Exploring the Effects of Attraction and Repulsion Parameters on the Bacterial Foraging Algorithm through Benchmark Functions

RIOS-WILLARS ERNESTO*, REYES-ACOSTA ALFREDO VALENTIN
Faculty of Systems,
Autonomous University of Coahuila,
Blvd. Fundadores Km. 13, Ciudad Universitaria, Arteaga, Coahuila,
MÉXICO

**Corresponding Author*

Abstract: - Metaheuristics are essential when working with complex problems from different fields. However, a suitable tuning scheme for these parameters is necessary to facilitate the search for potential solutions. This tuning is a challenging task. This work aims to develop a tuning method for the BFOA algorithm regarding attraction and repulsion values. In some cases, the parameter values are taken from previous works, while in other cases, the parametrization scheme comes from an automated or dynamic process. This work explores the Bacterial Foraging Algorithm (BFOA) within its parameters related to attraction and repulsion among bacteria, using 18 well-known benchmark functions from the literature. For this purpose, multiple BFOA executions were made, and averages were calculated for each test with repetitions for 24k BFOA executions. The interest variables for contrasting performance were the number of evaluated functions (NFE), the required time for the execution (time), and the associated cost to the achieved solution by BFOA (cost). Results: BFOA produced a different performance corresponding to each benchmark function. From this, four tuning schemes are proposed and validated by repetition, also contrasted by t-test. The conclusions show that the BFOA algorithm is susceptible to tuning, and the attraction and repulsion parameters must be according to the optimization problem. In terms of execution time, scheme III showed remarkable results. Regarding the obtained solution cost, scheme II outperformed the other three schemes.

Key-Words: - BFOA, tuning, bacterial foraging algorithm, benchmark functions, bioinspired, metaheuristics

Received: May 16, 2022. Revised: August 19, 2023. Accepted: September 19, 2023. Available online: October 25, 2023.

1 Introduction

The parameter setting for nature-inspired algorithms is an open problem, [1]. The control parameters significantly influence the algorithm's performance; their correct setting is crucial when obtaining optimal solutions. Setting them is difficult since their values are problem-dependent, [2]. In this work, we describe a method for exploring the Bacterial Foraging Algorithm in terms of performance based on benchmark functions in an extensive field of possible parameter values and then integrating a set of four schemas and a statistical validation. In this context, the benefit of this work is a novel tuning procedure based on attraction and repel (*d_attr, h_rep, w_attr, and w_rep*) BFOA parameters. The interest variables for comparison are the number of evaluated functions (NFE), the required time for execution (time), and the cost of the obtained solutions (cost). This BFOA paradigm has proven efficient as an optimization strategy in different areas, although it has the inconvenience of requiring several parameters tuning, [3]. This study explores the parametrization of the algorithm parting from the default considered values reported by, [4]. Figure 1 is a conceptual scheme of the process in this study.

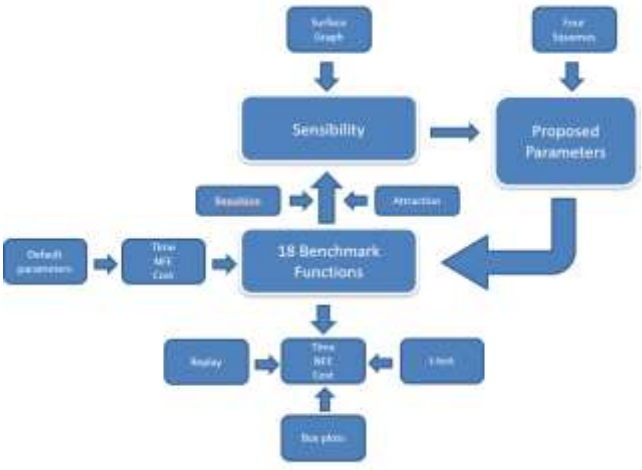


Fig. 1: Conceptual scheme of the process in the study.

Next is a list of the steps taken in this work:

1. Establish a BFOA with default parameters
2. Run BFOA on 18 benchmark functions
3. From the results, define the range of parameters and run BFOA on each range value with repetitions.
4. From the results, generate four fixed parameter schemes and run BFOA for further analysis.
5. Validate results by t-test and generate the final tuning scheme regarding NFE, time, and cost variables.

1.1 BFOA Background

The Bacterial Foraging Optimization Algorithm (BFOA) proposed by, [5], is based on the natural movement of the Escherichia coli bacteria (*E.coli*), which, aided by its scourges, can use a translatory motion in pursuit of nutrients into the environment, or if needed to get out of the way of adverse factors. It is a unicellular organism that tends to form colonies in which the individuals are attracted to and repelled each other by chemical substances segregated in a mutual interaction with swarming characteristics. The scourges are an extension in the form of tentacles that can rotate in both ways, allowing the bacteria to move in the environment in two ways: a) swim and b) tumble. A description of the results of the translatory movement of the bacteria in the function of its scourges' action and the environmental conditions can be found in the original Passino document.

1.2 BFOA Usage Diversity

The BFOA is a widely used tool in engineering and optimization areas. It helps to solve numerous problems, making it an essential part of these fields, for instance, In, [6], an application of the BFOA in turbine design. Alternatively, the one presented in, [7], where the author describes a BFOA application that solves vehicle routing problems. In, [8], BFOA was used as a strategy for designing active filters in electrical engineering and in, [9], for the power transference problem. In, [10], [11], it is applied to creating a PID controller in a DC-DC electric converter.

Moreover, in, [12], the proposal regards an application for solving the problem of robotic movement planning. On the other hand, in ,[13], an application is used as a placement strategy in a laser radar. In, [14], the BFOA is used in the Hydro Power Dispatch problem. In, [15], The BFOA application optimizes picture recording on multi-core hardware processors. In, [16], there is an application for the pattern recognition problem. The Passino's algorithm has generated numerous proposals and adaptations in various contexts. Hereafter, they are classified and briefly described. *Adaptations and improvements*: For example, in, [17], the authors describe a modified version of the Bacterial Foraging Optimization Algorithm (BFOA) developed to expedite the convergence to the optimal solution. However, it has been observed that this algorithm sometimes keeps oscillating when it is close to the objective. To tackle this issue, the researchers have proposed an operator that dynamically adjusts the chemotaxis steps based on the fitness values obtained. In, [18], the reproduction phase of the BFOA is analyzed as decisive in the convergence through two differential equations and in two bacteria in a one-level environment. In, [19], a version of the BFOA is proposed, where the best bacteria remain intact (elitism) while the others, which are a few, reboot themselves. In, [20], a parallelization of the BFOA is applied to the task planning problem. *Mathematical Models*: In [3], [21], [22], a detailed description of the algorithm is made from the mathematical point of view. *Performance Benchmarking:* In, [12], [23], [24], [25], [26], a performance benchmarking of the BFOA is made, comparing it with other optimization algorithms and within different problem contexts. *Hybridizations:* In, [27], there is a hybridization description between PSO, BFOA, and Differential Evolution (DE). In, [13] the authors present a hybrid between BFOA and Firefly Optimization proposed for the vehicle routing problem. In, [28], BFOA and ant colony hybridization are described and applied to the labor programming problem. The work in, [29], is a hybrid between GA and BFOA applied to the manufacturing cells distribution problem. In, [30], another hybridization between BFOA and PSO was used for the global numeric optimization problem.

