A Hybrid Computational Chemotaxis in Bacterial Foraging Optimization Algorithm for Global Numerical Optimization

Yosra JARRAYA, Souhir BOUAZIZ, Adel M. ALIMI
REsearch Group on Intelligent Machines (REGIM),
University of Sfax, National School of Engineers (ENIS),
BP 1173, Sfax 3038, Tunisia,
yosra.jarraya@ieee.org, souhir.bouaziz@ieee.org,
adel.alimi@ieee.org

Ajith Abraham[1,2]
[1]Machine Intelligence Research Labs, WA, USA
[2]IT4Innovations, VSB-Technical University of Ostrava,
Czech Republic
ajith.abraham@ieee.org

***Abstract*— This paper first proposes a simple scheme for adapting the chemotactic step size of the Bacterial Foraging Optimization Algorithm (BFOA), and then this new adaptation and two very popular optimization techniques called Partic le Swarm Optimization (PSO) and Differential Evolution (DE) are coupled in a new hybrid approach named Adaptive Chemotactic Bacterial Swarm Foraging Optimization with Differential Evolution Strategy (ACBSFO_DES). This novel technique has been shown to overcome the problems of premature convergence and slow of both the classical BFOA and the other BFOA hybrid variants over several benchmark problems.**

Keywords*— *Adaptive Bacterial Foraging Optimization Algorithm, Particle Swarm Optimization, Differential Evolution, Hybrid Computational Chemotaxis.

I. INTRODUCTION

Recently, to solve complex search problems of the real world, researchers have been drawing inspiration from nature especially bacterial growth. In recent years, bacterial foraging behavior of the E.Coli bacterium cells has been mimicked to become a rich source of potential engineering applications and computational model. Based on this concept, in 2002, Passino proposed a new evolutionary computation technique known as Bacterial Foraging Optimization Algorithm (BFOA) for distributed optimization and control [1], [2]. So far, BFOA has successfully been applied in many research and application areas such as optimal control [1], harmonic estimation [3], transmission loss reduction [4], active power filter synthesis [5], Grid Scheduling [6] and learning of artificial neural networks [7]. However, Computer simulations over several numerical benchmarks [8] indicate that BFOA possesses a less efficient convergence behavior over multi-modal and rough fitness landscapes as compared to other algorithms inspired from evolution and natural genetics like genetic algorithms (GA) [9] and Differential Evolution (DE) [10], and algorithms inspired from natural swarm like Particle Swarm Optimization (PSO) [11]. Its performance is also affected with the growth of search space dimensionality.

In order to overcome the drawbacks of this algorithm, hybridization of BFOA with other intelligence optimization algorithms appears to find approximate solutions to such problems. In 2007, Kim et al. proposed a hybrid algorithm involving GA and BFOA for function optimization [12]. Furthermore, in 2007 Biswas et al. proposed an improved hybrid BFOA namely Bacterial Swarm Optimization (BSO) [13]. BFOA was also hybridized by Biswas et al. in 2007 with another very popular optimization technique of current interest DE. This leads to a new algorithm named Chemotactic Differential Evolution (CDE) [14].Then, in 2009, Dasgupta et al. proposed the adaptive BFOA (ABFOA) which ameliorates the chemotactic step height [15]. Panda and Naik [16] also presented, in 2012, a modified BFOA named Crossover Bacterial Foraging Otimization Algorithm (CBFOA) that inherits the crossover operator of genetic algorithm.

In this work, a new hybrid approach involving BFOA named Adaptive Chemotactic Bacterial Swarm Foraging Optimization with Differential Evolution Strategy (ACBSFO_DES) is introduced. This approach is centered essentially on the chemotaxis step of the BFOA process by creating a new proposed adaptive chemotactic step size, and by integrating the ideas of Particle Swarm Optimization velocity and Differential Evolution operators to update the movement of the bacterium. The ACBSFO_DES has shown its efficiency mainly with multi-modal and high dimensional functions and also in overcoming the problem of premature convergence. The performance of the proposed method is evaluated using several numerical benchmark problems [8] and is compared with those of related methods.

