasco, Cunduacán, Tabasco, México
[2] Artificial Intelligence Research Center, University of Veracruz, Xalapa, Veracruz, México

Received: 17 Aug. 2015, Revised: 4 Nov. 2015, Accepted: 5 Nov. 2015
Published online: 1 Mar. 2016

Abstract: This paper presents an improved version of the modified bacterial foraging optimization algorithm to solve constrained numerical optimization problems. Four mechanisms are added: (1) two swim operators, one to favor the exploration and another one to focus on the exploitation of the search space, where a dynamic mechanism is considered to deal with the stepsize value, (2) a skew mechanism for a more suitable initial swarm where bacteria are divided in three groups, two of them close to the boundaries of the search space and one distributed in all the search space, (3) a local search operator and (4) a decrease in the usage of the reproduction step to deal with premature convergence. 60 well-known test problems from two benchmarks are solved along three experiments. The first experiment aims to provide preliminary evidence on the suitable behavior of the new mechanism added. The second experiment provides an in-depth comparison of the new version against its previous one based on final results and four performance measures. The third experiment compares the performance of the proposed algorithm against five state-of-the-art nature-inspired algorithms designed to deal with constrained continuous search spaces. The results show that the proposed algorithm clearly provides a better performance against its predecessor by increasing its ability to reach the feasible region and generating better solutions, while obtaining a competitive performance against those compared state-of-the-art algorithms.

Keywords: Nature-inspired optimization; evolutionary algorithms; swarm intelligence; constrained optimization

1 Introduction

Nowadays, meta-heuristic algorithms are a popular choice for solving complex optimization problems [41]. Evolutionary algorithms (EAs) are based on emulating the process of natural evolution and survival of the fittest. On the other hand, in the mid 1990's a new group of algorithms emerged, those based on social and cooperative behaviors of simple organisms such as insects, birds, fish and bacteria, among others, to which are called Swarm Intelligence Algorithms (SIAs) [10]. Originally designed to deal with unconstrained search spaces, SIAs and EAs can be extended with constraint-handling techniques with the purpose of solving constrained numerical optimization problems (CNOPs) [25]. Without loss of generality, a CNOP can be defined as to:

$$\begin{gathered}\text{Minimize } f(\mathbf{x}) \\ \text{subject to:} \\ g_i(\mathbf{x}) \leq 0,\ i = 1,...,m \\ h_j(\mathbf{x}) = 0,\ j = 1,...,p\end{gathered}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n] \in R^n$, is the solution vector and each decision variable x_i, $i = 1,...,n$ is bounded by lower and upper limits $L_i \leq x_i \leq U_i$, which define the search space S; m is the number of inequality constraints and p is the number of equality constraints (in both cases, the constraints can be linear or nonlinear). If F denotes the feasible region, then it must be clear that $F \subseteq S$. As it is commonly found in the specialized literature of nature-inspired algorithms to solve CNOPs [5,25,28] an equality constraint is transformed into an inequality constraint by using a small tolerance ε as follows: $|h_j(\mathbf{x})| - \varepsilon \leq 0, j = 1, \ldots, p$.

* Corresponding author e-mail: betania.hernandez@ujat.mx

Some of the most popular SIAs in the specialized literature to solve CNOPs is the Particle Swarm Optimization (PSO) algorithm [18,29], which simulates the cooperative behavior of bird flocks when looking for food or refuge. Such cooperation is represented in each solution (called particle) with a velocity vector whose values combine the cognitive information of each solution, i.e., its best position reached so far, and the social information, i.e., the position of the best particle in the swarm. The Artificial Bee Colony (ABC) [15,26] is another popular SIA, based on the behavior of honey bees. Three types of bees, which are variation operators, are considered: employed, onlooker and scouts. Food sources are the solutions of the optimization problem and they are improved by bees. The cooperation is reached by the information employed bees share with onlooker bees by means of waggle dances. In this way, the most rich food sources are improved with a higher probability with respect to low quality food sources.

Passino, inspired in previous studies [4], proposed Bacterial Foraging Optimization Algorithm (BFOA), a SIA to originally solve unconstrained numerical optimization problems [36]. Moreover, recent versions of BFOA have been adapted to solve CNOPs [11]. BFOA's main difference with respect to other representative SIAs is the way the cooperative behavior is computed. In BFOA such process is based on attractants and repellants among bacteria emulated by a penalty-like mechanism which increases the fitness of bacteria located in promising regions of the search space, while decreasing the fitness of bacteria located in poor zones.

This algorithm emulates the behavior of bacterium E.Coli in the search of nutrients in its environment. Each bacterium tries to maximize its obtained energy per each unit of time spent on the foraging process while avoiding noxious substances. In fact, bacteria can communicate among themselves. A swarm of bacteria presents the following process [36]:

1. Bacteria are distributed at random in the map of nutrients.
2. Bacteria locate high-nutrient regions in the map by chemotaxis movements (tumble and swim).
3. Bacteria in regions with noxious substances or low-nutrient regions will die and disperse, respectively.
4. Bacteria in high-nutrient regions will reproduce by splitting. They will also attempt to attract other bacteria by generating chemical attractors.
5. Bacteria then disperse to seek new nutrient regions in the map.

Those processes were summarized in four steps in BFOA: (1) chemotaxis, (2) swarming, (3) reproduction and (4) elimination-dispersal. Moreover, this algorithm has been adapted to solve CNOPs in [30], which in turn was extended in order to solve multi-objective CNOPs [31, 32], given the Modified Bacterial Foraging Optimization Algorithm (MBFOA). MBFOA inherits the four main processes used in BFOA. However, they were adapted with the aim to eliminate one loop and some parameters.

BFOA has been also combined with other search algorithms such as the Hooke-Jeeves Pattern Search Method as local search operator [31]. Moreover, BFOA was combined with Armijo rules for local search in the proposal called Spiral Bacterial Foraging Optimization (SBFO) which is a multi-agent, gradient-based algorithm that minimizes both the main objective function (local cost) and the distance between each agent and a temporary central point (global cost). Random parameter values were adopted in this proposal to cope with premature convergence, which is a feature of swarm-based optimization methods [16]. Modifications to BFOA have been reported as well, such as the elimination of the reproduction process [1], and the combination with other meta-heuristic algorithms like Genetic Algorithms (GA) [19], PSO [3,20], and Differential Evolution (DE) [2].

A review of BFOA to solve CNOPs was presented in [11], where the penalty function was detected as the most used constraint-handling technique. Moreover, such report concluded that BFOA is particularly sensitive to the step size parameter, which is used in the chemotaxis process with the tumble-swim movement. There are different ways to control the step size: (1) keeping it static during the search process (as in the original BFOA) [30, 13,46,12], (2) using random values [38,45,12], (3) using a dynamic variation [35,34,12], or (4) adopting an adaptive mechanism [40,31,12]. However, such proposals were stated mainly for specific optimization problems. On the other hand, there is a recent study of the step size in MBFOA [12] where different approaches were compared, being the dynamic control mechanism slightly superior with respect to static, random, and adaptive versions.

Motivated by the above mentioned, the aim of this work is to propose an improved bacterial foraging optimizer to solve CNOPs, which takes MBFOA as the search algorithm and two types of swims are proposed to deal with the sensitivity to the stepsize value. Furthermore, the initial swarm of bacteria is generated in three groups depending on the boundaries of each decision variable so as to take advantage of the random search directions used in the chemotaxis cycle. Finally, the reproduction step is limited to favor diversity and a local search operator [37] is added in two times of the search process. The proposed algorithm is tested on two sets of well-known problems with different features [21], [23] and its behavior is analyzed with different performance measures taken from the specialized literature [26]. Furthermore, the results obtained are compared against state-of-the-art nature-inspired algorithms.

Begin
Create a random initial swarm of bacteria $\theta^i(j,k,l)\ \forall i,\ i=1,\ldots,S_b$
Evaluate $f(\theta^i(j,k,l))\ \forall i,\ i=1,\ldots,S_b$
For l=1 to N_{ed} **Do**
For k=1 to N_{re} **Do**
For j=1 to N_c **Do**
For i=1 to S_b **Do**
Update $f(\theta^i(j,k,l))$ to emulate the **swarming process**
Perform the **chemotaxis process** (tumble-swim) with Eq. 1 and Eq.2 for bacteria $\theta^i(j,k,l)$ controlled by N_s
End For
End For
Perform the **reproduction process** by sorting all bacteria in the swarm based on $f(\theta^i(j+1,k,l))$, deleting the S_r worst bacteria and duplicating the remaining $S_b - S_r$
End For
Perform the **elimination-dispersal** process by eliminating each bacteria $\theta^i(j,k,l)\ \forall i,\ i=1,\ldots,S_b$ with probability $0 \leq P_{ed} \leq 1$
End For
End

Fig. 1: BFOA pseudocode. Input parameters are number of bacteria S_b, chemotaxis loop limit N_c, swim loop limit N_s, reproduction loop limit N_{re}, number of bacteria for reproduction S_r (usually $S_r = S_b/2$), elimination-dispersal loop limit N_{ed}, stepsize C_i and probability of elimination dispersal p_{ed}.

The document is organized as follows: Section 2 describes the original BFOA and MBFOA. Section 3 introduces the proposed algorithm. Section 4 presents the set of test problems to resolve, the performance measures to evaluate the results, a behavior analysis of the proposal with the new mechanisms, the results obtained in the test problems, and the analysis of them compared against state-of-the-art algorithms. Finally, Section 5 presents the general conclusions of this work.

2 Bacterial Foraging Optimization Algorithm

2.1 BFOA

Recalling from Section 1, BFOA is based on four processes: (1) chemotaxis, (2) swarming, (3) reproduction and (4) elimination-dispersal [36] and was designed to solve unconstrained numerical single-objective optimization problems. The algorithm is detailed in Figure 1.

A bacterium i represents a potential solution to the optimization problem (a n-dimensional real-value vector named as $\mathbf{x}$ in Section 1), and it is defined as $\theta^i(j,k,l)$, where j is its chemotaxis loop value, k is its reproduction loop value, and l is its elimination-dispersal loop value.

2.1.1 Chemotaxis

In this process, each bacterium in the current swarm performs one movement by tumbling (finding a search direction) and swimming (moving along such direction). The tumble, as proposed by Passino [36], consists of a search direction $\phi(i)$ generated at random with uniform distribution as presented in Equation 1:

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{1}$$

where $\Delta(i)$ is a uniformly distributed random vector of size n with elements within the following interval: $[-1,1]$. Once the search direction was generated (i.e. the tumble was carried out) each bacterium i modifies its position by a swimming step as indicated in Equation 2.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(i) \tag{2}$$

where $\theta^i(j+1,k,l)$ is the new position of bacterium i, which is based on its previous position $\theta^i(j,k,l)$ and its search direction $\phi(i)$ scaled by the stepsize $C(i)$. The swim will be repeated N_s times if and only if the new position is better than the previous one, i.e., (assuming minimization) $f(\theta^i(j+1,k,l)) < f(\theta^i(j,k,l))$. Otherwise, a new tumble is computed. The chemotaxis loop stops when the chemotaxis loop limit N_c is reached for all bacteria in the swarm.

2.1.2 Swarming

The swarming process is based on a modification to the fitness landscape by making more attractive the boundaries of the location of a given bacterium with the aim to attract more bacteria to such region. This process requires the definition of a set of parameter values by the user [36].

2.1.3 Reproduction

The reproduction process consists on sorting all bacteria in the swarm $\theta^i(j,k,l), \forall i,\ i=1,\ldots,S_b$ based on their objective function value $f(\theta^i(j,k,l))$ and eliminating half of them with the worst values. The remaining half will be duplicated so as to maintain a fixed swarm size.

