Adaptation and Local Search in the Modified Bacterial Foraging Algorithm for Constrained Optimization

Efrén Mezura-Montes, *Member, IEEE*
Laboratorio Nacional de
Informática Avanzada (LANIA) A.C.
Rébsamen 80, Centro, Xalapa, Veracruz, 91000, MEXICO
Email: emezura@lania.mx

Elyar A. López-Dávila
Instituto Tecnológico de Orizaba
Departamento de Ingeniería en Sistemas Computacionales
Avenida Oriente-9 852, Colonia Emiliano Zapata,
Orizaba, Veracruz, 94320, MEXICO
Email: elyar.rt.spdr@hotmail.com

***Abstract*—This paper presents the addition of an adaptive stepsize value and a local search operator to the modified bacterial foraging algorithm (MBFOA) to solve constrained optimization problems. The adaptive stepsize is used in the chemotactic loop for each bacterium to promote a suitable sampling of solutions and the local search operator aims to promote a better trade-off between exploration and exploitation during the search. Three MBFOA variants, the original one, another with only the adaptive stepsize and a third one with both, the adaptive stepsize and also the local search operator are tested on a set of well-known benchmark problems. Furthermore, the most competitive variant is compared against some representative nature-inspired algorithms of the state-of-the-art. The results obtained provide evidence on the utility of each added mechanism, while the overall performance of the approach makes it a viable option to solve constrained optimization problems.**

I. Introduction

Among the different sources of difficulty found in numerical optimization problems there is the presence of constraints [1]. Nonetheless, nature-inspired algorithms (NIAs) have been successfully applied to solve complex optimization problems [2], [3]. To deal with a constrained search space, NIAs, which in their original versions were designed to sample unconstrained search spaces, require a constraint-handling technique to bias the algorithm to the feasible region of such search space [4].

In a recent review of the state-of-the-art in constraint-handling in nature-inspired optimization, the importance of the search algorithm has been highlighted [5], as well as the fact that there are some NIAs which has been barely used to solve constrained numerical optimization problems (CNOPs). This is the case of the Bacterial Foraging Optimization Algorithm (BFOA), originally proposed by Passino [6] to solve unconstrained numerical optimization problems. BFOA emulates bacterial behavior when looking for regions with a high amount of nutrients while avoiding noxious substances. More specifically, bacteria follow the steps below:

1) Bacteria are distributed at random in the map of nutrients.
2) By means of a tumble-swim movement, some bacteria locate in high-nutrient regions and they will reproduce. In contrast, those located in regions with noxious substances or low amounts of nutrients will die and disperse, respectively.
3) Bacteria located in high-nutrient regions will attract other bacteria by using chemical attractors.
4) Bacteria are located in the highest-nutrient region.
5) Finally, bacteria disperse to seek new nutrient regions.

From the aforementioned steps followed by bacteria, four processes are identified: (1) chemotaxis, (2) swarming, (3), reproduction and (4) elimination-dispersal.

BFOA has been successfully applied in several domains, such as computer vision [7], [8], rfid networks [9], mechanical design [10], stock market prediction [11], unconstrained high-dimensional optimization problems [12], multimodal optimization [13]. There is also a micro version of BFOA for unconstrained global optimization [14]. Furthermore, there are studies on the effect of the stepsize value in unconstrained search spaces [15], [16]. However, the research efforts regarding BFOA specifically to solve CNOPs are still scarce [17] and no local search operators have been considered.

BFOA in its original version requires several parameters to be defined by the user and some of them are not easy to be fine-tuned. Based on this shortcoming, in [10] the so-called Modified Bacterial Foraging Optimization Algorithm (MBFOA) was proposed. Besides decreasing the number of input parameters, MBFOA included a simplification of the swarming, reproduction and elimination-dispersal processes. The goal was to get a simpler variant of the algorithm. MBFOA adopted the feasibility rules as a constraint-handling technique [5] and has been used to solve some engineering design problems [10].

The little research on BFOA to solve CNOPs combined with the interest to extend the knowledge about MBFOA are the main motivation of the research presented in this paper, in which an adaptive stepsize control and a local search operator are added to MBFOA so as to improve its performance when solving CNOPs.

The paper is organized as follows: Section II formally states the CNOP, while in Section III MBFOA is briefly described.

After that, Section IV presents the adaptive adjustment for the stepsize and the local search operator adopted. Section V describes the experimental design, the obtained results and their corresponding discussion. Finally, Section VI includes the conclusions of this work and its future work.

II. Statement of the problem

A CNOP, without loss of generality, can be defined as to:

Find $\vec{x}$ which minimizes

$$f(\vec{x}) \tag{1}$$

subject to

$$g_i(\vec{x}) \leq 0, \;\; i = 1, \ldots, m \tag{2}$$

$$h_j(\vec{x}) = 0, \;\; j = 1, \ldots, p \tag{3}$$

where $\vec{x} \in \mathbb{R}^n$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots, x_n]^T$, m is the number of inequality constraints, and p is the number of equality constraints. Each x_k, $k = 1, ..., n$ is bounded by lower and upper limits $L_k \leq x_k \leq U_k$ which define the search space $\mathcal{S}$. $\mathcal{F}$ comprises the set of all solutions which satisfy the constraints of the problems and it is called the feasible region. Both, the objective function and the constraints can be linear or nonlinear. To handle equality constraints in NIAs, they are usually transformed into inequality constraints as follows [18]: $|h_j(\vec{x})| - \varepsilon \leq 0$, where ε is the tolerance allowed (a very small value). $\varepsilon =$ 1E-4 was used in this paper.

