ACTA UNIVERSITARIA
Multidisciplinary Scientific Journal

| VOLUMEN 29 | 2019 |
ISSN online 2007-9621
http://doi.org/10.15174/au.2019.2335

Bacterial foraging optimization algorithm with mutation to solve constrained problems

Algoritmo basado en el Forrajeo de Bacterias con mutación para resolver problemas con restricciones

Betania Hernández-Ocaña[1], José Hernández-Torruco[1], Oscar Chávez-Bosquez[1]*, Juana Canul-Reich[1], Luis Gerardo Montané-Jiménez[2]

[1] División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco. Carr. Cunduacán-Jalpa Km. 1. Cunduacán, Tabasco, México. CP 86690. +52(914)336-0616. *E-mail: oscar.chavez@ujat.mx.
[2] Facultad de Estadística e Informática, Universidad Veracruzana.
*Corresponding author

Abstract

A simple version of a Swarm Intelligence algorithm called bacterial foraging optimization algorithm with mutation and dynamic stepsize (BFOAM-DS) is proposed. The bacterial foraging algorithm has the ability to explore and exploit the search space through its chemotactic operator. However, premature convergence is a disadvantage. This proposal uses a mutation operator in a swim, similar to evolutionary algorithms, combined with a dynamic stepsize operator to improve its performance and allows a better balance between the exploration and exploitation of the search space. BFOAM-DS was tested in three well-known engineering design optimization problems. Results were analyzed with basic statistics and common measures for nature-inspired constrained optimization problems to evaluate the behavior of the swim with a mutation operator and the dynamic stepsize operator. Results were compared against a previous version of the proposed algorithm to conclude that BFOAM-DS is competitive and better than a previous version of the algorithm.

Keywords: Metaheuristic; mutation operator; dynamic stepsize; engineering problem; performance measures.

Resumen

Se propone una versión simplificada de un algoritmo de Inteligencia Colectiva denominado algoritmo de optimización basado en el forrajeo de bacterias con mutación y tamaño de paso dinámico (BFOAM-DS). Este algoritmo tiene la habilidad de explorar y explotar el espacio de búsqueda mediante su operador quimiotáxico. Sin embargo, la convergencia prematura es una desventaja particular. Esta propuesta implementa un operador de mutación en el nado, similar al utilizado por los algoritmos evolutivos, y un tamaño de paso dinámico para mejorar el desempeño del algoritmo. BFOAM-DS se probó en tres problemas de optimización de diseño ingenieril. Los resultados obtenidos fueron analizados con estadísticas básicas y medidas de rendimiento comunes para evaluar el comportamiento del operador de nado con mutación y el operador de tamaño de paso dinámico. Se concluye que BFOAM-DS obtiene soluciones mejores que una versión previa del algoritmo y similares a la mejor solución conocida en la literatura especializada.

Palabras clave: Metaheurística; operador de mutación; tamaño de paso dinámico; problema de diseño ingenieril; medidas de desempeño.

Recibido: 6 de julio de 2018 Aceptado: 11 de febrero de 2019 Publicado: 23 de octubre de 2019

Como citar: Hernández-Ocaña, B., Hernández-Torruco, J., Chávez-Bosquez, O., Canul-Reich, J., & Montané-Jiménez, L. G. (2019). Bacterial foraging optimization algorithm with mutation to solve constrained problems. *Acta Universitaria 29*, e2335. doi. http://doi.org/10.15174.au.2019.2335

ISSN online 2007-9621

Introduction

Nature-inspired metaheuristics have gained popularity solving constrained numerical optimization problems (CNOP) over mathematical programming, due to their easy implementation and quick execution. Moreover, metaheuristics generally provide a set of feasible solutions that the user can use according to their preferences. A CNOP can be defined as to:

Minimize $f(\vec{x})$

subject to:

$$g(\vec{x}) \leq 0, i, \ldots, m$$

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

where $\vec{x} = [x_1, x_2, \ldots, x_n] \in R^n$ is the solution vector and each decision variable, $\mathrm{x_k}, \mathrm{k} = 1, \ldots, \mathrm{n}$ is bounded by the lower and upper limits $\mathrm{L_k} \leq \mathrm{x_k} \leq \mathrm{U_k}$ 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 CNOP (Coello-Coello, 2002; Mezura-Montes, 2009; Mezura-Montes & Coello-Coello, 2011), equality constraints are transformed into inequality constraints by using a small tolerance $\varepsilon > 0$ as follows: $\left|h_j(\vec{x})\right| - \varepsilon \leq 0, j = 1, \ldots, p$.

Nature-inspired metaheuristic algorithms are classified in two classes:

1. Evolutionary Algorithms (EA): emulate the evolution of species and the survival of the fittest. Some well-known EA are genetic algorithms (GA) (Eiben & Smith, 2003), evolution strategies (ES) (Schwefel, 1993), evolutionary programming (Fogel, 1999), genetic programming (GP) (Koza *et al.*, 2003), and differential evolution (DE) (Price, Storn & Lampinene, 2005), which have been successfully applied in CNOP such as mechanical design (Calva-Yáñez, Niño-Suárez, Villarreal-Cervantes, Sepúlveda-Cervantes & Portilla-Flores, 2013).

2. Swarm Intelligence Algorithms (SIA): have the capability of emulating the collaborative behavior of some simple species when searching for food or shelter (Engelbrecht, 2007). Some SIA are particle swarm optimization (PSO) (Eberhart, Shi & Kennedy 2001) and ant colony optimization (ACO) (Dorigo, Maniezzo & Colorni, 1996).