2.1.4 Elimination-dispersal

The elimination-dispersal process consists on eliminating each bacteria $\theta^i(j,k,l), \forall i,\ i=1,\ldots,S_b$ with a probability $0 \leq P_{ed} \leq 1$.

2.2 Modified BFOA (MBFOA)

MBFOA was proposed as one of the first attempts of adapting BFOA for solving CNOPs. In this way, the original algorithm was simplified and some parameters were eliminated so as to make it easier to use particularly

in engineering design problems [30].

The modifications made in MBFOA with respect to BFOA, detailed in [30], are the following:

1. **Algorithm simplification**: A generational loop (G) is considered, where three inner processes are carried out: chemotaxis, reproduction and elimination-dispersal. Based on the fact that in MBFOA there are no reproduction and elimination-dispersal loops, because they are replaced by the generational loop, a bacterium i in generation G and in chemotaxis step j is defined as $\theta^i(j,G)$. With this change, the nomenclature of the swim movement is modified as indicated in Equation 3.

$$\theta^i(j+1,G) = \theta^i(j,G) + C(i)\phi(i) \quad (3)$$

where $\theta^i(j+1,G)$ is the new position of bacterium i (new solution), $\theta^i(j,G)$ is the current position of bacterium i and $C(i)$ is the stepsize, which is computed by considering the limits per each design variable k [30] as indicated in Equation 4.

$$C(i)_k = R * (\frac{\Delta \mathbf{x}_k}{\sqrt{n}}), k = 1, ..., n \quad (4)$$

where $\Delta \mathbf{x}_k$ is the difference between upper and lower limits for design parameter x_k: $U_k - L_k$, n is the number of design variables and R is a user-defined percentage of the value used by the bacteria as stepsize.

The criteria to sort the swarm of bacteria for the reproduction process are the three rules of the constraint-handling technique (detailed in the next item of this list), instead of using only the objective function value as in BFOA.

Finally, instead of eliminating each bacterium based on a probability, only the worst bacterium $\theta^w(j,G)$ based on the feasibility rules is eliminated and a new randomly generated bacterium with uniform distribution takes its place.

2. **Constraint-Handling technique**: MBFOA adopts a parameter free constraint-handling technique to bias the search to the feasible region of the search space by using three feasibility rules proposed by Deb [7] as selection criteria when bacteria are compared. The rules are the following: (1) between two feasible bacteria, the one with the best objective function value is chosen, (2) between one feasible bacterium and another infeasible one, the feasible is chosen, and (3) between two infeasible bacteria, the one with the lowest sum of constraint violation is chosen.

The sum of constraint violation is computed as: $\sum_{i=1}^{m} max(0, g_i(\mathbf{x}))$, where m is the number of constraints of the problem. Each equality constraint is converted into an inequality constraint: $\| h_i(x) \| - \varepsilon \leq 0$, where ε is the tolerance allowed (a very small value, in the literature this value is usually $1E-04$).

The violation of each constraint can be normalized to eliminate significant differences of values among constraints when computing the sum of constraint violation. However, in the experiments of this paper such process was not required.

3. **Swarming operator**: Instead of the swarming process which requires four user-defined parameters, an attractor movement was included to MBFOA so as to let each bacterium in the swarm to follow the bacterium located in the most promising region of the search space in the current swarm, i.e., the best current solution. The attractor movement is defined in Equation 5.

$$\theta^i(j+1,G) = \theta^i(j,G) + \beta(\theta^B(G) - \theta^i(j,G)) \quad (5)$$

where $\theta^i(j+1,G)$ is the new position of bacterium i, $\theta^i(j,G)$ is the current position of bacterium i, $\theta^B(G)$ is the current position of the best bacterium in the swarm so far at generation G, and β defines the closeness of the new position of bacterium i with respect to the position of the best bacterium $\theta^B(G)$. The attractor movement applies twice in a chemotaxis loop, while in the remaining steps the tumble-swim movement is carried out. The aim is to promote a balance between exploration and exploitation in the search. Finally, if the value of a design variable generated by the tumble-swim or attractor operators is outside the valid limits defined by the optimization problem, the following modification, taken from [39] is made so as to get the value within the valid range as showed in Equation 6.

$$x_i = \begin{cases} 2 * \mathrm{L}_i - x_i & \text{if } x_i < \mathrm{L}_i \\ 2 * \mathrm{U}_i - x_i & \text{if } x_i > \mathrm{U}_i \\ x_i & \text{otherwise} \end{cases} \quad (6)$$

where x_i is the design variable value i generated by any bacterium movement, and L_i and U_i are the lower and upper limits for design variable i, respectively.

The complete pseudocode of MBFOA is detailed in Figure 2.

3 Improved MBFOA (IMBFOA)

IMBFOA is an improved version where different modifications were made in order to obtain better results in a wider set of CNOPs: (1) two swim movements within the chemotaxis process, one for exploration and another one for exploitation, (2) a skew mechanism for the initial swarm of bacteria, (3) a local search operator, and (4) a reduction on the usage of the reproduction process.

Begin
Create a random initial swarm of bacteria $\theta^i(j,0)\ \forall i,\ i=1,\ldots,S_b$
Evaluate $f(\theta^i(j,0))\ \forall i,\ i=1,\ldots,S_b$
For G=1 to *GMAX* **Do**
For i=1 to S_b **Do**
For j=1 to N_c **Do**
Perform the **chemotaxis process** (tumble-swim) with Eq. 1 and 3 and **attractor operator** with Eq. 5 for bacteria $\theta^i(j,G)$ by considering the constraint-handling technique
End For
End For
Perform the **reproduction process** by sorting all bacteria in the swarm based on the constraint-handling technique, deleting the S_r worst bacteria and duplicating the remaining $S_b - S_r$
Perform the **elimination-dispersal process** by eliminating the worst bacterium $\theta^w(j,G)$ in the current swarm by considering the constraint-handling technique
End For
End

Fig. 2: MBFOA pseudocode. Input parameters are number of bacteria S_b, chemotaxis loop limit N_c, number of bacteria for reproduction S_r (usually $S_r = S_b/2$), scaling factor β, percentage of initial stepsize R and number of generations *GMAX*.

3.1 Two updated swim operators

In the chemotaxis process, the range for the tumble is modified. Originally, the range boundaries are fixed values [-1,1] (see Equation 1). To favor finer movements (as suggested in the stepsize study in [12]) a narrow range $[\upsilon,\ \tau]$ is proposed; υ and τ are user-defined parameters, where $-1 \leq \upsilon < 0$, $0 < \tau \leq 1$ and $\upsilon < \tau$. Values between -0.25 and -0.05 are recommended for υ, as well as values between 0.15 and 1.0 for τ.

Considering the above mentioned narrow range, two swim operators are proposed to improve the search capabilities of bacteria. This first swim, focused on exploration, is computed as indicated in Equation 7:

$$\theta^i(j+1,G) = \theta^i(j,G) + \phi(i) \tag{7}$$

This second swim, focused on exploitation is computed as indicated in Equation 8:

$$\theta^i(j+1,G) = \theta^i(j,G) + C(i,G)\phi(i) \tag{8}$$

where $C(i,G)$ is the dynamic step size vector, inspired by a previous study presented in [12], and computed using Equation 1. Each value k of the vector $C(i,G)$ decreases dynamically at each generation of the algorithm using Equation 9:

$$C(i,G+1)_k = C(i,G)_k \frac{G}{GMAX}, k = 1,...,n \tag{9}$$

where $C(i,G+1)_k$ is the new step size value for variable k and *GMAX* is the maximum number of generations of the algorithm. The initial $C(i,G)$ is computed as indicated in Equation 4 but the R parameter is no longer used. υ and τ take random values with uniform distribution between [-1,0) and (0,1], respectively.

It is important to remark that the first swim (Equation 7) performs larger movements as the $\phi(i)$ vector is not scaled. On the other hand, the second swim (Equation 8) contributes early in the exploration task due to its dynamic behavior on the stepsize. However, such second swim will promote exploitation by small movements later in the search process.

It is important to notice as well that a given bacterium will not necessarily interleave exploration and exploitation swims, because if the new position of a given swim, $\theta^i(j+1,G)$ has a better fitness (based on the feasibility rules) than the original position $\theta^i(j,G)$, another similar swim in the same direction will be carried out . Otherwise, a new tumble for the other swim will be computed. The process stops after N_c attempts.

3.2 Skew mechanism for the initial swarm

Motivated by (1) the increasing interest on the initialization techniques for nature-inspired algorithms [17], and (2) the effect of the two swim operators already introduced, a skew mechanism to create the swarm of bacteria is added to IMBFOA.

The finer movements promoted by the combination of the two swims proposed in this work, besides the swarming operator already present in MBFOA, may lead to a loss of diversity early in the search. Moreover, such diversity lack in a constrained space may cause a difficulty to reach the feasible region of the search space. Therefore, a different way to generate the initial swarm of bacteria is proposed in this work, where just some bacteria are uniformly distributed in the search space, and the positions of the others are skewed to the boundaries of the search space as detailed below.

The initial swarm of bacteria S_b is generated by three groups. In the first group there are randomly located bacteria skewed to the lower limit of the decision variables. In the second group there are randomly located bacteria skewed to the upper limit of the decision variables. Finally, a group of randomly located bacteria without skew, as in MBFOA, is considered. The three groups use random numbers with uniform distribution. The formulas to set the limits for the first and second group per variable are presented in Equations 10 and 11.

$$[L_i, L_i + ((U_i - Li)/ss)] \tag{10}$$

$$[U_i - ((U_i - Li)/ss), U_i] \tag{11}$$

where *ss* is the skew size, whose large values decrease the skew effect and small values increase the skew effect. Figure 3 shows an example with a swarm of thirty bacteria, in a two-dimensional search space with $-5 \leq x_1 \leq 5$ and $-3 \leq x_2 \leq 10$, respectively. Considering $ss = 8$, the first group for x_1 is generated between $[-5, -5 + ((5-(-5)/8)] = [-5, -3.75]$ and the second group for x_1 is generated between $[5 - ((5-(-5)/8), 5] = [3.75, 5]$. Following the same formula, the range of the first group for x_2 is $[-3, -1.375]$ and the second group for x_2 is $[8.375, 10]$. The third group is generated by using the original limits for both variables. The aim of this skew in the initial swarm is to keep the algorithm from converging prematurely (it was observed in MBFOA motivated by its swarming process and the fixed stepsize). Combined with the two swim operators and the dynamic stepsize control, a better exploration of the search space in the initial phase of the search to promote better final results is expected.

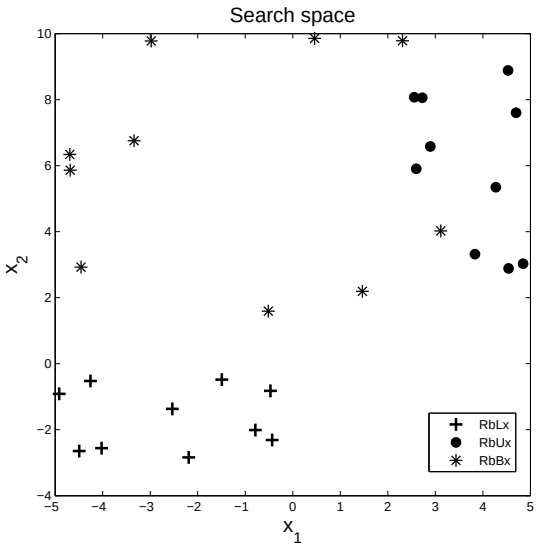


Fig. 3: Initial swarm of bacteria: RbLx are the random bacteria skewed to the lower limits of decision variables, RbUx are the random bacteria skewed to the upper limits and RbBx are random bacteria within the boundary of decision variables.