III. MBFOA

MBFOA, as other swarm intelligence algorithms, is inspired on social and cooperative behaviors observed in simple living organisms [3]. As the original BFOA, MBFOA is also based on the four processes mentioned in Section I, but combining the chemotaxis and swarming into one loop and simplifying the reproduction and elimination-dispersal.

MBFOA uses real-encoding to represent a solution, which is called *bacterium* and is represented by its position as $\theta^i(j, G) = \vec{x}_i$, where i is the number of bacterium, j represents the chemotactic loop number, and G is the cycle number of the algorithm.

The process starts with an initial swarm of S_b bacteria generated at random with a uniform distribution $\theta^i(j, G), \forall_i = 1, \ldots, S_b$. After that, each bacterium $\theta^i(j, G)$ will perform its chemotactic loop, which consists on either a tumble-swim movement (taken from BFOA) or a swarming movement. The first one looks to explore the search space while the second one aims to promote exploitation of promising regions already found. After that, the reproduction step takes place by replicating those better bacteria and, finally, the elimination-dispersal step eliminates those worst bacteria so as to let the new ones to be part of the swarm and keep its size constant at each cycle.

The tumble-swim operator consists on a search direction generated at random (tumble) and a movement in such direction by a bacterium (swim). If the new position is better than the previous one, another swim in the same direction will be carried out. Otherwise, a new tumble is computed so as to find a promising search direction. The tumble, i.e. search direction chosen at random is computed as indicated in Equation 4:

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}} \tag{4}$$

where $\Delta(i) \in \mathbb{R}^n$ is a vector generated at random with the same dimensionality of a solution in the CNOP and with each one of its elements within the range:$[-1, 1]$. After the tumble calculation, the swim generates a new position $\theta^i(j+1, G)$ for bacterium $\theta^i(j, G)$ as showed in Equation 5.

$$\theta^i(j+1, G) = \theta^i(j, G) + C_{new} \cdot \phi(i) \tag{5}$$

where $C_{new} \in \mathbb{R}^n$ is the stepsize vector, where each one of its elements is computed by using Equation 6:

$$C_{new} = R \cdot (\Delta x_k / \sqrt{n}) \tag{6}$$

where Δx_k is the difference between upper and lower limits for variable x_k, i.e., $\Delta x_k = (U_k - L_k)$ and R is a user-defined parameter. Unlike the $\Delta(i)$ vector which is generated at random each time a tumble is required, the stepsize vector remains fixed during all the execution of MBFOA.

The swarming operator works on some iterations within the chemotactic loop and replaces the tumble-swim operator in such iterations. The operator emulates the attraction exerted by those bacteria located in high-nutrient regions. In this MBFOA implementation the swarming operator is used at half and at the end of the chemotactic loop. The operator is detailed in Equation 7:

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

where the new position $\theta^i(j+1, G)$ for bacterium i is computed by its current position $\theta^i(j, G)$ and the position of the current best bacterium in the swarm so far at cycle G ($\theta^B(G)$) scaled by a factor β defined by the user.

MBFOA adopted the feasibility rules [19], one of the most known constraint-handling techniques [5], to deal with the constraints of the problem. The rules work as criteria in the chemotactic loop, in the reproduction step and in the elimination of the worst bacterium in the swarm.

The parameters used in MBFOA are the number of bacteria in the swarm S_b, the number of cycles (generations) $GMAX$, the number of cycles for the chemotactic loop N_c, the number of best bacteria for reproduction S_r, scaling factor for the swarming operator β and the percentage for the initial stepsize calculation R.

The complete pseudocode of MBFOA is presented in Fig. 1.

1: Initialize input parameters
2: Create a random initial swarm of bacteria $\theta^i(0,0), i = 1\ldots, S_b$
3: Evaluate $f(\theta^i(0,0))$, $g_j(\theta^i(0,0)), \forall_j, j = 1,\ldots,m$, $h_k(\theta^i(0,0)), \forall_k, k = 1,\ldots,p$, $\forall_i, i = 1,\ldots,S_b$.
4: **for** $G = 1$ to $GMAX$ **do**
5: **for** $i = 1$ to S_b **do**
6: **for** $j = 1$ to N_c **do**
7: Perform the chemotactic step (Equation 5 or 7) for bacterium $\theta^i(j,G)$
8: **end for**
9: **end for**
10: Perform the reproduction step by eliminating the worst S_r bacteria (based on feasibility criteria) and duplicating the best S_r bacteria
11: Eliminate the worst bacterium (based on feasibility criteria) $\theta^w(j,G)$ in the current population and generate one at random
12: **end for**

Fig. 1: MBFOA algorithm

IV. Modifications to MBFOA

With the aim to enhance MBFOA's performance when solving CNOPs, the stepsize used in the tumble-swim operator (Equation 6) is adapted based on the behavior of the swarm so as to favor convergence. Furthermore, to improve the local search capabilities of the approach, a local search operator, based on the Hooke-Jeeves mathematical programming method, was applied to some bacteria in the swarm at certain cycles. The two modifications are explained in the following sections.