Both PSO and ACO have gained popularity because of their great performance in solving CNOP. In 2002, another SIA-type algorithm called bacterial foraging optimization (BFOA) was introduced by Passino (2002), emulating the behavior of the *E.Coli* bacteria in the search of nutrients in its environment. This behavior is summarized in four processes: (1) chemotaxis (swim-tumble movements), (2) swarming (communication between bacteria), (3) reproduction (cloning of the best bacteria), and (4) elimination-dispersal (replacement of the worst bacteria). In BFOA, each bacterium tries to maximize the energy obtained per each unit of time spent on the foraging process while avoiding noxious substances. BFOA was used initially to solve unconstrained optimization problems; however, recent approaches add some constraints-handling technique to solve CNOP, where the penalty function is the most used technique (Hernández-Ocaña, Mezura-Montes & Pozozs -Parra, 2013).

Further investigations have addressed the fact that BFOA is particularly sensitive to the stepsize parameter, which is used in the chemotaxis process with the swim-tumble movement to determine the distance that a bacterium can move. In specialized literature, there are different ways to control the stepsize: (1) by keeping it static during the search process (as in the original BFOA) (Hernández-Ocaña, Pozos-Parra & Mezura-Montes, 2014; Huang, Chen & Abraham, 2010; Mezura-Montes & Hernández-Ocaña, 2009), (2) by using random values (Hernández-Ocaña *et al.*, 2014; Praveena, Vaisakh & Mohana Rao, 2010), (3) by using a dynamic variation (Hernández-Ocaña *et al.*, 2014; Niu, Fan, Xiao & Xue, 2012; Pandit, Tripathi, Tapaswi & Pandit, 2012), or (4) by adopting an adaptive mechanism (Hernández-Ocaña *et al.*, 2014; Mezura-Montes & López-Davila, 2012; Saber, 2012). However, such approaches were stated mainly for specific optimization problems. In a recent study of the stepsize by Hernández-Ocaña *et al.* (2014), different mechanisms were compared, and the dynamic control mechanism was found to be slightly superior to static, random, and adaptive versions.

Mezura-Montes & Hernández-Ocaña (2009) adapted BFOA in a proposal called modified bacterial foraging optimization algorithm (MBFOA) to solve CNOP. This approach inherits the four processes of BFOA, featuring individual and independent processes with sequential interaction. Therefore, the parameters that determine the number of swim movements, number of tumbles, and the swarming loop were eliminated. Moreover, the feasible rules proposed by Deb (2000) are used as a constraint-handling technique. Unlike BFOA, where the stepsize is static, in MBFOA the stepsize used in the swim movements was adapted according to the boundary of the decision variables.

BFOA has been combined with other algorithms, particularly with EA, to improve its performance, e.g., with a GA in Kim, Abraham & Cho (2007), Kushwaha, Bisht & Shah (2012), Luo & Chen (2010) and DE in Biswas, Dasgupta, Das & Abraham (2007). Mutation operators have been added to BFOA in Nouri & Hong (2012); nevertheless, no proposal using mutation within the chemotaxis process was found.

Currently, a new version based on MBFOA was proposed in Hernández-Ocaña *et al.* (2016) where two swims and a random stepsize are used in the chemotaxis process in order to improve the foraging performance. This proposal was called TS-MBFOA and implemented successfully to solve real-word problems: It minimizes the optimal synthesis of four-bar mechanisms and solves an instance of the menu planning problem (Hernández-Ocaña, Chávez-Bosquez, Hernández-Torruco, Canul-Reich & Pozos-Parra, 2018). However, this authors mention that stepsize is a sensitivity parameter, and it requires more study. In most cases, the stepsize is randomly updated.

This paper aims to test TS-MBFOA using mutation mechanisms and a simple dynamic stepsize. Results obtained are compared against state-of-the-art algorithms and validated with commonly performance measures found in the literature. It is worth mentioning that this approach uses fewer parameters than the original MBFOA due to both the dynamic stepsize implemented and the reproduction process applied to half the swarm bacteria, as initially introduced in BFOA. To the best of one's knowledge, this is the first time that mutation is used as a swim operator.

This paper is organized as follows: Materials and methods outlines the MBFOA, describes the three test problems to be solved and the performance measures used to evaluate the results obtained by this study's proposal; the results section details the experiments conducted in order to analyze the behavior of this approach against those of one of the best state-of-the-art algorithm; finally, the general conclusions and future works are presented.

ISSN online 2007-9621

Materials and Methods

Bacterial foraging optimization algorithm with mutation and dynamic stepsize (BFOAM-DS)

BFOAM-DS is an algorithm derived from MBFOA in order to improve the performance in constrained spaces. Two modifications to the MBFOA have been made:

- Two different swim operators are applied in the chemotaxis process: (1) exploration using a dynamic stepsize easy to implement, and (2) exploitation using a mutation operator, in order to improve the exploration and exploitation of the bacteria.
- The reproduction process is applied to half of the swarm bacteria.