1.3 Algorithm Tuning

The BFOA algorithm has the characteristic of requiring precise tuning, and this may be a severe disadvantage. However, every metaheuristic algorithm requires a set of parameters tuning according to the problem in the case. In this area, the central part of the metaheuristic applications uses the default values proposed by the metaheuristic developer. In other cases, the parameters are adjusted based on different strategies, looking for better solutions in an intuitive or automated iteration process.

For the BFOA algorithm, a case of auto-tuning is made on, [31], where the search process is made by enabling the bacterial foraging algorithm to adjust the run-length unit parameter dynamically during algorithm execution to balance the exploration/exploitation tradeoff. However, the

authors point out that developing and experimenting with various methods and tuning them for each particular problem is necessary. Several procedures have been made to parametrize other algorithms, such as GRASP, genetic algorithms, and scatter search, [32], with significant findings in this context. However, tuning or performing a parametrization of a metaheuristic is still under experimentation, especially for relatively new metaheuristics such as BFOA. In this context, a proposed method is based on the "functions reuse" considering parameters standard to several metaheuristics, for example, the number of elements in the initial set or the number of cycles without improving the best solution in the end condition, [33]. In other cases, metaheuristics are built from blocks, taking from other algorithms those parts that might be useful in a particular problem, [34]. Optimizing a BFOA remains complex due to the varying types of problems it addresses. A contribution is generated in this field through a base model with an Adaptive Neuro-Fuzzy Inference System, [35]. Also, in the search for efficient parameter tuning, a computational tool has been developed. In this case, it implements the iterated racing procedure for automatic algorithm configuration, [36]. However, in some cases, it is feasible to adopt parameter values from prior research where they have been established based on the tuning conducted by others, which allows for the rapid development of new features and strategies, [37]. In this work, we explore an extensive area of possible parameter values, which allows us to contrast and choose the most adequate.

Table 1 lists the most common practices for tuning a metaheuristic like BFOA.

Table 1. Most common tuning strategies.

Dynamic auto-tuning and self-adaptive process	The BFOA can change its parameters along the execution based on rules relative to the optimization problem.
Fuzzy Logic	A series of fuzzy rules are associated with the BFOA to explore parameter values.
Independent Computational tool	A tool consists of three phases: sampling new parameter configurations with a specific distribution, selecting the most competitive configurations through racing, and then updating the sampling distribution to favor better configurations.
Manual experimentation	It consists of a manual trial and error process where the parameter values are explored first.
Taken from others	The parameter values are taken from other implementations where a tuning process is already made.

2 Bacterial Foraging Algorithm

The optimization in the BFOA algorithm is based on the chemotactic behavior of the Escherichia Coli Bacteria (E. Coli). Although using chemotaxis as a model for optimization was proposed first in, [38], Passino's work includes some modifications, such as agents' reproduction and dispersion. E. Coli is the best-understood microorganism, [5], since its behavior and genetic structure are well-studied. This unicellular organism consists of a capsule with its organs and scourges used for locomotion; it can reproduce by dividing itself and exchanging genetic information with its peers. In addition, it can detect food (nutrients) and avoid toxic substances, making a random search based on two locomotion states: the translation (swim) and the rotation (tumble). The decision to remain in one of these states depends on the nutrients or toxic substances concentration in the environment. This behavior is called chemotaxis.

The following describes the optimization steps with the algorithm, [10]. Step 1: The inner cycle of the chemotaxis is illustrated in Figure 2. In this process, the E. coli movement is simulated. The move is made in two ways: lurching (tumbling) or swimming. One operation at a time. The value of the objective function is calculated. The bacteria change its position if the value of the modified objective function is worse than before. Once the chemotaxis is completed, the bacteria will circulate a new interest point in the search space.

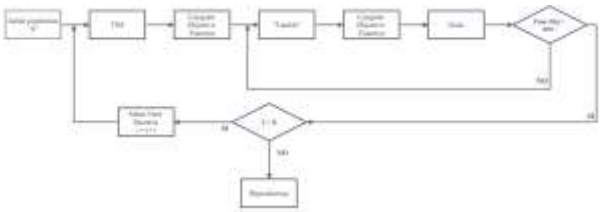


Fig. 2: Chemotaxis process in the BFOA algorithm, [5].

Step 2: In the reproduction process, the value of the objective function is calculated for each of the bacteria in the organized population. The worst half of the population is disregarded, and the best half is duplicated. The chemotaxis process starts and continues for this new bacteria generation according to the number of reproductive cycles. Step 3: In the elimination-dispersion cycle, some bacteria are eliminated with low probability and dispersed randomly. This process maintains the bacteria numbers constant. The entry parameter is the number of bacteria $_{(Sb)}$ Chemotaxis steps limit $_{(Nc)}$, Swim-steps limit $_{(Ns)}$, Reproductive cycles limit $_{(Nre)}$, bacteria number aimed to produce elimination-dispersion

cycle limit (Ned), step size (Ci), and elimination and dispersion probability (Ped). The cost of each bacteria is optimized by its interaction with other bacteria. The interaction function is calculated according to the expression *g()*.

$$g(cell_k) = \sum_{i=1}^{S}\left[-d_{attr} * e^{(-w_{attr}*\sum_{m=1}^{P}(cell_m^k - other_m^i)^2)}\right] + \sum_{i=1}^{S}\left[h_{repel} * e^{(-w_{repel}*\sum_{m=1}^{P}(cell_m^k - other_m^i)^2}\right]$$

Where the *cell* is the bacteria of the issue, and *the other* is a neighboring cell. d_{attr} y w_{attr} are attraction coefficients. h_{repel} and w_{repel} are repulsion coefficients. S is the number of bacteria in the population. P is the number of dimensions in the optimization problem, [4].

2.1 Bacterial Interaction Parameters

These are significant since this interaction sustains the optimization process in the paradigm that states that bacterial colonies aim to move towards new positions where nutrients are in better availability. These parameters represent the attraction extent and depth between bacteria (d_attr y w_attr) and the repulsion extent and depth between bacteria (h_rep y w_rep). These four parameters are part of the tuning assembly of the BFOA. In this sense, [3], expresses that the BFOA has the disadvantage of requiring more parameters than other optimization algorithms, which makes it more complicated from this point of view. However, [11], points out that researchers use the BFOA because it does not require precise mathematical models for tuning. Notably, [5], points out that if the extent of the attraction signal is long and very deep, the cells will have a solid tendency to swarm. Moreover, in the opposite sense, each cell will look for the optimization independently.

3 Methodology

For the development of this project, two phases are established.

1) First Phase:
   a) BFOA algorithm implantation in a set of 18 benchmark functions and result evaluation.
   b) Algorithm sensibility exploration within the bacterial interaction parameters on a benchmark function.
2) Second Phase:
   a) Algorithm Parameterization proposal from the sensibility exploration.
   b) Perform Results validations by repetition and t-test.

3.1 Parameters and Resources

The BFOA algorithm was developed in Ruby language with computer equipment, including an Intel Core i7 Processor, 10Gb in RAM Memory, and a Windows 7 operative system. The graphics were obtained from the Excel Microsoft Software and the Minitab Statistic Software. The BFOA parameters considered by default and reported by, [4], are described in Table 2.