3.3 Local Search Operator

To help IMBFOA to generate better results and based on the improved behavior observed by memetic algorithms [33], Sequential Quadratic Programming (SQP) [37] is incorporated to IMBFOA as a local search operator. This proposal has a simple structure (see Figure 4), where SQP is used only twice during the search process, once after the first cycle of the algorithm and once at half of the search process. This local search operator is applied to the best bacterium in the swarm after the chemotaxis, swarming, reproduction, and elimination-dispersal processes. However, the user can define the frequency of usage of the local search operator by defining the parameter Local Search frequency LS_G.

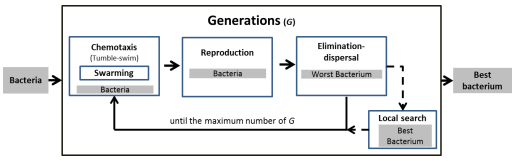


Fig. 4: IMBFOA general process

3.4 Scarce usage of the reproduction step

To reduce premature convergence due to bacteria duplication, the reproduction takes place only at certain cycles of the algorithm (defined by the *RepCycle* parameter).

The corresponding pseudocode of IMBFOA is presented in Figure 5 and in its caption the user-defined parameters are summarized.

4 Results and analysis

Three experiments were carried out to analyze the behavior of the IMBFOA to solve CNOPs. The first experiment aimed to preliminary show the behavior of the mechanisms added to IMBFOA with respect to the original MBFOA. The second experiment focused on an in-depth comparison of IMBFOA and MBFOA based on performance measures and final results. Finally, such final results obtained were compared against state-of-the-art nature-inspired algorithms to solve CNOPs. IMBFOA and the performance measures were coded in Matlab R2009b, and run on a PC with a 3.5 Core 2 Duo Processor, 4GB RAM, and Windows 7. 25 independent runs were carried out by the proposed approach.

The Wilcoxon Signed Rank Test (WSRT) [6], suggested for nature-inspired algorithms comparison in [8] (for paired samples), and the Friedman test [14] (for multiple sample comparison) were used as the statistical tests to validate the differences in the samples of runs. The scores for the WSRT were based on the best fitness values of each algorithm in each test problem, while the Friedman test used the means values. Both tests were applied with 95%-confidence.

Begin
Create an initial swarm of bacteria **by using the skew mechanism**
$\theta^i(j,0)\ \forall i,\ i=1,\ldots,S_b$
Evaluate $f(\theta^i(j,0))\ \forall i,\ i=1,\ldots,S_b$
For G=1 to *GMAX* **Do**
For i=1 to S_b **Do**
For j=1 to N_c **Do**
Perform the **chemotaxis process by interleaving both proposed** swims with Eqs 7 and Eq. 8 and the attractor operator with Eq. 5 using β for bacteria $\theta^i(j,G)$
End For
End For
If (G mod RepCycle == 0)
Perform the **reproduction process** by sorting the swarm based on the feasibility rules and deleting the S_r worst bacteria and duplicating the remaining $S_b - S_r$
End If
Perform the **elimination-dispersal process** by eliminating the worst bacterium $\theta^w(j,G)$ in the current swarm
Update the step size vector dynamically by using Eq. 9
If G mod LS_G==0
Apply SQP to the best bacterium in the swarm. If the obtained bacterium is better it replaces that best bacterium
End If
End For
End

Fig. 5: IMBFOA pseudocode. Input parameters are number of bacteria S_b, chemotaxis loop limit N_c, number of bacteria for reproduction S_r, scaling factor β, the reproduction cycle RepCycle, the number of cycles *GMAX*, the local search frequency LS_G, τ and υ for the search direction, and *ss* for the skew mechanism.

The input parameter values used by IMBFOA are shown in Table 1, they were fine-tuned by the iRace tool [22]. The only fixed parameter was *GMAX* related with the termination condition of *Max_FEs*, depending of the test problem as it will be detailed later. The parameters used by the original MBFOA, also fine-tuned by iRace, are shown in Table 1 as well. The maximum number of evaluations for the local search operator was $5,000$ FEs for the first benchmark and for the 10D test problems of the second benchmark, while a maximum number of $10,000$ FEs were allowed for the 30D test problems.

Table 1: IMBFOA and MBFOA parameter values obtained by the iRace tool [22].

	Value	
Parameter	**MBFOA**	**IMBFOA**
S_b	20	20
N_c	24	24
S_r	2	1
R	0.012	-
β	1.5	1.5
RepCycle	-	100
GMAX	value to reach Max_FEs	value to reach Max_FEs
LS_G	-	1 and (*GMAX*/2) generations
τ	-	-0.25
υ	-	0.15
ss	-	8

4.1 Test problems

24 well-known CNOPs found in [21] and 18 scalable CNOPs (10D and 30D test problems) found in [23] were used in the experiments. A summary of the features of the first 24 test problems is presented in Table 2. The main characteristics of the 18 test problems (10D and 30D) are presented in Table 3. The maximum number of evaluations (Max_FEs) allowed for each one of the first 24 test problems was 240,000. For the second set of test problems the Max_FEs was 200,000 for 10D and 600,000 for 30D. The tolerance for equality constraints was set to $\varepsilon = 1E-04$.

Table 2: Summary of the first 24 test problems. *n* is the number of variables, ρ is the estimated ratio between the feasible region and the search space, *li* is the number of linear inequality constraints, *ni* is the number of nonlinear inequality constraints, *le* is the number of linear equality constraints, *ne* is the number of nonlinear equality constraints, *a* is the number of active constraints and $f(\mathbf{x}^*)$ is the best known feasible solution

Problem	*n*	**type of function**	ρ	**li**	**ni**	**le**	**ne**	*a*	$f(\mathbf{x}^*)$
g01	13	quadratic	0.0111%	9	0	0	0	6	-15
g02	20	nonlinear	99.9971%	0	2	0	0	1	-0.803619104
g03	10	polynomial	0.0000%	0	0	0	1	1	-1.0005001
g04	5	quadratic	52.1230%	0	6	0	0	2	-30665.53867
g05	4	cubic	0.0000%	2	0	0	3	3	5126.496714
g06	2	cubic	0.0066%	0	2	0	0	2	-6961.813876
g07	10	quadratic	0.0003%	3	5	0	0	6	24.30620907
g08	2	nonlinear	0.8560%	0	2	0	0	0	-0.095825041
g09	7	polynomial	0.5121%	0	4	0	0	2	680.6300574
g10	8	linear	0.0010%	3	3	0	0	6	7049.248021
g11	2	quadratic	0.0000%	0	0	0	1	1	0.7499
g12	3	quadratic	4.7713%	0	1	0	0	0	-1
g13	5	nonlinear	0.0000%	0	0	0	3	3	0.053941514
g14	10	nonlinear	0.0000%	0	0	3	0	3	-47.76488846
g15	3	quadratic	0.0000%	0	0	1	1	2	961.7150223
g16	5	nonlinear	0.0204%	4	34	0	0	4	-1.905155259
g17	6	nonlinear	0.0000%	0	0	0	4	4	8853.539675
g18	9	quadratic	0.0000%	0	13	0	0	6	-0.866025404
g19	15	nonlinear	33.4761%	0	5	0	0	0	32.65559295
g20	24	linear	0.0000%	0	6	2	12	16	0.2049794
g21	7	linear	0.0000%	0	1	0	5	6	193.7245101
g22	22	linear	0.0000%	0	1	8	11	19	236.4309755
g23	9	linear	0.0000%	0	2	3	1	6	-400.0551
g24	2	linear	79.6556%	0	2	0	0	2	-5.508013272

Table 3: Summary of the 18 scalable test problems. ρ is the estimated ratio between the feasible region and the search space, *I* the number of inequality constraints, *E* the number of equality constraints and *D* number of decision variables.

Function	**Search range**	**Objective type**	**Number of constraints**		ρ	
			E	***I***	***10D***	***30D***
C01	$[0,10]^D$	Non separable	0	2 Non separable	0.997689	1.000000
C02	$[-5.12, 5,12]^D$	Separable	1 Separable	2 Separable	0.000000	0.000000
C03	$[-1000,1000]^D$	Non separable	1 Separable	0	0.000000	0.000000
C04	$[-50,50]^D$	Separable	2 Non separable	0	0.000000	0.000000
C05	$[-600,600]^D$	Separable	2 Separable	0	0.000000	0.000000
C06	$[-600,600]^D$	Separable	2 Rotated	0	0.000000	0.000000
C07	$[-140,140]^D$	Non separable	0	1 Separable	0.000000	0.000000
C08	$[-140,140]^D$	Non separable	0	1 Rotated	0.505123	0.503725
C09	$[-500,500]^D$	Non separable	1 Separable	0	0.379512	0.375278
C10	$[-500,500]^D$	Non separable	1 Rotated	0	0.000000	0.000000
C11	$[-100,100]^D$	Rotated	1 Non separable	0	0.000000	0.000000
C12	$[-1000,1000]^D$	Separable	1 Non separable	1 Separable	0.000000	0.000000
C13	$[-500,500]^D$	Separable	0	2 Separable, 1 Non separable	0.000000	0.000000
C14	$[-1000,1000]^D$	Non separable	0	3 Separable	0.003112	0.006123
C15	$[-1000,1000]^D$	Non separable	0	3 Rotated	0.003210	0.006023
C16	$[-10,10]^D$	Non separable	2 Separable	1 Separable, 1 Non separable	0.000000	0.000000
C17	$[-10,10]^D$	Non separable	1 Separable	2 Non separable	0.000000	0.000000
C18	$[-50,50]^D$	Non separable	1 Separable	1 Separable	0.000010	0.000000

4.2 Performance measures

The following measures were computed to evaluate the performance of IMBFOA. The first five were taken from [21]:

- **Feasible run**: a run where at least one feasible solution is found within Max_FEs.
- **Successful run**: a run where a feasible solution $\mathbf{x}$ satisfying $f(\mathbf{x}) - f(\mathbf{x}^*) \leq 0.0001$ is found within Max_FEs.
- **Feasible rate** = (number of feasible runs) / total runs.
- **Success rate** = (number of successful runs) / total runs.
- **Success performance**= mean (FEs for successful runs) $\times$ (# of total runs) / (# of successful runs).
- **Successful swim**: A swim movement where the new position is better (based on the feasibility rules of the constraint-handling technique) than the current position.
- **Successful swim rate** = (number of successful swims) / total swims, where total swims=$S_b \times N_c \times GMAX$.
- **Progress ratio (PR)**: Proposed in [27], the aim is to measure the improvement capability of the algorithm within the feasible region of the search space. For this measure, high values are preferred because they indicate a higher improvement of the first feasible solution found. It is calculated as shown in the Equation 12:

$$PR = \begin{cases} \left| In\sqrt{\frac{f_{min}(\theta^B(G_{ff}))}{f_{min}(\theta^B(GMAX))}} \right|, & \text{if } f_{min}(\theta^B(GMAX) > 0 \\ \left| In\sqrt{\frac{f_{min}(\theta^B(G_{ff})+1}{f_{min}(\theta^B(GMAX)+1}} \right|, & \text{if } f_{min}(\theta^B(GMAX) = 0 \\ \left| In\sqrt{\frac{f_{min}(\theta^B(G_{ff})+2|f_{min}(\theta^B(GMAX))|}{f_{min}(\theta^B(GMAX)+2|f_{min}(\theta^B(GMAX))|}} \right|, & \text{if } f_{min}(\theta^B(GMAX) < 0 \end{cases} \quad (12)$$

where $f_{min}(\theta^B(G_{ff})$ is the value of the objective function of the first feasible solution found and $f_{min}(\theta^B(GMAX)$ is the value of the objective function of the best solution found. The best, average result and standard deviation (St.d) on 25 independent runs are presented.

4.3 Results of the first experiment

As a first step to understand the behavior of IMBFOA, the effectiveness of the skew mechanism to create the initial swarm of bacteria was analyzed. The parameters used by IMBFOA are those in Table 1.