In BFOA, MBFOA, TS-MBFOA, and BFOAM-DS, a bacterium i represents a potential solution to the CNOP (i.e., a n-dimensional real-value vector identified as $\vec{x}$), and it is defined as $\theta^i(j,G)$, where j is the chemotaxis loop index and G is the current generation of the algorithm. The initial swarm bacteria are generated randomly with a uniform distribution in the range of the decision variables. Within a generation (G), three inner processes are carried out: (1) chemotaxis, (2) reproduction, and (3) elimination-dispersal. Swarming process is added into the chemotaxis process. Figure 1 outlines the process performed by BFOAM-DS.

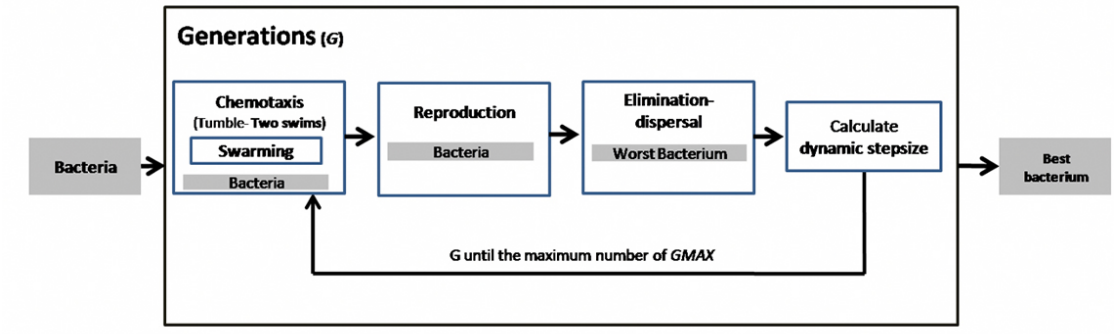

Figure 1. BFOAM-DS overall process.
Source: Hernández-Ocaña *et al.* (2018).

Chemotaxis

In this process, two swims are interleaved in each generation: Either the exploitation swim or exploration swim is performed. The process starts with the exploitation swim (classical swim). Yet, a 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 in the next generation. Otherwise, a new tumble for the other swim will be computed. The process stops after N_c attempts.

The exploration swim uses the mutation between bacteria and is computed as:

$$\theta^i(j+1,G) = \theta^i(j,G) + \beta(\theta_1^r(j,G) - \theta_2^r(j,G)) \quad (1)$$

ISSN online 2007-9621

where $\theta_1^r(j,G)$ and $\theta_2^r(j,G)$ are two different randomly selected bacteria from the swarm. β is a positive-value user-defined parameter defining the closeness of the new position of a bacterium concerning the position of the best bacterium $\theta^B(G)$, which is a value between a range $(0,1)$, i.e., it scales the area where a bacterium can move. This parameter is also used in the swarming process.

The exploitation swim is computed as:

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

where $\phi(i)$ is calculated with the original BFOA Equation 3, determining the direction of the swim or tumble:

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{3}$$

where $\Delta(i)$ is a uniformly distributed random vector of size n with elements within

$[-1,1]$.

Equation 4 determines the distance of movement of a bacterium:

$$C(i,G) = \Theta(i)\left(\frac{G}{Gmax}\right) \tag{4}$$

where $C(i,G)$ is the dynamic stepsize of each bacterium updated in each generation, $\Theta(i)$ is a randomly generated vector of size n with elements within the range of each decision variable: $[Upper_k, Lower_k], k = 1,\dots,n$, and $Gmax$ is the maximum number of generations of the algorithm.

It is important to remark that the exploration swim (Equation 1) performs larger movements due to the mutation operator which uses bacteria randomly. On the other hand, the exploitation swim (Equation 2) generates small movements using the dynamic stepsize in the search process.

Swarming

At the half number of the chemotaxis process, the swarming operator is applied with the Equation 5, where β is a user-defined positive parameter into $(0,1)$. In this proposal, unlike of MBFOA, if a solution violates the boundary of decision variables, then a new solution $x\ i$ is randomly generated, bounded by lower and upper limits $L_i \le x_i \le U_i$.

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

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 at generation G. If the population is feasible, this best bacterium has a better value (fitness) in the objective function. If the population is outside the feasible region, this best bacterium has a smaller amount of violation of constraints. The swarming operator movement applies twice in a chemotaxis loop, while in the remaining steps the tumble-swim movement is carried out.

Reproduction

In this process, half of worst bacteria (S_r) are replaced, and the remaining ones are duplicated.

Elimination-dispersal

The worst bacteria are eliminated, and new bacteria are randomly generated with uniform distribution between the ranges of the decision variables.

The corresponding BFOAM-DS pseudocode is presented in Algorithm 1. The user-defined parameters are summarized in the caption.

Algorithm 1: BFOAM-DS. Input parameters are number of bacteria S_b, chemotaxis loop limit N_c, scaling factor β, and number of iterations (generations) $GMax$.

```
Create an initial swarm of bacteria at random θ^i(j, G) ∀i = 1, ..., S_b
Evaluate each θ^i(j, G) ∀i = 1, ..., 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 exploration and
            exploitation using Eqs. 1 and 2, and the swarming operator in Eq. 5 for
            bacterium θ^i(j, G) by considering the three feasibility rules as the selection
            criteria.
        end
    end
    Perform the reproduction process by sorting all bacteria in the swarm, based on the
    feasibility rules, eliminating the half of worst bacteria and duplicating the best
    bacteria.
    Perform the elimination-dispersal process by eliminating the worst bacterium
    θ^i(j, G) in the current swarm.
    Update the dynamic stepsize using Eq. 4.
end
```

Source: Authors' own elaboration.

