ISSN 2007-9737

Comparative Analysis of the Bacterial Foraging Algorithm and Differential Evolution in Global Optimization Problems

Adrian García-López, Oscar Chávez-Bosquez,
José Hernández-Torruco, Betania Hernández-Ocaña

Universidad Juárez Autónoma de Tabasco,
División Académica de Ciencias y
Tecnologías de la Información,
México

221H18002@alumno.ujat.mx,
{oscar.chavez,jose.hernandezt, betania.hernandez}@ujat.mx

Abstract. There are bio-inspired metaheuristics in nature rarely used in areas where there is not domain or knowledge of computational algorithms, to mention some, medicine, finance and administration. TS-MBFOA, a bacteria-based algorithm and the Differential Evolution Algorithm (DEA), are metaheuristic algorithms proposed for the optimization of complex problems mathematically modeled as linear or non-linear problems. In this paper, these algorithms are implemented to analyze their performance in the search for better solutions in constrained optimization problems. Tests were conducted on four optimization problems known in the literature as benchmark problems. Both algorithms were run in 30 independent executions for each problem with the same number of generations and evaluations. Although the parameters of each algorithm are different, the number of evaluations was selected for a fair comparison. Results are similar for both algorithms, however, DEA obtains better results for the problem with the larger number of constraints. Additionally, DEA generates solutions in less time than TS-MBFOA. The nonparametric Wilcoxon Signed Rank Test indicates significant differences in only 3 problems. The convergence graph of both algorithms for each problem shows that after 50 generations, both algorithms are close to the best known solution in the state of the art.

Keywords. Bacterial foraging, differential evolution, global optimization, metaheuristics.

1 Introduction

Bio-inspired algorithms are computational techniques inspired by nature, primarily the simulation or emulation of simple and intelligent processes of certain animals, insects, or bacteria in search of food or shelter.

These algorithms arise to improving search algorithms to solve numerical and combinatorial optimization problems [9]. These algorithms are classified as metaheuristics and incorporate techniques and strategies to design or improve mathematical procedures aimed at obtaining high performance. [18].

Metaheuristics generate a set of results for a particular problem that is totally or approximately global optimum.

These algorithms are classified into two groups based on different natural phenomena: Evolutionary Algorithms (EAs) emulate the evolutionary process of the species [2] and Swarm Intelligence Algorithms (SIAs) emulate the collaborative behaviour of certain simple and intelligent species such as bacteria [14], bees [8], ants [1], among others [3]. Metaheuristics were created to solve unconstrained optimization problems.

However, to handle this problems, mechanisms such as feasibility rules, special operators, decoders, among others are implemented. The use of metaheuristics is an effective alternative for solving Constraint Numerical optimization Problems (CNOPs) [12].

Generally, a CNOP is known as a general nonlinear programming problem and can be defined as:

$$\text{minimize : } f(\vec{x})$$

subject to:

$$g_i(\vec{x}) = 0, \quad i = 1, 2, ..., m \text{ or} \tag{1}$$

$$h_j(\vec{x}) \leq 0, \quad j = 1, 2, ..., p, \tag{2}$$

where $\vec{x} \in \mathbb{R}^n$ such that $n \geq 1$, where $\vec{x}$ is the solution vector $\vec{x} = [x_1, x_2, x_3, ..., x_n]^T$, where each $x_i, i = 1, 2, 3, ..., n$ is delimited by the lower and upper limit $L_i \leq x_i \leq U_i, \quad k = 1, 2, ..., D$; D is the number of design variables, m is the number of inequality constraints, and p is the number of equality constraints (in both cases, the constraints can be linear or non-linear).

If we denote by F the feasible region (where all the solutions that satisfy the problem are found) and by S the entire search space, then $F \subseteq S$ [7].

There are different EAs techniques used to solve CNOP, highlighting: Genetic Programming (GP), Genetic Algorithms (GA), Evolutionary Programming (EP) and Differential Evolution Algorithm (DEA).

DEA is a simple and easy to implement technique using the basic operators of genetic algorithms: mutation, crossover and selection. Despite its simplicity and smallthe number of parameters, it generates good results in CNOPs.

Since then, DEA has been proven in competitions such as the International Contest on Evolutionary optimization (ICEO) of the IEEE [15, 16] and in a wide variety of real-world applications, such as the optimization of the four-bar mechanism [19] or for global optimization of engineering and chemical processes [10].