Two representative test problems were used in this experiment: g03 from the first benchmark and C04 in 30D from the second and scalable benchmark. These two test problems were chosen because their estimated ratio between the feasible region and the whole search space (ρ) is close to zero (i.e., the feasible region is quite small and difficult to find). The sums of constraint violation of the run located on the median value of the fitness value, out of twenty five runs, of each one of the twenty bacteria just after the chemotaxis, reproduction and elimination dispersal processes based on the initial swarm generated with and without the skew mechanism are plotted in Figures 6 and 7, for test problems g03 and C04, respectively. As it can be observed, almost half of the swarm is feasible in problem g03 (Figure 6) and in problem C04 the violation is significantly decreased (Figure 7). Moreover, 92% of the twenty five independent runs obtained feasible solutions in the initial swarm using IMBFOA with skew mechanism in test problem g03, while 100% of the independent runs obtained a solution similar to the best known feasible solution in such test problem. On the other hand, only 8% of the independent runs obtained feasible solutions in the initial swarm with IMBFOA without skew mechanism and 84% of the independent runs provided a solution similar to the best known solution.

A similar behavior was observed in test problem C04 with 30D, IMBFOA with skew mechanism generated an initial swarm with a lower sum of constraint violation with respect to IMBFOA without skew mechanism. Furthermore, 68% of the twenty-five independent runs computed by the version with skew mechanism obtained feasible solutions and 44% found similar results to the best known solution. In contrast, for IMBFOA without skew mechanism, only 56% of the twenty-five independent runs obtained feasible solutions and 32% reached similar results to the best known solution.

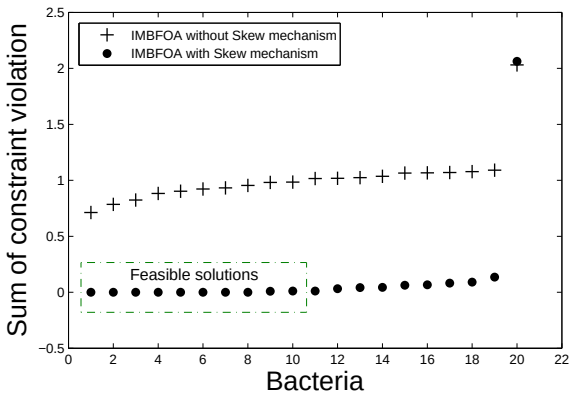


Fig. 6: IMBFOA with/without the skew mechanism in test problem g03. The sum of constraint violation of each bacterium in the swarm after the first generation is plotted.

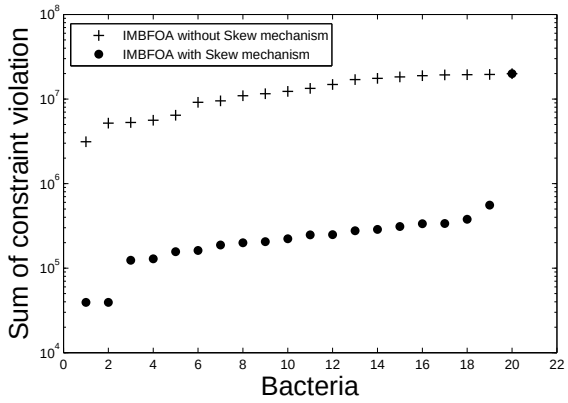


Fig. 7: IMBFOA with/without the skew mechanism in test problem C04 with 30D. The sum of constraint violation of each bacterium in the swarm after the first generation is plotted.

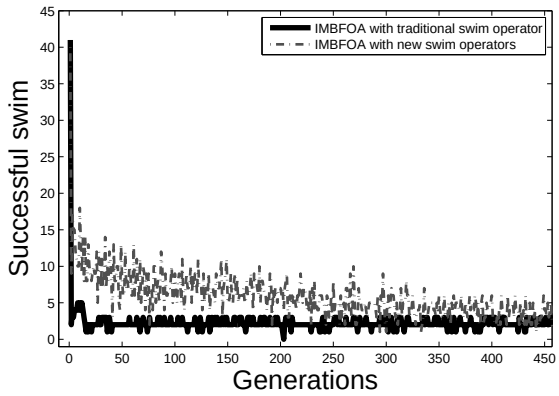


Fig. 8: Successful swims by IMBFOA with/without the two swims proposed in test problem g03 in the execution located in the median value of 25 independent runs.

To further analyze the effect of the step size control mechanism and the swim operators proposed, the number of successful swims were computed in test problems g03 and C04 for 30D. The successful swims by generation (G) obtained by IMBFOA and IMBFOA but with the traditional swim operator, of the run located in the median value of 25 independent runs per each one of the algorithms in the two test problems are plotted in Figures 8 (g03) and 9 (C04 for 30D). It is clear to note that IMBFOA always maintain a higher number of successful swims. The successful swim rates were 5.77% for IMBFOA, and 0.80% for the version with the traditional swim operator.

Finally, the convergence plots of IMBFOA and the variant with the traditional swim operator are shown in Figures 10 and 11 for test problem g03 and C04 for 30D, respectively. It is clear that the usage of the two proposed swims lead to better results.

From the overall results of this first experiment a preliminary conclusion is that IMBFOA seems to improve the constraint satisfaction, the ability to generate better solutions by the swims operators and a better convergence behavior. However, more evidence is provided in the next experiment.

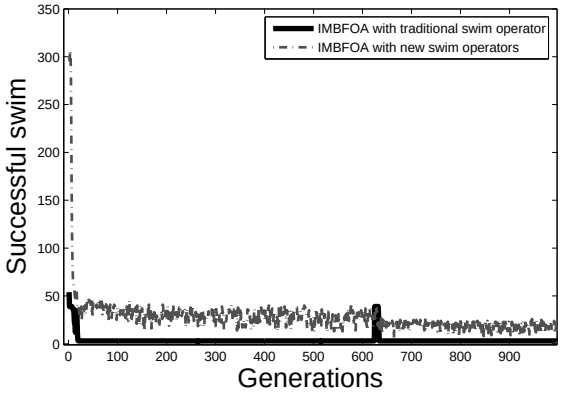


Fig. 9: Successful swims by IMBFOA with/without the two swims proposed in test problem C04 for 30D in the execution located in the median value of 25 independent runs.

4.4 Results of the second experiment

IMBFOA was compared against MBFOA using the two set of benchmark problems. The parameters adopted by IMBFOA and MBFOA are shown in Table 1. The tolerance for equality constraints ε was set to 1.0E-04.

The statistical results of the final fitness values obtained by each one of the two bacterial-based algorithms in the first set of 24 benchmark problems are summarized in Table 4. Based on such results the following findings are discussed: IMBFOA found feasible solutions to 21 of the 24 test problems, while MBFOA found feasible solutions in 17 of 24 test problems. Moreover, IMBFOA provided "better" best results in most test problems. Based on the WSRT, there are significant differences in all cases between IMBFOA and MBFOA.

Three performance measures (feasible rate, success rate and success performance) obtained by IMBFOA and BFOA in 25 independent runs are presented in Table 5. In terms of the feasible rate, IMBFOA obtained feasible solutions in all runs in most of the test problems, with the exception of test problems g20, g21 and g22. MBFOA could not find feasible solutions in those problems as well

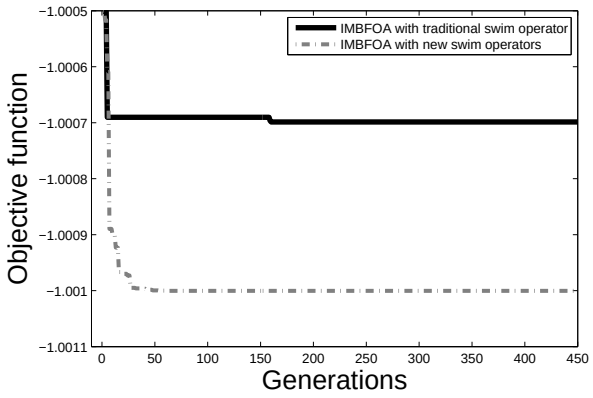


Fig. 10: Convergence plot by IMBFOA with/without the two swims proposed in test problem g03 in the execution located in the median value of 25 independent runs.

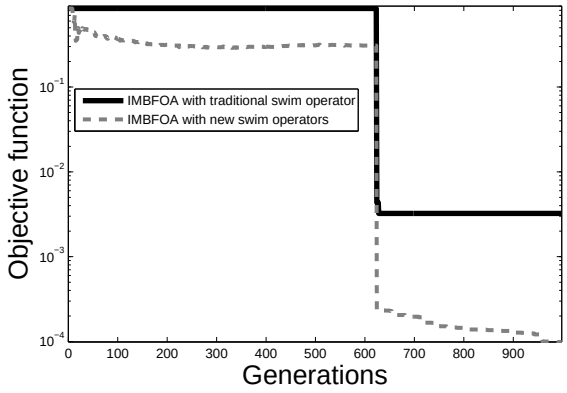


Fig. 11: Convergence plot by IMBFOA with/without the two swims proposed in test problem C04 for 30D in the execution located in the median value of 25 independent runs.

as in problems g05, g13 and g17. IMBFOA clearly outperformed MBFOA regarding success rate. Finally, it is also clear that IMBFOA required less computational cost with respect to MBFOA as suggested by the success performance values.

Table 6 includes the statistical results of the progress ratio measure on 25 independent runs. Unlike MBFOA, IMBFOA showed a very good ability to improve the first feasible solution found. Just in test problems g05, g09, g10, g11, g15, g17 and g24 the measure was close to zero. The common feature of such test problems is that they have a very small feasible region (see column ρ in Table 2).

Table 4: Statistical results obtained on 25 independent runs by IMBFOA and MBFOA in the first set of test problems. The 95%-confidence WSRT based on the best results reports significant differences between the algorithms.

Prob.	$f(\mathbf{x}^*)$	Criteria	MBFOA	IMBFOA
		FEs	240,000	240,000
		Runs	25	25
g01	-15	Best	-9.885358	**-15**
		Average	-4.913382	**-14.93**
		St.d	2.11E+00	**2.36E-01**
g02	-0.8036191	Best	-0.44923	**-0.8035462**
		Average	-0.332766	**-0.6801028**
		St.d	4.70E-02	**5.98E-02**
g03	-1.0005001	Best	-1.000472	**-1.001**
		Average	-1.00044	**-1.0009**
		St.d	**2.30E-05**	4.57E-05
g04	-30665.5387	Best	-30665.05071	**-30665.539**
		Average	-30663.33132	**-30665.539**
		St.d	4.19E+00	**0**
g05	5126.49671	Best	-	**5126.497**
		Average	-	**5126.496**
		St.d	-	**5.18E-05**
g06	-6961.81388	Best	-6960.833737	**-6961.813875**
		Average	-6950.804575	**-6961.81385**
		St.d	1.21E+01	**8.39E-05**
g07	24.3062091	Best	24.726052	**24.3062**
		Average	25.910263	**24.481**
		St.d	8.36E-01	**2.62E-01**
g08	-0.095825	Best	**-0.095825**	**-0.095825**
		Average	**-0.095825**	**-0.095825**
		St.d	1.60E-12	**8.01E-18**
g09	680.630057	Best	680.653436	**680.63**
		Average	680.706034	**680.64**
		St.d	4.25E-02	**3.70E-02**
g10	7049.24802	Best	7082.964009	**7049.24802**
		Average	7356.790777	**7320.5021**
		St.d	**4.89E+02**	8.10E+02
g11	0.7499	Best	**0.7499**	**0.7499**
		Average	0.749901	**0.7499**
		St.d	**2.12E-06**	3.48E-06
g12	-1	Best	-0.999999	**-1**
		Average	-0.999247	**-1**
		St.d	1.95E-03	**0**
g13	0.053941	Best	-	**0.053941**
		Average	-	**0.17709**
		St.d	-	**1.83E-01**
g14	-47.764888	Best	-42.534548	**-46.467894**
		Average	-38.684487	**-45.016895**
		St.d	2.52E+00	**9.84E-01**
g15	961.715022	Best	961.715343	**961.71502**
		Average	961.717716	**961.71502**
		St.d	1.57E-03	**1.10E-08**
g16	-1.905155	Best	-1.903357	**-1.905155**
		Average	-1.887545	**-1.904055**
		St.d	5.64E-02	**1.25E-03**
g17	8853.53967	Best	-	**8927.5917**
		Average	-	**8927.5918**
		St.d	-	**1.39E-04**
g18	-0.866025	Best	-0.859667	**-0.866025**
		Average	-0.730242	**-0.864223**
		St.d	1.18E-01	**2.46E-03**
g19	32.655592	Best	49.473018	**32.655593**
		Average	117.292903	**37.160608**
		St.d	7.33E+01	**8.45E+00**
g20	0.2049794	Best	-	-
		Average	-	-
		St.d	-	-
g21	193.72451	Best	-	-
		Average	-	-
		St.d	-	-
g22	236.430976	Best	-	-
		Average	-	-
		St.d	-	-
g23	-400.0551	Best	-	**-400.0023**
		Average	-	**-399.9196**
		St.d	-	**2.78E-01**
g24	-5.50801327	Best	-5.508006	**-5.508013**
		Average	-5.507687062	**-5.508013**
		St.d	2.83E-04	**2.70E-11**