Problem description

Three design engineering problems are used to test the performance of this study's proposal, called BFOAM-DS. It was coded using MATLAB R2009b and executed on a PC with a 3.5 Core 2 Duo Processor, 4GB of RAM, and 64-bit Windows 7 operating system.

P01. Tension/compression spring design optimization problem.

It minimizes the weight of a tension/compression spring, subject to constraints of minimum deflection, shear stress, surge frequency, and limits on the outside diameter and on design variables (Arora, 2012). There are three design variables: the wire diameter (x_1), the mean coil diameter (x_2), and the number of active coils (x_3). The mathematical model has the form:

Minimize: $f(x) = (x_3 + 2)x_2x_1^2$

subject to:

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566x_2x_1^3 - x_1^4} + \frac{1}{5108x_1^2} - 1 \leq 0$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

where $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$ and $2 \leq x_3 \leq 15$. Best solution: $x^* = (0.051690,\ 0.356750,\ 11.287126)$, where $f(x^*) = 0.012665$.

P02. Pressure vessel optimum design optimization problem

It involves the minimization of the entire cost, consisting of material cost, welding and forming costs (Sandgren, 1990). The mathematical model has the form:

Minimize: $f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$

subject to:

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(x) = -x_2 + 0.00954x_3 \leq 0$$

$$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3\pi x_3^3} + 1296000 \leq 0$$

$$g_4(x) = x_4 - 240 \leq 0$$

where $1 \leq x_1, x_2 \leq 99$ and $10 \leq x_3, x_4 \leq 200$. Best solution: $x^* = (0.8125,\ 0.4375,\ 42.098446, 176.636596)$, where $f(x^*) = 6059.714335$.

P03. Welded beam design optimization problem

It finds the minimum fabrication cost, considering four design variables: x_1, x_2, x_3, x_4 and constraints of shear stress τ, bending stress in the beam σ, buckling load on the bar P_c, and end deflection on the beam δ (Michalewicz & Fogel, 2004). The mathematical model has the form:

Minimize: $f(x) = 1.1047x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$

subject to:

$g_1(X) = \tau(X) - \tau\, max \leq 0$

$g_2(X) = \sigma(X) - \sigma\, max \leq 0$

$g_3(X) = X_1 - X_4 \leq 0$

$g_4(X) = 0.10471X_1^2 + 0.04811X_3X_4(14.0 + X_2) - 5.0 \leq 0$

$g_5(X) = 0.125 - X_1 \leq 0$

$g_6(X) = \delta(X) - \delta\,\max \leq 0$

$g_7(X) = P - P_c(X) \leq 0$

with

$\tau(X) = \sqrt{(\tau)^2 + 2\tau'\tau''\frac{x^2}{2R} + (\tau)^2}, \tau' = \frac{P}{\sqrt{2}X_1X_2}, \tau'' = \frac{MR}{J},$

$M = P(L + \frac{X_2}{2}), R = \sqrt{\frac{X_2^2}{4} + (\frac{X_1+X_3}{2})^2}$

$$J = 2\left\{\frac{X_1X_2}{\sqrt{2}}\left[\frac{X_2^2}{12} + \left(\frac{X_1 + X_3}{2}\right)^2\right]\right\}\ \sigma(X) = \frac{6PL}{X_4X_3^2}, \delta(X) = \frac{4PL^3}{EX_3^3X_4}$$

$$P_c(X) = \frac{4.013\sqrt{\frac{EGX_3^2X_4^6}{36}}}{L^2}\left(1 - \frac{X_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$P = 6000lb, L = 14in, E = 30 * 10^6psi, G = 12 * 10^6psi, \tau max = 13{,}600psi,$

$\sigma\, max = 30000psi, \delta\, max = 0.25\ in$

where $0.1 \leq x_1 \leq 2.0, 0.1 \leq x_2 \leq 10.0, 0.1 \leq x_3 \leq 10.0, 0.1 \leq x_4 \leq 2.0$.

Best solution: $x^* = (0.244369,\ 6.217520,\ 8.291471,\ 0.244369,\ -5741.176933,\ -0.000001,\ -0.0000000,\ -3.022955,\ -0.119369,\ -0.234241,\ -0.000309)$, where $f(x^*) = 2.380957$.

Performance measures

To evaluate the behavior of the compared algorithms, the following performance measures for nature-inspired constrained optimization problems, taken from Liang *et al.* (2006), were computed:

- *Feasible run*: a run where at least one feasible solution is found within the Maximum number of evaluations allowed per each problem (Max_{evals}).
- *Feasible rate* = (number of feasible runs) / (total runs).
- *Successful run*: a run where a feasible solution x^* satisfying $f(\vec{x}) - f(\vec{x}^*) \leq 0.0001$ is found within Max_{evals}.
- Success rate = (number of successful runs) / (total runs).
- *Success performance* = the mean of (feasible solutions for successful runs) × (number of total runs) / (number of successful runs).
- *Successful swim* = A swim movement where the new position is better (based on the feasibility rules) than the original position.
- *Successful swim rate* = (number of successful swims) / (total swims), where (total swims) = $S_b \times N_c \times GMax$.

Results

Parameter setting

The user-defined parameters of BFOAM-DS are shown in table 1. Those values were fine-tuned using the iRace tool (López-Ibáñez, Dubois-Lacoste, Pérez-Cáceres, Birattari & Stützle, 2011), except for $Gmax$, whose value was fixed to adjust the termination condition to the Max_{evals} value per each problem. The parameter values for MBFOA were taken from Hernández-Ocaña *et al.* (2014).