A. Adaptive Stepsize

One of the main changes promoted by MBFOA with respect to BFOA was to keep the user from defining a stepsize value [10] (see Equation 6). However, this value remained fixed during the search. With the aim to provide bacteria with the ability to increase or decrease the stepsize value based on their behavior, an adaptive control for the stepsize inspired in the "$1/5$-rule" from evolution strategies [20] is proposed in this paper. One desired behavior in an artificial bacterium in MBFOA is its improvement in each step of the tumble-swim operator within the chemotactic loop. In this way, a successful movement is counted when a bacterium improves its solution by an application of the tumble-swim or swarming operator. Based on those values, the successful rate (SR) is computed as the average number of successful movements over all movements made by all bacteria in a single cycle. The SR value is then used to update the stepsize value at cycle G ($C_{new}(G)$) as detailed in Equation 8

$$C_{new}(G) = \begin{cases} C_{new}(G-1) \cdot SSA & \text{if SR} < 0.2 \\ C_{new}(G-1) \cdot (1/SSA) & \text{otherwise} \end{cases} \quad (8)$$

1: Initialize input parameters
2: Perform an exploratory move with $\theta^{(k)}$ as the base point, let the result be θ
3: **if** Exploratory move is a success **then**
4: $\theta^{(k+1)} = \theta$, and go to step 13
5: **else**
6: Go to step 8
7: **end if**
8: **if** $\|\Delta\| < \epsilon$ **then**
9: **Terminate**
10: **else**
11: set $\Delta_i = \Delta_i/\alpha$ for $i = 1, 2, \ldots, N$, and go to step 2
12: **end if**
13: Set $k = k + 1$ and perform the pattern move: $\theta_p^{(k+1)} = \theta^{(k)} + (\theta^{(k)} - \theta^{(k-1)})$
14: Perform another exploratory move using $\theta_p^{(k+1)}$ as the base point, let the result be $\theta^{(k+1)}$
15: **if** $f(\theta^{(k+1)})$is better than$f(\theta^{(k)})$(based on the feasibility rules) **then**
16: Go to step 13
17: **else**
18: Go to step 8
19: **end if**

Fig. 2: Hooke-Jeeves pattern search method

where SSA is an adjustment parameter to control the level of change for the stepsize value. Also inspired by the "$1/5$-rule" [20], a value of 0.2 for SR was adopted in this work. The initial stepsize value is computed as indicated in Equation 9

$$C_{new}(0) = R \cdot (U_k - L_k) \quad (9)$$

B. Local Search

To improve the fine-grain search capability of MBFOA, an adaptation of a well-known mathematical programming method named *Hooke-Jeeves Pattern Search Method* [21] was employed as a local search method at certain periods of time during the search. This method was chosen because it does not require gradient calculations and its combination with the constraint-handling technique used in MBFOA is straightforward, as it will be detailed later in this paper. Furthermore, its implementation and integration with MBFOA do not require complicated programming efforts.

The Hooke-Jeeves Pattern Search Method creates search directions iteratively by combining exploratory moves and heuristic pattern moves. The exploratory move looks to sample the vicinity of a solutions and choose the best solution within it. Such solutions and the original one are used to make a pattern move.

The pseudocode of the local search method is detailed in Fig. 2. The exploratory movement used by such method is also detailed in Fig. 3.

The initial point required by the LS method ($\theta^{(0)}$) is indeed one bacterium in the current swarm, the increments for each variable of the optimization problem $\Delta_k(k = 1, 2, \ldots, n)$ are

Input: θ^c
Output: θ
1: set $\theta = \theta^c$, $i = 1$
2: Calculate $f = f(\theta), f^+ = f(\theta_i + \Delta_i)$ and $f^- = f(\theta_i - \Delta_i)$
3: Find f_{min} = the best one from(f, f^+, f^-) based on the feasibility rules.
4: Set θ corresponds to f_{min}
5: **if** $i = N$ **then**
6: θ is the result, and go to step 10
7: **else**
8: set $i = i + 1$ and go to step 2
9: **end if**
10: **if** $\theta \neq \theta^c$ **then**
11: **success**
12: **else**
13: **failure**
14: **end if**

Fig. 3: Exploratory move

defined by using a percentage of the difference between its upper and lower limits as follows: $0.5 \cdot (U_k - L_k), (k = 1, 2, \ldots, n)$, the reduction factor α was fixed to 2 and the termination criteria ϵ was set to 1E$-$8.

In a recent literature review on memetic algorithms [22], some key aspects, specially for constrained optimization, were remarked: (1) the integration of the algorithmic components to deal with the constraints, (2) the selection of the individuals to apply the LS, and (3) the LS application frequency. In this work, according with the tendency observed in [22], the integration of the components to deal with the constrained search space is made by the adoption of the set of feasibility rules as a criterion to choose solutions in the LS, i.e., feasible solutions with a better value of the objective function are preferred over infeasible ones and, among infeasible solutions, those with a lower sum of constraint violation are preferred. The LS search will be applied to the best 10% of the swarm size S_b, based on the feasibility rules. Finally, the LS will be applied every 25 cycles.

With the aim to favor convergence, the reproduction step, which replaces part of the swarm, and the elimination-dispersal step are only applied every 30 cycles.

The pseudocode of the MBFOA variant with adaptive stepsize and LS, called MBFOA-AS-LS is presented in Fig. 4.