The results obtained by IMBFOA and MBFOA in the second set of test problems with 10 and 30 dimensions, are presented in Table 7. The parameters used by IMBFOA and MBFOA are the same as in Table 1. The tolerance for equality constraints ε was set to 1.0E-04.

The results in Table 7 show that IMBFOA found feasible solutions in most of the 10D test problems with 10

Table 5: Feasible rate, success rate and success performance obtained by IMBFOA and MBFOA in the first set of test problems.

	Feasible rate		Success rate		Success performance	
Prob.	**MBFOA**	**IMBFOA**	**MBFOA**	**IMBFOA**	**MBFOA**	**IMBFOA**
g01	100 %	100 %	-	64 %	-	218,451
g02	100 %	100 %	-	4 %	-	3,104,700
g03	100 %	100 %	4 %	100 %	122,782	26,022
g04	100 %	100 %	-	100 %	-	7,707
g05	-	100 %	-	76 %	-	122,782
g06	100 %	100 %	-	96 %	-	49,496
g07	100 %	100 %	-	44 %	-	127,975
g08	100 %	100 %	12 %	100 %	68,024	601
g09	100 %	100 %	-	76 %	-	41,549
g10	100 %	100 %	-	84 %	210,201	68,537
g11	100 %	100 %	4 %	100 %	128,302	78,051
g12	100 %	100 %	12 %	100 %	-	3,991
g13	-	100 %	-	68 %	-	132,750
g14	33 %	100 %	-	-	-	-
g15	100 %	100 %	-	100 %	-	8,251
g16	100 %	100 %	-	20 %	-	588,396
g17	-	100 %	-	-	-	-
g18	90 %	100 %	-	32 %	-	85,422
g19	100 %	100 %	-	68 %	-	82,566
g20	-	-	-	-	-	10,525
g21	-	-	-	-	-	-
g22	-	-	-	-	-	-
g23	-	100 %	-	-	-	-
g24	100 %	100 %	32 %	100 %	20,400	4,089
Average	67.62 %	87.5 %	2.66 %	74 %	109,942	198,411

Table 6: Statistical values for the progress ratio measure obtained by IMBFOA and MBFOA in the first set of test problems.

	Best		Average		St.d	
Prob.	**MBFOA**	**IMBFOA**	**MBFOA**	**IMBFOA**	**MBFOA**	**IMBFOA**
g01	1.27E+00	1.61E+00	1.26E+00	1.43E+00	5.22E-03	6.86E-02
g02	1.60E+00	1.59E+00	1.60E+00	1.58E+00	1.88E-03	6.00E-03
g03	1.61E+00	1.95E+00	1.59E+00	1.60E+00	9.43E-03	1.11E-01
g04	5.17E+00	5.17E+00	5.17E+00	5.17E+00	6.02E-05	0.00E+00
g05	-	1.36E-07	-	1.28E-07	-	9.45E-09
g06	4.42E+00	4.48E+00	4.42E+00	4.43E+00	3.71E-03	1.61E-02
g07	0.00E+00	1.47E+00	0.00E+00	5.68E-01	0.00E+00	5.44E-01
g08	1.60E+00	1.60E+00	1.60E+00	1.60E+00	2.20E-05	1.13E-13
g09	0.00E+00	2.76E-01	0.00E+00	1.39E-01	0.00E+00	1.36E-01
g10	0.00E+00	4.11E-01	0.00E+00	8.79E-02	0.00E+00	1.43E-01
g11	6.38E-05	1.44E-01	2.51E-05	1.32E-01	2.83E-05	4.15E-02
g12	1.57E+00	1.57E+00	1.57E+00	1.57E+00	3.73E-10	2.22E-04
g13	-	1.05E+00	-	4.37E-01	-	5.40E-01
g14	1.86E+00	1.95E+00	1.84E+00	1.91E+00	2.04E-02	2.64E-02
g15	0.00E+00	7.79E-08	0.00E+00	7.79E-08	0.00E+00	4.67E-12
g16	1.96E+00	2.25E+00	1.84E+00	1.34E+00	1.42E-01	7.18E-01
g17	-	3.35E-07	-	3.30E-07	-	2.77E-09
g18	1.59E+00	1.70E+00	1.58E+00	1.67E+00	2.81E-03	4.96E-02
g19	0.00E+00	2.76E+00	0.00E+00	9.44E-01	0.00E+00	1.03E+00
g20	-	-	-	-	-	-
g21	-	-	-	-	-	-
g22	-	-	-	-	-	-
g23	-	2.99E+00	-	2.99E+00	-	7.28E-04
g24	6.27E-01	6.96E-01	6.25E-01	6.36E-01	1.80E-03	2.14E-02

dimensions (except C11). In contrast, MBFOA found feasible solutions only in 14 of 18 10D test problems. Regarding 30D, IMBFOA found feasible solutions in all test problems with the exception of test problem C03, while MBFOA found feasible solutions only in 13 out of 18 30D test problems. With respect to the best and average fitness values, IMBFOA outperformed MBFOA in most test problems, except C11. Finally, as it was the case on the first benchmark, the WSRT reported significant differences between the compared algorithms.

The feasible rates obtained by IMBFOA and MBFOA are presented in Table 8, where IMBFOA obtained again better results in most of the test problems. MBFOA obtained feasible solutions only in 14 out of 18 10D test problems, while that IMBFOA found feasible solutions to

Table 7: Statistical results obtained on 25 independent runs by IMBFOA and MBFOA in the second set of test problems. The 95%-confidence WSRT based on the best results reports significant differences between the algorithms.

		10D		30D	
Prob.	**Criteria**	**MBFOA**	**IMBFOA**	**MBFOA**	**IMBFOA**
	FEs	200,000	200,000	200,000	200,000
	Runs	25	25	25	25
C01	Best	-0.7441309	**-0.7473104**	-0.2651637	**-0.3996041**
	Average	-0.6682597	**-0.723775**	-0.2242018	**-0.2914506**
	St.d	6.99E-02	**3.14E-02**	**1.89E-02**	3.79E-02
C02	Best	1.2378498	**-2.2777066**	2.9831645	**-2.280652**
	Average	2.7822775	**-2.0803575**	4.0398068	**-2.219141**
	St.d	7.87E-01	**3.64E-01**	4.75E-01	**1.90E-01**
C03	Best	-	**7.20E-05**	-	-
	Average	-	**1.242E+11**	-	-
	St.d	-	**4.65E+11**	-	-
C04	Best	-	**-6.04E-04**	-	**3.50E-08**
	Average	-	**1.02E-01**	-	**1.44E-04**
	St.d	-	**2.76E-01**	-	**2.16E-04**
C05	Best	510.62142	**-483.599468**	451.47207	**-483.590859**
	Average	510.62142	**-296.059023**	518.458869	**-181.502159**
	St.d	**0.00E+00**	1.60E+02	**3.14E+01**	2.76E+02
C06	Best	559.64589	**-578.66231**	478.916572	**-530.543371**
	Average	559.64589	**-507.301756**	564.091995	**-527.240421**
	St.d	**0.00E+00**	1.13E+02	3.62E+01	**6.55E+00**
C07	Best	1.29E+00	**7.13E-09**	8.89E+09	**3.47E-10**
	Average	6.65E+01	**9.57E-01**	3.41E+10	**1.59E+00**
	St.d	1.13E+02	**1.74E+00**	1.75E+10	**1.99E+00**
C08	Best	4.02E-01	**2.95E-09**	4.62E+09	**2.96E-10**
	Average	1.47E+02	**7.37E+01**	2.95E+10	**1.13E+01**
	St.d	3.60E+02	**1.55E+02**	1.68E+10	**2.94E+01**
C09	Best	1.37E+12	**2.81E-11**	1.27E+13	**3.92E-11**
	Average	5.77E+12	**3.28E+07**	2.67E+13	**4.10E+06**
	St.d	3.16E+12	**1.33E+08**	6.65E+12	**6.22E+06**
C10	Best	1.44E+12	**4.08E+01**	1.31E+13	**3.67E-11**
	Average	7.43E+12	**4.29E+06**	2.54E+13	**3.21E+03**
	St.d	4.87E+12	**7.08E+06**	6.41E+12	**5.49E+03**
C11	Best	-	-	-	**-3.33E-04**
	Average	-	-	-	**-1.11E-04**
	St.d	-	-	-	**9.04E-05**
C12	Best	-	**-0.1992**	-	**-0.1992434**
	Average	-	**-1.95E-01**	-	**1.38E+00**
	St.d	-	**1.14E-02**	-	**6.33E+00**
C13	Best	-62.27577	**-62.27639**	-	**-62.751801**
	Average	-50.11026	**-58.15052**	-	**-59.22955**
	St.d	6.04E+00	**2.63E+00**	-	**2.20E+00**
C14	Best	1.58E+11	**4.08E-08**	6.32E+13	**1.26E-08**
	Average	7.59E+12	**2.22E+05**	2.31E+14	**2.28E+06**
	St.d	6.36E+12	**8.34E+05**	7.56E+13	**6.28E+06**
C15	Best	1.19E+13	**4.50E+00**	8.57E+13	**1.06E-08**
	Average	4.73E+13	**2.22E+07**	2.71E+14	**2.69E+04**
	St.d	2.03E+13	**5.45E+07**	6.20E+13	**8.79E+04**
C16	Best	4.95E-01	**0.00E+00**	1.14E+00	**1.44E-15**
	Average	9.65E-01	**3.81E-02**	1.20E+00	**5.16E-01**
	St.d	1.28E-01	**7.99E-02**	3.67E-02	**5.01E-01**
C17	Best	9.12E+01	**3.07E-16**	7.19E+02	**3.47E-15**
	Average	2.47E+02	**1.00E+00**	1.43E+03	**6.54E+01**
	St.d	1.20E+02	**1.50E+00**	3.14E+02	**2.60E+02**
C18	Best	3.75E+03	**5.61E-16**	1.59E+04	**2.78E-15**
	Average	6.20E+03	**4.09E-10**	2.71E+04	**4.58E-11**
	St.d	1.53E+03	**1.31E-09**	5.75E+03	**8.32E-11**

17 of 18 test problems. In 30D test problems the behavior was similar, IMBFOA found feasible solutions in 17 out of 18 test problems while MBFOA found feasible solutions only in 13 of 18 test problems.

