Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 4525294, 18 pages
http://dx.doi.org/10.1155/2016/4525294

Research Article

Two-Swim Operators in the Modified Bacterial Foraging Algorithm for the Optimal Synthesis of Four-Bar Mechanisms

Betania Hernández-Ocaña,[1] Ma. Del Pilar Pozos-Parra,[1] Efrén Mezura-Montes,[2] Edgar Alfredo Portilla-Flores,[3] Eduardo Vega-Alvarado,[3] and Maria Bárbara Calva-Yáñez[3]

[1]*División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco, 86690 Cunduacán, TAB, Mexico*
[2]*Centro de Investigación en Inteligencia Artificial, Universidad Veracruzana, Sebastián Camacho 5, Centro, 91000 Xalapa, VER, Mexico*
[3]*Instituto Politécnico Nacional (IPN-CIDETEC), U. Adolfo López Mateos, 07700 Ciudad de México, DF, Mexico*

Correspondence should be addressed to Betania Hernández-Ocaña; betania.h.o@gmail.com

Received 9 November 2015; Revised 11 January 2016; Accepted 12 January 2016

Academic Editor: Saeid Sanei

This paper presents two-swim operators to be added to the chemotaxis process of the modified bacterial foraging optimization algorithm to solve three instances of the synthesis of four-bar planar mechanisms. One swim favors exploration while the second one promotes fine movements in the neighborhood of each bacterium. The combined effect of the new operators looks to increase the production of better solutions during the search. As a consequence, the ability of the algorithm to escape from local optimum solutions is enhanced. The algorithm is tested through four experiments and its results are compared against two BFOA-based algorithms and also against a differential evolution algorithm designed for mechanical design problems. The overall results indicate that the proposed algorithm outperforms other BFOA-based approaches and finds highly competitive mechanisms, with a single set of parameter values and with less evaluations in the first synthesis problem, with respect to those mechanisms obtained by the differential evolution algorithm, which needed a parameter fine-tuning process for each optimization problem.

1. Introduction

Nature-inspired algorithms (NIAs) have been successfully used to solve Constrained Numerical Optimization Problems (CNOPs) using constraint-handling techniques [1] given that, originally, these algorithms were designed to deal solely with unconstrained search spaces. NIAs can be comprised of two groups: (1) Evolutionary Algorithms (EAs) [2] based on emulating the process of natural evolution and survival of the fittest and (2) Swarm Intelligence Algorithms (SIAs) [3] based on cooperative behaviors of simple organisms such as insects, birds, fish, or bacteria. EAs are one of the most used metaheuristics. However, SIAs have been gaining popularity among researchers and practitioners, mainly with the Particle Swarm Optimization (PSO) [4] and the Ant Colony Optimization (ACO) [5] algorithms.

Without loss of generality, a CNOP can be defined as

$$\begin{aligned} &\text{Minimize} \quad f(\vec{x}) \\ &\text{subject to:} \quad g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, m \\ &\qquad\qquad\quad h_j(\vec{x}) = 0, \quad j = 1, \ldots, p, \end{aligned} \tag{1}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n] \in R^n$ is the solution vector and each decision variable x_k, $k = 1, \ldots, n$ is bounded by lower and upper limits $L_k \leq x_k \leq U_k$, 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 CNOPs [1, 6, 7] equality constraints are transformed

into inequality constraints by using a small tolerance $\varepsilon > 0$ as follows: $|h_j(\vec{x})| - \varepsilon \leq 0$, $j = 1, \ldots, p$.

In the context of mechanical engineering, synthesis is the design process of mechanical systems [8]. Four-bar mechanisms are widely used in machinery design, since they are the simplest articulated mechanisms for controlled movement with one degree of freedom. The synthesis of these mechanisms is a well-known CNOP, and originally two classical approaches were used for this synthesis: graphical and analytical methods. However, implementing such solutions is a complicated issue and their results are quite limited; for this reason, the design of these mechanisms is a case of hard numerical optimization. There are several types of syntheses; this work addresses the dimensional design of a mechanism, that is, to calculate the length of the necessary links for generating a specific movement [9].

The synthesis of four-bar mechanisms has been carried out with different nature-inspired metaheuristics. In [10] a modified Genetic Algorithm (GA) with a penalty function as constraint-handler was proposed. differential evolution (DE) and a variable control method for deviations were applied in [11]. In [12] the synthesis of a mechanism for tracking a trajectory of n points, based on Simulated Annealing (SA), was developed. Regarding this same tracking problem, a performance comparison among three different metaheuristics, GA, DE, and PSO, was presented in [13], while the Artificial Bee Colony (ABC), PSO, a binary GA (BGA), and a hybrid GA-PSO approach were used in [14]. Finally, in [15] the synthesis of a planar four-bar mechanism for position control using the Harmony Search (HS) algorithm was carried out. From the abovementioned literature review, the diversity of nature-inspired algorithms to solve the synthesis of four-bar mechanisms is noticeable.

On the other hand, there are algorithms whose usage in this type of optimization problems has been less explored, as it is the case of the Bacterial Foraging Optimization Algorithm (BFOA), which is a SIA proposed by Passino in 2002 to solve unconstrained numerical optimization problems [16]. BFOA emulates the behavior of bacterium *E. coli* in the search of nutrients in its environment. The goal of each bacterium is to maximize the energy it obtains per each unit of time spent on the foraging process while avoiding noxious substances. BFOA is considered a SIA because bacteria can communicate among them. Such behavior can be summarized in four processes: (1) chemotaxis (swim and tumble movements are performed), (2) swarming (bacteria can communicate with each other to direct their search for nutrients), (3) reproduction (the best bacteria are duplicated and these replaced other bacteria), and (4) elimination-dispersal (the worst bacteria are eliminated and new bacteria are randomly dispersed).

Regarding unconstrained optimization, BFOA has been combined with other algorithms, particularly with EAs, to improve its performance, for example, with a GA in [17–19] and DE in [20]. Mutation operators have been added to BFOA in [21]. Moreover, hybrids with other SIAs [22], and particularly with PSO, are found in the specialized literature [23, 24]. Furthermore, BFOA was also combined with artificial immune system's clonal selection and fuzzy logic within its chemotaxis process in [25].

When dealing with constrained search spaces there are different approaches [26]; for example, in [27], BFOA was adapted to solve CNOPs in a proposal called Modified-BFOA (MBFOA) which used a set of feasibility rules [28] as constraint-handler. Moreover, MBFOA was simplified and improved in its step size handling in the swim operator in [29]. Further modifications were proposed in [30, 31] to solve multiobjective CNOPs, where some four-bar mechanisms were tackled. More recently, MBFOA was further improved to solve CNOPs in the so-called Improved MBFOA (IMBFOA for short) [32] where the idea of two-swim movements within the chemotaxis process was initially explored. However, IMBFOA heavily depends on a local search based on sequential quadratic programming. IMBFOA is the starting point of this research, focused on the optimal synthesis of four-bar mechanisms. Finally, BFOA has been combined with two NIAs BFOA-DE-PSO in [33, 34], and the idea of using micropopulations was explored in [35].

From the above literature review, it was found that NIAs are a valid option to solve four-bar mechanisms. However, the usage of BFOA-based approaches is still scarce. Furthermore, the incorporation of additional variation operators in those BFOA-based approaches is more frequent in unconstrained optimization. The abovementioned is the main motivation of this work, where two improved swims (and different from those in IMBFOA) are proposed to enhance the capabilities of MBFOA to deal with four-bar mechanisms optimization. Furthermore, such two-swim mechanism performance allows eliminating the second-order local search operator. Therefore, the contribution of this work consists in getting knowledge about the type of operators which provide better results in those constrained search spaces defined by the four-bar mechanisms tackled in this research.

The proposed approach is compared against those BFOA-based approaches for constrained optimization (IMBFOA and MBFOA) and also against a DE-based approach designed to solve mechanical design problems. To the best of the authors' knowledge, this is the first time that the variation operators of a BFOA-based approach are studied in such a way that they improve the capabilities of the algorithm to solve a set of four-bar synthesis problems.

The document is organized as follows: in Section 2 the general synthesis of a four-bar mechanism is explained, including the kinematics of the mechanism and its coupler, as well as the specifications of the three case studies to solve. In Section 3, brief descriptions of MBFOA and IMBFOA are presented and the new proposal "Two-Swim MBFOA" (TS-MBFOA) is introduced. Section 4 shows the results obtained by TS-MBFOA and their comparison against those obtained by other NIAs. Finally, Section 5 presents the conclusions and future work of this research.

2. Synthesis of Four-Bar Mechanisms

Figure 1 shows a planar four-bar mechanism formed by a reference bar r_1, an input bar r_2 (crank), a coupler r_3, and an output bar r_4 (rocker). In order to analyze this mechanism two coordinate systems are established: a system that is fixed to the real world (OXY) and another for self-reference

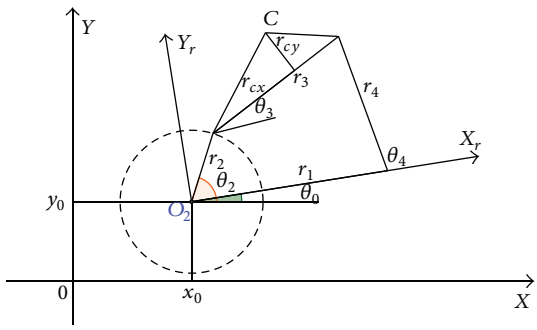

Figure 1: Four-bar mechanism.

(OX_rY_r). (x_0, y_0) is the distance between the origin points of both systems, θ_0 is the rotation angle of the reference system, and θ_i $(i = 2, 3, 4)$ corresponds to the angle for every bar in the mechanism; finally, the coordinate pair (r_{cx}, r_{cy}) determines the position C of the coupler.