Table 2. Default parameters in the BFOA algorithm.

Population size	50
Step Size (Ss)	0.1
Elimination – dispersion cycles (Ned)	1
Reproductive cycles (Nre)	4
Quimiotaxis cycles (Nc)	70
Swim length (Ns)	4
Elimination probability (Ped)	0.25
Attraction depth (d_attr)	0.1
Attraction wide (w_attr)	0.2
Repeland depth (h_rep)	d_attr
Repeland wide (w_ewp)	10

3.2 First Phase: Benchmark Functions

To ensure evidence diversity, in this study, we use a group of 18 functions relative to known problems in the optimization area, [39], [40]. These are classified according to similarity and search space form, [41]. Moreover, some functions have multiple local minimums, others have a plane form, valley form, or bowl-like depth, and others are staggered and of various shapes. For this benchmark, all functions are two-dimensional (d=2) and carried out on 30 repetitions.

Cross In Tray: Has multiple global minima. The function is commonly evaluated on the square xi ∈ [-10, 10] for all i = 1, 2, [42].

$$f(\boldsymbol{x}) = -\mathbf{0.0001}\left\{\left|\boldsymbol{\sin x_1 \sin x_2\, e}^{\left(\left|\mathbf{100}-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right|\right)}\right| + \mathbf{1}\right\}^{\mathbf{0.1}} \quad (2)$$

Search Space: xi ∈ [-10, 10]
Global Min: -2.06261

Drop Wave: This is a multimodal test function. The given form of function has only two variables and the following definition. The test area is usually restricted to the square $-5.12 \le x1 \le 5.12, -5.12 \le x2 \le 5.12$, [40].

$$f(x) = -\frac{1+cos\left(12\sqrt{(x_1^2+x_2^2)}\right)}{0.5(x_1^2+x_2^2)+2} \quad (3)$$

Search Space: $x_i \in [-5.12, 5.12]$
Global Min: -1

Holder Table: Is a Continuous, Differentiable, Separable, Non-Scalable, Multimodal function. The four global minima are at *x*=f(±9.646168, ±9.646168), f(x*)= −26.920336*, [41].

$$f(x) = -\left|sin(x_1)\, cos\left(x_2\right) e^{\left(\left|1-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right|\right)}\right| \quad (4)$$

Search Space: xi ∈ [-10, 10]
Global Min: -19.2085

EggHolder: This has a deceptive landscape and is a challenging function to optimize because it is characterized by an uneven plane having several dozen local minimums that easily mislead the search agents. The function is usually evaluated on the square xi ∈ [-512, 512], for all i = 1, 2, [42].

$$f(x) = -(x_2+47)\, sin\left(\sqrt{\left|x_2+{}^{x_1}\!/_{2}+47\right|}\right) - x_1\, sin\left(\sqrt{|x_1-(x_2+47)|}\right) \quad (5)$$

Search Space: xi ∈ [-512, 512]
Global Min: -959.6407

Rastrigin: This is based on the function of *De Jong* with the addition of cosine modulation to produce frequent local minima. The test area is usually restricted to hypercube *−5.12≤xi≤5.12,i=1,...,n.* Its global minimum equal *f(x)=0* is obtainable for *xi=0,i=1,...,n.,* [40].

$$f(x) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\, cos\, 2\pi x_i\right] \quad (6)$$

Search Space: xi ∈ [-5.12, 5.12]
Global Min: 0

Schwefel: It is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction. The function has the following definition. The test area is usually restricted to a hypercube

−500 ≤ xi ≤ 500, i=1,...,n.

Its global minimum *f(x)=−418.9829n* is obtainable for *xi=420.9687,i= 1,...,n.,* [43].

$$f(x) = 418.9829d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{|x_i|}\right) \quad (7)$$

Search Space: xi ∈ [-500, 500]
Global Min: 0

Schaffer2: The function is usually evaluated on the square xi ∈ [-100, 100], for all i = 1, 2, [41].

$$f(x) = 0.5 + \frac{sin^2(x_1^2-x_2^2)-0.5}{\left[1+0.001(x_1^2+x_2^2)\right]^2} \quad (8)$$

Search Space: xi ∈ [-100, 100]
Global Min: 0

Ackley has a flat outer region and a large hole at the center. The function poses a risk for optimization algorithms, particularly hill climbing algorithms, to be trapped in one of its many local minima, [44].

$$f(x) = \alpha e^{\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)} - e^{\left(\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)\right)} + \alpha + e^1 \quad (9)$$

Var: α=20, b=0.2, c=2π
Search Space: xi ∈ [-32.768, 32.768]
Global Min: 0

Booth: Is usually evaluated on the square xi ∈ [-10, 10], for all i = 1, 2, [41].

$$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \quad (10)$$

Search Space: xi ∈ [-10, 10]
Global Min: 0

Matyas: Has no local minima except the global one. The function is usually evaluated on the square *xi ∈ [-10, 10] for all i = 1, 2,* [41].

$$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48 x_1 x_2 \quad (11)$$

Search Space: xi ∈ [-10, 10]
Global Min: 0

Zakharov: Is continuous and unimodal. The suggested search area is the hypercube [−10, 10]D. The global minimum is f28(x*) = 0 at x* = {0, 0, …, 0}. The general formulation of the function is, [45].

$$f(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{d} 0.5 i x_i\right)^4 \quad (12)$$

Search Space: xi ∈ [-5, 10]
Global Min: 0

Sphere: Is continuous, convex, unimodal, differentiable, separable, highly symmetric, and rotationally invariant. The suggested search area is the hypercube [−100, 100]D. The global minimum is f01(x*) = 0 at x* = {0, 0, …, 0}, [45].

$$f(x) = \sum_{i=1}^{d} x_i^2 \quad (13)$$

Search Space: xi ∈ [-5.12, 5.12]
Global Min: 0

Rosenbrok: This is often used as a test problem for optimization algorithms (where a variation with 100 replaced by 105 is sometimes used. It has a global minimum of 0 at the point (1, 1), [46].

$$f(x) = \sum_{i=1}^{d-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \quad (14)$$

Search Space: xi ∈ [-5, 10]
Global Min: 0

Michaelwicz: Has *d!* local minima, and it is multimodal. The parameter *m* defines the steepness of the valleys and ridges; a larger *m* leads to a more complex search. The recommended value of *m* is *m = 10,* [41].

$$f(x) = -\sum_{i=1}^{d} sin(x_i)\, sin^{2m}\left(ix_i^2/\pi\right) \quad (15)$$

Var: m=10

Search Space: xi ∈ [0, π]

Global Min: -1.8013

Easom: This is an unimodal test function where the global minimum has a

small area relative to the search space. The function was inverted for minimization, [40].

It has only two variables

$$f(x) = -cos(x_1)\, cos(x_2) e^{(-(x_1-\pi)^2-(x_2-\pi)^2)} \quad (16)$$

Search Space: xi ∈ [-100, 100]

Global Min: 0

Beale: is multimodal, with sharp peaks at the corners of the input domain. The function is usually evaluated on the square *xi ∈ [-4.5, 4.5] for all i = 1, 2*, [41].