The remainder of this paper is organized as follows: Section 2 provides a brief literature overview of the bacterial foraging optimization algorithm. Section 3 describes the proposed hybrid optimization algorithm and its implementation in detailed. Section 4 describes the experimental set-up, the simulation strategies and discusses the numerical results on a suite of different numerical optimization problems. Finally, a conclusion is drawn in Section 5.

II. THE BACTERIAL FORAGING OPTIMIZATION ALGORITHM

It's a foraging strategy, where the bacteria try to climb up the nutrient concentration to avoid noxious substance and search for ways out of neutral media. The bacterial swarm proceeds through four principal mechanisms namely chemotaxis, swarming, reproduction and elimination-dispersal.

A. Chemotaxis

This process simulates the movement of an E.coli cell through swimming and tumbling via flagella. In this step, the bacterium changes its position only if the modified objective function value is less than the previous one. Biologically, an E.coli bacterium can move in two different ways. It can swim for a period of time in the same direction or it may tumble, and alternate between these two modes of operation for the entire lifetime. Suppose $\theta(i,j,k,l)$ represents *i*-th bacterium at *j*-th chemotactic, *k*-th reproductive and *l*-th elimination-dispersal step. *C*(*i*) is the size of the step taken in the random direction specified by the tumble (run length unit). Then in computational chemotaxis the movement of the bacterium may be represented by:

$$\theta(i,j+1,k,l) = \theta(i,j,k,l) + C(i) * \frac{\Delta(i)}{\sqrt{\Delta(i)\,\Delta(i)}} \qquad (1)$$

Where Δ indicates a vector in the random direction whose elements lie in [-1, 1].

B. Swarming

A group of E.coli cells arrange themselves in a traveling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemo-effecter. The cells, when stimulated by high level of succinate, release an attractant aspertate, which helps them to aggregate into groups and thus move as concentric patterns of swarms of high bacterial density. The cell to cell, signaling in E.coli swarm may be represented by the following function:

$$J_{cc}(\theta, P(i,j,k,l)) = \sum_{i=1}^{S}\left[-d_{attractant} * \exp\left(-w_{attractant} * \sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right] + \sum_{i=1}^{S}\left[-h_{repellant} * \exp\left(-w_{repellant} * \sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right] \qquad (2)$$

Where $J_{cc}(\theta, P(i,j,k,l))$ is the cost function value to be added to the actual cost function. *S* is the total number of bacteria and *p* is the number of parameters to be optimized which are present in each bacterium. $\theta = [\theta_1, \theta_2, \ldots\ldots, \theta_D]^T$ is a point in the *D*-dimensional search domain. $d_{attractant}$, $w_{attractant}$, $h_{repellant}$ and $w_{repellant}$ are different coefficients that should be chosen properly [1, 9].

C. Reproduction

To keep a constant population size, the least healthy bacteria die. The remaining bacteria (those yielding higher value of fitness function) are allowed to split into two bacteria in the same place. This step helps to ignore nutrient-poor regions and focuses on the right areas.

D. Elimination and dispersal

Gradual or sudden changes in the local environment where a bacterium population lives may occur due to various reasons, e.g. a significant local rise in temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients. Events can take place in such a fashion that all the bacteria in a region are killed or a group is dispersed into a new location. To simulate this phenomenon in the BFOA, some bacteria are liquidated at random with a very small probability while the new replacements are randomly initialized over the search space.

III. THE ADAPTIVE CHEMOTACTIC BACTERIAL SWARM FORAGING OPTIMIZATION WITH DIFFERENTIAL EVOLUTION STRATEGY

In this work, to ameliorate the convergence behavior of bacteria near the global optima, the classical BFOA was hybridized with some other well known evolutionary algorithms like PSO and DE. Also, the chemotactic step size taken by a bacterium was adjusted to avoid false convergence. This combination of hybridization can prove to be very effective in tackling many difficult optimization problems on which the classical algorithms perform poorly.