V. Results and discussion

Two experiments were carried out to analyze the performance of MBFOA-AS-LS: (1) A comparison against the original MBFOA and MBFOA with only adaptive stepsize (MBFOA-AS) so as to determine the importance of each new element in MBFOA-AS-LS, and (2) A comparison against some state-of-the-art approaches in nature-inspired constrained optimization. Thirteen well-known test problems were used in the comparisons proposed. The features of the problems are

1: Initialize input parameters
2: Create a random initial swarm of bacteria $\theta^i(0,0), i = 1 \ldots, S_b$
3: Evaluate $f(\theta^i(0,0))$, $g_j(\theta^i(0,0)), \forall_j, j = 1, \ldots, m$, $h_k(\theta^i(0,0)), \forall_k, k = 1, \ldots, p$, $\forall_i, i = 1, \ldots, S_b$.
4: **for** $G = 1$ to $GMAX$ **do**
5: **for** $i = 1$ to S_b **do**
6: **for** $j = 1$ to N_c **do**
7: Perform the chemotactic step (tumble-swim, tumble-tumble or swarming) for bacterium $\theta^i(j, G)$
8: **end for**
9: **end for**
10: Perform the stepsize adjustment (Equation 8)
11: **if** $G \bmod 25 = 0$ **then**
12: Apply the LS operator for the 10% best bacteria in the swarm and replace the 10% worst bacteria with those obtained by the LS
13: **end if**
14: **if** $G \bmod 30 = 0$ **then**
15: Perform the reproduction step by eliminating the S_r worst S_r bacteria (based on the feasibility rules) and duplicating the S_r) best bacteria
16: Eliminate the worst bacterium $\theta^w(j, G)$ in the current population, based on feasibility criteria and replace it with a one generated at random
17: **end if**
18: **end for**

Fig. 4: MBFOA-AS-LS algorithm

TABLE I
Main features of test problems. n indicates the dimensionality of the problem, ρ is the estimated size of the feasible region with respect to the whole search space, LI and NI are the number of linear and nonlinear inequality constraints respectively and LE and NE are the number of linear and nonlinear equality constraints. Finally A indicates the number of active constraints.

P	n	Function	ρ	LI	NI	LE	NE	A
g01	13	quadratic	0.0111%	9	0	0	0	6
g02	20	nonlinear	99.9971%	0	2	0	0	1
g03	10	polynomial	0.0000%	0	0	0	1	1
g04	5	quadratic	52.1230%	0	6	0	0	2
g05	4	cubic	0.0000%	2	0	0	3	3
g06	2	cubic	0.0066%	0	2	0	0	2
g07	10	quadratic	0.0003%	3	5	0	0	6
g08	2	nonlinear	0.8560%	0	2	0	0	0
g09	7	polynomial	0.5121%	0	4	0	0	2
g10	8	linear	0.0010%	3	3	0	0	6
g11	2	quadratic	0.0000%	0	0	0	1	1
g12	3	quadratic	4.7713%	0	1	0	0	0
g13	5	nonlinear	0.0000%	0	0	0	3	3

summarized in Table I. Due to space restrictions the detailed definitions for the test problems were not included, but the reader can find them in [23].

In both experiments, 30 independent runs per each MBFOA variant per each test problem were computed. The parameters employed, obtained after several preliminary tests, were the following for the three MBFOA variants: $S_b = 50$, $N_c = 50$.

For MBFOA $S_r = 25$, while for MBFOA-AS and MBFOA-AS-LS $S_r = 2$.

The number of cycles was GMAX$= 80$ for MBFOA and MBFOA-AS (approximately 200,000 total evaluations per independent run). For MBFOA-AS-LS the GMAX value was reduced to 65 due to the variable number of evaluations made by the LS. The aim was twofold: (1) to keep MBFOA-AS-LS from exceeding the 200,000 evaluations and (2) to test if this variant is able to find better results with a lower number of evaluations with respect to the other two MBFOA variants.

The value for the R parameter in the original MBFOA was set as follows: the value used were 0.5 for problems g01 and g02, and a value of 0.015 was used for the remaining problems. Moreover, the β value was 0.6 for problems g03, g04, g08, g09, g11, and g12 while a value of 0.005 was used for problems g01, g02, g05, g06, g07, g10, and g13.

Regarding MBFOA-AS and MBFOA-AS-LS, the value for R was fixed to 0.65 and SSA was assigned a value of 0.817 for problems g01-g03, g07-g09, g11 and g12, while a value of 0.717 was set for problems g04, g05, g06, g10 and g13. Finally, the β value was set to 0.001 for problems g01, g04, g05, g06, g07, g08, g10, and g13 and 0.9 for problems g02, g03, g09, g11, and g12. It is worth mentioning that the parameter selection for the three MBFOA variants was made with the aim to favor their best overall performance in the testbed adopted.

The results in this first experiment are discussed based on quality (best result obtained so far), consistency (best average result with a lower standard deviation value) and computational cost (average number of evaluations per run).

The results obtained in the first experiment are summarized in Table II, where the three MBFOA variants were able to find feasible solutions in all independent runs per test problems, showing their ability to find the feasible region in different landscapes. The 95%-confidence Wilcoxon test was applied to the samples of runs regarding the final results obtained by each algorithm so as to get statistical support to the findings. Based on the Wilcoxon test results, there were no significant differences among the results obtained by the three MBFOA variants in problems g03, g08, g09 and g12. However, in those four test problems MBFOA-AS-LS required the lowest average number of evaluations to reach its best results.