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.62 - x_1 + x_1x_2^3)^2 \quad (17)$$

Search Space: xi ∈ [-4.5, 4.5]

Global Min: 0

Styblinski-Tang: Has three local minimums in addition to its global minimum. The function is continuous, not convex, defined on n-dimensional space, and multimodal, [41].

$$f(x) = \frac{1}{2}\sum_{i=1}^{d}\left(x_i^4 - 16x_i^2 + 5x_i\right) \quad (18)$$

Search Space: xi ∈ [-5, 5]

Global Min: -39.16599d

3.2.1 Sensibility Analysis

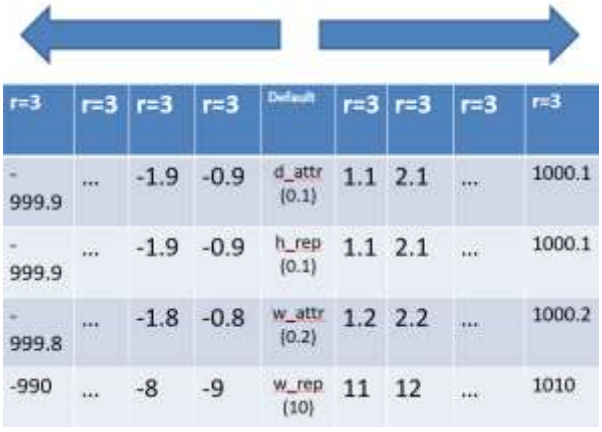
As pointed out before, the *w_attr, h_rep, d_attr, and w_rep* parameters represent the attraction extent and depth between bacteria (*d_attr, w_attr*) and the repulsion extent and depth between bacteria (*h_rep, w_rep*). In this section, we describe how parameter values are explored to observe the results for the benchmark function when running on each parameter setting.

Consider the set

{(w_attr, h_rep, d_attr, w_rep) | w_attr = i, h_rep = i, d_attr = i, w_rep = k}

Where *w_attr, h_rep*, and *d_attr* take values from integer *i {i | -999.9 ≤ i ≤ 1000.1}* and *w_rep* take values from integer *k* in the established range *{k | -990 ≤ k ≤ 1010}*. From the definition of (*i,k*) values, we run a "scenario", a BFOA algorithm running on the Michalewicz benchmark function for a workspace with multiple local minima and multimodal characteristics.

The sensibility analysis consists of executing a set of BFOA algorithms *n=8000* such that the values of the bacterial interaction parameters are different in each one. Establishing as a start the default values in, [4]. Then, gradual and unitary changes are made to each parameter (one at a time) incrementally in one case and decremental in the other, considering three repetitions for each experiment. See Figure 3 for a graphic description. From the repetitions, the averages are calculated.



r=3	r=3	r=3	r=3	Default	r=3	r=3	r=3	r=3
-999.9	...	-1.9	-0.9	d_attr (0.1)	1.1	2.1	...	1000.1
-999.9	...	-1.9	-0.9	h_rep (0.1)	1.1	2.1	...	1000.1
-999.8	...	-1.8	-0.8	w_attr (0.2)	1.2	2.2	...	1000.2
-990	...	-8	-9	w_rep (10)	11	12	...	1010

Fig. 3: Graphic description of the variation parameters in the BFOA sensibility analysis.

This series of combinations produces 2000 adjustments for each parameter, 1000 for the incremental variation case, and 1000 for the decremental variation. Each experiment had three repetitions. In sum, 24,000 BFOA algorithm executions. For this test, the interest variables to be measured in each run were the execution time (*t*), the number of evaluated functions (*NFE*), and the cost of the found solution by the algorithm (*cost*) to measure the algorithm's efficiency and robustness, [47].

Figure 4 describes the workflow used for this exploration.

3.3 Second Phase

From the results graphic in the sensibility exploration, we selected the regions where the algorithm has a noticeable performance regarding the interest variables. (NFE, t, cost). For this, the surface graphics are also generated to determine by observation a beneficial point to tune the algorithm and validate. The validation consists of executing new runs for the algorithm in the 18 benchmark functions and a group (n=4) of parametrization schemes proposed from the analysis.

As a part of the validation process, the t-tests are executed for each parametrization scheme and in each optimization benchmark function for the results of the 30 runs in terms of the cost variable. In the same way, box graphics are presented as a contrast of the results before and after parametrization with the proposed scheme.

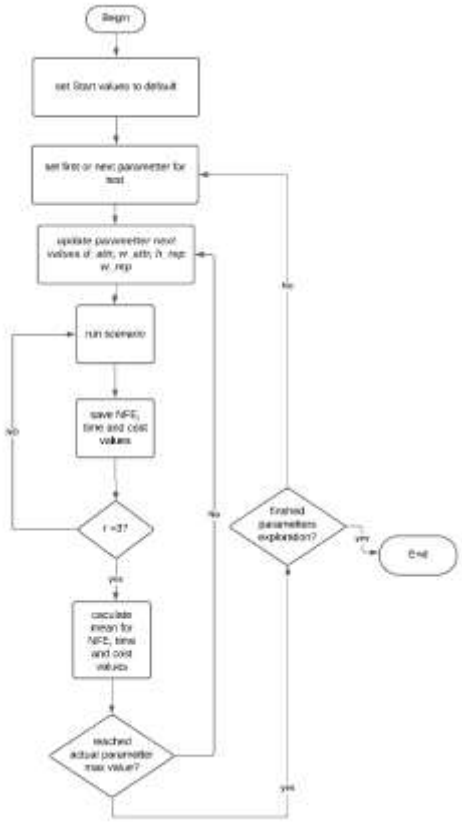


Fig. 4: The workflow used for BFOA algorithm parameters exploration. Each of the *w_attr, h_rep, d_attr, and w_rep* parameters are updated in turn, and three algorithm executions are performed for average value calculation.

4 Results

The results from the algorithm executions from each benchmark function are presented.

4.1 First Phase: Benchmark Results

Table 3 shows the BFOA algorithm's results in 18 benchmark functions. Each box corresponds to the average result of the 30 algorithm's repetitions in the respective function and with the default parameters.

Table 3. BFOA Results in the 18 benchmark functions.

Opt-Cost	Function	NFE	t	Cost
0	Ackley	28616.8	3764.51	9.344239
-1.80	Michaelwi	29372.3	3873.65	-
0	Rastrigin	28588.8	3762.24	0.710712
0	Rosenbro	32261.7	4275.71	0.000652
0	Schwefel	40398.7	2474.17	201.1314
0	Sphere	38576.7	4499.39	2.74135E-
-78.33	Styblinski	32229.9	1959.00	-
0	Zakharov	38901.1	5398.30	5.33258E-
0	Beale	31692.7	5485.38	4.7815E-
0	Booth	37141.1	4843.21	6.12336E-
-2.06	Cross In	43420.5	5238.76	-
-1	Drop	33297.1	4514.52	-
-1	Easom	56335.3	6101.51	-
-959.64	Egg	40293.5	5974.64	-
-19.20	Holder	35989.3	4941.58	-
0	Matyas	40177	5371.40	1.69966E-
0	Schaffer 2	26183.6	3376.09	0.002326
-186.73	Shubert	28487.7	3935.19	-

4.2 Sensibility Analysis Results

Regarding the four variables of bacterial interactions and the three interest variables. (*time, NFE, cost*). A noticeable difference can be observed between the sides of the graphic parting from the center, where the default value corresponds. To the left are the incremental values from the default, and to the right are the decremental ones. That is to say that the value 1 is the most positive and the value 2000 the most negative. These tests were made, simultaneously varying one of the four parameters and leaving the other three in default value.