2.1. Kinematics of the Mechanism. The kinematics of four-bar mechanisms have been extensively treated; a detailed explanation is found in [36, 37]. For analyzing the mechanism position, the closed loop equation can be established as follows:

$$\overrightarrow{r_1} + \overrightarrow{r_4} = \overrightarrow{r_2} + \overrightarrow{r_3}. \tag{2}$$

Applying polar notation to each term of (2),

$$r_1 e^{j\theta_1} + r_4 e^{j\theta_4} = r_2 e^{j\theta_2} + r_3 e^{j\theta_3}. \tag{3}$$

Using the equation of Euler on (3) and separating the real and imaginary parts,

$$\begin{aligned} r_1 \cos\theta_1 + r_4 \cos\theta_4 &= r_2 \cos\theta_2 + r_3 \cos\theta_3, \\ r_1 \sin\theta_1 + r_4 \sin\theta_4 &= r_2 \sin\theta_2 + r_3 \sin\theta_3. \end{aligned} \tag{4}$$

Expressing the equation system (4) in terms of θ_4,

$$\begin{aligned} r_4 \cos\theta_4 &= r_2 \cos\theta_2 + r_3 \cos\theta_3 - r_1 \cos\theta_1, \\ r_4 \sin\theta_4 &= r_2 \sin\theta_2 + r_3 \sin\theta_3 - r_1 \sin\theta_1. \end{aligned} \tag{5}$$

The compact form of Freudenstein's equation is obtained by squaring system (5) and adding its terms as follows:

$$A_1 \cos\theta_3 + B_1 \sin\theta_3 + C_1 = 0, \tag{6}$$

where

$$\begin{aligned} A_1 &= 2r_3 \left(r_2 \cos\theta_2 - r_1 \cos\theta_1\right), \\ B_1 &= 2r_3 \left(r_2 \sin\theta_2 - r_1 \sin\theta_1\right), \\ C_1 &= r_1^2 + r_2^2 + r_3^2 - r_4^2 - 2r_1 r_2 \cos\left(\theta_1 - \theta_2\right). \end{aligned} \tag{7}$$

Table 1: Sign of radical in relation to the type of mechanism.

Configuration	θ_3	θ_4
Open	$+\sqrt{}$	$-\sqrt{}$
Crossed	$-\sqrt{}$	$+\sqrt{}$

Then the angle θ_3 can be calculated as a function of the parameters A_1, B_1, C_1, and θ_2; this solution is generated by expressing $\sin\theta_3$ and $\cos\theta_3$ in terms of $\tan(\theta_3/2)$:

$$\begin{aligned} \sin\theta_3 &= \frac{2\tan\left(\theta_3/2\right)}{1 + \tan^2\left(\theta_3/2\right)}, \\ \cos\theta_3 &= \frac{1 - \tan^2\left(\theta_3/2\right)}{1 + \tan^2\left(\theta_3/2\right)}. \end{aligned} \tag{8}$$

A second-order lineal equation is obtained by substitution on (6):

$$\begin{aligned} &\left[C_1 - A_1\right] \tan^2\left(\frac{\theta_3}{2}\right) + \left[2B_1\right] \tan\left(\frac{\theta_3}{2}\right) + A_1 + C_1 \\ &\quad = 0. \end{aligned} \tag{9}$$

From the solution of (9), the angular position θ_3 is given by (10):

$$\theta_3 = 2\arctan\left[\frac{-B_1 \pm \sqrt{B_1^2 + A_1^2 - C_1^2}}{C_1 - A_1}\right]. \tag{10}$$

A similar process is carried out to get θ_4 from (4) using Freudenstein's equation. The correct sign for the radical must be selected in the equations for θ_3 and θ_4, according to the configuration of the mechanism. Table 1 indicates the signs related with both configurations.

2.2. Kinematics of the Coupler. Since the point of interest in the coupler is C, to determine its position in the reference system OX_rY_r it has to be established that

$$\begin{aligned} C_{xr} &= r_2 \cos\theta_2 + r_{cx} \cos\theta_3 - r_{cy} \sin\theta_3, \\ C_{yr} &= r_2 \sin\theta_2 + r_{cx} \sin\theta_3 + r_{cy} \cos\theta_3. \end{aligned} \tag{11}$$

In the global coordinate system, this point is expressed as

$$\begin{bmatrix} C_x \\ C_y \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 \\ \sin\theta_0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} C_{xr} \\ C_{yr} \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}. \tag{12}$$

Equations (11) and (12) and the expressions from the kinematics of the mechanism are sufficient to calculate the position of C along the trajectory.

2.3. Design Constraints. One of the most important aspects involved in a mechanism design is to accomplish the constraints on its performance, which are related to mobility criteria and the size and shape of the mechanism itself.

2.3.1. Grashof's Law. Grashof's law is a fundamental consideration when designing a four-bar mechanism, since it defines the criteria to ensure complete mobility for at least one link of that mechanism. This law establishes that *for a planar four-bar linkage, the sum of the shortest and the largest bars cannot be larger than the sum of the remaining bars, if a continual relative rotation between two elements is desired* [8]. If s is the length of the shortest link, l represents the largest bar, and p, q indicate the remaining elements, it is established that

$$l + s \leq p + q. \tag{13}$$

In this work, Grashof's law is given by

$$r_1 + r_2 \leq r_3 + r_4. \tag{14}$$

Therefore, to ensure that the solution method fulfills this law, the following constraints were established:

$$\begin{aligned} r_2 &< r_3, \\ r_3 &< r_4, \\ r_4 &< r_1. \end{aligned} \tag{15}$$

2.3.2. Sequence of Input Angles. Since the general problem of synthesis addressed in this work is the generation of trajectories based on sequences of successive precision points representing different positions of the coupler, the values of the crank angles have to be ordered in correspondence with these sequences. If the angle for a specific point i is denoted as θ_2^i, it is required that

$$\theta_2^1 < \theta_2^2 < \cdots < \theta_2^K, \tag{16}$$

where K is the number of precision points.

2.4. Optimization Strategies. After properly establishing the kinematics of the mechanism, the design problem can be defined as a numerical optimization case, and then it is necessary to specify the appropriate mathematical expressions for evaluating the performance of the system.

2.4.1. Objective Function. This work addresses the synthesis of a planar mechanism in order to calculate the length of its bars, the rotation angle in respect to the reference system, the distance between the coordinate systems, and the set of angles for the input bar to generate a trajectory corresponding to a sequence of precision points. In the global coordinate system OXY, the point of the precision pair C_d^i is indicated as

$$C_d^i = \left[C_{xd}^i, C_{yd}^i\right]^T. \tag{17}$$

The set of K pairs of precision points is defined as

$$\Omega = \left\{C_d^i \mid i \in K\right\}. \tag{18}$$

Then, given a set of values of the mechanism bars and their parameters x_0, y_0, θ_0, each point of the coupler can be expressed as a function of the input bar position:

$$C^i = \left[C_x\left(\theta_2^i\right), C_y\left(\theta_2^i\right)\right]^T. \tag{19}$$

Accordingly, it is desired to minimize the distance (error) between the precision point C_d^i and the calculated point C^i. To quantify the overall error the following function is proposed:

$$\text{error} = \sum_{i=1}^{K}\left[\left(C_{xd}^i - C_x^i\right)^2 + \left(C_{yd}^i - C_y^i\right)^2\right]. \tag{20}$$

2.4.2. Case Studies

(1) M01. It is the design of a four-bar mechanism that follows a linear vertical path defined by a sequence of six precision points, without a previously established synchronization. The set of precision points is defined as

$$\begin{aligned} \Omega = \{&(20, 20), (20, 25), (20, 30), (20, 35), (20, 40), \\ &(20, 45)\}. \end{aligned} \tag{21}$$

The vector of design variables is

$$\begin{aligned} \vec{p} = \{&p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, \\ &p_{15}\}, \end{aligned} \tag{22}$$

where

$$\vec{p} = \left\{r_1, r_2, r_3, r_4, r_{cx}, r_{cy}, \theta_0, x_0, y_0, \theta_2^1, \theta_2^2, \theta_2^3, \theta_2^4, \theta_2^5, \theta_2^6\right\}. \tag{23}$$

The first four variables correspond to the lengths of the bars in the mechanism presented in Figure 1, the following two are the position of the coupler, θ_0 is the orientation angle of the system with respect to the horizontal, $O_2 = (x_0, y_0)$ is its coordinate position, and the last six are the angle values for the input bar r_2. The boundaries for each design variable are defined as

$$\begin{aligned} p_1, p_2, p_3, p_4 &\in [0, 60], \\ p_5, p_6, p_8, p_9 &\in [-60, 60], \\ p_7, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15} &\in [0, 2\pi]. \end{aligned} \tag{24}$$

The single-objective numerical optimization problem for this case is described by the following:

$$\begin{aligned} \min \quad & f(\vec{p}) \\ & = \sum_{i=1}^{N}\left[\left(C_{xd}^i - C_x^i\right)^2 + \left(C_{yd}^i - C_y^i\right)^2\right], \\ & \qquad p \in \mathbb{R}^{15} \\ \text{subject to:} \quad & g_1(\vec{p}) = p_1 + p_2 - p_3 - p_4 \leq 0, \\ & g_2(\vec{p}) = p_2 - p_3 \leq 0, \\ & g_3(\vec{p}) = p_3 - p_4 \leq 0, \\ & g_4(\vec{p}) = p_4 - p_1 \leq 0, \\ & g_5(\vec{p}) = p_{10} - p_{11} \leq 0, \\ & g_6(\vec{p}) = p_{11} - p_{12} \leq 0, \\ & g_7(\vec{p}) = p_{12} - p_{13} \leq 0, \end{aligned}$$

$$g_8(\vec{p}) = p_{13} - p_{14} \leq 0,$$
$$g_9(\vec{p}) = p_{14} - p_{15} \leq 0. \tag{25}$$

(2) M02. It is the design of a four-bar mechanism that follows a trajectory defined by a sequence of five unaligned precision points, with a previously established synchronization for each point. The set of precision points is defined as

$$\Omega = \{(3,3), (2.759, 3.363), (2.372, 3.663), (1.89, 3.862), (1.355, 3.943)\}. \tag{26}$$

For this case it is considered that $x_0, y_0, \theta_0 = 0$. The restriction given by (16) is not considered since the sequence of input angles is set by

$$\theta_2^i = \left\{\frac{2\pi}{12}, \frac{3\pi}{12}, \frac{4\pi}{12}, \frac{5\pi}{12}, \frac{6\pi}{12}\right\}. \tag{27}$$

The vector of design variables is

$$\vec{p} = \{p_1, p_2, p_3, p_4, p_5, p_6\}, \tag{28}$$

where

$$\vec{p} = \{r_1, r_2, r_3, r_4, r_{cx}, r_{cy}\}. \tag{29}$$

The upper and lower values for the design variables are defined as

$$p_1, p_2, p_3, p_4 \in [0, 50], \quad p_5, p_6 \in [-50, 50]. \tag{30}$$

The objective function is defined by the following:

$$\min \quad f(\vec{p}) = \sum_{i=1}^{N} \left[\left(C_{xd}^i - C_x^i\right)^2 + \left(C_{yd}^i - C_y^i\right)^2\right], \quad p \in \mathbb{R}^6 \tag{31}$$

$$\begin{aligned} \text{subject to:} \quad & g_1(\vec{p}) = p_1 + p_2 - p_3 - p_4 \leq 0, \\ & g_2(\vec{p}) = p_2 - p_3 \leq 0, \\ & g_3(\vec{p}) = p_3 - p_4 \leq 0, \\ & g_4(\vec{p}) = p_4 - p_1 \leq 0. \end{aligned} \tag{32}$$

(3) M03. It is the design of a four-bar mechanism for tracking a trajectory delimited by pairs of precision points. This case considers a sequence with ten pairs of precision points given by the coordinates shown in Table 2.