On the other hand, the results obtained in problems g01, g04, g05 and g06 were significantly different among the three MBFOA variants. MBFOA-AS-LS was better (based on quality and consistency) in those four problems with respect to MBFOA. Moreover, in test problems g01 and g05, MBFOA-AS-LS provided better quality and consistency of results with respect to MBFOA-AS. Both modified variants (MBFOA-AS and MBFOA-AS-LS) provided almost similar quality and consistent results in problems g04 and g06. It is worth remarking that, in those four test problems MBFOA-AS-LS required less evaluations to reach the best results.

In test problems g02, g07, g10, g11, and g13 there were significant differences between MBFOA-AS and the original MBFOA and also between MBFOA-AS-LS and the original MBFOA. In test problems g02, g07, g10, and g13 the modified variants were clearly better (based on quality and consistency) than the original BFOA. In problem g11 a slightly more consistent performance was provided by MBFOA. In this same subset of test problems (g02, g07, g10, g11, and g13), no significant differences were obtained between the two modified variants (MBFOA-AS and MBFOA-AS-LS) regarding final results. However, MBFOA-AS-LS required less evaluations to reach its best results.

After analyzing the performance of MBFOA-AS-LS with respect to its elements and previous variants, its final results are compared against some highly competitive nature-inspired algorithms for constrained optimization in Table III. It can be seen that MBFOA-AS-LS provided quality results in problems g01, g03, g04, g05, g06, g08, g11, and g12. Regarding its consistency, it was competitive, with respect to the algorithms compared, in problems g04, g06, g08, g11 and g12. In the remaining ones, MBFOA-AS-LS was trapped in local optimum solutions, specially in high-dimensional test problems (g01 and g02).

From the results in the two experiments carried out in the proposed set of benchmark problems, the following findings are remarked:

- The three MBFOA variants showed a strong ability to find the feasible region of the search space in the testbed adopted in this paper.
- In contrast, the three MBFOA variants were sensitive to two of their parameters (β for the three variants, SSA for MBFOA-AS and MBFOA-LS and R for the original MBFOA). Particularly, the β parameter required almost opposite values for different test problems.
- There is not a clear pattern in the relationship between a high/low β value and the features of the CNOP solved. A high β value means that the new location of a bacterium (derived from the swarming operator) will be closer to the position of the best bacterium so far. In contrast, a low β value locates the new position of the bacterium closer to its current position, but oriented to the location of the best bacterium in the swarm. This issue requires further analysis and it is part of the future work.
- The adaptive stepsize mechanism provided a considerable improvement in MBFOA's ability to reach competitive results when solving CNOPs.
- Limiting the duplication of those best bacteria and the corresponding elimination of the worst bacteria in the swarm (i.e. S_r=2) had a beneficial effect on the performance of the two new MBFOA variants (with or without LS). The decreasing effect on the stepsize values limits the exploration capability of a bacterium, mostly in late cycles of the process. Therefore, the massive duplication promoted by the original MBFOA might cause premature convergence in MBFOA-AS and MBFOA-AS-LS.
- The improvement observed in MBFOA-AS-LS was mainly in test problems with more than one equality constraints (g05 and g13) and problems with a high dimensionality (g01 and g02)

TABLE II

STATISTICAL RESULTS OF THE COMPARISON BETWEEN MBFOA, MBFOA-AS AND MBFOA-AS-LS. A RESULT IN BOLDFACE INDICATES A BETTER RESULT. A $\surd$ MEANS A SIGNIFICANT DIFFERENCE BETWEEN MBFOA AND MBFOA-AS, AN X MEANS A SIGNIFICANT DIFFERENCE BETWEEN MBFOA AND MBFOA-AS-LS, AND AN $*$ MEANS A SIGNIFICANT DIFFERENCE BETWEEN MBFOA-AS AND MBFOA-AS-LS