The Bacterial Foraging Optimization Algorithm (BFOA) is a SIA based on foraging of *Escherichia Coli* bacteria [14], which simulates the process of chemotaxis (swim and tumble), swarming, reproduction, and elimination-dispersal.

Algorithm 1: TS-MBFOA pseudocode. S_b is the number of bacteria, N_c is the number of chemotaxis cycles, β is the scaling factor, R is the stepsize, S_r is the number of bacteria to reproduce, $Repcycle$ is the reproduction frequency and $GMAX$ is the number of generations.

```
1  Create an population of random bacteria θ^i(j,0) ∀i, i = 1,...,S_b.
2  Evaluate f(θ^i(j,0)) ∀i, i = 1,...,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              Chemotaxis process: Interleaving the
               proposed swims with Eqs. 3 and 4.
7              Apply grouping (Eq. 7) using β for the
               bacteria θ^i(j,G).
8          end
9      end
10     if G mod RepCycle == 0 then
11         Reproduction process.
12         Ordering the population (follow the feasibility
           rules).
13         Duplicate the best bacteria S_r.
14         Eliminate S_b − S_r worst bacteria.
15     end
16     Elimination-dispersion process.
17     Eliminating the worst bacteria θ^w(j,G) from the
       current population considering the technique of
       handling constraint.
18     Update the step size vector using Eq. 6.
19 end
```

These bacteria in the process face several problems in their search for food [11]. From this algorithm, significant modifications were made: the number of parameters was reduced, an operator for handling constraints was incorporated [13] and a mutation operator similar to an EAs [6]; this is called Two-Swim Modified-BFOA (TS-MBFOA).

This metaheuristic allows competitive and favourable results when solving CNOPs, with a good configuration of its parameters. This approach has been successfully used to solve engineering design problems, such as the well-known Tension Compression Spring [7], the generation of healthy menus [4] and optimization of a Smart-Grid [5].

In the real world, TS-MBFOA and DEA have implementations for solving optimization problems,

Algorithm 2: DEA pseudocode. The parameters are as follows: $Population$ of individuals, F is mutation, CR is crossover and $GMAX$ is the number of generations.

```
Create a random initial population
x_{j,i}, i = 1, ..., NP, j = 1, ..., D
Evaluate population
f(x_{j,i}), i = 1, ..., NP, j = 1, ..., D
for g = 0 to GMAX do
    for i = 1 to NP do
        Randomly select r1 ≠ r2 ≠ r3 ≠ i
        j_rand = randint[1, D]
        for j = 1 to D do
            if rand_j[0,1) < CR  or  j = j_rand
            then
                v_{j,i,g} =
                x_{j,i,g}, r1 + F(x_{j,i,g}, r2 − x_{j,i,g} r3)
            end
            else
                v_{j,i,g} = x_{j,i,g}
            end
        end
        if f(v_{j,i,g}) ≤ f(x_{j,i,g}) then
            x_{g+1,j} = v_{j,i,g}
        end
        else
            x_{g+1,j} = x_{j,i,g}
        end
    end
end
```

Table 1. TS-MBFOA and DEA Parameters

TS-MBFOA		DEA	
Parameter	Value	Parameter	Value
S_b	15	Population	50
R	0.0005	F	0.7
N_c	8	CR	0.8
β	1.95	$GMAX$	500
S_r	1		
$Repcycle$	100		
$GMAX$	500		

however, these algorithms are not fully exploited in different areas where researchers are not aware of their adaptation and implementation.

Therefore, this research is motivated to explore the capabilities of both algorithms in the solution of particular CNOPs known as: Pressure Vessel, Process Synthesis MINLP, Tension Compression Spring and Quadratically constrained quadratic program. These algorithms are implemented in a free and cross-platform programming language.

Results obtained were validated using basic statistics such as best value, mean, median, standard deviation and worst value.

Also, the nonparametric Wilcoxon Signed Rank Test (WSRT) was applied to measure the consistency of the results.

Finally, convergence graphs of the median number of executions for each algorithm in each problem are presented to notice the performance of the algorithms.

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

The TS-MBFOA is a proposed algorithm for solving CNOPs [7], where bacteria i is a potential solution and is denoted as $\theta^i(j, G)$, where j is the chemotaxis loop and G is the generational loop (chemotaxis, swarming, reproduction and elimination-dispersal).

The chemotaxis process is interleaved with exploitation or exploration swim in each cycle.