The vector of design is

$$\vec{p} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}\}, \tag{33}$$

Table 2: Pairs of precision points for case study 3.

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

where

$$\vec{p} = \left\{r_1, r_2, r_3, r_4, r_{cx}, r_{cy}, \theta_0, x_0, y_0, \theta_2^1, \ldots, \theta_2^{10}\right\} \tag{34}$$

and its variables are limited by

$$p_1, p_2, p_3, p_4 \in [0, 60], \quad p_5, p_6, p_8, p_9 \in [-60, 60], \quad p_7, p_{10} \cdots p_{19} \in [0, 2\pi]. \tag{35}$$

Because in this case the trajectory is defined by pairs of precision points, the objective function in (20) is modified in order to consider the error with respect to each point. Therefore, the new function is given by the following:

$$\begin{aligned} \min \quad & f(\vec{p}) = \text{error}_1 + \text{error}_2, \\ & \text{error}_1 = \sum_{i=1}^{K} \left[\left(C_{1xd}^i - C_x^i\right)^2 + \left(C_{1yd}^i - C_y^i\right)^2\right], \\ & \text{error}_2 = \sum_{i=1}^{K} \left[\left(C_{2xd}^i - C_x^i\right)^2 + \left(C_{2yd}^i - C_y^i\right)^2\right] \end{aligned} \tag{36}$$

$$\begin{aligned} \text{subject to:} \quad & g_1(\vec{p}) = p_1 + p_2 - p_3 - p_4 \leq 0, \\ & g_2(\vec{p}) = p_2 - p_3 \leq 0, \\ & g_3(\vec{p}) = p_3 - p_4 \leq 0, \\ & g_4(\vec{p}) = p_4 - p_1 \leq 0, \\ & g_5(\vec{p}) = p_{10} - p_{11} \leq 0, \\ & g_6(\vec{p}) = p_{11} - p_{12} \leq 0, \\ & g_7(\vec{p}) = p_{12} - p_{13} \leq 0, \\ & g_8(\vec{p}) = p_{13} - p_{14} \leq 0, \end{aligned}$$

$$g_9(\vec{p}) = p_{14} - p_{15} \leq 0,$$
$$g_{10}(\vec{p}) = p_{15} - p_{16} \leq 0,$$
$$g_{11}(\vec{p}) = p_{16} - p_{17} \leq 0,$$
$$g_{12}(\vec{p}) = p_{17} - p_{18} \leq 0,$$
$$g_{13}(\vec{p}) = p_{18} - p_{19} \leq 0. \quad (37)$$

Finally, it is important to note that the complexity of the study cases presented in the paper is high, due to two aspects. (1) A large number of precision points that must touch the mechanism: in the state of the art of synthesis mechanisms, cases with four precision points maximum are solved using graphic methods or MPM's. (2) Values of design variables θ_i^2 with $i = 1, \ldots, 6$ for case $M01$ and $i = 1, \ldots, 10$ for case $M02$: these must have an ascending or descending order, which implies a strong constraint for finding solution vectors to ensure a proper mechanism functioning. Additionally, cases $M01$ and $M02$ have not previously undergone a synchronization on the mechanism input bar.

3. Two-Swim Modified Bacterial Foraging Optimization Algorithm (TS-MBFOA)

TS-MBFOA is inspired by the ideas of IMBFOA, a recently proposed BFOA-based algorithm to solve CNOPs by using two-swim operators, a skew mechanism for the initial swarm of bacteria, a second-order local search operator, and a limited usage of the reproduction step [32]. To get a self-contained paper, in the next subsections, MBFOA, IMBFOA's base algorithm, is presented. After that, IMBFOA is detailed. Finally, TS-MBFOA is introduced.

3.1. Modified Bacterial Foraging Optimization Algorithm (MBFOA). MBFOA is based on the original BFOA [16], but it was proposed to solve CNOPs. Each one of its elements is detailed as follows.

(i) A bacterium i represents a potential solution to the CNOP (i.e., a n-dimensional real-value vector identified as $\vec{x}$ in Section 1), and it is denoted as $\theta^i(j, G)$, where j is its chemotaxis loop index and G is a generational (cycle) loop index. Within a cycle, three inner processes are carried out: chemotaxis, reproduction, and elimination-dispersal. Swarming process is added to the chemotaxis process.

(ii) *Chemotaxis*. In this process, each bacterium in the current swarm performs a tumble-swim movement. The tumble, as proposed by Passino [16], consists of a search direction $\phi(i)$ generated at random with uniform distribution as presented in the following:

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}}, \quad (38)$$

where $\Delta(i)$ is a n-dimensional real-value vector generated at random with uniform distribution where each one of its elements has values between $[-1, 1]$.

The swim allows the bacterium $\theta^i(j, G)$ to follow the search direction and move to a new position $\theta^i(j+1, G)$. The swim is computed as indicated in the following:

$$\theta^i(j+1, G) = \theta^i(j, G) + C(i)\phi(i), \quad (39)$$

where $C(i)$ is the step size vector and its values are calculated by considering the limits of each design variable k, defined by the expression in the following:

$$C(i)_k = R * \left(\frac{\Delta x_k}{\sqrt{n}}\right), \quad k = 1, \ldots, n, \quad (40)$$

where Δx_k is the difference between the upper and lower limits of each variable x_k: $U_k - L_k$, n is the number of variables, and $R \in [0, 1]$ is a user-defined parameter that scales the step size value of bacteria. This vector remains fixed during the search process. MBFOA, like other BFOA-based algorithms as shown in [26], is particularly sensitive to this parameter and such behavior has motivated further studies as that in [29].

If the new position, $\theta^i(j+1, G)$, is better with respect to the previous position $\theta^i(j, G)$ [28] (i.e., (1) both positions are feasible, but the new position has a better objective function value, (2) the new position is feasible while the previous one is not, or (3) both positions are infeasible, but the new position has a lower sum of constraint violation), another swim in the same direction will be carried out by taking this better solution as the new starting position. Otherwise, a new tumble is computed. The process stops after N_c attempts (parameter defined by the user).

(iii) *Swarming*. MBFOA includes an attractor movement within the chemotaxis process, which lets each bacterium in the swarm follow the bacterium located in the most promising region of the search space, that is, either the feasible bacterium with the best objective function value or the bacterium with the lowest sum of constraint violation if no feasible bacteria are found in the current swarm. Such information is given by the three feasibility rules used in the chemotaxis process. The movement is detailed in the following:

$$\theta^i(j+1, G) = \theta^i(j, G) + \beta\left(\theta^B(G) - \theta^i(j, G)\right), \quad (41)$$

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 cycle G, and β (user-defined parameter) 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 is applied once within the chemotaxis loop. In the remaining steps, the tumble-swim movement is used. Both the tumble-swim and swarming movements can generate variable values outside their limits. Therefore, a simple repair mechanism is used as in [38], where the violated value is multiplied by 2 and the violated limit is subtracted (i.e., $x_{\text{valid}} = 2 * \text{violated\_limit} - x_{\text{invalid}}$).

(iv) *Reproduction*. The swarm is sorted based on the same three rules adopted in the chemotaxis process and the first S_r are cloned (these bacteria are considered as the best ones), and the remaining $S_b - S_r$ (the worst bacteria) are eliminated (S_b is the swarm size).

(v) *Elimination-Dispersal*. This process eliminates only the worst bacterium $\theta^w(j, G)$ based on the already mentioned feasibility rules, and a new randomly generated bacterium is inserted as a replacement.

In Algorithm 1 the corresponding MBFOA pseudocode is presented, and its user-defined parameters are summarized in the caption.

3.2. Improved MBFOA (IMBFOA). IMBFOA was designed to improve MBFOA in its performance to solve CNOPs. Four changes were promoted: (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 based on sequential quadratic programming, and (4) a reduction on the usage of the reproduction process. Below is the description of each change.

(i) *Two-Swim Operators*. In the chemotaxis process, instead of the $[-1, 1]$ interval, the range for the tumble was set to $[\upsilon, \tau]$, where υ and τ are user-defined parameters, $-1 \leq \upsilon < 0, 0 < \tau \leq 1$.

The first swim, focused on exploration, is computed as indicated in the following:

$$\theta^i(j+1, G) = \theta^i(j, G) + \phi(i), \tag{42}$$

where $\phi(i)$ is computed as in (38), but now considering the updated range and not using the step size vector. The second swim, focused on exploitation, is computed as indicated in the following:

$$\theta^i(j+1, G) = \theta^i(j, G) + C(i, G)\,\phi(i), \tag{43}$$

where $C(i, G)$ is a dynamic step size vector [29]. However, each value k of vector $C(i, G)$ decreases dynamically at each cycle of the algorithm as indicated in the following:

$$C(i, G+1)_k = C(i, G)_k \frac{G}{G\text{MAX}}, \quad k = 1, \ldots, n, \tag{44}$$

where $C(i, G+1)_k$ is the new step size value for variable k, while G and GMAX are the current and maximum number of cycles of the algorithm, respectively. The initial $C(i, 0)$ is computed as indicated in (40), but the R parameter is no longer used.

The first swim is applied until no improvement is obtained, and then the second swim takes place and so on. The process stops, as in the chemotaxis process in MBFOA, after N_c attempts.

(ii) *Skew Mechanism for the Initial Swarm*. The initial swarm of bacteria S_b is generated by considering three groups. In the first group there are randomly generated bacteria but with their location skewed to the lower limit of the decision variables L_k. In the second group there are randomly generated bacteria but with their location skewed to the upper limit of the decision variables U_k. Finally, a third group of randomly generated bacteria are created as in the original MBFOA (i.e., without any skew). The three groups use random values with uniform distribution. The details to set the limits per variable for the first and second group are presented in the following:

$$\begin{aligned} &\left[L_k, L_k + \left(\frac{(U_k - L_k)}{\text{ss}}\right)\right], \\ &\left[U_k - \left(\frac{(U_k - L_k)}{\text{ss}}\right), U_i\right], \end{aligned} \tag{45}$$

where ss is the skew size. A high value decreases the skew effect, while a low value increases it.

(iii) *Local Search Operator*. Sequential Quadratic Programming (SQP) [39] is the local search operator in IMBFOA. This search is applied to the best bacterium in the swarm after the chemotaxis, swarming, reproduction, and elimination-dispersal processes. The user can define the local search operator usage frequency with the LS_G parameter.