Table 1. Parameter values for the MBFOA and BFOAM-DS comparison.

Parameter	MBFOA	BFOAM-DS
S_b	50	40
N_c	12	20
S_r	$S_b/2$	$S_b/2$
β	1.76	0.68
R	1.62E-2	-
$GMax$	value to reach Max_{evals}	value to reach Max_{evals}

Note: '-' indicates that the corresponding parameter is not required by the algorithm.
Source: Authors' own elaboration.

Statistics

The proposed BFOAM-DS was tested solving the three problems with 15 000 Max_{evals} in a set of 30 independent runs. Results are presented in table 2 and discussed based on three measures: (1) Quality, i.e., the best solution found so far; (2) Consistency, i.e., the mean value closer to the best known solution x^* and the lowest standard deviation value; and (3) Computational cost, i.e., the number of evaluations required by each given problem.

For the Tension/compression spring design optimization problem (P01), the proposed BFOAM-DS found the best result when compared to MBFOA, and this solution is similar to the best-known solution x^*. In addition to these results, BFOAM-DS has better consistency than MBFOA.

For the pressure vessel optimum design optimization problem (P02), BFOAM-DS found the best result, which was similar to x^*. However, the result across 30 runs of MBFOA showed more consistency.

Finally, for the welded beam design optimization problem (P03), BFOAM-DS showed better results and higher consistency than those of MBFOA.

It is important to mention that MBFOA found these results at a computational cost of 48 000 evaluations, in contrast to BFOAM-DS which only cost 15 000 evaluations. The number of evaluations is calculated using $S_b \times N_c \times GMax$. For example, $40\text{x}20\text{x}18 = 14400$ evaluations for BFOAM-DS according to the values shown in table 1, where $Gmax$ is a value to reach Max_{evals}, in this case $15\ 000\ /(40\text{x}20)\ = 18Gmax$.

Table 2. Statistical results of MBFOA and BFOAM-DS.

Problem	Criteria	MBFOA	BFOAM-DS
P01	Best	0.012671	0.012665233
	Average	0.012759	0.012681938
	Std.	1.36E-04	5.08E-05
P02	Best	6060.46	6059.701609
	Average	6074.625	6173.535938
	Std.	1.56E+01	2.01E+02
P03	Best	2.386	2.380952906
	Average	2.404	2.380957824
	Std.	1.6E-02	1.19E-05
	Max_{evals}	48000	14400

Source: Authors' own elaboration.

Figures 2, 3, and 4 show the convergence graphs of MBFOA and BFOAM-DS in each of the engineering problems. Graphs depict the convergence in the median of all runs per problem. BFOAM-DS has a similar behavior in the three problems; it reaches the optimum before ten generations. On the other hand, MBFOA converges prematurely in local optima in all problems, it also requires more generations, with 48 000 evaluations as a stop condition.

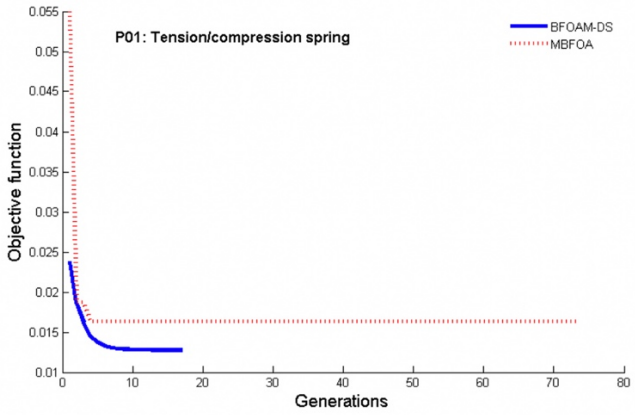

Figure 2. Convergence graphic of MBFOA and BFOAM-DS in the P01 problem.
Source: Authors' own elaboration.

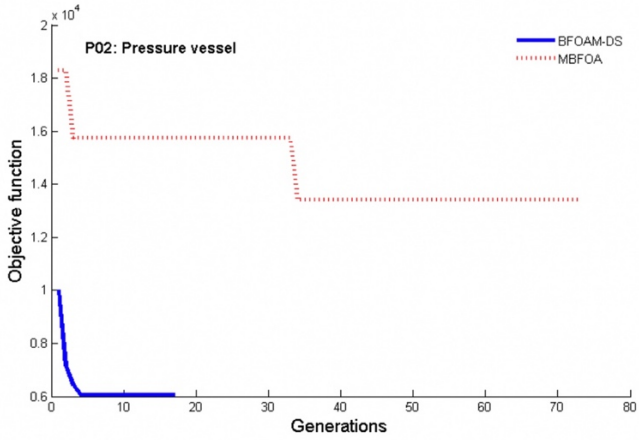

Figure 3. Convergence graphic of MBFOA and BFOAM-DS in the P02 problem.
Source: Authors' own elaboration.