In the execution time comparison between the four parameters in Figure 5, an increase for the range 987 to 1161 stands out in the values from w_attr, while d_attr and h_rep remain in stable fields. In the same way, w_rep presents an increase in the range 1103 to 1451 to stabilize subsequently.

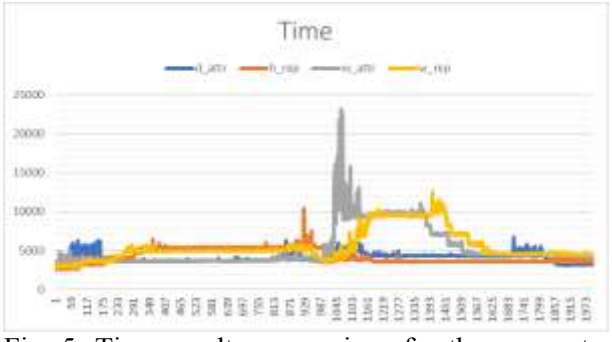


Fig. 5: Time results comparison for the parameters d_attr, h_rep, w_attr, w_rep.

Figure 6 compares the number of the evaluated function (NFE) for the four parameters d_attr, h_rep, w_attr, and w_rep. The range 987 to 1161 stands out because the evaluated functions increase between the parameters w_rep and w_attr. In contrast, the parameter h_rep demonstrates a decrease in the functions being assessed.

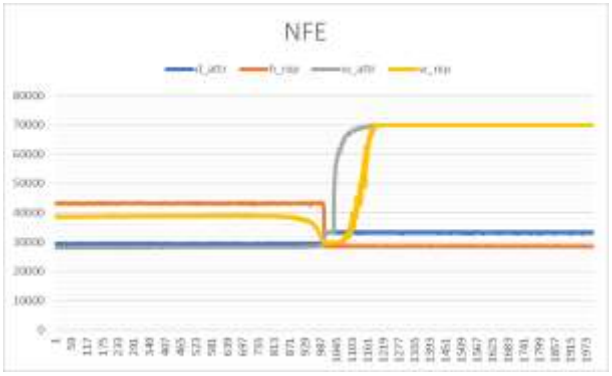


Fig. 6: Comparison of the evaluated functions number (NFE) for the parameters d_attr, h_rep, w_attr, w_rep.

Figure 7 compares the costs of the functions according to the four parameters d_attr, h_rep, w_attr, and w_rep. An increase is to be noted from the value 1009 for the parameters h_rep y d_attr. In the same way, the parameter w_attr presents an increase parting from said value to stabilize parting from the value 1121 subsequently.

Fig. 7: Cost comparison according to parameters d_attr, h_rep, w_attr, w_rep.

Surface Graphics**.** Figure 8 shows the surface generated by parametrizations in terms of interest variables. The favorable point is located in the region where the cost, execution time, and evaluated function number are minimum.

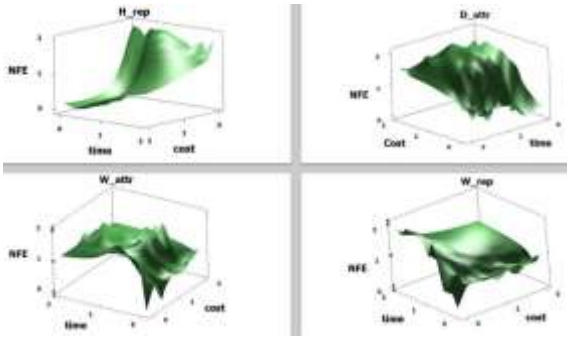


Fig. 8: Surface graphics in the sensibility analysis.

4.3 Second Phase: Proposed Parameters

The sensibility analysis and surface graphics observation establish the following parametrization schemes for validation by repetition and t-test. See Table 4.

Table 4. Proposed schemes.

Scheme	d_attr	h_rep	w_attr	w_rep
I	995.1	0.1	0.2	10
II	0.1	-0.9	0.2	10
III	0.1	0.1	655.2	10
IV	0.1	0.1	0.2	967

4.4 Validation by Repetition and T-Test

Once the four schemes for tuning BFOA are established, the next step is to run the algorithm for the 18 benchmark functions with 30 repetitions to have a general panorama of the algorithm's performance in terms of the cost value.

Also, as a part of the validation process, it is known that a *t*-test can be used to determine whether two groups differ from each other in terms of independent samples. At this point, we use *t*he test to determine whether the average result (n=30) differs from the execution on the default parameter values vs. the execution with each proposed scheme.

Table 5 shows results from the validation by repetition and t-test of the parametrization proposed scheme number I. The value in the t column corresponds to the *p* significance of the statistical test from the cost with default values vs. the value achieved with the scheme mentioned.

Table 5. Validation by repetition in the scheme I.

		Cost w/	**Scheme I**	
Optimal	**Function**	**defaults**	**Cost**	**t**
0.0000	Ackley	9.3442	7.7122	0.0600
-1.8013	Michaelwikcz	-1.8012	-1.7987	0.0010
0.0000	Rastrigin	0.7107	0.0424	0.0000
0.0000	Rosenbrok	0.0007	0.0744	0.0140
0.0000	Schwefel	201.1314	213.7595	0.6610
0.0000	Sphere	0.0000	0.0814	0.3220
-78.3320	Styblinski	-78.3323	-75.9263	0.0040
0.0000	Zakharov	0.0001	1.0591	0.0050
0.0000	Beale	0.0000	0.1003	0.0060
0.0000	Booth	0.0001	3.1139	0.0000
-2.0636	Cross In Tray	-2.0626	-2.0491	0.0260
-1.0000	Drop Wave	-0.8708	-0.9873	0.0000
-1.0000	Easom	-0.0333	-0.0407	0.8730
-959.6407	Egg Holder	-719.0541	-677.5126	0.2330
-19.2085	Holder Table	-19.2081	-18.1454	0.1000
0.0000	Matyas	0.0000	0.1210	0.0000
0.0000	Schaffer 2	0.0023	0.0026	0.7420
-186.7309	Shubert	-178.8117	-175.8331	0.6160

Table 6 shows results from the validation by repetition and t-test of the parametrization proposed scheme number II. The value in the t column corresponds to the *p* significance of the statistical test from the cost with default values vs. the value achieved with the scheme mentioned.

Table 6. Validation by repetition in the scheme II.