(iv) *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.

Algorithm 2 includes the IMBFOA pseudocode and its parameters are in the caption.

3.3. Two-Swim MBFOA (TS-MBFOA). TS-MBFOA revisits the two swims originally proposed in IMBFOA to enhance its search capabilities and to simplify them as well. In this way, two new swims to be applied within the chemotaxis process are proposed in this work. The first one of them aims to complement the swarming operator by letting a bacterium to explore other areas of the search space with the guide of randomly chosen bacteria. The second swim focuses on slight movements of the bacterium in its vicinity by using the original swim proposed by Passino [16], but with very small step size values.

The details of each one of the two proposed swims are presented below.

(1) Create an initial swarm of bacteria at random $\theta^i(j,0)\ \forall i,\ i = 1,\ldots,S_b$
(2) Evaluate each $\theta^i(j,0)\ \forall i, i = 1,\ldots,S_b$
(3) **for** $G = 1$ to GMAX **do**
(4) **for** $i = 1$ to S_b **do**
(5) **for** $j = 1$ to N_c **do**
(6) Perform the *chemotaxis process* (tumble-swim) with (38), (39) and the *attractor operator* in (41) for bacteria $\theta^i(j,G)$ by considering the three feasibility rules as selection criteria
(7) **end**
(8) **end**
(9) Perform the *reproduction process* by sorting all bacteria in the swarm based on the feasibility rules, duplicating the S_r best bacteria and eliminating the remaining $S_b - S_r$
(10) Perform the *elimination-dispersal process* by eliminating the worst bacterium $\theta^w(j,G)$ in the current swarm
(11) **end**

Algorithm 1: MBFOA. 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 cycles (generations) GMAX.

(1) Create an initial swarm of bacteria *by using the skew mechanism* $\theta^i(j,0)\ \forall i,\ i = 1,\ldots,S_b$
(2) Evaluate $\theta^i(j,0)\ \forall i,\ i = 1,\ldots,S_b$
(3) **for** $G = 1$ to GMAX **do**
(4) **for** $i = 1$ to S_b **do**
(5) **for** $j = 1$ to N_c **do**
(6) Perform the *chemotaxis process by using the two swims* in (42) and (43) and the attractor operator in (41) for bacteria $\theta^i(j,G)$ by considering the three feasibility rules as selection criteria
(7) **end**
(8) **end**
(9) **if** (G mod RepCycle == 0) **then**
(10) Perform the *reproduction process* by sorting all bacteria in the swarm based on the feasibility rules, duplicating the S_r best bacteria and eliminating the remaining $S_b - S_r$
(11) **end**
(12) Perform the *elimination-dispersal process* by eliminating the worst bacterium $\theta^w(j,G)$ in the current swarm
(13) *Update the step size vector* with (44)
(14) **if** (G mod LS_G == 0) **then**
(15) Apply the local search operator (i.e., SQP) to the best bacterium in the swarm. If the obtained bacterium is better than the original best bacterium, it takes its place in the swarm.
(16) **end**
(17) **end**

Algorithm 2: 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.

(1) Exploration Swim. The first swim is computed as indicated in the following:

$$\theta^i\left(j+1,G\right) = \theta^i\left(j,G\right) + \beta - 1\left(\theta^{r_1}\left(j,G\right) - \theta^{r_2}\left(j,G\right)\right), \tag{46}$$

where β is the user-defined parameter utilized in MBFOA's swarming operator and its value is now greater than 1. $\theta^{r_1}(j,G)$ and $\theta^{r_2}(j,G)$ are two bacteria randomly selected from the swarm ($i \neq r_1 \neq r_2$). This swim operator uses the position of such two bacteria to determine a search direction considering the current position of the bacterium ready to swim $\theta^i(j,G)$ as the starting point.

Figure 2 shows the behavior of this swim operator using a space of two decision variables, each one into a range of $[-5,5]$. In this example, the new position of the bacterium after the swim will fall in the purple spot defined by bact1 and bact2, which are $\theta_1^r(j,G)$ and $\theta_2^r(j,G)$, respectively. The best bacterium is included so as to remark that this operator aims to find different regions of the search space (i.e., not those on the neighborhood of the best current solution as the swarming movement promotes).

(2) Exploitation Swim. The second swim returns to be original swim based on random search directions but is now coupled with small random step size values to precisely favor fine movements, as indicated in the following:

$$\theta^i\left(j+1,G\right) = \theta^i\left(j,G\right) + C\left(i,G\right)\phi\left(i\right), \tag{47}$$

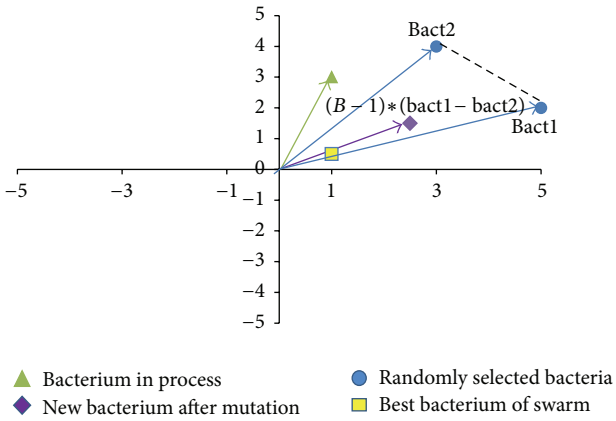

Figure 2: Graphical example of the first exploration swim.

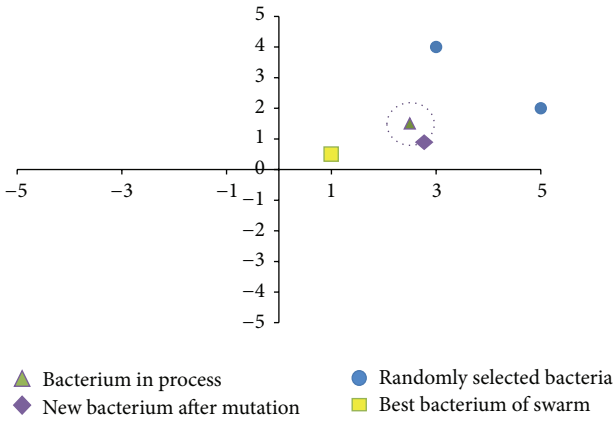

Figure 3: Random swim behavior.

where the step size values comprise a n-dimensional random vector called again $C(i, G)$ [29], calculated at each generation as shown in the following:

$$C(i,G)_k = R * \Delta_{(i_k)}, \quad k = 1, \ldots, n, \tag{48}$$

where $\Delta_{(i_k)}$ is a randomly generated value with uniform distribution within $[L_k, U_k]$ of decision variable k. R is a user-defined parameter to scale the step size, and its value should be close to zero, for example, $5.00E-03$. At the first cycle, the step size is calculated using just $\Delta_{(i_k)}$ to allow bacteria in the initial swarm to move in different directions within the search space while avoiding attractors at the start of the process, as suggested in [40].

Figure 3 shows the swim behavior, where the bacterium represented as a green triangle will move by using a random search direction but close to its current position, regardless of the positions of other bacteria in the swarm.

Algorithm 3 presents TS-MBFOA pseudocode, and its parameters are detailed in the caption.

The combined expected effect of both proposed swims with the swarming operator, all three inside the chemotaxis process, is an enhanced ability to avoid local optimum solutions and a promotion of a faster convergence. Such effect is possible because of the fact that TS-MBFOA has a swim for exploration (first proposed swim), a swim to favor convergence (swarming operator from MBFOA), and a fine-swim to further improve good quality solutions (second proposed swim).

Finally, it is important to remark that the local search based on SQP is not used in TS-MBFOA. Therefore, second-order information is not required as it was the case with IMBFOA.

Table 3: Main features of each four-bar synthesis problem. Max_Evals is the maximum number of evaluations allowed per each of the problems, n is the number of design variables, and c is the number of constraints.

Problem	Max_Evals	n	c
$M01$	500,000	15	5
$M02$	100,000	6	4
$M03$	500,000	19	5

4. Results and Analysis

TS-MBFOA was used to solve the three four-bar synthesis design problems stated in Section 2. A summary of their main features is presented in Table 3. Four experiments were designed to (1) assess the effectiveness of the proposed swims compared with the swims in IMBFOA and MBFOA, (2) compare the final results of TS-MBFOA against those of IMBFOA and MBFOA, (3) compare the final results of TS-MBFOA now against those obtained by a DE-based approach to solve mechanical engineering problems [41], and (4) simulate the best four-bar systems obtained by each one of the three algorithms in each one of the optimization problems to analyze their behavior from a mechanical point of view. The Wilcoxon Signed-Rank Test (WSRT) [42] was used to validate the differences observed in the samples of 30 independent runs computed per algorithm per test problem in the experiments. TS-MBFOA was coded in MATLAB R2009b and executed on a PC with a 3.5 Core 2 Duo Processor, 4 GB of RAM, and 64-bit Windows 7 operating system.

4.1. Performance Measures. To evaluate the behavior of the compared algorithms, the following performance measures for nature-inspired constrained optimization, taken from [43], were computed:

(i) *Feasible run*: a run where at least one feasible solution is found within Max_Evals.

(ii) *Feasible rate* = (number of feasible runs)/total runs.

(iii) *Successful swim*: A swim movement where the new position is better (based on the feasibility rules) than the original position.

(iv) *Successful swim rate* = (number of successful swims)/total swims, where total swims = $S_b \times N_c \times GMAX$.

4.2. Parameter Setting. The parameter setting for nature-inspired algorithms is an open problem [44]. Therefore, to get suitable parameter values for the proposed algorithm,

(1) Create an initial swarm of bacteria by using the skew mechanism $\theta^i(j,0)\ \forall i, i = 1,\ldots,S_b$
(2) Evaluate $\theta^i(j,0)\ \forall i,\ i = 1,\ldots,S_b$
(3) **for** $G = 1$ to GMAX **do**
(4) **for** $i = 1$ to S_b **do**
(5) **for** $j = 1$ to N_c **do**
(6) Perform the *chemotaxis process by using the two swims* in (46) and (47) and the attractor operator in (41) for bacteria $\theta^i(j,G)$ by considering the three feasibility rules as selection criteria
(7) **end**
(8) **end**
(9) **if** (G mod RepCycle == 0) **then**
(10) Perform the reproduction process by sorting all bacteria in the swarm based on the feasibility rules, duplicating the S_r best bacteria and eliminating the remaining $S_b - S_r$
(11) **end**
(12) Perform the elimination-dispersal process by eliminating the worst bacterium $\theta^w(j,G)$ in the current swarm
(13) *Calculated the step size vector* with (48)
(14) **end**

Algorithm 3: TS-MBFOA pseudocode. Input parameters are number of bacteria S_b, chemotaxis loop limit N_c, scaling stepsize R, number of bacteria for reproduction S_r, scaling factor β, the reproduction cycle RepCycle, ss for the skew mechanism, and the number of cycles GMAX.