Function/ Optimal		**Algorithms**			Wilcoxon rank-sum test
		MBFOA	**MBFOA-AS**	**MBFOA-AS-LS**	
g01	Best	-13.589	-10.504	**-15.000**	$\surd$ X *
-15.000	Mean	-10.895	-8.710	**-14.685**	
	Worst	-7.513	-7.147	**-12.833**	
	Std. Dev.	1.43E+00	8.46E-01	**7.28E-01**	
	FE's Avg.	200130	200052	**184957.93**	
g02	Best	-0.557452	-0.775588	**-0.792584**	$\surd$ X
-0.803619	Mean	-0.399145	-0.574781	**-0.622200**	
	Worst	-0.295688	-0.312250	**-0.409785**	
	Std. Dev.	**7.08E-02**	1.36E-01	1.09E-01	
	FE's Avg.	200130	200052	**185786.03**	
g03	Best	**-1.000**	**-1.000**	**-1.000**	
-1.000	Mean	**-1.000**	-0.923	**-1.000**	
	Worst	**-1.000**	-0.189	-0.999	
	Std. Dev.	**7.08E-02**	1.36E-01	1.09E-01	
	FE's Avg.	200130	200052	**178065.7**	
g04	Best	30664.348	**-30665.539**	**-30665.539**	$\surd$ X *
-30665.539	Mean	-30659.449	**-30665.539**	**-30665.539**	
	Worst	-30546.472	**-30665.539**	**-30665.539**	
	Std. Dev.	2.13E+01	2.22E-11	**1.48E-11**	
	FE's Avg.	200130	200052	**167692.16**	
g05	Best	5126.727	**5126.498**	**5126.498**	$\surd$ X *
5126.497	Mean	5292.677	5126.819	**5126.627**	
	Worst	6018.415	5127.565	**5127.434**	
	Std. Dev.	2.13E+02	3.64E-01	**2.54E-01**	
	FE's Avg.	200130	200052	**186285.46**	
g06	Best	-6961.401	**-6961.814**	**-6961.814**	$\surd$ X *
-6961.814	Mean	-6942.298	**-6961.814**	**-6961.814**	
	Worst	-6423.888	**-6961.814**	**-6961.814**	
	Std. Dev.	9.79E+01	**4.66E-07**	3.14E-06	
	FE's Avg.	200130	200052	**169011.03**	
g07	Best	24.584	**24.339**	24.349	$\surd$ X
24.306	Mean	24.912	24.530	**24.461**	
	Worst	25.539	24.886	**24.623**	
	Std. Dev.	1.96E-01	1.38E-01	**8.14E-02**	
	FE's Avg.	200130	200052	**179165.4**	
g08	Best	**-0.095825**	**-0.095825**	**-0.095825**	
-0.095825	Mean	**-0.095825**	**-0.095825**	**-0.095825**	
	Worst	-0.095824	**-0.095825**	**-0.095825**	
	Std. Dev.	1.83E-07	**4.23E-17**	**4.23E-17**	
	FE's Avg.	200130	200052	**164683.4**	
g09	Best	680.636	**680.631**	680.633	
-680.630	Mean	680.754	**680.684**	680.690	
	Worst	681.086	680.912	**680.837**	
	Std. Dev.	1.34E-01	7.75E-02	**6.61E-02**	
	FE's Avg.	200130	200052	**171547.6**	
g10	Best	7095.584	**7050.761**	7051.648	$\surd$ X
7049.248	Mean	7237.317	7079.940	**7077.555**	
	Worst	8003.925	7150.168	**7142.653**	
	Std. Dev.	1.75E+02	2.58E+01	**2.5E+01**	
	FE's Avg.	200130	200052	**178213.63**	
g11	Best	**0.75**	**0.75**	**0.75**	$\surd$ X
0.75	Mean	**0.75**	**0.75**	**0.75**	
	Worst	**0.75**	**0.75**	**0.75**	
	Std. Dev.	**4.98E-07**	8.95E-05	3.46E-06	
	FE's Avg.	200130	200052	**164794.86**	
g12	Best	**-1.000**	**-1.000**	**-1.000**	
-1.000	Mean	**-1.000**	**-1.000**	**-1.000**	
	Worst	**-1.000**	**-1.000**	**-1.000**	
	Std. Dev.	**0.0E+00**	**0.0E+00**	**0.0E+00**	
	FE's Avg.	200130	200052	**165164**	
g13	Best	0.057703	0.054659	**0.054063**	$\surd$ X
0.053942	Mean	1.341246	**0.137253**	0.152251	
	Worst	16.231837	0.471096	**0.448168**	
	Std. Dev.	2.93E+00	**9.54E-02**	1.06E-01	
	FE's Avg.	200130	200052	**184875.2**	

TABLE III

STATISTICAL RESULTS (B: BEST, M: MEAN, W: WORST, SD: STANDARD DEVIATION) OBTAINED BY MBFOA-AS-LS WITH RESPECT TO THOSE PROVIDED BY STATE-OF-THE-ART APPROACHES ON THIRTEEN BENCHMARK PROBLEMS. VALUES IN **boldface** MEAN THAT THE GLOBAL OPTIMUM OR BEST KNOW SOLUTION WAS REACHED, VALUES IN *italics* MEAN THAT THE OBTAINED RESULT IS BETTER (BUT NOT THE OPTIMAL OR BEST KNOWN) WITH RESPECT TO THE APPROACHES COMPARED.