The process begins with the classic swim (exploration and mutation between bacteria) and is calculated with Eq. 3, where a bacterium will not necessarily interleave exploration and exploitation swims, because if the new position of a given swim $\theta^i(j+1, G)$ has better fitness than the original position $\theta^i(j, G)$, then another swim at the same direction will occur in the next loop.

Otherwise, a new tumble will be calculated. The process stops after N_c attempts. The exploration swim uses the mutation between bacteria and is calculated by:

$$\theta^i(j+1, G) = \theta^i(j, G) + (\beta)(\theta^r_1(j, G) - \theta^r_2(j, G)). \quad (3)$$

The swim operator is calculated with Eq. 4:

$$\theta^i(j+1, G) = \theta^i(j, G) + C(i, G)\phi(i), \quad (4)$$

where $\phi(i)$ is calculated with the original BFOA tumble operator defined in Eq. 5:

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}}, \tag{5}$$

where $\Delta(i)^T$ is a random vector generated with elements inside an interval $[-1, 1]$. $C(i, G)$ is the random step size of each bacteria updated with Eq. 6:

$$C(i, G) = R * \Theta(i), \tag{6}$$

where $\Theta(i)$ is a random vector of size n with elements within the range of each decision variable: $[U_k, L_k]$, $k = 1, ...n$, and R is a user-defined parameter for scaling the step size.

In the middle cycle of the chemotaxic process, the swarming operator is applied with Eq. 7:

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

Bacteria are ordered in the reproduction process, eliminating the worst bacteria $S_b - S_r$ and the best bacteria are duplicated every certain number of loops.

In elimination-dispersion, the worst bacteria are eliminated from the population $\theta^w(j, G)$ and a new one is random generated. In this proposal, the original bias mechanism of the TS-MBFOA is not used to consume less computational cost. TS-MBFOA pseudocode is presented in Algorithm 1.

3 Differential Evolution Algorithm (DEA)

DEA, developed by Storn and Price in 1995, was proposed to solve numerical optimization problems [17].

This algorithm is competitive in global optimization problems, its strategy is based on population search. The algorithm starts from a population of NP D-dimensional individuals, also called parent vectors.

Table 2. Best result found by the TS-MBFOA and DEA in 30 independent executions to each CNOPs

CNOP	Nv	Alg	Bv	T
Problem 1 $f(\vec{x})^* = 6059.946$	4	TS-MBFOA	**8796.84951**	44 **20**
		DEA	9375.75820	
Problem 2 $f(\vec{x})^* = 4.579582$	7	TS-MBFOA	4.26075	55 **31**
		DEA	**3.73269**	
Problem 3 $f(\vec{x})^* = 0.012681$	3	TS-MBFOA	**0.012665**	96 **21**
		DEA	0.012665	
Problem 4 $f(\vec{x})^* = -118.704$	2	TS-MBFOA	-118.70485	**22** 27
		DEA	-118.70485	

An individual of the NP population represents a solution to the problem and is computed as in Eq. 8:

$$\begin{aligned} & x_{j,i,g}, \\ & i = 1, 2, 3, ..., NP, \\ & j = 1, ..., D, \\ & g = 1, 2, ..., GMAX, \end{aligned} \tag{8}$$

where $x_{j,i,g}$ is an individual with j dimensions or number of variables. i is the individual in the population NP and g is the number of generations in the process the algorithm to the maximum number GMAX. The process of the DE algorithm is described below.

Initialization process: individuals are randomly generated within the search space limited by the upper and lower limit of each problem variable, i.e.: $L_j \leq x_j \leq U_j$. In each generation, individuals mutate, recombine and select to produce new offspring. If the descendant performs better than the parent, it is integrated into the next generation.

Mutation process: the search direction controls the magnitude of displacement in the search space and the speed of convergence to the optimal solution. The mutated vector is constructed from two vectors weighted by a scaling factor, as shown in Eq. 9:

$$\begin{aligned} v_{j,i,g} = x_{j,i,g}, r1 + F(x_{j,i,g}, r2 - x_{j,i,g} r3), \\ r1 \neq r2 \neq r3 \neq i, \end{aligned} \tag{9}$$

Table 3. Basic statistics of the best results of the 30 iterations of the TS-MBFOA and DEA. The best values are highlighted in bold