		Cost w/	Scheme II	
Optimal	**Function**	**defaults**	**Cost**	**t**
0.0000	Ackley	9.3442	8.7900	0.5640
-1.8013	Michaelwikcz	-1.8012	-1.8005	0.0000
0.0000	Rastrigin	0.7107	0.9730	0.2170
0.0000	Rosenbrok	0.0007	0.0009	0.3540
0.0000	Schwefel	201.1314	171.9608	0.2430
0.0000	Sphere	0.0000	0.0001	0.0170
-78.3320	Styblinski	-78.3323	-78.3307	0.0000
0.0000	Zakharov	0.0001	0.0001	0.0110
0.0000	Beale	0.0000	0.0001	0.0080
0.0000	Booth	0.0001	0.0004	0.0050
-2.0636	Cross In Tray	-2.0626	-2.0586	0.0810
-1.0000	Drop Wave	-0.8708	-0.8939	0.3530
-1.0000	Easom	-0.0333	-0.1333	0.1680
-959.6407	Egg Holder	-719.0541	-777.8452	0.0950
-19.2085	Holder Table	-19.2081	-19.2077	0.1380
0.0000	Matyas	0.0000	0.0005	0.2750
0.0000	Schaffer 2	0.0023	0.0013	0.1400
-186.7309	Shubert	-178.8117	-167.4650	0.1680

Table 7 shows results from the validation by repetition and t-test of the parametrization proposed scheme number III. The value in the t column corresponds to the *p* significance of the statistical test from the cost with default values vs. the value achieved with the scheme mentioned.

Table 7. Validation by repetition in the scheme III.

		Cost w/	Scheme	
Optimal	**Function**	**defaults**	**Cost**	**t**
0.0000	Ackley	9.3442	8.3033	0.237
-1.8013	Michaelwikc	-1.8012	-1.7891	0.001
0.0000	Rastrigin	0.7107	0.7487	0.843
0.0000	Rosenbrok	0.0007	0.0363	0.000
0.0000	Schwefel	201.1314	191.4861	0.693
0.0000	Sphere	0.0000	0.0250	0.000
-78.3320	Styblinski	-78.3323	-78.3307	0.020
0.0000	Zakharov	0.0001	0.0235	0.000
0.0000	Beale	0.0000	0.0071	0.000
0.0000	Booth	0.0001	0.0161	0.006
-2.0636	Cross In Tray	-2.0626	-1.9844	0.000
-1.0000	Drop Wave	-0.8708	-0.8301	0.168
-1.0000	Easom	-0.0333	0.0000	0.326
-	Egg Holder	-	-738.2348	0.584
-19.2085	Holder Table	-19.2081	-19.1903	0.000
0.0000	Matyas	0.0000	0.1508	0.000
0.0000	Schaffer 2	0.0023	0.0073	0.000
-	Shubert	-	-159.4473	0.050

Table 8 shows results from the validation by repetition and t-test of the parametrization proposed scheme number IV. The value in the t column corresponds to the *p* significance of the statistical test from the cost with default values vs. the value achieved with the scheme mentioned.

Table 8. Validation by repetition in the scheme IV.

		Cost w/	Scheme	
Optimal	**Function**	**defaults**	**Cost**	**t**
0.0000	Ackley	9.3442	9.5079	0.829
-1.8013	Michaelwikc	-1.8012	-1.–	0.000
0.0000	Rastrigin	0.7107	0.6810	0.865
0.0000	Rosenbrok	0.0007	0.0011	0.153
0.0000	Schwefel	201.1314	184.9300	0.546
0.0000	Sphere	0.0000	0.0000	0.000
-78.3320	Styblinski	-78.3323	-78.3323	0.637
0.0000	Zakharov	0.0001	0.0000	0.000
0.0000	Beale	0.0000	0.0000	0.020
0.0000	Booth	0.0001	0.0000	0.003
-2.0636	Cross In Tray	-2.0626	-2.0626	0.682
-1.0000	Drop Wave	-0.8708	-0.8681	0.920
-1.0000	Easom	-0.0333	-0.0395	0.918
-	Egg Holder	-	-767.3098	0.195
-19.2085	Holder Table	-19.2081	-19.1103	0.327
0.0000	Matyas	0.0000	0.0001	0.122
0.0000	Schaffer 2	0.0023	0.0023	0.993
-	Shubert	-	-159.7021	0.033

The average results for the NFE and time values are calculated for the 30 runs on each of the four schemes.

Table 9 shows the values for schemes I and II on each benchmark function.

Table 9. Average results of the different schemes regarding the NFE variables and execution time.

	Scheme I		**Scheme II**	
Function	**NFE**	**time**	**NFE**	**time**
Ackley	55527.7	4860.7	31556.0	3270.5
Michaelwikcz	40781.2	4406.9	38876.2	3967.6
Rastrigin	41811.0	4540.4	28611.6	3273.9
Rosenbrok	42265.7	4664.5	32620.5	3618.4
Schwefel	56504.8	4155.0	41796.4	3414.1
Sphere	41535.9	3330.1	42232.6	3428.6
Styblinski	41545.8	3307.1	39536.2	3443.3
Zakharov	43772.7	3561.1	41164.5	3497.8
Beale	41089.8	3839.0	35473.4	3553.2
Booth	47924.9	4277.0	40768.2	3878.8
Cross In Tray	48067.7	4306.3	53430.4	4688.1
Drop Wave	41860.6	3905.2	45815.9	4311.8
Easom	56376.8	4742.3	57087.8	4945.9
Egg Holder	56469.9	5403.0	41484.3	4075.8
Holder Table	47909.8	4967.1	45461.4	4359.0
Matyas	48012.7	4944.9	47669.3	4530.5
Schaffer 2	47086.3	4844.4	28157.3	3075.7
Shubert	41740.7	3412.9	28503.6	2617.2

Table 10 shows schemes III and IV values on each benchmark function.

Table 10. Average results of the different schemes regarding the NFE variables and execution time.

	Scheme III		Scheme IV	
Function	**NFE**	**time**	**NFE**	**time**
Ackley	28397.0	2642.3	34970.3	3378.5
Michaelwikcz	27624.6	3258.6	37688.3	3982.1
Rastrigin	28518.5	3399.1	29088.9	3284.6
Rosenbrok	31482.9	3555.4	33113.6	3678.9
Schwefel	38489.8	2882.3	42143.2	3058.0
Sphere	30898.6	2607.4	37131.7	2695.0
Styblinski	29133.1	2450.5	38598.5	2861.5
Zakharov	34235.4	2752.8	43078.6	3141.3
Beale	30177.7	2788.7	35717.8	3263.7
Booth	35541.1	3108.8	38808.1	3424.3
Cross In Tray	28032.6	2611.1	44224.6	3737.6
Drop Wave	28062.0	2667.1	47192.7	4012.9
Easom	28000.0	2563.6	50800.7	4254.3
Egg Holder	38283.2	3848.3	41806.9	4239.0
Holder Table	28787.8	3346.8	45688.5	4369.4
Matyas	31181.9	3522.4	48440.2	4600.1
Schaffer 2	28123.9	3317.0	27974.6	3082.4
Shubert	28444.3	2424.9	28712.0	2186.1

4.5 Box Diagrams

Here are the box diagrams for the cases where the tuning changed the results (Figure 9).

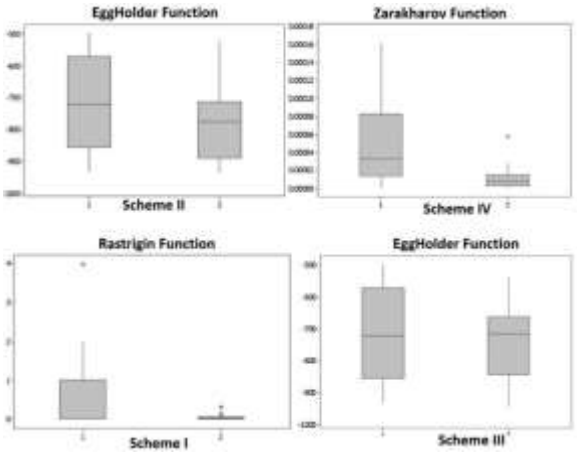


Fig. 9: Cases where the parametrization proposal improved the cost variable results in contrast with the default parametrization results in the independent runs group of 30 repetitions for the BFOA algorithm.