Table 4: Parameter values for the three BFOA-based compared algorithms. "—" indicates that the corresponding parameter is not required by the algorithm located in the column.

Parameter	MBFOA	IMBFOA	TS-MBFOA
S_b	40	20	60
N_c	24	20	10
R	$1.2E-02$	—	$5.00E-03$
S_r	1	2	1
β	1.75	1.5	1.75
RepCycle	—	100	100
GMAX	Value to reach Max_Evals	Value to reach Max_FEs	Value to reach Max_FEs
LS_G	—	1 and (GMAX/2) generations	—
υ	—	0.15	—
τ	—	−0.25	—
ss	—	8	8

a tuning process was carried out by the iRace tool [45]. iRace implements the iterated racing procedure for automatic algorithm configuration. Iterated racing is a generalization of the iterated F-race and consists of three phases: (1) sampling new parameter configurations with a particular distribution, (2) choosing the most competitive configurations by means of racing, and (3) updating the sampling distribution to favor better configurations. For details about iRace the reader is referred to [45].

The user-defined parameter of TS-MBFOA is shown in Table 4. The parameter values for MBFOA and IMBFOA were taken from [29] and [32], respectively.

The set of parameters for TS-MBFOA in Table 4 is one out of four sets provided by iRace. The other three are the following: (1) $S_b = 60$, $N_c = 12$, $R = 1.89E-2$, $S_r = 1$, $\beta = 1.32$, RepCycle = 100, ss = 5, (2) $S_b = 40$, $N_c = 22$, $R = 1.50E+0$, $S_r = 2$, $\beta = 1.75$, RepCycle = 80, ss = 8, and (3) $S_b = 40$, $N_c = 24$, $R = 1.34E-1$, $S_r = 5$, $\beta = 1.54$, RepCycle = 100, ss = 8. As it can be seen, four parameters in the sets have different values: N_c, R, S_r, and β. This suggests that TS-MBFOA is not very sensitive to those parameters. From those four parameters, R has been reported as very sensitive in previous MBFOA versions [29]. However, TS-MBFOA shows less sensitivity to its value. On the other hand, the parameters with similar values in the sets are S_b, RepCycle, and ss. This suggests that TS-MBFOA requires a more careful tuning of the swarm size, the reproduction frequency, and the initial skew in the population. However, the tuning process could deal with such sensitivity.

4.3. Experiment 1: Effectiveness of the Proposed Swims. The number of successful swims per generation obtained by TS-MBFOA, IMBFOA, and MBFOA on the three four-bar synthesis problems ($M01$, $M02$, and $M03$) is presented in Figures 4, 5, and 6, where the run located in the median value of 30 independent runs is plotted. In the three figures, the effectiveness of the two proposed swims included in TS-MBFOA was superior in most of the process and particularly

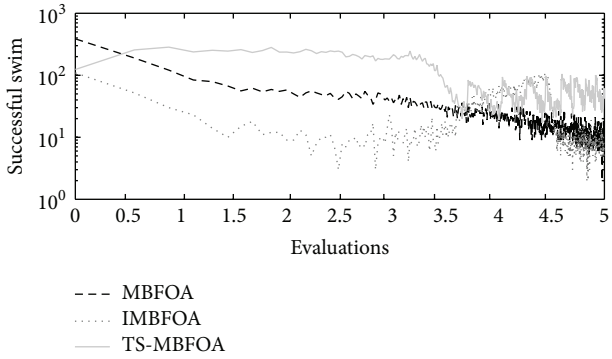

Figure 4: Successful swims by MBFOA, IMBFOA, and TS-MBFOA in $M01$ problem in the execution located in the median value of 30 independent runs.

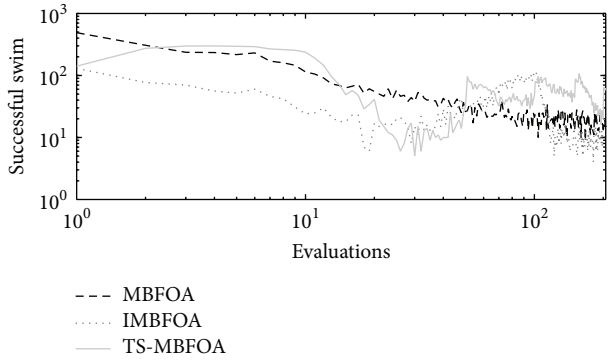

Figure 5: Successful swims by MBFOA, IMBFOA, and TS-MBFOA in $M02$ problem in the execution located in the median value of 30 independent runs.

late in the search. On the other hand, the number of successful swims in MBFOA showed a decreasing tendency in the three synthesis problems. Finally, the number of successful swims in IMBFOA was the lowest in the three problems but showed some improvement at the end of the search, but it did not outperform those successful swims by TS-MBFOA. To provide further evidence to the above finding, the successful swim rates in problem $M01$ were 3.68% by MBFOA, 2.16% by IMBFOA, and 7.26% by TS-MBFOA. In problem $M02$ the successful swim rates were 2.31%, 3.76%, and 4.36%, by MBFOA, IMBFOA, and TS-MBFOA, respectively. Finally, in problem $M03$, the rates were 2.14%, 3.76% and 6.53% by MBFOA, IMBFOA, and TS-MBFOA, respectively. As a conclusion of this first experiment, the two proposed swims were able to generate a greater number of better solutions during the search, mainly in late generations, unlike the swims in IMBFOA and BFOA. It remains to be seen if such behavior leads to better final results.

4.4. Experiment 2: Final Results Comparison among BFOA-Based Approaches. As a first element of analysis, the feasible rates obtained by TS-MBFOA, IMBFOA, and MBFOA in the three four-bar synthesis problems are shown in Table 5. It is clear that the three algorithms were able to consistently reach the feasible region of the search space.

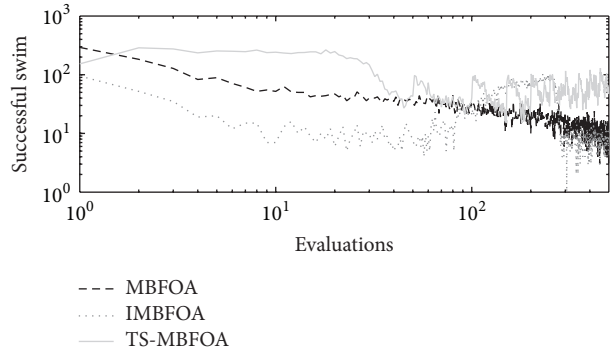

Figure 6: Successful swims by MBFOA, IMBFOA, and TS-MBFOA in $M03$ problem in the execution located in the median value of 30 independent runs.

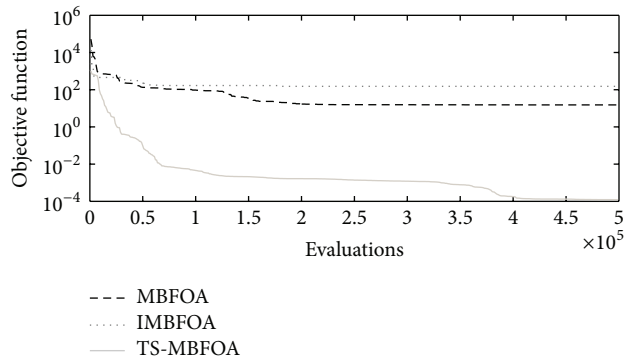

Figure 7: Convergence plots by each BFOA-based algorithm in problem $M01$.

Table 5: Feasible rate obtained by MBFOA, IMBFOA, and TS-MBFOA on 30 independent runs.

Prob.	MBFOA	IMBFOA	TS-MBFOA
$M01$	100%	100%	100%
$M02$	100%	100%	100%
$M03$	100%	100%	100%

The statistical results obtained by MBFOA, IMBFOA, and TS-MBFOA on the three four-bar synthesis problems are presented in Table 6 in terms of best, average, and standard deviation values of 30 independent runs. According to the 95%-confidence Wilcoxon Signed-Rank Test, the differences observed in the samples of runs in Table 6 are significant. Based on such information, TS-MBFOA outperformed IMBFOA and BFOA in the three optimization problems.

To further understand the behavior of each BFOA-based algorithm, the convergence plots for each optimization problem are shown in Figures 7, 8, and 9 using the run located in the median value of the 30 independent runs. To complement the information, in Table 7, the objective function value of the best solution found in such run is presented per algorithm per optimization problem.

Those results suggest that the combination of the two proposed swims allowed TS-MBFOA to avoid local optimum

Table 6: Statistical results obtained in 30 independent runs by MBFOA, IMBFOA, and TS-MBFOA when solving the three four-bar synthesis problems. Best results are remarked in boldface. All differences are significant based on the 95%-confidence Wilcoxon test.

Problem	Stat	MBFOA	IMBFOA	TS-MBFOA
$M01$	Evaluations	500,000	500,000	500,000
	Best	$1.20E+00$	$1.21E-02$	**1.26E − 29**
	Average	$2.50E+01$	$3.38E+01$	**2.40E − 02**
	Std.	$2.57E+01$	$6.93E+01$	**9.15E − 02**
$M02$	Evaluations	100,000	100,000	100,000
	Best	0.002997125	0.003726955	**0.002628079**
	Average	$3.69E+00$	$4.34E-03$	**2.63E − 03**
	Std.	$2.98E-04$	$9.29E-04$	**1.31E − 17**
$M03$	Evaluations	500,000	500,000	500,000
	Best	0.5630512	3.537282325	**0.2750193**
	Average	$1.66E+01$	$1.35E+01$	**1.07E + 00**
	Std.	$2.72E+01$	$1.03E+04$	**1.10E + 00**

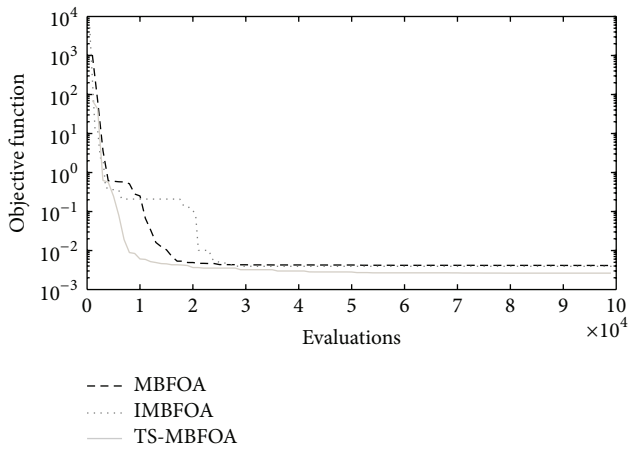

Figure 8: Convergence plots by each BFOA-based algorithm in problem $M02$.