CNOP	Measure	TS-MBFOA	DEA
Problem 1 $f(\vec{x})^* = 6059.946$	Media	**8796.84982**	9375.75820
	Median	**8796.84951**	9375.75820
	Std. dev.	0.00166	**5.45696E-12**
	Worst	**8796.85879**	9375.75820
Problem 2 $f(\vec{x})^* = 4.579582$	Media	4.27758	**3.73305**
	Median	4.27485	**3.73305**
	Std. dev.	0.01223	**2.01891E-4**
	Worst	4.32189	**3.73348**
Problem 3 $f(\vec{x})^* = 0.012681$	Media	**0.012665**	**0.012665**
	Median	**0.012665**	**0.012665**
	Std. dev.	0.0000013	**7.110684E-12**
	Worst	0.012672	**0.012665**
Problem 4 $f(\vec{x})^* = -118.704$	Media	**-118.7048**	**-118.70485**
	Median	**-118.7048**	**-118.70485**
	Std. dev.	7.21821E14	**4.26325E-14**
	Worst	**-118.7048**	**-118.7048**

where $v_{j,i,g}$ is the descendant generated by crossing three individuals from the population $r1, r2, r3$ totally different from each other and randomly selected with a uniform distribution.

Crossover process: controls the recombination of the mutation of individuals to generate a new descendant, where the *CR* operator is an end-user-defined parameter, which can be randomly or statically defined.

In the *selection* process: the descendant is evaluated in the problem function. If the descendant obtains a better result than the parent, then it replaces the parent in the next generation of the algorithm, otherwise the parent is kept (Eq. 10):

$$x_{g+1,i} = \begin{cases} v_{j,i,g} & \text{if } f(v_{j,i,g}) \leq f(x_{j,i,g}), \\ x_{j,i,g} & \text{otherwise}, \end{cases} \quad (10)$$

where f is the objective function of the problem to optimize. The Algorithm 2 shows the classic DEA.

4 Experimentation and Results

TS-MBFOA and DEA were implemented in the Java programming language, a free and cross-platform language.

The CNOPs to solve by both algorithms have their own characteristics, such as: different numbers of variables, number and types of constraints, ranges of variables, among others; as presented below in its mathematical model.

Problem 1: Pressure Vessel.

Minimize: $0.6224x_1x_3x_4 + 1.7781x_2{x_3}^2 + 3.1661{x_1}^2x_4 + 19.84{x_1}^2x_3$.

subject to:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0, \quad (11)$$

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \leq 0, \quad (12)$$

$$g_3(\vec{x}) = -\pi{x_3}^2x_4 - \frac{4}{3}\pi{x_3}^3 + 1296000 \leq 0, \quad (13)$$

$$g_4(\vec{x}) = x_4 - 240 \leq 0, \quad (14)$$

where: $1 \leq x_1 \leq 99$, $1 \leq x_2$ 99, $10 \leq x_3 \leq 200$ and $10 \leq x_4 \leq 200$:

$$f(\vec{x})^* = 6059.946. \quad (15)$$

Problem 2: Process Synthesis MINLP.

Minimize: $(y_1-1)^2+(y_2-2)^2+(y_3-1)^2-log(y_4+1)+(x_1-1)^2+(x_2-2)^2+(x_3-3)^2$

subject to:

$$y_1 + y_2 + y_3 + x_1 + x_2 + x_3 \leq 5, \quad (16)$$

$$y_3^2 + x_1^2 + x_2^2 + x_3^2 \leq 5.5, \quad (17)$$

$$y_1 + x_1 \leq 1.2, \quad (18)$$

$$y_2 + x_2 \leq 1.8, \quad (19)$$

$$y_3 + x_3 \leq 2.5, \quad (20)$$

$$y_4 + x_1 \leq 1.2, \quad (21)$$

$$y_2^2 + x_2^2 \leq 1.64, \quad (22)$$

$$y_3^2 + x_3^2 \leq 4.25, \quad (23)$$

$$y_2^2 + x_3^2 \leq 4.64, \quad (24)$$

where: $0 \leq x_i \leq (1.2, 1.8, 2.5)$ $i = 1, 2, 3$
$y_i = 0, 1,\ i = 1, 2, 3, 4$:

$$f(\vec{x})^* = 4.579582. \quad (25)$$

Problem 3: Tension Compression Spring. Minimize: $(N+2)Dd^2$ subject to:

$$g_1(\vec{x}) = 1 - \frac{D^3 N}{71785 d^4} \leq 0, \tag{26}$$

$$g_2(\vec{x}) = \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{1}{5108d^2} - 1 \leq 0, \tag{27}$$