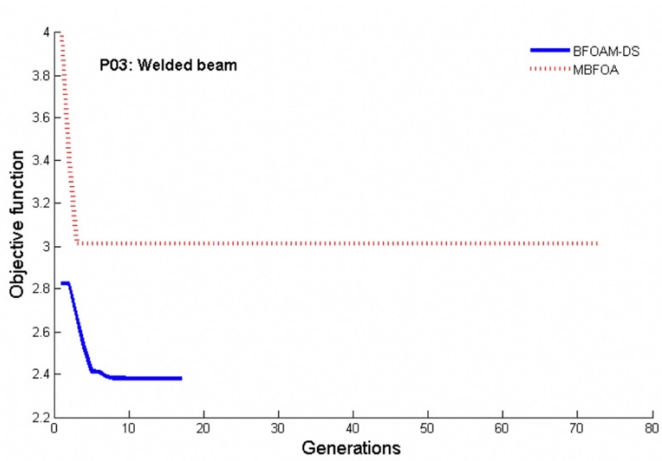

Figure 4. Convergence graphic of MBFOA and BFOAM-DS in the P03 problem.
Source: Authors' own elaboration.

ISSN online 2007-9621

Effectiveness results of BFOAM-DS in solving CNOP are shown in table 3. According to the results of the feasible rate measure, for all three problems, the proposed algorithm found feasible solutions within the maximum number of evaluations allowed across all independent runs. However, only in problem P03, this study's algorithm obtained 100% feasible solutions, which is similar to the best-known solution. For problem P01, this algorithm obtained 96.66% and 63.33% in problem P02. The computational cost to obtain a feasible solution similar to the best-known solution for each problem is 4860, 13 977, and 10 120, respectively, according to the measure success performance where P02 is the most complex problem, followed by P03 and P01.

Table 3. Performance of BFOAM-DS according to the runs and successful swims.

Criteria	P01	P02	P03
Feasible rate	100%	100 %	100%
Success rate	96.66%	63.33%	100%
Success performance	4.86E+03	1.39E+04	1.01E+04

Source: Authors' own elaboration.

The effectiveness of the stepsize was measured using the performance measures successful swim and successful swim rate, taken from Hernández-Ocaña *et al.* (2014). The goal of these measures is to obtain the number of successful swims in a run. The algorithm BFOAM-DS carried out 13 600 swims ($40S_b \times 20N_c \times 17GMax$). According to the results in table 4, 14% of swims are successful in each run of BFOAM-DS, which are similar results to those mentioned in Hernández-Ocaña *et al.* (2014) for solving other CNOP. Nevertheless, the version of BFOAM-DS without mutation only obtained the 3.18% of successful swims in average.

Table 4. Performance of BFOAM-DS according to the success of the runs and swims.

Problem	Successful swim	Successful swim rate
	BFOAM-DS without mutation	
P01	735	5.40%
P02	324	2.38%
P03	241	1.77%
	BFOAM-DS with mutation	
P01	2166	15.92%
P02	1998	14.69%
P03	1899	13.96%

Source: Authors' own elaboration.

Figures 5, 6, and 7 show the behavior of the BFOAM-DS algorithm with and without the mutation swim in each of the engineering problems. Graphs depict the swim in the median of all runs per problem. The version with mutation swim generates more successful swims during all the generations of the algorithm, on average 110 successful swims in the three problems. Concerning the algorithm without a mutation swim, in P01 and P03 problems, successful swims are few in the first generation (40 successful swims on average), and these gradually decrease until the sixth generation, where the algorithm remains on average with five successful swims per generation. In the P02 problem, the swims without mutation remain stable and, on average, 38 successful swims per generation are generated until the execution of the algorithm is completed. In general, swims with mutation improve the performance of the algorithm, allowing better solutions with less computational cost.

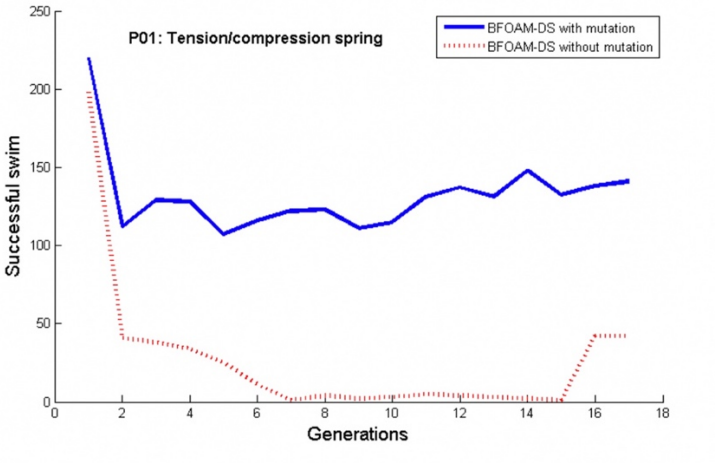

Figure 5. Successful swims of BFOAM-DS in the P01 problem.
Source: Authors' own elaboration.

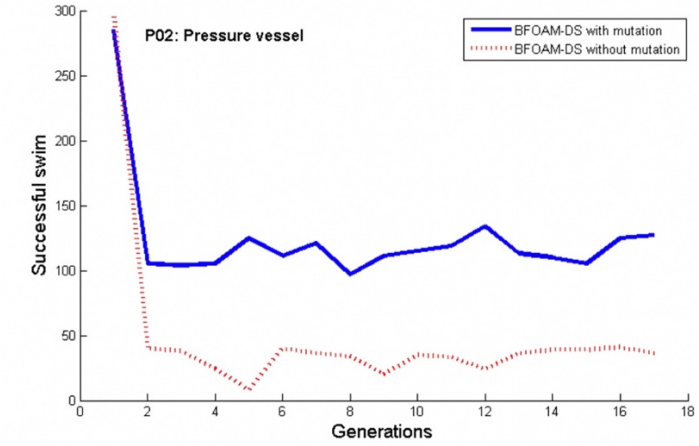

Figure 6. Successful swims of BFOAM-DS in the P02 problem.
Source: Authors' own elaboration.