5 Discussion

Phase 1, developed in this study, allowed us to observe an overview of the behavior and performance of BFOA in the 18 benchmark functions described. From this phase, it was possible to make a performance comparison considering the variables NFE, time, and cost. As a next step of phase 1, additional tests were made with the algorithm considering a set of parameterization values in a benchmark function. This part of the process allowed us to observe a new panorama of performance concerning the multiplicity of values between the four parameters of attraction and repulsion in BFOA and proceed to phase 2, in which from the results obtained between the 18 benchmark functions and the observation of the results of parameter values on landscape graphs, four work schemes were integrated.

With these schemes, more specific tests were carried out with the fact that 30 runs were again made for each benchmark function in each proposed scheme.

This finding is consistent with the literature that the performance of an optimization algorithm depends on the problem to be attacked, as well as the parameter values and limits that direct the search for solutions. Likewise, as part of phase 2, t-tests were carried out to determine significant differences between sets of 30 runs of each benchmark function, being a set corresponding to the runs of the algorithm with the default values, in contrast to another independent set of runs made with the different values of the proposed schemes. This contrast was made by calculating the significance value of the t-test, with which it is possible to reject the null hypothesis that there is no significant difference between the mean of the independent samples. In this sense, Table 11 shows those cases in which a significant difference was found between said pair of sets of runs. Likewise, the values of significance *p* marked in bold for the corresponding cases are observed.

Table 11. The p-value for each benchmark function where any scheme made a significant difference.

Function	**Scheme I**	**Scheme II**	**Scheme III**	**Scheme IV**
Michaelwikcz	0.001	0	0.001	0
Rastrigin	0	0.217	0.843	0.865
Rosenbrok	0.014	0.354	0	0.153
Sphere	0.322	0.017	0	0
Styblinski	0.004	0	0.02	0.637
Zakharov	0.005	0.011	0	0
Beale	0.006	0.008	0	0.02
Booth	0	0.005	0.006	0.003
Cross In Tray	0.026	0.081	0	0.682
Drop Wave	0	0.353	0.168	0.92
Holder Table	0.1	0.138	0	0.327
Matyas	0	0.275	0	0.122
Schaffer 2	0.742	0.14	0	0.993
Shubert	0.616	0.168	0.05	0.033

It is noteworthy in the case of benchmark functions in which no difference was found. As are the functions Ackley, Schwefel, Easom, and Egg

Holder. This finding might be due to the search space and the nature of the algorithm based on foraging bacteria. Finally, the box diagrams visually describe cases from the EggHolder, Zarackarov, and Rastrigin functions in which an improvement was found regardless of the statistic *t*-test. It is recommended to perform a more extensive test on the EggHolder function.

6 Conclusion

The results presented in this study conclude that the BFOA algorithm is susceptible to particular tuning according to the problem aimed to solve. The benchmark functions used for this experiment can be taken as a reference for future tunings. As a prospective study, the proposed schemes can be used as a reference. It is essential to notice that in terms of time and NFE, scheme III has remarkable results regarding the other three. In regards to the cost, by adding the absolute differences of the averages throughout the 18 benchmark functions from the optimal vs. the result of each scheme, results outstanding scheme II; however, to exploit the usefulness of this study, the results of the interest function must be considered (any of the 18 presented) and in the light of the explored parametrization schemes.

The limitations of this study are related to the parameter values explored. Only one of the four parameters was adjusted, while the other three remained on the default value. It is recommended to perform a more extensive test for the generated schemes in this work. In future directions, BFOA might be tested on schemes III and II, leaving behind benchmark functions to work on problems from a specific field.

Acknowledgement:
The authors would like to acknowledge the Systems Faculty from the Autonomous University of Coahuila for research support.

References:

[1] A. Eiben M. C. Schut. New ways to calibrate evolutionary algorithms. Advances in Metaheuristics for Hard Optimization, P. Siarry and Z. Michalewicz, Eds., Natural Computing Series, 2008, pp.153-177.

[2] Dragoi, E. D. Parameter control and hybridization techniques in differential evolution: a survey. Artif Intell , 2016 https://doi.org/10.1007/s10462-015-9452-8.

[3] S. Das, S. Dasgupta, A. Biswas, A. Abraham, A. Konar, On Stability of the Chemotactic Dynamics in Bacterial-Foraging Optimization Algorithm, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, Vol.39, No.3, 2009, pp. 670-679.

[4] J. Brownlee, *Clever Algorithms*, Brownlee, 2011.

[5] K. Passino, Bacterial Foraging Optimization, *International Journal of Swarm Intelligence Research*, Vol.1, No.1, 2010, pp.1-16.

[6] P. Kou, J. Zhou, C. Li, Y. He, H. He, Dentification of Hydraulic Turbine Governor System Parameters Based on Bacterial Foraging Optimization Algorithm, *2010 6th International Conference on Natural Computation*, 2010, pp.3339-3334.

[7] K. Vaisakh, P. Praveena, R. S. M. Rao, PSO-DV and Bacterial Foraging Optimization Based Dynamic Economic Dispatch with Non-Smooth Cost Functions, *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies*, 2009, pp.135-139.

[8] N. Dehghan, I. Ziari, Optimization Placement APF Based on Bacterial Foraging Algorithm, *2011 46th International Universities' Power Engineering Conference (UPEC)*, 2011, pp.1-4.

[9] E. S. Ali, S. M. Abd-Elazim, Hybrid BFOA-PSO Approach for SSSC Damping Controller Design, *2013 International Conference on Control, Decision and Information Technologies (CoDIT)*, 2013, pp.464-469.

[10] D. M. S. Regis, S. P. Kumar, G. Devadhas, An Optimum Setting of Controller for a dc-dc converter Using Bacterial Intelligence Technique, *Innovative Smart Grid Technologies-India (ISGT India)*, 2011, pp.204-210.

[11] G. Wu, Application of Adaptive PID Controller Based on Bacterial Foraging Optimization Algorithm, *2013 25th Chinese Control and Decision Conference (CCDC)*, 2013, pp.2353-2356.

[12] A. Jati, G. Singh, P. Rakshit, A. Konar, E. Kim, A. K. Nagar, A Hybridisation of Improved Harmony Search and Bacterial Foraging for Multi-Robot Motion Planning, *2012 IEEE Congress on Evolutionary Computation*, 2012, pp.1-8.

[13] M. Minshed, LADAR Signal Modeling Using Bacterial Foraging Optimization Algorithm

(BFOA), *2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012)*, 2012, pp.352-355.

[14] G. Wu, Economic Dispatch of Hydro Power System Based on Bacterial Foraging Optimization Algorithm, *2013 25th Chinese Control and Decision Conference (CCDC)*, 2013, pp.865-868.

[15] S. I. Bejinariu, Image Registration Using Bacterial Foraging Optimization Algorithm on Multi-Core Processors, *2013 4th International Symposium on Electrical and Electronics Engineering (ISEEE)*, 2013, pp.1-6.