$$g_3(\vec{x}) = 1 - \frac{140.45d}{D^2 N} \leq 0, \tag{28}$$

$$g_4(\vec{x}) = \frac{D+d}{1.5} - 1 \leq 0, \tag{29}$$

$$\tag{30}$$

where: $0.05 \leq d \leq 2$, $0.25 \leq D$ 1.3 and $2 \leq N \leq 15$:

$$f(\vec{x})^* = 0.012681. \tag{31}$$

Problem 4: Quadratically constrained quadratic program.
Minimize: $x_1^4 - 14x_1^2 + 24x_1 - x_2^2$
subject to:

$$-x_1 + x_2 - 8 \leq 0, \tag{32}$$

$$x_2 - x_1^2 - 2x_1 + 2 \leq 0, \tag{33}$$

where: $(-8, 0) \leq x_i \leq (10, 10) i = 1, 2$:

$$f(\vec{x})^* = -118.7048. \tag{34}$$

Each algorithm has parameters that must be calibrated to generate competitive results.

Previously, we performed tests in search of a good calibration, to run each algorithm on the test problems.

Tab. 1 presents the parameter settings of the TS-MBFOA and DEA. TS-MBFOA and DEA, where a number of 500 fixed generations were established for each algorithm.

This yields to around 60,500 evaluations for each algorithm, which allows a fair comparison between the results and to plot the convergence of the algorithms.

In the tests conducted with each algorithm, 30 independent iterations were adjusted to measure the consistency of the results.

The results of the independent runs of each algorithm with the four CNOPs are presented in Tab. 2, where **CNOP** is the problem to solve, $f(\vec{x})^*$

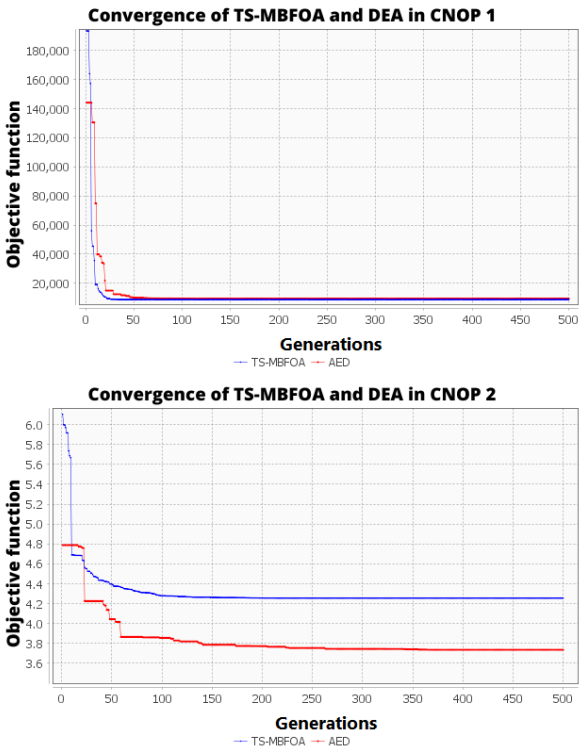

Fig. 1. Typical convergence in CNOP 1 and 2

is the best known value in the literature, **Nv** is the number of design variables, **Alg** are the algorithms used, **BV** is the best value found and, **T** is the time in seconds of total iterations.

TS-MBFOA obtained better results than DEA in most of the runs. However, the runtime of DEA was smaller in many cases.

In problem 1, the TS-MBFOA obtained a result of 8796.84951, better than the one obtained by DEA. However, this solution is feasible but not competitive with the best known optimal solution, which is $f(\vec{x})^* = 6059.946$.

For problem 2, DEA finds the best optimal solution of 3.73269. With TS-MBFOA, the best value found is 4.26075, a non-competitive solution. For problems 3 and 4, both algorithms find the global optimum of the problem similar to the optimal solution known in the state-of-the-art.