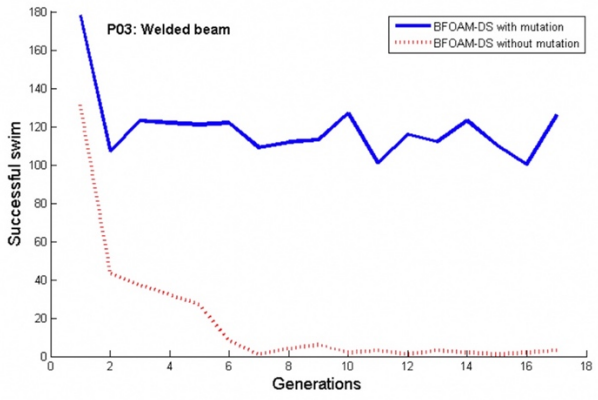

Figure 7. Successful swims of BFOAM-DS in the P03 problem.
Source: Authors' own elaboration.

ISSN online 2007-9621

Discussion

In this new version of the bacterial foraging optimization algorithm, the mutation power of evolutionary algorithms was added to the chemotaxis process of the bacterial foraging in order to improve the performance of the algorithm. Moreover, an easy dynamic stepsize was used, which led to a decrease in the number of parameters to be defined by the user.

The statistical tests and performance measures applied to the results of 30 independent runs of each one of the three optimization problems showed that BFOAM-DS outperforms the previous version of the algorithm. Moreover, BFOAM-DS requires 33 000 evaluations fewer than the previous version of the algorithm (48 000-14 400) to achieve these competitive results. The stepsize, a user-defined parameter, has been replaced in the algorithm by a proposed dynamic stepsize. The exploration and exploitation capacity of the algorithm was improved with the swim with a mutation operator that, on average, increased the effectiveness of swims by 10%.

In general, the proposed BFOAM-DS performed better than the original MBFOA, obtaining results with less computational cost. Moreover, the consistency was competitive, and the quality of results was similar to the best-known solution in each problem. Another advantage is that BFOAM-DS requires less tuning of parameters due to the use of a dynamic stepsize.

Conclusions

Bacterial foraging optimization is a metaheuristic included in the group of intelligence swarm algorithms used to solve complex problems. This algorithm is considered younger and less known than evolutionary algorithms. A proposal based on bacterial foraging has been made, and it has been tested solving three constraint numerical optimization problems. This version has been called the bacterial foraging optimization algorithm with mutation and dynamic stepsize (BFOAM-DS).

BFOAM-DS was tested in three engineering design optimization problems. Results were analyzed with basic statistics (best, average and standard deviation). In addition, common measures for nature-inspired constrained optimization problems are used to evaluate the behavior of the swim with a mutation operator (feasible run, feasible rate, successful run, success rate, success performance, successful swim, and successful swim rate) and the dynamic stepsize operator. Then, results were compared against a previous version of the algorithm (MBFOA) to observe the effectiveness of the proposed improvements.

BFOAM-DS solved effectively all test problems with fewer evaluations than the previous version of the algorithm. This new version of the algorithm requires fewer parameters to calibrate, so it would be easier for the final user to tune up. The proposed operators improve the overall performance of the algorithm, as demonstrated by the performance tests.

As future work, this study's proposal will be tested against other complex problems and an analysis of the frequency of reproduction process will be carried out.

Acknowledgements

To *Consejo Nacional de Ciencia y Tecnología* (Conacyt) for supporting the joint doctoral program in Computer Science at the Universidad Juárez Autónoma de Tabasco and Universidad Veracruzana.

References

Arora, J. (2012). *Introduction to Optimum Design*. New York, NY: McGraw-Hill.

Biswas, A., Dasgupta, S., Das, S., & Abraham, A. (2007). A Synergy of Differential Evolution and Bacterial Foraging Optimization for global optimization. *Neural Network World*, *17*(6), 607-626.

Calva-Yáñez, M. B., Niño-Suárez, P. A., Villarreal-Cervantes, M. G., Sepúlveda-Cervantes, G., & Portilla-Flores, E. A. (2013). Differential evolution for the control gain's optimal tuning of a four-bar mechanism. *Polibits*, *47*, 67-73.

Coello-Coello, C. A. (2002). 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. doi: https://doi.org/10.1016/S0045-7825(01)00323-1

Deb, K. (2000). An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2-4), 311-338. doi: https://doi.org/10.1016/S0045-7825(99)00389-8

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions of Systems, Man and Cybernetics, Part B*, *26*(1), 29-41. doi: https://doi.org/10.1109/3477.484436

Eiben, A. E., & Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Berlin Heidelberg: Springer Verlag.

Eberhart, R.C., Shi, Y., & Kennedy, J. (2001). *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann.

Engelbrecht, A. P. (2007). *Computational Intelligence. An Introduction*. New York, NY: John Wiley & Sons.

Fogel, L. J. (1999). *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming*. New York, NY: John Wiley & Sons.