[16] S. Dasgupta, A. Biswas, S. Das, A. Abraham, Automatic Circle Detection on Images with an Adaptive Bacterial Foraging Algorithm, *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, 2008, pp.1695-1696.

[17] S. Dasgupta, S. Das, A. Abraham, A. Biswas, Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis, *IEEE Transactions on Evolutionary Computation*, Vol.13, No.4, 2009, pp.919-941.

[18] A. Abraham, A. Biswas, S. Dasgupta, S. Das, Analysis of Reproduction Operator in Bacterial Foraging Optimization Algorithm, *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008, pp.1476-1483.

[19] S. Dasgupta, A. Biswas, S. Das, B. K. Panigrahi, A. Abraham, A Micro-Bacterial Foraging Algorithm for High-Dimensional Optimization, *2009 IEEE Congress on Evolutionary Computation*, 2009, pp.785-792.

[20] P. Borovska, E. Asenov, V. Gancheva, Exploring the Efficiency of Parallel Bacteria Foraging Metaheuristics for Job Shop Scheduling Problem Optimization, *Proceedings of the 12th International Conference on Computer Systems and Technologies*, 2011, pp.88-94.

[21] S. Das, A. Biswas, S. Dasgupta, A. Abraham, Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications, *Foundations of Computational Intelligence*, Vol.3, 2009, pp.23-55.

[22] R. M. Thomas, Survey of Bacterial Foraging Optimization Algorithm, *International Journal of Science and Modern Engineering*, Vol.1, No.4, 2013, pp.11.

[23] U. K. Rout, R. K. Sahu, S. Panda, Gravitational Search Algorithm based Automatic Generation Control for Interconnected Power System, *2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT)*, 2013, pp.558-563.

[24] E. Mezura-Montes, B. Hernández-Ocaña, Modified Bacterial Foraging Optimization for Engineering Design, In: *Intelligent Engineering Systems Through Artificial Neural Networks,* ASME Press, 2009, pp.269-281.

[25] A. Baijal, V. S. Chauhan, T. Jayabarathi, Application of PSO, Artificial Bee Colony and Bacterial Foraging Optimization Algorithms to Economic Load Dispatch: An Analysis, *arXiv:111129882*011.

[26] E. Mezura-Montes, B. Hernández-Ocana, Bacterial Foraging for Engineering Design Problems: Preliminary Results, *Memorias del 4o Congreso Nacional de Computacion Evolutiva (COMCEV'2008), CIMAT*, 2008, pp.227-232.

[27] P. Praveena, K. Vaisakh, S. Rao, A Bacterial Foraging and Pso-De Algorithm for Solving Dynamic Economic Dispatch Problem with Valve-Point Effects, *2010 1st International Conference on Integrated Intelligent Computing*, 2010, pp.227-232.

[28] S. Narendhar, T. Amudha, A Hybrid Bacterial Foraging Algorithm For Solving Job Shop Scheduling Problems, *arXiv preprint 121149712*012.

[29] C. M. Moncayo, D. A. G. Alvarado, J. M. A. Osorio, " Discrete Methods Based on Bacterial Chemotaxis and Genetic Algorithms to Solve the Problem of Plant Distribution in Manufacturing, Science and Engineering Cells in New Granada " (Métodos Discretos Basados en Quimiotaxis de Bacterias y Algoritmos Genéticos para Solucionar el Problema de la Distribución de Planta en Celdas de Manufactura, *Ciencia e Ingeniería Neogranadina)*, Vol.24, No.1, 2014, pp.6-28.

[30] H. Shen, Y. Zhu, X. Zhou, H. Guo, C. Chang, Bacterial Foraging Optimization Algorithm with Particle Swarm Optimization Strategy for Global Numerical Optimization, *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, 2009, pp.497-504.

[31] Chen, H. Z, Adaptive Bacterial Foraging Optimization. Abstract and Applied Analysis,2011, pp.248-272.

[32] Almeida, F. G.-E, Parameterized Schemes of Metaheuristics: Basic Ideas and Applications

With Genetic Algorithms, Scatter Search, and GRASP. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2013, pp.570-586

[33] Cutillas-Lozano, J, Determination of the Kinetic Constants of a Chemical Reaction in Heterogeneous Phase Using Parameterized Metaheuristics, 2013, pp.787-796,

[34] Cruz-Duarte, J. O.-B.-M. Towards a Generalised Metaheuristic Model for Continuous Optimisation Problems, 2020, pp.8-2046

[35] Cabrera-Hernández, M. A.-F.-A.-S.-H. Parameters Influencing the Optimization Process in Airborne Particles PM10 Using a Neuro-Fuzzy Algorithm Optimized with Bacteria Foraging (BFOA). International Journal of Intelligence Science, 2019, pp.67-91 36. M. López-Ibáñez, J. D.-L. The irace package, iterated race for automatic algorithm configuration. Operations Research Perspectives, 2016, pp.43-58.

[37] Hernández-Ocaña, B. P.-M.-F.-A.-Y. Two-Swim Operators in the Modified Bacterial Foraging Algorithm for the Optimal Synthesis of Four-Bar Mechanisms. Computational Intelligence and Neuroscience, 2016, pp 158-19638.H. Bremermann, Chemotaxis and Optimization, *Journal of the Franklin Institute*, Vol.297, 1974, pp.397-404.

[39] N. Chase, M. Redemacher, E. Goodman, R. Averill, R. Sidhu, *A Benchmark Study of Optimization Search Algorithms*, Red Cedar Technology, 2010.

[40] M. Molga, C. Smutnicki, *Test Functions for Optimization Needs*, 2005.

[41] S. Surjanovic, D. Binham, Virtual Library of Simulation Experiments, 2013, http://www.sfu.ca/~ssurjano/index.html (Accessed Date: 17/8/2023)

[42] A. R. Al-Roomi, *Unconstrained Single-Objective Benchmark Functions Repository*, Dalhousie University, Electrical and Computer Engineering, 2023.

[43] M. Jamil, X.-S. Yang, A Literature Survey of Benchmark Functions for Global Optimization Problems, *International Journal of Mathematical Modelling and Numerical Optimization*, Vol.4, No.2, 2013, pp.150-194.

[44] W. Cai, L. Yang, Y. Yu, Solution of Ackley Function Based on Particle Swarm Optimization Algorithm, *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications*, 2020, pp.563-566.

[45] V. Plevri, G. Solorzano, A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking, *Data*, Vol.7, No.4, 2022, p.46.

[46] E. W. Weisstein, Rosenbrock Function, 2023, https://mathworld.wolfram.com/RosenbrockFunction.html (Accessed Date: 17/8/2023)

[47] A. Bonilla-Petriciolet, J. C. Tapia-Picazo, C. Soto-Becerra, J. G. Zapiain-Salinas, "Perfiles de Comportamiento Numérico de los Métodos Estocásticos Simulated Annealing y Very Fast Simulated Annealing en Cálculos Termodinámicos, *Ingeniería Investigación y Tecnología*" (Perfiles de Comportamiento Numérico de los Métodos Estocásticos Simulated Annealing y Very Fast Simulated Annealing en Cálculos Termodinámicos, *Ingeniería Investigación y Tecnología)*, Vol.12, No.I, 2011, pp.51-62.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed to the present research at all stages, from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare relevant to this article's content.