Generally speaking, both algorithms generate results in less than 60 seconds, except for problem 3. With TS-MBFOA this experiment tooked 96

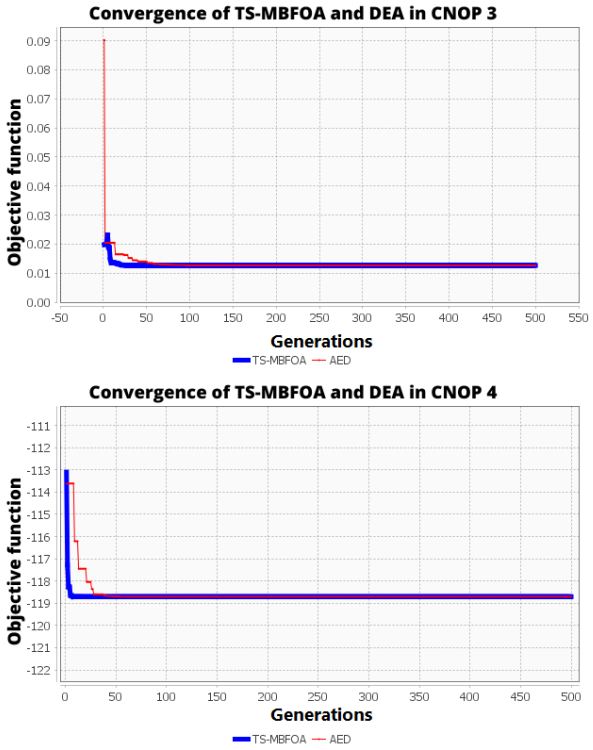

Fig. 2. Typical convergence in CNOP 3 and 4

seconds, but it is a reasonable time with a competitive result.

For each experiment performed with both metaheuristics, basic statistics were applied to check the consistency of the results obtained.

Tab. 3 shows the basic statistics of the 30 iterations performed for each CNOPs with both algorithms. The TS-MBFOA obtains better results in problem 1. In problems 3 and 4, both algorithms obtain equal results.

In problem 2, DEA obtains a better result. According to the standard deviation, the consistency of the solutions found by DEA is better than TS-MBFOA.

Figs. 1 and 2 present the behavior of both algorithms at each CNOP of iteration number 15 of the 30 runs performed independently (median).

The convergence of both algorithms is similar in 3 of the 4 CNOPs, both algorithms from the first 50 generations already reach optimal solutions.

Only in problem 2 both algorithms require more than 200 generations to converge to an optimal solution. It is worth mentioning that CNOP 2 is a highly constrained problem. The WSRT non-parametric test was applied with a 95% confidence level, being 5% the significant level.

The result obtained in CNOP 1, 2, and 3 is *p-value* = 0.00001, a value lower than the significant level.

This indicates a significant difference between the results of both algorithms. Therefore, the lower the *p-value*, the more significant the result. For CNOP 4 the *p-value* = 0, this indicates that there is no significant difference.

5 Conclusion and Future Work

In this work, two metaheuristics were implemented to solve a set of numerical optimization problems with constraints: TS-MBFOA, a swarm intelligence algorithm, and DEA, an evolutionary algorithm. Four benchmark problems were tested on both algorithms, programmed in Java Language.

30 independent runs were performed by each algorithm on each test problem with a number of 500 generations and approximately 60,500 evaluations.

The parameters of each algorithm were adjusted to the number of evaluations allowed in this work. Basic statistics and a non-parametric test called Wilcoxon Signed Rank Test was applied to know the quality and consistency of the results, where both algorithms obtained similar results.

DEA has better consistency of results according to the standard deviation obtained in each problem. TS-MBFOA and DEA obtained better quality results in 1 of the 4 problems. The non-parametric test indicates that there is no significant difference between the results of both algorithms in problem 4.

In the remaining three problems, there is a significant difference (problem 1, 2 and 3). With respect to execution times, both algorithms generate solutions in seconds, however, DEA is the one that generates results in less time with respect to TS-MBFOA and this is due to the simplicity of the processes and the small number of parameters of the evolutionary algorithm.

The convergence graph of each algorithm shows that the TS-MBFOA and DEA after 50 generations begin to converge, but in problem 2, both algorithms converge after 200 generations.

It is necessary to perform a finer adjustment of the parameters of each algorithm and to test both algorithms in more benchmark problems.

Acknowledgments

We thank CONACYT (Ministry of Science in México) for supporting the Doctoral program in Computer Science at the Universidad Juárez Autónoma de Tabasco.