Hernández-Ocaña, B., Mezura-Montes, E., & Pozos-Parra, P. (2013). A review of the bacterial foraging algorithm in constrained numerical optimization. In *Proccedings of the Congress on Evolutionary Computation (CEC'2013).* Cancún, México.

Hernández-Ocaña, B., Pozos-Parra, M. P., & Mezura-Montes, E. (2014). 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)*. Vancouver, DB, Canada.

Hernández-Ocaña, B., Pozos-Parra, M. P., Mezura-Montes, E., Portilla-Flores, E., Vega-Alvarado, E., & Calva-Yáñez, M. (2016). Two swim operators in the modified bacterial foraging algorithm for the optimal synthesis of four-bar mechanisms. *Computational Intelligence and Neuroscience*, *2016*(1), 1-18. doi: http://dx.doi.org/10.1155/2016/4525294

Hernández-Ocaña, B., Chávez-Bosquez, O., Hernández-Torruco, J., Canul-Reich, J., & Pozos-Parra, P. (2018). Bacterial foraging optimization algorithm for menu planning. *IEEE Access*, *6*, 8619-8629. doi: https://doi.org/10.1109/ACCESS.2018.2794198

Huang, H. C., Chen, Y. H., & Abraham, A. (2010). Optimized watermarking using swarm-based bacterial foraging. *Information Hiding and Multimedia Signal Processing, 1*(1), 51-58.

Kim, D. H., Abraham, A., & Cho, J. H. (2007). A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Information Sciences*, *177*(18), 3918-3937. doi: https://doi.org/10.1016/j.ins.2007.04.002

Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., & Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Verlag US: Springer.

Kushwaha, N., Bisht, V. S., & Shah, G. (2012). Genetic algorithm based bacterial foraging approach for optimization. In *Proceedings on National Conference on Future Aspects of Artificial intelligence in Industrial Automation (NCFAAIIA 12).*

Liang, J. J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P. N., Coello-Coello, C. A., & Deb, K. (2006). *Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization.* Technical report, Nanyang Technological University, Singapore.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., & Stützle,T. (2011). The iRace package, Iterated Race for Automatic Algorithm Configuration. *Operations Research Perspectives*, *3*, 43-5. doi: https://doi.org/10.1016/j.orp.2016.09.002

Luo, Y., & Chen, Z. (2010). Optimization for PID control parameters on hydraulic servo control system based on the novel compound evolutionary algorithm. In *Second International Conference on Computer Modeling and Simulation.* Sanya, Hainan, China. doi: https://doi.org/10.1109/ICCMS.2010.53

Mezura-Montes, E. (2009). *Constraint-Handling in Evolutionary Optimization.* Berlin Heidelberg: Springer-Verlag.

Mezura-Montes, E., & Coello-Coello,C. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, *1*(4), 173-194. doi: https://doi.org/10.1016/j.swevo.2011.10.001

Mezura-Montes, E., & Hernández-Ocaña, B. (2009). Modified bacterial foraging optimization for engineering design. In: H. Cihan, K. Dagli, M. Bryden, S. M. Corns, M. Gen, K. Tumer, & G. Suer. (Eds.). *Intelligent Engineering Systems through Artificial Neural Networks* (pp. 357-364). USA: ASME Press. doi: https://doi.org/10.1115/1.802953.paper45

Mezura-Montes, E., & López-Davila, E. A. (2012). Adaptation and local search in the modified bacterial foraging algorithm for constrained optimization. In *Proccedings of the IEEE Congress on Evolutionary Computation (CEC'12).* Brisbane, QLD, Australia. doi: https://doi.org/10.1109/CEC.2012.6256172

Michalewicz, Z., & Fogel, D. B. (2004). *How to Solve It: Modern Heuristics*. Berlin Heidelberg: Springer-Verlag.

Niu, B., Fan, Y., Xiao, H., & Xue, B. (2012). Bacterial foraging based approaches to portfolio optimization with liquidity risk. *Neurocomputing*, *98*, 90-100. doi: https://doi.org/10.1016/j.neucom.2011.05.048

Nouri, H., & Hong, T. S. (2012). A bacterial foraging algorithm based cell information considering operation time. *Journal of Manufacturing Systems*, *31*(3), 326-336. doi: https://doi.org/10.1016/j.jmsy.2012.03.001

Pandit, N., Tripathi, A., Tapaswi, S., & Pandit, M. (2012). An improved bacterial foraging algorithm for combined static/dynamic environmental economic dispatch. *Applied Soft Computing*, *12*(11), 3500-3513. doi: https://doi.org/10.1016/j.asoc.2012.06.011

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, *22*(3), 52-67. doi: https://doi.org/10.1109/MCS.2002.1004010

Praveena, P, Vaisakh, K., & Mohana Rao, S. (2010). 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*. Bangalore, India. doi: https://doi.org/10.1109/ICIIC.2010.26

Price, K., Storn, R. M., & Lampinen, J. A. (2005). *Differential Evolution: A Practical Approach to Global Optimization*. Berlin Heidelberg: Springer-Verlag.

Saber,A. Y.(2012). Economic dispatch using particle swarm optimization with bacterial foraging effect. *International Journal of Electrical Power & Energy Systems*, *34*(1), 38-46. doi: https://doi.org/10.1016/j.ijepes.2011.09.003

Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, *112*(1), 223-229. doi: https://doi.org/10.1115/1.2912596

Schwefel, H. P.(1993). *Evolution and Optimization Seeking: The Sixth Generation.* New York, NY: John Wiley & Sons.