In terms of success rate and success performance, the results of both algorithms are presented in Table 9 and 10, respectively. The reference results to compute these two measures were taken from [43]. From these two tables it is clear to see that IMBFOA outperformed MBFOA because the latter was unable to get success runs in any given test problem. The progress ratio measure results are shown in Table 11 where in most 10D test problems IMBFOA was able to improve the first feasible solution found (i.e. it was able to move inside the feasible region of the search space); just in test problems C03, C09, C10 and C15 IMBFOA was unable to show such behavior. In 30D test problems IMBFOA was also able to move for

Table 8: Feasible rate obtained by IMBFOA and MBFOA in the second set of test problems.

	10D		30D	
Prob.	MBFOA	IMBFOA	MBFOA	IMBFOA
C01	100 %	100 %	100 %	100 %
C02	100 %	100 %	100 %	100 %
C03	-	100%	-	-
C04	-	92%	-	68 %
C05	98 %	100 %	96 %	100 %
C06	100 %	100 %	96 %	100 %
C07	100 %	100 %	100 %	100 %
C08	100 %	100 %	100 %	100 %
C09	100 %	100 %	100 %	100 %
C10	100 %	100 %	100 %	100 %
C11	-	-	-	32 %
C12	-	76%	-	92 %
C13	100 %	100 %	-	100 %
C14	100 %	100 %	100 %	100 %
C15	100 %	100 %	100 %	100 %
C16	100 %	100 %	88 %	100 %
C17	100 %	100 %	100 %	100 %
C18	100 %	100 %	100 %	100 %
Average	77.67 %	98.12 %	71.11 %	93.65 %

improvement inside the feasible region in most test problems, with the exception of test problems C02, C04, C07, C09 and C14. Unlike IMBFOA, MBFOA scarcely showed some improvement inside the feasible region in 10D and 30D test problems.

Table 9: Success rate obtained by IMBFOA and MBFOA in the second set of test problems.

	10D		30D	
Prob.	MBFOA	IMBFOA	MBFOA	IMBFOA
C01	-	16 %	-	-
C02	-	28 %	-	-
C03	-	4%	-	-
C04	-	80%	-	44 %
C05	-	-	-	-
C06	-	24 %	-	-
C07	-	76 %	-	60 %
C08	-	32 %	-	80 %
C09	-	8 %	-	12 %
C10	-	-	-	-
C11	-	-	-	48 %
C12	-	-	-	24 %
C13	-	-	-	-
C14	-	52 %	-	4 %
C15	-	-	-	28 %
C16	-	68 %	-	12 %
C17	-	32 %	-	16 %
C18	-	100 %	-	100 %
Average	-	43.33 %	-	38.91 %

The findings of the second experiment confirm those of the first experiment, because IMBFOA improved the final results obtained by the original MBFOA in the two sets of benchmark problems, even in those with a high dimensionality (i.e. 30D test problems). Moreover, IMBFOA outperformed MBFOA in its abilities to find the feasible region and to reach the vicinity of the best known solution in most test problems. Furthermore, IMBFOA showed capacity to move inside the feasible region of the search space, even in those quite small feasible regions.

Table 10: Success performance obtained by IMBFOA and MBFOA in the second set of test problems.

	10D		30D	
Prob.	MBFOA	IMBFOA	MBFOA	IMBFOA
C01	-	687,359	-	-
C02	-	269,591	-	-
C03	-	3,757,000	-	-
C04	-	33,049	-	965,601
C05	-	-	-	-
C06	-	280,688	-	-
C07	-	39,556	-	63,231
C08	-	171,753	-	173,904
C09	-	131,325	-	420,908
C10	-	-	-	631,556
C11	-	-	-	10,592,075
C12	-	-	-	2,139,503
C13	-	-	-	-
C14	-	195,961	-	760,875
C15	-	-	-	916,357
C16	-	84,313	-	2,797,425
C17	-	77,255	-	881,653
C18	-	10,769	-	50,782
Average	-	478,218	-	1,699,489

Table 11: Statistical values for the progress ratio measure obtained by IMBFOA and MBFOA in the second set of test problems.

	10D						30D					
	Best		Average		St.d		Best		Average		St.d	
Prob.	MBFOA	IMBFOA	MBFOA	IMBFOA	MBFOA	IMBFOA	MBFOA	IMBFOA	MBFOA	IMBFOA	MBFOA	IMBFOA
C01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C02	0.00E+00	0.00E+00	1.24E-03	6.41E-01	0.00E+00	5.33E-06	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C03	-	0.00E+00	-	0.00E+00	-	0.00E+00	-	0.00E+00	-	0.00E+00	-	0.00E+00
C04	-	1.61E+00	-	1.61E+00	-	2.37E-05	-	1.99E+00	-	1.66E+00	-	2.32E-01
C05	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C06	6.27E-01	3.18E+00	6.25E-01	3.18E+00	1.80E-03	4.64E-07	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C07	0.00E+00	5.11E-02	0.00E+00	3.54E-02	0.00E+00	1.01E-02	0.00E+00	4.78E-01	0.00E+00	4.36E-01	0.00E+00	2.86E-02
C08	0.00E+00	5.91E-03	0.00E+00	4.89E-03	0.00E+00	1.45E-03	0.00E+00	9.96E-01	0.00E+00	4.55E-01	0.00E+00	2.43E-01
C09	6.38E-05	2.99E+00	2.51E-05	2.99E+00	2.83E-05	0.00E+00	0.00E+00	2.50E+00	0.00E+00	2.36E+00	0.00E+00	2.04E-01
C10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C11	-	-	-	-	-	-	-	0.00E+00	-	0.00E+00	-	0.00E+00
C12	-	0.00E+00	-	0.00E+00	-	0.00E+00	-	1.60E+00	-	1.60E+00	-	2.02E-07
C13	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	-	0.00E+00	-	0.00E+00	-	0.00E+00
C14	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C15	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.07E-01	0.00E+00	1.07E-01	0.00E+00	0.00E+00
C16	5.91E-03	1.37E+01	4.89E-03	2.82E+00	1.45E-03	6.10E+00	0.00E+00	3.69E+00	0.00E+00	2.29E+00	0.00E+00	1.08E+00
C17	0.00E+00	5.46E+00	0.00E+00	5.46E+00	0.00E+00	0.00E+00	0.00E+00	1.57E+00	0.00E+00	4.62E-01	0.00E+00	7.42E-01
C18	7.95E-01	9.47E+00	7.63E-01	4.21E+00	3.12E-01	3.10E+00	0.00E+00	7.52E+00	0.00E+00	2.06E+00	0.00E+00	2.58E+00

4.5 Results of the third experiment

The final results obtained by IMBFOA were compared with those obtained by different state-of-the-art nature-inspired algorithms to solve CNOPs. Table 12 presents a comparison in the first set of test problems against the Memetic Self-Adaptive Multi-Strategy Differential Evolution (Memetic-SAMSDE), which also uses SQP as local search [9], the Adaptive Penalty Formulation with GA (APF-GA) [44], and the Modified Differential Evolution (MDE) [24]. The number of FEs computed by APF-GA and MDE was 500,000 while for Memetic-SAMSDE and IMBFOA was 240,000 FEs.

Based on the results in Table 12, IMBFOA was able to provide similar results with respect to the three compared algorithms. Regarding the computational cost measured by the number of FEs, IMBFOA required less than half of the FEs required by APF-GA and MDE to reach competitive results. The Friedman test confirmed that there were not significant differences among the algorithms compared. The p−value in this set of test problems, based on the mean values of each test problem, was 0.8591. Only test problems where all algorithms found feasible solutions (twenty one) were considered in such test.

Table 12: Statistical comparison of IMBFOA and state-of-the-art nature-inspired algorithms in the first set of test problems.

Prob.	$f(x^*)$	Criteria	IMBFOA	Memetic-SAMSDE	APF-GA	MDE
		FEs	240,000	240,000	500,000	500,000
		Runs	25	30	25	25
g01	-15	Best	**-15**	**-15**	**-15**	**-15**
		Average	-14.93	**-15**	**-15**	**-15**
		St.d	2.36E-01	**0**	**0**	**0**
g02	-0.8036191	Best	-0.8035462	**-0.8036191**	-0.803601	**-0.8036191**
		Average	-0.6801028	**-0.8036191**	-0.803518	-0.78616
		St.d	5.98E-02	**0**	1.00E-04	1.20E-02
g03	-1.0005001	Best	**-1.001**	1.0005	**-1.001**	-1.0005
		Average	-1.0009	-1.0005	**-1.001**	-1.0005
		St.d	4.57E-05	**0**	**0**	**0**
g04	-30665.5387	Best	**-30665.539**	**-30665.539**	**-30665.539**	-30665.5386
		Average	**-30665.539**	**-30665.539**	**-30665.539**	-30665.5386
		St.d	**0**	**0**	1.00E-04	**0**
g05	5126.49671	Best	**5126.497**	**5126.497**	**5126.497**	**5126.497**
		Average	5126.496	**5126.497**	5127.5423	**5126.497**
		St.d	5.18E-05	**0**	1.43E+00	**0**
g06	-6961.81388	Best	**-6961.813875**	**-6961.813875**	**-6961.814**	**-6961.814**
		Average	-6961.813851	**-6961.813875**	**-6961.814**	**-6961.814**
		St.d	8.39E-05	**0**	**0**	**0**
g07	24.3062091	Best	**24.3062**	**24.3062**	**24.3062**	**24.3062**
		Average	24.481	**24.3062**	**24.3062**	**24.3062**
		St.d	2.62E-01	**0**	**0**	**0**
g08	-0.095825	Best	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
		Average	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
		St.d	8.01E-18	**0**	**0**	**0**
g09	680.630057	Best	**680.63**	**680.63**	**680.63**	**680.63**
		Average	680.64	**680.63**	**680.63**	**680.63**
		St.d	3.70E-02	**0**	**0**	**0**
g10	7049.24802	Best	**7049.24802**	**7049.24802**	**7049.24802**	**7049.24802**
		Average	7320.5021	**7049.24802**	7077.6821	**7049.24802**
		St.d	8.10E+02	**0**	5.12E+01	**0**
g11	0.7499	Best	**0.7499**	**0.7499**	**0.7499**	**0.7499**
		Average	**0.7499**	**0.7499**	**0.7499**	**0.7499**
		St.d	3.48E-06	**0**	**0**	**0**
g12	-1	Best	**-1**	**-1**	**-1**	**-1**
		Average	**-1**	**-1**	**-1**	**-1**
		St.d	**0**	**0**	**0**	**0**
g13	0.053941	Best	0.053941	**0.053942**	**0.053942**	**0.053942**
		Average	0.17709	**0.053942**	**0.053942**	**0.053942**
		St.d	1.83E-01	**0**	**0**	**0**
g14	-47.764888	Best	-46.467894	**-47.764888**	-47.76479	-47.764887
		Average	-45.016895	**-47.764888**	-47.76479	-47.764874
		St.d	9.84E-01	**0**	1.00E-04	1.40E-05
g15	961.715022	Best	**961.71502**	**961.71502**	**961.71502**	**961.71502**
		Average	**961.71502**	**961.71502**	**961.71502**	**961.71502**
		St.d	1.10E-08	**0**	**0**	**0**
g16	-1.905155	Best	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**
		Average	-1.904055	**-1.905155**	**-1.905155**	**-1.905155**
		St.d	1.25E-03	**0**	**0**	**0**
g17	8853.53967	Best	8927.5917	**8853.5397**	8853.5398	**8853.5397**
		Average	8927.5918	**8823.5397**	8888.4876	**8853.5397**
		St.d	1.39E-04	**0**	2.90E+01	**0**
g18	-0.866025	Best	**-0.866025**	**-0.866025**	**-0.866025**	**-0.866025**
		Average	-0.864223	**-0.866025**	-0.865925	**-0.866025**
		St.d	2.46E-03	**0**	1.00E-04	**0**
g19	32.655592	Best	**32.655593**	**32.655593**	**32.655593**	32.64827
		Average	37.160608	**32.655593**	**32.655593**	33.34125
		St.d	8.45E+00	**0**	**0**	8.47E-01
g20	0.2049794	Best	-	-	-	-
		Average	-	-	-	-
		St.d	-	-	-	-
g21	193.72451	Best	-	**193.72451**	196.63301	**193.72451**
		Average	-	**193.72451**	199.51581	**193.72451**
		St.d	-	**0**	2.36E+00	**0**
g22	236.430976	Best	-	**236.370313**	-	-
		Average	-	**245.738829**	-	-
		St.d	-	**9.05E+00**	-	-
g23	-400.0551	Best	-400.0023	**-400.0551**	-399.7624	**-400.0551**
		Average	-399.9196	**-400.0551**	-394.7627	**-400.0551**
		St.d	2.78E-01	**0**	3.87E+00	**0**
g24	-5.50801327	Best	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**
		Average	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**
		St.d	2.70E-11	**0**	**0**	**0**

IMBFOA was compared in Tables 13 (10D) and 14 (30D) against Memetic-SAMSDE, ε constrained Differential Evolution [43], and a Genetic Algorithm based algorithm (IEMA) [42] in the second set of benchmark problems. All algorithms required 500,000 FEs for the reported results. Based on Table 13 IMBFOA obtained very competitive results in most of the 10D test problems.