References

1. **Dorigo, M., Maniezzo, V., Colorni, A. (1996).** The ant system: Optimization by a colony of cooperating agents. IEEE Transactions of Systems, Man and Cybernetics-Part B, Vol. 26, No. 1, pp. 29–41. DOI: 10.1109/3477.484436.
2. **Eiben, A. E., Smith, J. E. (2003).** Introduction to evolutionary computing. Natural Computing Series.
3. **Engelbrecht, A. (2005).** Fundamentals of computational swarm intelligence. John Wiley & Sons.
4. **Hernández-Ocaña, B., Chávez-Bosquez, O., Hernández-Torruco, J., Canul-Reich, J., Pozos-Parra, P. (2018).** Bacterial foraging optimization algorithm for menu planning. IEEE Access, Vol. 6, pp. 8619–8629. DOI: 10.1109/ACCESS.2018.2794198.
5. **Hernández-Ocaña, B., Hernández-Torruco, J., Chávez-Bosquez, O., Calva-Yáñez, M. B., Portilla-Flores, E. A. (2019).** Bacterial foraging-based algorithm for optimizing the power generation of an isolated microgrid. Applied Sciences, Vol. 9, No. 6. DOI: 10.3390/app9061261.
6. **Hernández-Ocaña, B., Pozos-Parra, M. D. P., Mezura-Montes, E. (2016).** Improved modified bacterial foraging optimization algorithm to solve constrained numerical optimization problems. Applied Mathematics and Information Sciences, Vol. 10, No. 2, pp. 607–622. DOI: 10.18576/amis/100220.
7. **Hernández-Ocaña, B., Pozos-Parra, M. P., Mezura-Montes, E., Portilla-Flores, E. A., Vega-Alvarado, E., Calva-Yáñez, M. B. (2016).** Two-swim operators in the modified bacterial foraging algorithm for the optimal synthesis of four-bar mechanisms. Computational Intelligence and Neuroscience, Vol. 2016, pp. 1–18. DOI: 10.1155/2016/4525294.
8. **Karaboga, D., Basturk, B. (2007).** Powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization, Vol. 39, No. 3, pp. 459–471. DOI: 10.1007/s10898-007-9149-x.
9. **León, J. A. (2009).** Diseño e implementación en hardware de un algoritmo bio-inspirado. Ph.D. thesis, Instituto Politécnico Nacional. Centro de Investigación en Computación.
10. **Martínez-Zecua, M. Y., Salamanca-Vázquez, L. A., Flores-Pulido, L., Portilla-Flores, E. A., Ortız-Arroyo, A. (2019).** Evolución diferencial para la optimización global de procesos de ingenierıa quımica. Research in Computing Science, Vol. 148, No. 8. DOI: 10.13053/rcs-148-8-2.
11. **Mezura-Montes, E., Cetina-Domínguez, O., Hernández-Ocaña, B. (2010).** Nuevas heurísticas inspiradas en la naturaleza para optimización numérica. , pp. 249–272.
12. **Mezura-Montes, E., Coello-Coello, C. A. (2011).** Constraint-handling in nature-inspired numerical optimization: Past, present and future. Swarm and Evolutionary Computation, Vol. 1, No. 4, pp. 173–194. DOI: 10.1016/j.swevo.2011.10.001.
13. **Mezura-Montes, E., Hernández-Ocaña, B. (2008).** Bacterial foraging for engineering design problems: Preliminary results. Memorias del 4o Congreso Nacional de Computación Evolutiva (COMCEV).
14. **Passino, K. (2002).** Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine, Vol. 22, No. 3, pp. 52–67. DOI: 10.1109/MCS.2002.1004010.
15. **Price, K. (1997).** Differential evolution vs. the functions of the 2nd ICEO. Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC 97), pp. 153–157. DOI: 10. 1109/ICEC.1997.592287.
16. **Price, K., Storn, R. M., Lampinen, J. A. (2006).** Differential evolution: A practical approach to global optimization. Springer Science & Business Media.

17. **Storn, R., Price, K. (1997).** Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, Vol. 11, No. 4, pp. 341–359. DOI: 10.1023/A:1008202821328.

18. **Suarez, O. (2011).** Una aproximación a la heurística y metaheurísticas. INGE@ UAN-Tendencias en la Ingeniería, Vol. 1, No. 2.

19. **Zapata-Zapata, M. F., Mezura-Montes, E., Portilla-Flores, E. A. (2017).** Evolución diferencial con memoria de parámetros para la optimización de mecanismos de cuatro barras. Research in Computing Science, Vol. 134, No. 1, pp. 9–22.

Article received on 02/10/2022; accepted on 15/12/2022.. Corresponding author is Adrian García-López.