Table 7: Best solution obtained by MBFOA, IMBFOA, and TS-MBFOA in the run located in the median value out of 30 independent runs. Best values are remarked in boldface.

Algorithm	$M01$	$M02$	$M03$
MBFOA	$1.53E+01$	$4.10E-03$	$1.26E+01$
IMBFOA	$1.51E+02$	$3.93E-03$	$7.75E+01$
TS-MBFOA	**1.21E − 04**	**2.63E − 03**	**4.80E − 01**

solutions and find even more promising areas in the feasible region of the search space. In contrast, IMBFOA and MBFOA got trapped in those local attractors. Finally, based on the best solution found in the run located in the median value out of the 30 independent runs, TS-MBFOA was the most consistent algorithm to reach competitive values.

4.5. Experiment 3: Comparison between TS-MBFOA and an Evolutionary Algorithm for Mechanical Design. The results of TS-MBFOA were compared against those obtained by

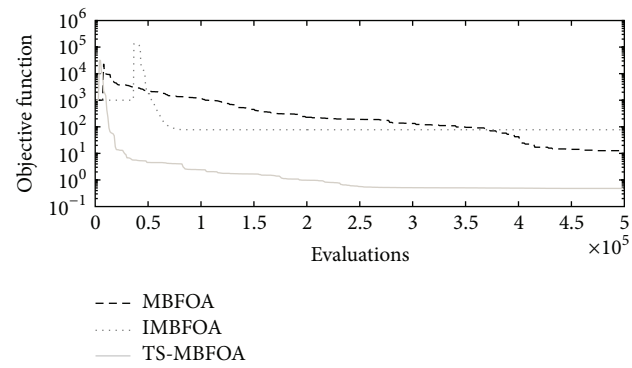

Figure 9: Convergence plots by each BFOA-based algorithm in problem $M03$.

a differential-evolution-based approach designed to solve mechanical design problems [41]. The parameter values used for such algorithm were the following: 100 individuals and 7500 generations for problem $M01$, 100 individuals and 1000 generations for problem $M02$, and 100 individuals and 5000 generations for problem $M03$. F and CR values were randomly generated at each generation within the following intervals: [0.3, 0.9] and [0.8, 1.0], respectively.

Table 8 includes the statistical results of 30 independent runs carried out by TS-MBFOA and the DE-based approach. It is important to mention that in the 30 runs, both algorithms found feasible solutions. Moreover, the 95%-confidence Wilcoxon test indicated that the differences between the algorithms in the final results were not significant.

Despite the fact that no significant differences were observed in the results obtained by TS-MBFOA and the DE-based approach, TS-MBFOA was able to find those competitive results by using a single parameter setting, while the DE-based approach required a fine-tuning for each optimization problem. Furthermore, TS-MBFOA required less evaluations to reach such results in problem $M01$ (500,000 against 750,000 evaluations in problem $M01$).

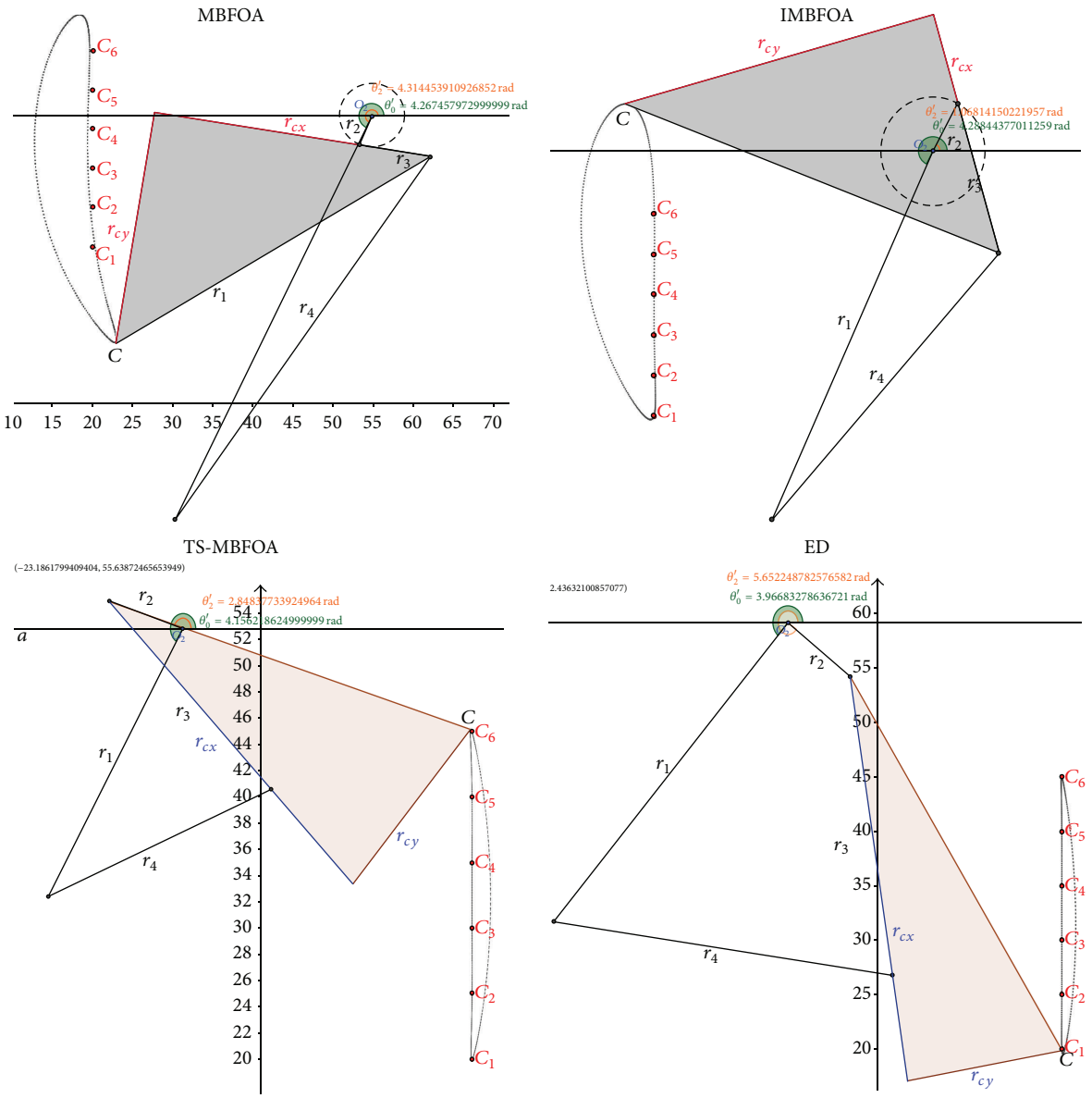

Figure 10: Problem $M01$ best solution simulation.

4.6. Experiment 4: Simulation of Best Solutions. This final experiment simulated the mechanisms corresponding to the best solutions found by MBFOA, IMBFOA, TS-MBFOA, and the DE-based algorithms for the three problems. The results are analyzed from a mechanical point of view. The decision variable values of each best solution per algorithm per optimization problem are presented in Table 9. The graphical representations of the simulations are shown in Figures 10, 11, and 12 for problems $M01$, $M02$, and $M03$, respectively.

Regarding problem $M01$, Figure 10 indicates that the four algorithms found mechanisms (solutions) whose trajectories pass over the six precision points. However, the mechanisms generated by MBFOA and IMBFOA are less efficient in terms of time and energy consumption because their recovering loops to start tracking the points again are much longer than those obtained by TS-MBFOA and the DE-based approach.

The mechanisms provided by TS-MBFOA, the DE-based approach, and MBFOA in problem $M02$ (Figure 11) were equally good from a mechanical point of view, that is, the trajectories of the three mechanisms pass over the five precision points and their elements vary in less than 25% among them. The exception was the mechanism obtained by IMBFOA because it was deficient; that is, its transmission did not pass over the trajectory specified by the five precision points.

For the most complex problem $M03$ (Figure 12), the mechanisms provided by TS-MBFOA and the DE-based approach showed a similar path through the precision point pairs. Furthermore, the length of the bars is quite uniform (see Table 9). In contrast, the mechanism found by MBFOA fails in some precision point pairs and the length of its bars is not as uniform as those of the mechanisms obtained by TS-MBFOA and the DE-based approach. Large bars may

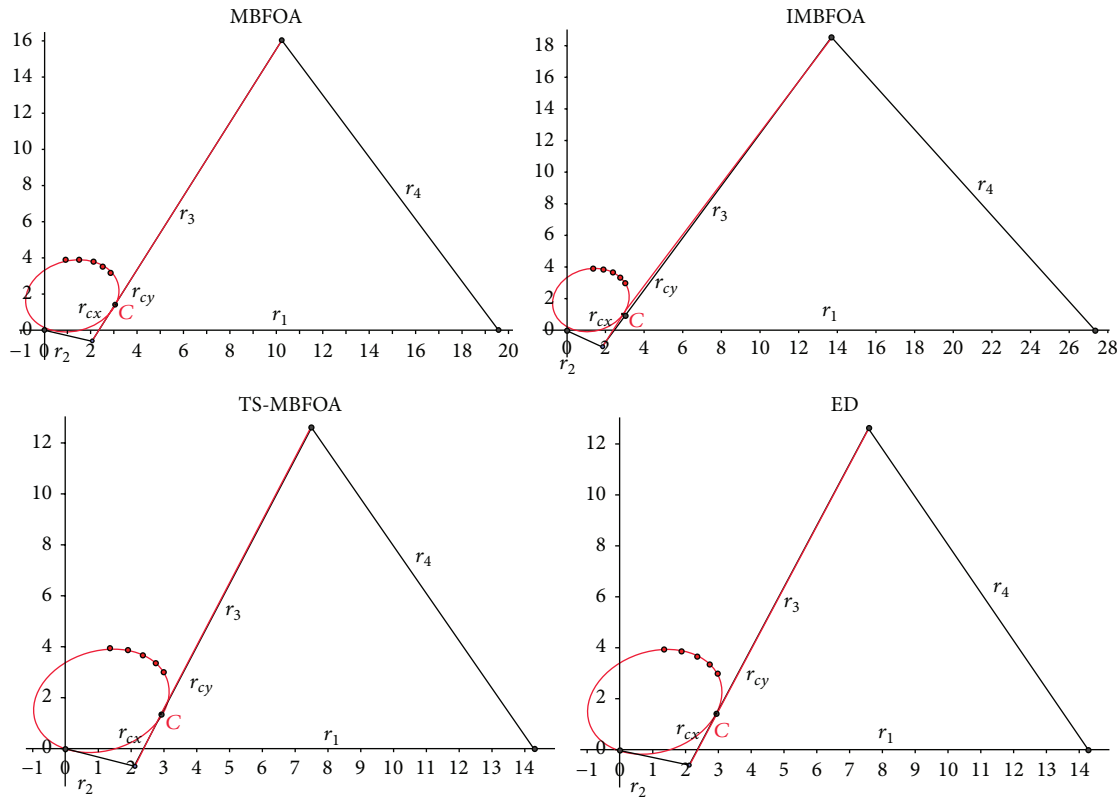

Figure 11: Problem $M02$ best solution simulation.