A. Particle Swarm Optimization

Particle swarm optimization is a population-based stochastic optimization method developed by Eberhart and Kennedy [11] in 1995. This technique was inspired by social behavior of bird flocking and fish schooling. The particles in the swarm cooperate. They exchange information about what they've discovered in the places they have visited. In each time step, a particle has to move to a new position. It does this by adjusting its velocity. This velocity is updated according to the global best position and the best position of each particle. The velocity and position update equations of the *i*-th particle in the swarm may be given as follows:

$$V_i(t+1) = w * V_i(t) + C_1 * R_1 * \left(p_{best} - X_i(t)\right) + C_2 * R_2 * (g_{best} - X_i(t)) \qquad (3)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \qquad (4)$$

Where:

V_i (t): Velocity of agent *i* at iteration *t*.
w: Weighting Function.
C_1 and C_2: acceleration/weighting factors.
R_1 and R_2: random number between 0 and 1.
$X_i(t)$: Current position of agent *i* at iteration *t*.
p_{best}: best position found by *i*-th particle (local best).
g_{best}: best position found by swarm (global best).

B. The Proposed Adaptive Bacterial Foraging Optimization Algorithm

In computational chemotaxis, the bacterium moves by an amount of *"C*Δ"* if the objective function value is reduced for new location. *C* is the size of a chemotactic step taken by the bacterium (size of movement), and Δ define the random direction of the movement. According to the original BFOA, *C* is assumed to be constant. A chemotactic step size *C* varying as the function of the current fitness value and the global best fitness value is expected to provide better convergence behavior as compared to a fixed step size. So to get a better improvement, the parameter J_{best} was introduced in

the adaptation equation of the step size C in the following way:

$$S = (|J(\theta) - J_{best}| + 1)/\lambda \quad (5)$$

S is the new adaptive chemotactic step size and J_{best} is the objective function value for the globally best bacterium. From (5), we can see that if $|J(\theta) - J_{best}|$ is large, then the bacterium is far away from the global best, so S will also be large. On the other hand, if $|J(\theta) - J_{best}|$ is small, then the bacterium is very close to the global best and consequently S will also be small. Using this new adaptation of the chemotaxis step size, the bacterium with better function value (in a nutrient-rich zone) will try to take a smaller step and retain its current position. However, the bacterium located at a poor nutrient region of the fitness landscape will take large step sizes to attain better fitness.

C. The adaptive differential evolution algorithm

Differential Evolution (DE) is a stochastic evolutionary optimization technique which has been proposed by Storn and Price [10]. It has been applied to different engineering problems in different areas for many reasons, such as its simplicity, its faster convergence and its robustness. Seen that DE is a population-based algorithm, its process is based on the typical evolutionary operators: mutation, crossover and selection. In this study, the DE algorithm is used as follows:

1) Mutation

At each iteration, new vectors are generated by the combination of vectors randomly chosen from the current population. In our contribution, the strategy "\DE/best/1" is used according to the equation:

$$V_{i,G} = X_{best,G} + F * (X_{r_1^i,G} - X_{r_2^i,G}) \quad (6)$$

Where $V_{i,G}$ is the mutant vector and $X_{i,G}$ is the target vector. $X_{best,G}$ is the best individual vector with the best fitness function value in the population at generation G. $X_{r_1^i,G}$ and $X_{r_2^i,G}$ are randomly chosen from the current population such that $r_1^i \neq r_2^i \neq i$. F is a scaling factor.

2) Crossover (Recombination)

The new vectors are then mixed with a predetermined target vector to get a trial vector according to the equation:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j & if\ (rand(0,1) \leq CR\ or\ j = j_{rand}) \\ x_{i,G}^j & otherwise \end{cases} \quad (7)$$

Where: $u_{i,G}^j$ is a trial vector generated, j_{rand} is a randomly chosen integer in the range [1, D] and CR is a constant within the range [0, 1].