Problem BKS	Stat	LCA [24]	EXDE [25]	AIS [26]	A-DDE [27]	DSS-MDE [28]	DECV [29]	MBFOA-AS-LS
g01	B	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**
-15.000	M	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**	-14.855	-14.685
	W	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**	-13.000	-12.833
	SD	0	NA	0	7.00E-06	1.30E-10	4.59E-01	7.28E-01
g02	B	-0.803616	NA	-0.803611	-0.803605	**-0.803619**	-0.704009	-0.792584
-0.803619	M	-0.801793	NA	*-0.803183*	-0.771090	-0.786970	-0.569458	-0.6222002
	W	-0.792602	NA	*-0.802295*	-0.609853	-0.728531	-0.238203	-0.409785
	SD	*3.8E-03*	NA	1.501E-02	3.66E-02	1.50E-02	9.51E-02	1.09E-01
g03	B	**-1.000**	**-1.025**	**-1.000**	**-1.000**	**-1.005**	-0.461	**-1.000**
-1.000	M	**-1.000**	**-1.025**	-0.995	**-1.000**	**-1.005**	-0.134	**-1.000**
	W	-0.999	**-1.025**	-0.991	**-1.000**	**-1.005**	-0.002	-.999
	SD	1.8E-04	NA	6.22E-02	*9.30E-12*	1.90E-08	1.17E-01	1.07E-04
g04	B	**-30665.539**	-31025.600	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
-30665.539	M	**-30665.539**	-31025.600	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
	W	**-30665.539**	-31025.600	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
	SD	1.09E-11	NA	0	3.20E-13	2.70E-11	1.56E-06	1.48E-11
g05	B	**5126.497**	**5126.484**	**5126.497**	**5126.497**	**5126.497**	**5126.497**	5126.498
5126.497	M	**5126.497**	**5126.484**	5127.073	**5126.497**	**5126.497**	**5126.497**	5126.627
	W	**5126.497**	**5126.484**	5127.361	**5126.497**	**5126.497**	**5126.497**	5127.433
	SD	5.07E-13	NA	5.37E-01	2.10E-11	0	0	2.54E-01
g06	B	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**
-6961.814	M	**-6961.814**	**-6961.814**	-6961.811	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**
	W	**-6961.814**	**-6961.814**	-6961.810	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**
	SD	1.85E-12	NA	2.98E-02	2.11E-12	0	0	3.13E-06
g07	B	**24.306**	**24.306**	**24.306**	**24.306**	**24.306**	**24.306**	24.349
24.306	M	**24.306**	**24.306**	24.963	**24.306**	**24.306**	24.794	24.461
	W	**24.306**	24.307	26.101	**24.306**	**24.306**	29.511	24.623
	SD	1.50E-04	NA	7.30E-02	4.20E-05	*7.50E-07*	1.37E+00	8.14E-02
g08	B	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
-0.095825	M	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
	W	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
	SD	2.82E-17	NA	0	9.10E-10	4.00E-17	4.23E-17	4.23E-17
g09	B	**680.630**	**680.630**	**680.630**	**680.630**	**680.630**	**680.630**	680.633
680.630	M	**680.630**	**680.630**	680.917	**680.630**	**680.630**	**680.630**	680.690
	W	**680.630**	**680.630**	681.480	**680.630**	**680.630**	**680.630**	680.837
	SD	1.25E-10	NA	1.05E-01	1.15E-10	*2.90E-13*	3.45E-07	6.61E-02
g10	B	**7049.248**	**7049.248**	7052.183	**7049.248**	**7049.248**	**7049.248**	7051.648
7049.248	M	7049.271	**7049.248**	7052.671	**7049.248**	7049.249	7103.548	7077.555
	W	7049.518	**7049.248**	7153.581	**7049.248**	7049.255	7808.980	7142.653
	SD	4.91E-02	NA	1.37E+00	*3.23E-4*	1.40E-03	1.48e+02	2.50E+01
g11	B	**0.75**	**0.75**	**0.75**	**0.75**	**0.749**	**0.75**	**0.75**
0.75	M	**0.75**	**0.75**	**0.75**	**0.75**	**0.749**	**0.75**	**0.75**
	W	**0.75**	**0.75**	0.76	**0.75**	**0.749**	**0.75**	**0.75**
	SD	1.12E-16	NA	1.22E-03	5.35E-15	0	1.12E-16	3.46E-06
g12	B	**-1.000**	NA	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**
-1.000	M	**-1.000**	NA	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**
	W	**-1.000**	NA	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**
	SD	0	NA	0	4.10E-9	0	0	0
g13	B	**0.053942**	NA	**0.053940**	**0.053942**	**0.053942**	0.059798	0.054063
0.053942	M	**0.053942**	NA	0.054900	0.079627	**0.053942**	0.382401	0.152251
	W	**0.053942**	NA	0.058170	0.438803	**0.053942**	0.999094	0.448168
	SD	3.44E-08	NA	9.00E-04	9.60E-02	*1-00E-13*	2.68E-01	1.06E-01

- The LS operator, based on the Hooke-Jeeves mathematical programming method, reduced MBFOA's number of evaluations to reach high quality results. However, no significant improvement in such final results were observed with respect to the MBFOA variant without LS.
- MBFOA-AS-LS provided competitive results with respect to some state-of-the-art nature-inspired algorithms for CNOPs. However, the consistency is still an issue in its performance.

VI. CONCLUSIONS AND FUTURE WORK

The addition of an adaptive stepsize mechanism and a local search (LS) operator to the modified bacterial foraging algorithm (MBFOA) was presented in this paper with the goal to improve its performance in the solution of CNOPs. The adaptive stepsize was inspired in the adaptive stepsize mechanism of evolution strategies, while the Hooke-Jeeves method was adopted as the LS operator. The stepszie modification was based on the success movements (i.e., solution improvement) by bacteria in their chemotactic loop, where the tumble-swim and swarming operators are used. The LS operator worked intensively in only the 10% best bacteria (based on the feasibility rules) in the population at every 25 cycles. The original MBFOA, a MBFOA variant with only the adaptive stepsize mechanism and another variant with both, the adaptive stepsize and also the LS operator were tested

on thirteen well-known test problems. The results obtained suggested that the adaptive stepsize mechanism significantly improves the performance of MBFOA in numerical constrained search spaces. Furthermore, the LS operator decreases the number of evaluations required by the algorithm to reach competitive results. Such improvement was more remarked in problems with more than one equality constraint and problems with a high dimensionality. Even MBFOA-AS-LS showed a competitive performance against nature-inspired approaches for constrained optimization, it showed a significant sensitivity to its parameters and was trapped in local optimum solutions in some test problems. Those limitations are the initial motivation for the future work, which includes revisiting the adaptive stepsize mechanism to decrease the sensitivity to its parameters and also exploring other LS operators taken from the mathematical programming literature. Finally, a suitable mechanism to deal with parameter β will be designed.