Table 8: Statistical results obtained in 30 independent runs by TS-MBFOA and the DE-based approach when solving the three four-bar synthesis problems. Best results are remarked in boldface. No significant differences were found based on the 95%-confidence Wilcoxon test.

Problem	Stat	TS-MBFOA	DE
$M01$	Evaluations	500,000	750,000
	Best	**1.26217E − 29**	**1.26218E − 29**
	Average	$2.40E-02$	**1.99E − 03**
	Std.	$9.15E-02$	**5.39E0 − 03**
$M02$	Evaluations	100,000	100,000
	Best	**0.002628079**	**0.002628079**
	Average	**2.63E − 03**	$2.63E-03$
	Std.	**1.31E − 17**	$4.473E-10$
$M03$	Evaluations	500,000	500,000
	Best	0.2750193	**0.274968745**
	Average	**1.07E + 00**	$1.31E+00$
	Std.	**1.10E + 00**	$3.27E+00$

produce undesired effects such as bad alignment, weak balancing because of a transverse flexion produced by internal loads, and a bigger stress between mechanical elements on the tightening points related with the transmission angle of the mechanism. Finally, IMBFOA failed to provide a competitive mechanism based on both number of precision point pairs covered and bar length uniformity.

From this last experiment, the simulation of the best solutions suggests that TS-MBFOA was able to find, from a mechanical point of view, high-quality four-bar mechanisms in the three instances presented in this work. Particularly in problem $M01$, TS-MBFOA was able to find a very competitive solution with less evaluations with respect to the DE-based approach, an algorithm whose performance has been highly competitive when solving mechanical design problems [41].

5. Conclusions and Future Work

This work proposed two-swim operators for the modified bacterial foraging optimization algorithm (TS-MBFOA), to solve three instances of the synthesis of four-bar planar mechanisms: (1) design of a four-bar mechanism that follows a linear vertical path defined by a sequence of six precision points, without a previously established synchronization ($M01$), (2) design of a four-bar mechanism that follows a trajectory defined by a sequence of five unaligned precision points, with a previously established synchronization for each point ($M02$), and (3) design of a four-bar mechanism for tracking a trajectory delimited by a sequence of ten pairs of precision points ($M03$). The first swim favored exploration of the search space by using locations of other bacteria, while the second swim promoted fine movements in the vicinity of the bacterium position by using small step size values.

Table 9: Best feasible solutions found by MBFOA, IMBFOA, TS-MBFOA, and the DE-based algorithms in each four-bar synthesis problem.

Variables	$M01$				$M02$				$M03$			
	MBFOA	IMBFOA	TS-MBFOA	DE	MBFOA	IMBFOA	TS-MBFOA	DE	MBFOA	IMBFOA	TS-MBFOA	DE
x_1	56.96046969	49.93205601	24.0706441	37.35322401	19.59633374	27.35371004	14.31454786	14.31436975	38.12083651	18.60412365	2.632965423	2.614591501
x_2	4.060834527	6.690293689	7.250066369	8.414037276	2.140600536	2.083629796	2.211165812	2.211165657	0.774854018	0.113660019	1.056973251	1.034801583
x_3	8.923787074	19.2082537	20.94483824	27.79863521	18.56633857	22.85644937	14.31454786	14.3143696	19.29477637	5.578117599	1.763888952	1.826884381
x_4	56.08270471	43.78525807	22.57639254	37.00803944	18.5842461	23.02591436	14.31454786	14.31436974	24.68930882	14.30112295	2.205482703	2.207891726
x_5	−25.83505865	−11.38723361	31.53789197	37.61267067	2.239379687	2.286600675	2.174361597	2.174363923	8.333794141	−11.78460943	1.206113171	1.250924301
x_6	−29.75007805	−41.16507909	16.20250155	16.97308561	0.023572299	0.153197064	0.022209768	0.022212213	−6.463530675	−32.31550869	0.363242643	0.447340178
x_7	4.267457973	4.28844377	4.156218625	3.966832786					5.93886	0.033123647	5.813750828	5.826803696
x_8	54.8103088	55.71366791	−7.286682385	−9.678596639					−8.625076191	−8.396013249	0.084076204	0.099169603
x_9	36.62395408	52.75753663	52.85173965	59.18270796					6.384329872	35.28021924	1.437990445	1.328798537
x_{10}	1.120012858	0.082820007	1.24	1.717902384					0.038575927	0.029010814	0.396045164	0.410281221
x_{11}	1.362491164	0.640982496	2.309698447	2.451658775					0.42340301	0.053824757	1.029004626	1.039364984
x_{12}	1.621239336	0.964618395	2.879833244	2.966271497					0.971029032	0.588872867	1.625114804	1.650008311
x_{13}	1.855988457	1.242039407	3.422015391	3.464656513					1.704933689	0.863809971	2.219097833	2.260034392
x_{14}	2.11850087	1.505902613	3.997488978	4.013579749					2.378218281	0.882403592	2.803908752	2.865981127
x_{15}	2.453816162	1.774143421	5.243554444	5.124608233					3.063472839	1.764509265	3.403882079	3.490250618
x_{16}									3.67844246	2.024993472	4.09753211	4.163941276
x_{17}									3.938125257	2.860256871	4.853550562	4.905546281
x_{18}									4.286898695	4.250445558	5.380182303	5.416480223
x_{19}									5.042731775	4.584225722	6.052924519	6.06760838
$f(\vec{x})$	1.206824913	0.012052859	**1.26217E − 29**	**1.26218E − 29**	0.002997125	$3.73E-03$	**0.002628079**	**0.002628079**	0.5630512	3.537282325	0.275019399	**0.274968745**

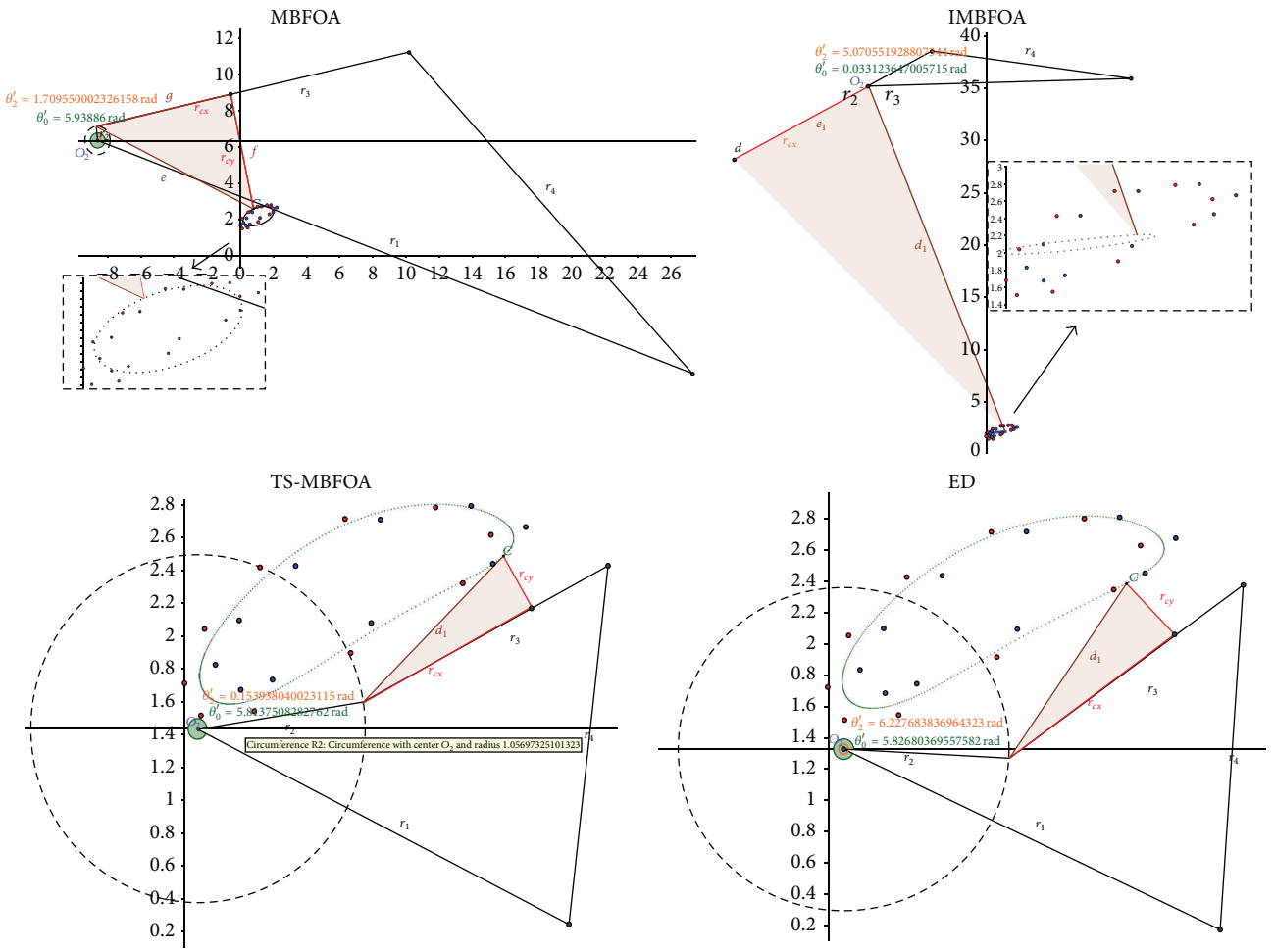

Figure 12: Problem $M03$ best solution simulation.