The Friedman test indicated significant differences among the algorithms compared with a p−value of 0.04.Test problem C11 was not considered because IMBFOA did not find feasible solutions.

In the case of 30D test problems (Table 14) the Friedman test suggested significant differences among the algorithms in the comparison with a p−value of 0.0293. Test problems C03, C04, C11 and C12 were not considered because IEMA did not find feasible solutions and IMBFOA did not reach the feasible region in test problem C03 as well.

Table 13: Statistical comparison of IMBFOA and state-of-the-art nature-inspired algorithms in the second set of test problems (10D).

Prob.	Criteria	IMBFOA	Memetic-SAMSDE	εDEag	IEMA
	Fes	200,000	200,000	200,000	200,000
	Runs	25	25	25	25
C01	Best	**-0.7473104**	**-0.7473104**	**-0.7473104**	-0.74731
	Average	-0.723775	**-0.7473104**	-0.7470402	-0.743189
	St.d	3.14E-02	**0**	1.32E-03	4.33E-03
C02	Best	-2.2777066	-2.2777099	-2.277702	**-2.27771**
	Average	-2.0803575	-2.2776477	-2.25887	**-2.27771**
	St.d	3.64E-01	5.9704E-05	2.39E-02	**1.82E-07**
C03	Best	7.20E-05	**0**	**0**	1.47E-16
	Average	1.242E+11	4.8159E-21	**0**	6.23E-07
	St.d	4.65E+11	6.862E-21	**0**	1.40E-06
C04	Best	**-6.04E-04**	-1.00E-05	-9.99E-06	-9.99E-06
	Average	1.02E-01	**-1.00E-05**	-9.92E-06	-9.91E-06
	St.d	2.76E-01	**2.28E-10**	1.55E-07	8.99E-08
C05	Best	-483.599468	-483.610625	-483.6106	**-483.611**
	Average	-296.059023	**-483.610625**	-483.6106	-379.156
	St.d	1.60E+02	1.4883E-07	**3.89E-13**	1.79E+02
C06	Best	-578.66231	**-578.66236**	-578.6581	-578.662
	Average	-507.301756	**-578.6622**	-578.6528	-551.47
	St.d	1.13E+02	**1.18E-04**	3.63E-003	7.36E+01
C07	Best	7.13E-09	**0**	**0**	1.75E-08
	Average	9.57E-01	9.30E-24	**0**	3.26E-09
	St.d	1.74E+00	1.71E-23	**0**	3.39E-09
C08	Best	2.95E-09	**0**	**0**	1.01E-10
	Average	7.37E+01	**9.51E-20**	6.73E+00	4.07E+00
	St.d	1.55E+02	**3.49E-19**	5.56E+00	6.38E+00
C09	Best	2.81E-11	**0**	**0**	1.20E-09
	Average	3.28E+07	1.29E-21	**0**	1.95E+12
	St.d	1.33E+08	2.87E-21	**0**	5.40E+12
C10	Best	4.08E+01	**0**	**0**	5.40E-09
	Average	4.29E+06	7.46E-23	**0**	2.56E+12
	St.d	7.08E+06	1.56E-22	**0**	3.97E+12
C11	Best	-	**1.52E-03**	**1.52E-03**	**1.52E-03**
	Average	-	**1.52E-03**	**1.52E-03**	-1.15E-03
	St.d	-	1.36E-08	**6.34E-11**	2.73E-08
C12	Best	-0.1992	**-570.0899**	**-570.0899**	-10.9735
	Average	-0.1948	-3.3553	**-336.7349**	-0.648172
	St.d	**1.14E-02**	1.16E+02	1.78E+02	2.20E+00
C13	Best	-62.27639	-68.42937	-68.42937	**-68.4294**
	Average	-58.15052	**-68.42937**	**-67.42937**	-68.0182
	St.d	2.63E+00	**0**	1.03E-06	1.40E+00
C14	Best	4.08E-08	**0**	**0**	8.04E-10
	Average	2.22E+05	9.78E-21	**0**	5.63E+01
	St.d	8.34E+05	2.10E-20	**0**	1.83E+02
C15	Best	4.50E+00	**0**	**0**	9.35E-10
	Average	2.22E+07	**2.48E-20**	1.80E-01	1.58E+08
	St.d	5.45E+07	**1.01E-19**	8.81E-01	6.04E+08
C16	Best	**0**	**0**	**0**	4.44E-16
	Average	3.81E-02	**0**	3.70E-01	3.30E-02
	St.d	7.99E-02	**0**	3.71E-01	2.26E-02
C17	Best	3.07E-16	**0**	1.46E-017	9.48E-15
	Average	1.00E+00	**8.17E-16**	1.25E-01	3.15E-03
	St.d	1.50E+00	**1.45E-15**	1.94E-01	1.58E-02
C18	Best	5.61E-16	**0**	3.73E-20	2.24E-15
	Average	4.09E-10	**3.04E-25**	9.68E-19	1.62E-14
	St.d	1.31E-09	**1.50E-24**	1.81E-18	3.82E-14

It is worth noticing that the competitive performance IMBFOA showed in the 10D test problems is not significantly affected when solving them on 30D. One shortcoming of IMBFOA is that the number of FEs to deal with the 30D test problems increased considerably with respect to the number computed on 10D (e.g. test problem C04).

The conclusions of this third experiment are that IMBFOA provided competitive results against five state-of-the-art nature-inspired algorithms to solve CNOPs on 60 test problems. The performance of IMBFOA slightly decreased in presence of high dimensionality, but it remained as competitive. However,

Table 14: Statistical comparison of IMBFOA and state-of-the-art nature-inspired algorithms in the second set of test problems (30D).

Prob.	Criteria	IMBFOA	Memetic-SAMSDE	εDEag	IEMA
	FEs	600,000	600,000	600,000	600,000
	Runs	25	25	25	25
C01	Best	-0.3996041	**-0.8218843**	-0.8218255	-0.821883
	Average	-0.2914506	-0.8156324	**-0.8208687**	-0.817769
	St.d	3.79E-02	4.41E-03	**7.10E-04**	4.79E-03
C02	Best	-2.2806521	**-2.2880962**	-2.169248	-2.28091
	Average	-2.2191412	**-2.2777017**	-2.151424	-1.50449
	St.d	1.90E-01	**9.85E-04**	1.20E-02	2.14E+00
C03	Best	-	**1.15E-20**	2.87E+01	-
	Average	-	**9.85E-18**	2.88E+01	-
	St.d	-	**3.48E-17**	8.05E-01	-
C04	Best	3.50E-08	**-3.33E-06**	4.70E-03	-
	Average	1.44E-04	**1.51E-05**	8.16E-03	-
	St.d	2.16E-04	**9.11E-05**	3.07E-03	-
C05	Best	-483.59086	**-483.61062**	-453.1307	-286.678
	Average	-181.502159	**-483.61058**	-449.546	-270.93
	St.d	2.76E+02	**9.60E-05**	2.90E+00	1.41E+01
C06	Best	-530.543	**-530.637**	-528.575	-529.593
	Average	-527.2404	**-530.098**	-527.9068	-132.876
	St.d	6.55E+00	3.08E-01	**4.75E-01**	5.61E+02
C07	Best	3.47E-10	**5.02E-25**	1.15E-15	4.82E-10
	Average	1.59E+00	**9.34E-20**	2.60E-15	8.49E-10
	St.d	1.99E+00	**1.67E-19**	1.23E-15	4.84E-10
C08	Best	2.96E-10	**3.40E-21**	2.52E-14	1.12E-09
	Average	1.13E+01	**1.65E-17**	7.83E-14	1.77E+01
	St.d	2.94E+01	**3.89E-17**	4.86E-14	4.08E+01
C09	Best	3.92E-11	**9.82E-23**	2.77E-16	7.31E+03
	Average	4.10E+06	**1.38E-14**	1.07E+01	2.99E+07
	St.d	6.22E+06	**4.59E-14**	2.82E+01	4.50E+07
C10	Best	3.67E-11	**4.88E-25**	3.25E+01	2.77E+04
	Average	3.21E+03	**1.62E-15**	3.33E+01	1.58E+07
	St.d	5.49E+03	**3.64E-15**	4.55E-01	1.68E+07
C11	Best	-3.33E-04	**-3.92E-04**	-3.27E-04	-
	Average	-1.11E-04	**-3.92E-04**	-2.86E-04	-
	St.d	9.04E-05	**4.80E-07**	2.71E-05	-
C12	Best	-0.1992434	**-0.1992611**	-0.1991453	-
	Average	1.38E+00	**-1.99E-01**	3.56E+02	-
	St.d	6.33E+00	**2.01E-06**	2.89E+02	-
C13	Best	-62.7518	**-68.42936**	-66.42473	**-68.4294**
	Average	-59.2295	**-68.2398**	-65.3531	-67.4872
	St.d	2.20E+00	3.45E-01	5.73E-01	**9.84E-01**
C14	Best	1.26E-08	**1.57E-19**	5.02E-14	3.29E-09
	Average	2.28E+06	2.81E-12	**3.09E-13**	7.38E-09
	St.d	6.28E+06	6.25E-12	**5.61E-13**	3.07E-01
C15	Best	1.06E-08	**7.47E-21**	2.16E+01	3.12E+04
	Average	2.69E+04	**3.94E-15**	2.16E+01	2.29E+08
	St.d	8.79E+04	**1.31E-14**	1.10E-04	4.64E+08
C16	Best	1.44E-15	**0**	**0**	6.16E-12
	Average	5.16E-01	**0**	2.17E-21	1.63E-03
	St.d	5.01E-01	**0**	1.06E-20	8.16E-03
C17	Best	**3.47E-15**	6.91E-11	2.17E-01	9.28E-10
	Average	6.54E+01	**1.23E-07**	6.33E+00	8.84E-02
	St.d	2.60E+02	**1.48E-07**	4.99E+00	1.51E-01
C18	Best	2.78E-15	**3.00E-20**	1.23E+00	1.38E-14
	Average	4.58E-11	**8.36E-15**	8.75E+01	4.74E-14
	St.d	8.32E-11	3.84E-14	1.66E+02	**6.57E-14**

the number of FEs required by IMBFOA in such large scale search spaces increased significantly.