ACKNOWLEDGMENT

The first author acknowledges support from CONACyT through project No. 79809.

REFERENCES

[1] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, 2nd ed. Germany: Springer, 2004.

[2] A. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer Verlag, 2003.

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

[4] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence. Springer-Verlag, 2009, vol. 198.

[5] E. Mezura-Montes and C. A. Coello Coello, "Constraint handling in nature-inspired numerical optimization: Past present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[6] K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.

[7] S. Dasgupta, A. Biswas, S. Das, and A. Abraham, "Automatic circle detection on imges with and adaptive bacterial foraging algorithm," in *2008 Genetic and Evolutionary Computation Conference (GECCO'2008)*. Atlanta, USA: ACM Press, July 2008, pp. 1695–1696.

[8] H. Nguyen and B. Bhanu, "Tracking pedestrians with bacterial foraging optimization swarms," in *2011 Congress on Evolutionary Computation (CEC'2011)*. New Orleans, LA, USA: IEEE Service Center, June 2011, pp. 491–495.

[9] H. Chen, Y. Zhu, and K. Hu, "Multi-colony bacteria foraging optimization with cell-to-cell communication for rfid network planning," *Applied Soft Computing*, vol. 10, no. 2, pp. 539–547, 2010.

[10] E. Mezura-Montes and B. Hernández-Ocaña, "Modified bacterial foraging optimization for engineering design," in *Proceedings of the Artificial Neural Networks in Enginnering Conference (ANNIE'2009)*, ser. Intelligent Engineering Systems Through Artificial Neural Networks, C. H. Dagli and et al., Eds., vol. 19. St. Louis, MO, USA: ASME Press, November 2009, pp. 357–364.

[11] R. Majhi, G. Panda, G. Sahoo, P. Dash, and D. Das, "Stock market prediction of s&p 500 and djia using bacterial foraging optimization technique," in *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*. Singapore: IEEE Press, September 2007, pp. 2569–2575.

[12] Y. Chu, H. Mi, H. Liao, Z. Ji, and Q. Wu, "A fast bacterial swarming algorithm for high-dimensional function optimization," in *2008 Congress on Evolutionary Computation (CEC'2008)*. Hong Kong: IEEE Service Center, June 2008, pp. 3134–3139.

[13] K.M.Bakwad, S.S.Pattniak, B.S.Sohi, S.Devi, S.Gollapudi, C.Sagar, and P. Patra, "Synchronous bacterial foraging optimization for multimodal and high dimensional functions," in *Proceedings of the 2008 International Conference on Computing, Communication and Networking (ICCCN 2008)*, 2008, pp. 1–8.

[14] A. Dasgupta, A. Biswas, S. Das, B. K. Panigrahi, and A. Abraham, "A micro-bacterial foraging algorithm for high-dimensional optimization," in *2009 Congress on Evolutionary Computation (CEC'2009)*. Tronheim, Norway: IEEE Service Center, May 2009, pp. 3134–3139.

[15] S. Dasgupta, S. Das, A. Abraham, and A. Biswas, "Adaptive computational chemotaxis in bacterial foraging optimization: An analysis," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 919–941, 2009.

[16] H. Supriyono and M. Tokhi, "Bacterial foraging algorithm with adaptable chemotactic step size," in *Proceedings of the Second International Conference on : Computational Intelligence, Communication Systems and Networks (CICSyN 2010)*. Liverpool, UK: IEEE Service Center, July 2010, pp. 72–77.

[17] Q. Wang, X.-Z. Gao, and C. Wang, "An Adaptive Bacterial Foraging Algorithm For Constrained Optimization," *International Journal of Innovative Computing Information and Control*, vol. 6, no. 8, pp. 3585–3593, August 2010.

[18] T. P. Runarsson and X. Yao, "Stochastic Ranking for Constrained Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, September 2000.

[19] K. Deb, "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2/4, pp. 311–338, 2000.

[20] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

[21] K. Deb, *Optimization for Engineering Design*. India: Prentice-Hall, 1995.

[22] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, no. 1, pp. 1–14, 2012.

[23] E. Mezura-Montes and C. A. Coello Coello, "A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 1–17, February 2005.

[24] A. Husseinzadeh-Kashan and B. Karimi, "A new algorithm for constrained optimization inspired by the sport league championship," in *2010 Congress on Evolutionary Computation (CEC'2010)*. Barcelona, Spain: IEEE Service Center, July 2010, pp. 487–494.

[25] J. Lampinen, "A Constraint Handling Approach for the Differential Evolution Algorithm," in *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, vol. 2. Piscataway, New Jersey: IEEE Service Center, May 2002, pp. 1468–1473.

[26] K. M. Woldemariam and G. G. Yen, "Constrained optimization using artificial immune system," in *2010 Congress on Evolutionary Computation (CEC'2010)*. Barcelona, Spain: IEEE Service Center, July 2010, pp. 1689–1696.

[27] E. Mezura-Montes and A. G. Palomeque-Ortiz, "Parameter Control in Differential Evolution for Constrained Optimization," in *2009 Congress on Evolutionary Computation (CEC'2009)*. Tronheim, Norway: IEEE Service Center, May 2009, pp. 1375–1382.

[28] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences*, vol. 178, no. 15, pp. 3043–3074, August 1 2008.

[29] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization. an empirical study," *Information Sciences*, vol. 180, no. 22, pp. 4223–4262, 2010.