TS-MBFOA was analyzed in four experiments, where it was found that the two swims, unlike those of the other two BFOA-based algorithms, provided a larger number of better solutions along the search, even in its last cycles. Moreover, TS-MBFOA was able to consistently generate feasible solutions in the three optimization problems and its final results clearly outperformed those of MBFOA and IMBFOA because of its ability to avoid local optimum solutions. Furthermore, TS-MBFOA was able to obtain, with a single set of parameter values, competitive results in the three synthesis problems, with respect to one evolutionary algorithm designed for mechanical design optimization which required being fine-tuned for each synthesis problem. TS-MBFOA also found a similar competitive result for problem $M01$ but with less evaluations than the DE-based approach. Finally, from a mechanical point of view, the best solutions obtained by TS-MBFOA for the three optimization problems were highly competitive, suitable, and better than those found by MBFOA and IMBFOA.

The future work consists in revisiting the design of TS-MBFOA for studying the sensitivity to some of its parameters and trying to adapt their values. Finally, optimization problems of other mechanisms will be stated and solved.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Betania Hernández-Ocaña acknowledges support from *Universidad Juárez Autónoma de Tabasco (UJAT)* and *Consejo Nacional de Ciencia y Tecnología (CONACyT)* through a scholarship to pursue graduate studies at UJAT, Mexico. Efrén Mezura-Montes acknowledges support from CONACyT through Project no. 220522. Edgar Alfredo Portilla-Flores, Eduardo Vega-Alvarado, and Maria Bárbara Calva-Yáñez are grateful for Instituto Politécnico Nacional (IPN) for its support via Secretería de Investigación y Posgrado (SIP) with Project SIP-20151320.

References

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

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

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

[4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, Perth, Australia, November-December 1995.

[5] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[6] C. 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*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

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

[8] J. E. Shigley and J. J. Uicker, *Teoría de Máquinas y Mecanismos*, Mc-GrawHill, 1988.

[9] R. L. Norton, *Diseño de Maquinaria, una Introducción a la Síntesis y al Análisis de Mecanismos y Máquinas*, McGrawHill, New York, NY, USA, 1995.

[10] J. A. Cabrera, A. Simon, and M. Prado, "Optimal synthesis of mechanisms with genetic algorithms," *Mechanism and Machine Theory*, vol. 37, no. 10, pp. 1165–1177, 2002.

[11] R. R. Bulatović and S. R. Dordević, "On the optimum synthesis of a four-bar linkage using differential evolution and method of variable controlled deviations," *Mechanism and Machine Theory*, vol. 44, no. 1, pp. 235–246, 2009.

[12] H. Martínez-Alfaro, "Four-bar mechanism synthesis for n desired path points using simulated annealing," in *Advances in Metaheuristics for Hard Optimization*, Natural Computing Series, pp. 23–37, Springer, Berlin, Germany, 2008.

[13] S. K. Acharyya and M. Mandal, "Performance of EAs for four-bar linkage synthesis," *Mechanism and Machine Theory*, vol. 44, no. 9, pp. 1784–1794, 2009.

[14] H. Emdadi, M. Yazdanian, M. M. Ettefagh, and M. Feizi-Derakhshi, "Double four-bar crank-slider mechanism dynamic balancing by meta-heuristic algorithms," *International Journal of Artificial Intelligence & Applications*, vol. 4, no. 5, pp. 1–18, 2013.

[15] A. Sanchez-Marquez, E. Vega-Alvarado, E. A. Portilla-Flores, and E. Mezura-Montes, "Synthesis of a planar four-bar mechanism for position control using the harmony search algorithm," in *Proceedings of the 11th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE '14)*, pp. 1–6, IEEE, September 2014.

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

[17] D. H. Kim, A. Abraham, and J. H. Cho, "A hybrid genetic algorithm and bacterial foraging approach for global optimization," *Information Sciences*, vol. 177, no. 18, pp. 3918–3937, 2007.

[18] N. Kushwaha, V. S. Bisht, and G. Shah, "Genetic algorithm based bacterial foraging approach for optimization," *IJCA Proceedings on National Conference on Future Aspects of Artificial Intelligence in Industrial Automation*, vol. NCFAAIIA, no. 2, pp. 11–14, 2012.

[19] Y. Luo and Z. Chen, "Optimization for PID control parameters on hydraulic servo control system based on the novel compound evolutionary algorithm," in *Proceedings of the 2nd International Conference on Computer Modeling and Simulation (ICCMS '10)*, pp. 40–43, IEEE, January 2010.

[20] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "A synergy of differential evolution and bacterial foraging optimization for global optimization," *Neural Network World*, vol. 17, no. 6, pp. 607–626, 2007.

[21] H. Nouri and T. S. Hong, "A bacteria foraging algorithm based cell formation considering operation time," *Journal of Manufacturing Systems*, vol. 31, no. 3, pp. 326–336, 2012.

[22] H.-C. Huang, Y.-H. Chen, and A. Abraham, "Optimized watermarking using swarm-based bacterial foraging," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 1, no. 1, pp. 51–58, 2010.

[23] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "Synergy of PSO and bacterial foraging optimization—a comparative study on numerical benchmarks," in *Innovations in Hybrid Intelligent Systems*, E. Corchado, J. M. Corchado, and A. Abraham, Eds., vol. 44 of *Advances in Soft Computing*, pp. 255–263, Springer, Berlin, Germany, 2007.

[24] W. Korani, "Bacterial foraging oriented by particle swarm optimization strategy for PID tuning," in *Proceedings of the 10th Annual Genetic and Evolutionary Computation Conference (GECCO '08)*, pp. 1823–1826, IEEE, Atlanta, Ga, USA, July 2008.

[25] D. H. Kim and J. H. Cho, "Advanced bacterial foraging and its application using fuzzy logic based variable step size and clonal selection of immune algorithm," in *Proceedings of the 6th IEEE International Conference on Hybrid Information Technology (ICHIT '06)*, vol. 1, pp. 293–298, Cheju Island, Republic of Korea, November 2006.

[26] B. Hernández-Ocaña, E. Mezura-Montes, and P. Pozos-Parra, "A review of the bacterial foraging algorithm in constrained numerical optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 2695–2702, IEEE, Cancun, Mexico, June 2013.

[27] E. Mezura-Montes and B. Hernández-Ocaña, "Modified bacterial foraging optimization for engineering design," in *Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE '09)*, C. H. Dagli, K. M. Bryden, S. M. Corns, M. Gen, K. Tumer, and G. Süer, Eds., vol. 19 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pp. 357–364, ASME Press, St Louis, Miss, USA, November 2009.

[28] 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.

[29] B. Hernández-Ocaña, M. Del Pilar Pozos-Parra, and E. Mezura-Montes, "Stepsize control on the modified bacterial foraging algorithm for constrained numerical optimization," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO '14)*, pp. 25–32, ACM, Vancouver, Canada, July 2014.

[30] E. Mezura Montes and E. A. Lopez-Davila, "Adaptation and local search in the modified bacterial foraging algorithm for constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '12)*, pp. 1–8, IEEE, Queensland, Australia, June 2012.

[31] E. Mezura-Montes, E. A. Portilla-Flores, and B. Hernández-Ocaña, "Optimum synthesis of a four-bar mechanism using the modified bacterial foraging algorithm," *International Journal of Systems Science*, vol. 45, no. 5, pp. 1080–1100, 2014.

[32] B. Hernández-Ocaña, M. D. P. Pozos-Parra, and E. Mezura-Montes, "Improved modified bacterial foraging optimization algorithm to solve constrained numerical optimization problems," *Applied Mathematics and Information Sciences*, vol. 10, no. 2, pp. 607–622, 2016.

[33] P. Praveena, K. Vaisakh, and S. R. M. Rao, "A bacterial foraging and PSO-DE algorithm for solving dynamic economic dispatch problem with valve-point effects," in *Proceedings of the 1st International Conference on Integrated Intelligent Computing (ICIIC '10)*, pp. 227–232, IEEE, Bangalore, India, August 2010.

[34] K. Vaisakh, P. Praveena, S. Rama Mohana Rao, and K. Meah, "Solving dynamic economic dispatch problem with security constraints using bacterial foraging PSO-DE algorithm," *International Journal of Electrical Power and Energy Systems*, vol. 39, no. 1, pp. 56–67, 2012.

[35] N. Pandit, A. Tripathi, S. Tapaswi, and M. Pandit, "An improved bacterial foraging algorithm for combined static/dynamic environmental economic dispatch," *Applied Soft Computing Journal*, vol. 12, no. 11, pp. 3500–3513, 2012.

[36] P. Siarry and Z. Michalewicz, Eds., *Análisis de Mecanismos y Problemas Resueltos*, Alfaomega, 2006.

[37] E. Vega-Alvarado, E. Santiago-Valentín, A. Sánchez-Márquez, A. Solano-Palma, E. A. Portilla-Flores, and L. Flores-Pulido, "Síntesis óptima de un mecanismo plano para seguimiento de trayectoria utilizando evolución diferencial," *Research in Computing Science*, vol. 72, pp. 85–98, 2014.

[38] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 207–214, IEEE, British Columbia, Canada, July 2006.

[39] M. J. D. Powell, "Algorithms for nonlinear constraints that use Lagrangian functions," *Mathematical Programming*, vol. 14, no. 2, pp. 224–248, 1978.

[40] A. Kasaiezadeh, A. Khajepour, and S. L. Waslander, "Spiral bacterial foraging optimization method: algorithm, evaluation and convergence analysis," *Engineering Optimization*, vol. 46, no. 4, pp. 439–464, 2014.

[41] E. A. Portilla-Flores, E. Mezura-Montes, J. Alvarez-Gallegos, C. A. Coello-Coello, C. A. Cruz-Villar, and M. G. Villarreal-Cervantes, "Parametric reconfiguration improvement in non-iterative concurrent mechatronic design using an evolutionary-based approach," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 5, pp. 757–771, 2011.

[42] G. Corde and D. Foreman, *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*, John Wiley & Sons, Hoboken, NJ, USA, 2009.

[43] J. Liang, T. P. Runarsson, E. Mezura-Montes et al., "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization," Tech. Rep., School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 2006.

[44] A. Eiben and M. C. Schut, "New ways to calibrate evolutionary algorithms," in *Advances in Metaheuristics for Hard Optimization*, P. Siarry and Z. Michalewicz, Eds., Natural Computing Series, pp. 153–177, Springer, Berlin, Germany, 2008.

[45] M. López-Ibáñez, J. Dubois-Lacoste, T. Sttzle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," IRIDIA TR/IRIDIA/2011-004, Université libre de Bruxelles, Brussels, Belgium, 2011.