5 Conclusions

An improved modified bacterial foraging optimization algorithm was presented in this paper. Two swim operators to be applied within the chemotaxis cycle were proposed. Both swims aimed to improve the capabilities of the algorithm to explore and exploit the search space, while tackling the already documented sensitivity to the stepsize used in the tumble-swim operator of the original MBFOA by adopting a dynamic value which decreases along the search. Moreover, a skew mechanism to divide the initial swarm in three groups, one skewed to the lower bounds of the search space, another one located in the upper bounds and finally one located in all the search space, was added to favor the exploration/exploitation of the search space combined with the two-swim operator. Finally, a local search operator based on SQP was applied twice during the search (one time after the first cycle of the algorithm and another one at half of the GMAX cycles). Finally, the reproduction step was scarcely applied so as to discourage premature convergence due to the duplication of bacteria.

Three experiments were carried out, where two well-known benchmarks with a total of 60 test problems were solved (1) to provide preliminary evidence of the behavior from the new mechanisms added to the algorithm, (2) to provide an in-depth comparison between IMBFOA and the original MBFOA based on final results and also on four performance measures, and (3) to compare the results of IMBFOA against those provided by five state-of-the-art nature-inspired algorithms to solve CNOPs.

From the preliminary experiments it was found that IMBFOA improved the MBFOA ability to reach the feasible region of the search space (even in tiny feasible regions), and it was also found that the combination of the two swims improved the generation of better solutions during all the search. Such findings were confirmed by the results of the second experiment, where IMBFOA clearly outperformed MBFOA based on final results, besides showing a better capacity to move inside the feasible region after reaching it. Finally, IMBFOA provided a competitive performance against state-of-the-art algorithms, and, to the best of the authors' knowledge, this is the first attempt to design a BFOA-based algorithm to solve a wide set of CNOPs.

One shortcoming found in IMBFOA was the high number of FEs required to find good results in high-dimensional CNOPs. This is the starting point of the future work, where a more suitable way to combine IMBFOA with the local search operator will be studied. Furthermore, the swim operators will be revisited to increase their probability to generate better solutions with less evaluations per swim.

Acknowledgement

The first author acknowledges support from Universidad Juárez Autónoma de Tabasco (UJAT) and the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through a scholarship to pursue Ph.D studies at UJAT, México. Third author acknowledges support from CONACyT through project No. 220522.

References

[1] Arijit Biswas, Swagatam Das, Ajith Abraham, and Sambarta Dasgupta. Analysis of the reproduction operator in an

artificial bacterial foraging system. *Applied Mathematics and Computation*, 0(215):3343–3355, 2010.

[2] Arijit Biswas, Sambarta Dasgupta, Swagatam Das, and Ajith Abraham. A Synergy of Differential Evolution and Bacterial Foraging Optimization for global optimization. *Neural Network World*, 17:607–626, 2007.

[3] Arijit Biswas, Sambarta Dasgupta, Swagatam Das, and Ajith Abraham. Synergy of pso and bacterial foraging optimization - a comparative study on numerical benchmarks. In E. Corchado et al., editor, *Innovations in Hybrid Intelligent Systems 2007*, pages 255–263. Springer-Verlag, 2007.

[4] Hans J. Bremermann. Chemotaxis and optimization. *J. Franklin Inst*, 297:397–404, 1974.

[5] Carlos A. Coello Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[6] G.W. Corde and D.I. Foreman. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. John Wiley, Hoboken, NJ, 2009.

[7] Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.

[8] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[9] Saber M. Elsayed, Ruhul A. Sarker, and Daryl L. Essam. On an evolutionary approach for constrained optimization problem solving. *Applied soft computing*, 12(0):3208–3227, 2012.

[10] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2005.

[11] Betania Herández-Ocaña, Efrén Mezura-Montes, and Pilar Pozos-Parra. A review of the bacterial foraging algorithm in constrained numerical optimization. In *Proccedings of the Congress on Evolutionary Computation (CEC'2013)*, pages 2695–2702. IEEE, 2013.

[12] Betania Hernández-Ocaña, Ma. Del Pilar Pozos-Parra, and Efrén Mezura-Montes. Stepsize control on the modified bacterial foraging algorithm for constrained numerical optimization. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 25–32, New York, NY, USA, 2014. ACM.

[13] Hsiang-Cheh Huang, Yueh-Hong Chen, and Ajith Abraham. Optimized watermarking using swarm-based bacterial foraging. *Information Hiding and Multimedia Signal Processing*, 1(1):51–58, 2010.

[14] The MathWorks Inc. Friedman. URL http://www.mathworks.com/help/stats/friedman.html, 2015. Accessed 8-05-2015.

[15] D. Karaboga and B. Basturk. powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[16] Alireza Kasaiezadeh, Amir Khajepour, and Steven L. Waslander. Spiral bacterial foraging optimization method: Algorithm, evaluation and convergence analysis. *Engineering Optimization*, 46(4):439–464, 2014.

[17] Borhan Kazimipour, Xiaodong Li, and AK. Qin. A review of population initialization techniques for evolutionary algorithms. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2585–2592, July 2014.

[18] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, UK, 2001.

[19] Dong Hwa Kim, Ajith Abraham, and Jae Hoon Cho. A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Information Sciences*, 177(18):3918–3937, 2007.

[20] Wael Korani. Bacterial foraging oriented by particle swarm optimization strategy for pid tuning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 1823–1826, Atlanta, GA, USA, 2008. ISBN:978-1-60558-131-6.

[21] J.J. Liang, Thomas Philip Runarsson, Efrn Mezura-Montes, Maurice Clerc, P.N. Suganthan, Carlos A. Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, School of EEE Nanyang Technological University, Singapore, September 2006.

[22] Manuel López-Ibáñez, Jéremie Dubois-Lacoste, Thomas Sttzle, and Mauro Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[23] R. Mallipeddi and P.N. Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. Technical report, School of EEE Nanyang Technological University, Singapore, April 2010.

[24] E. Mezura-Montes, J. Velazquez-Reyes, and C.A. Coello Coello. Modified differential evolution for constrained optimization. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 25–32, 2006.

[25] Efrén Mezura-Montes, editor. *Constraint-Handling in Evolutionary Optimization*, volume 198 of *Studies in Computational Intelligence*. Springer-Verlag, 2009.

[26] Efrén Mezura-Montes and Omar Cetina-Domínguez. Empirical analysis of a modified artificial bee colony for constrained numerical optimization. *Applied Mathematics and Computation*, 218(18):10943 – 10973, 2012.

[27] Efrén Mezura-Montes and Carlos A. Coello Coello. Identifying on-line behavior and sources of difficulty in constrained optimization using evolutionary algorithms. In *In Proceedings of the IEEE Congress on Evolutionary Computation 2005 (CEC2005)*, pages 1477–1484, Edinburgh, UK., 2005. IEEE Press.

[28] Efrén Mezura-Montes and Carlos A. Coello Coello. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[29] Efren Mezura-Montes and Jorge Isacc Flores-Mendoza. Improved particle swarm optimization in constrained numerical search spaces. In Raymond Chiong, editor, *Nature-Inspired Algorithms for Optimization*, volume 193, pages 299–332. Springer-Verlag, Studies in Computational Intelligence Series, 2009, ISBN: 978-3-540-72963-1., 2009.

[30] Efrén Mezura-Montes and Betania Hernández-Ocaña. Modified bacterial foraging optimization for engineering design. In Cihan H. Dagli and et al., editors, *Proceedings of*

the Artificial Neural Networks in Enginnering Conference (ANNIE'2009), volume 19 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 357–364, St. Louis, MO, USA, November 2009. ASME Press.

[31] Efrén Mezura-Montes and Elyar A. López-Davila. Adaptation and local search in the modified bacterial foraging algorithm for constrained optimization. In *Proccedings of the IEEE Congress on Evolutionary Computation 2012*, pages 497–504. ISBN:978-1-4673-1508-1, 2012.

[32] Efrén Mezura-Montes, Edgar Alfredo Portilla-Flores, and Betania Hernández-Ocaña. Optimum synthesis of a four-bar mechanism using the modified bacterial foraging algorithm. *International Journal of Systems Science*, 2013. DOI:10.1080/00207721.2012.745023.

[33] Ferrante Neri and Carlos Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[34] Ben Niu, Yan Fan, Han Xiao, and Bing Xue. Bacterial foraging based approaches to portfolio optimization with liquidity risk. *Neurocomputing*, 98(0):90 – 100, 2012.

[35] Nicole Pandit, Anshul Tripathi, Shashikala Tapaswi, and Manjaree Pandit. An improved bacterial foraging algorithm for combined static/dynamic enviromental economic dispatch. *Applied Soft Computing*, 0(12):3500–3513, 2012.

[36] Kevin M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67, 2002.

[37] M. J. D. Powell. Algorithms for Nonlinear Constraints that use Lagrangian Functions. *Mathematical Programming*, 14:224–248, 1978.

[38] P. Praveena, K. Vaisakh, and S. Rama Mohana Rao. A bacterial foraging and pso-de algorithm for solving dynamic economic dispatch problem with valve-point effects. In *First International Conference on Integrated Intelligent Computing*, pages 227–232. IEEE, 2010.

[39] Kukkonen S. and Lampinen J. Constrained real-parameter optimization with generalized differential evolution. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 207–214, Vancouver, BC, Canada, July 2006. IEEE.

[40] Ahmed Yousuf Saber. Economic dispatch using particle swarm optimization with bacterial foraging effect. *Electrical Power and Energy Systems*, 0(34):38–46, 2012.

[41] Patrick Siarry and Zbigniew Michalewicz, editors. *Advances in Metaheuristic Methods for Hard Optimization*. Springer, Berlin, 2008. ISBN 978-3-540-72959-4.

[42] H.K. Singh, T. Ray, and W. Smith. Performance of infeasibility empowered memetic algorithm for cec 2010 constrained optimization problems. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, July 2010.

[43] T. Takahama and S. Sakai. Constrained optimization by e constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–9, July 2010.

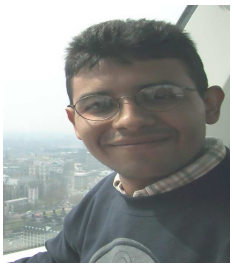
[44] B. Tessema and G.G. Yen. An adaptive penalty formulation for constrained evolutionary optimization. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(3):565–578, May 2009.

[45] K. Vaisakh, P. Praveena, S. Rama Mohana Rao, and Kala Meah. Solving dynamic economic dispatch problem with security constraints using bacterial foraging pso-de algorithm. *Electrical Power and Energy Systems*, 0(39):56–67, 2012.

[46] Yudong Zhang, Lenan Wu, and Shuihua Wang. Bacterial foraging optimization based neural network for short-term load forecasting. *Computational Information Systems*, 6(7):2099–2105, 2010.

Betania Hernández Ocaña graduated with honors from the MsC on Applied Computing at LANIA A.C. in Xalapa, Veracruz, México, and is currently pursuing Ph.D studies at the Juárez Autonomous University of Tabasco, México. Her research interests are nature-inspired algorithms to solve complex optimization problems.

Ma. Del Pilar Pozos-Parra received her M.S. (1998) and Ph.D (2002) both in Computer Sciences (Knowledge Representation and Formalization of Reasoning) from SUPAERO College, France. She is currently a Professor in the Department of Informatics and Systems, University of Tabasco, Mexico. Her research interests include classical logic, belief merging and revision, automated reasoning and meta-heuristic algorithms.



Efrén Mezura Montes is a full-time researcher at the Artificial Intelligence Research Center, University of Veracruz, México, His research interests are the design, analysis and application of bio-inspired algorithms to solve complex optimization problems. He has published over 100 papers in peer-reviewed journals and conferences. He also has one edited book and several book chapters published by international publishing companies.