3) Adaptive Selection

After undergoing the crossover step and before doing selection, the trial vector $U_{i,G}^j$ obtained was inserted into (1). In the same time, hybridization with PSO and amelioration of chemotactic step size are made separately then together as follows:

- First of all, we integrate the updated velocity operator of the PSO algorithm defined in (3) into (1):

$$h_{i,G+1}^j = \boldsymbol{u_{i,G+1}^j} + C(i) * \boldsymbol{velocity} \quad (8)$$

- Second, the amelioration of the chemotactic step size S described by (5) was also introduced and coupled with the trial vector $u_{i,G}^j$ as follows:

$$k_{i,G+1}^j = \boldsymbol{u_{i,G+1}^j} + \boldsymbol{S(i)} * \frac{\Delta(i)}{\sqrt{\Delta(i)\,\Delta(i)}} \quad (9)$$

- Third, the trial vector $u_{i,G}^j$ obtained by the crossover step of DE, the new adaptive chemotactic step size $S(i)$ and the velocity operator borrowed from PSO to update the tumble direction are all integrated together in (1):

$$l_{i,G+1}^j = \boldsymbol{u_{i,G+1}^j} + \boldsymbol{S(i)} * \boldsymbol{velocity} \quad (10)$$

- Finally, an adaptive DE selection step is made. Consequently, the best vector obtained (between X, U, H, K and L) is accepted for the next generation if and only if it yields a reduction in the value of the objective function (F).

$$X_{i,G+1} = \begin{cases} U_{i,G+1}\ if\ F(U_{i,G+1}) \leq F(X_{i,G+1}) \\ H_{i,G+1}\ elseif\ F(H_{i,G+1}) \leq F(X_{i,G+1}) \\ K_{i,G+1}\ elseif\ F(K_{i,G+1}) \leq F(X_{i,G+1}) \\ L_{i,G+1}\ elseif\ F(L_{i,G+1}) \leq F(X_{i,G+1}) \end{cases} \quad (11)$$

D. The Adaptive Chemotactic Bacterial Swarm Foraging Optimization with Differential Evolution Strategy

To get the best convergence for our functions, we hybridize all the principal ideas already mentioned. After undergoing a chemotactic step of the BFO algorithm, each bacterium takes the mutation and crossover steps of the DE algorithm. Then, the result vector obtained was firstly coupled with the velocity operator of PSO algorithm, and then with the updated step size factor. Finally, a selection step of DE algorithm was introduced in a modified way to select the best vector result. Consequently, this combination leads to a new global hybrid optimization algorithm named ACBSFO_DES.

This new hybridization aims to gather and make use of the BFOA ability in finding a new solution by elimination and dispersal steps, the PSO ability to exchange social information, the robustness of the DE in the global search and the convergence speed of the adaptive BFOA.

In ACBSFO_DES, the bacterium moves by an amount of ameliorated *"C*Δ"* if the objective function value is reduced for new location. The updating of movement is determined by (8), (9) or (10).

Below, we briefly provide a description of the parameters of ACBSFO_DES algorithm (Table 1) and its pseudo-code.

TABLE I. DESCRIPTION OF THE PARAMETERS

Parameters	Description
P	Dimension of the search space
N	The number of bacteria in the population

Nc	The number of chemotactic steps
Nre	The number of reproduction steps
Ned	The number of elimination-dispersal events
Ns	Swimming length
Ped	Elimination-dispersal probability
C	The size of the step taken in the random direction specified by the tumble
θ (i, j, k, l)	Represents *i*-th bacterium at *j*-th chemotactic, *k*-th reproductive and *l*-th elimination-dispersal step
F, CR	DE parameters
C_1, C_2, R_1, R_2, *w*, velocity	PSO parameters

ACBSFO_DES algorithm

[Step 1]: Initialization (Table 1).
[Step 2]: Elimination-dispersal loop
[Step 3]: Reproduction loop
[Step 4]: Chemotaxis loop
For each bacterium
[a] Take a chemotactic step for every bacterium (*i*).
[b] Compute fitness function $J(i,j,k,l)$.
$J(i,j,k,l) = J(i,j,k,l) + J_{cc}(\theta(i,j,k,l), P(i,j,k,l))$
[c] $J_{last} = J(i,j,k,l)$
[d]Tumble: generate a random vector Δ(*i*) with each element(*i*), m = 1…,*p*, a random number on [-1, 1].
[e] Move:
$$\theta(i,j+1,k,l) = \theta(i,j,k,l) + C(i) * \frac{\Delta(i)}{\sqrt{\Delta(i)\,\Delta(i)}}$$
[f] Compute $J(i,j+1,k,l)$
$J(i,j+1,k,l) = J(i,j,k,l) + J_{cc}(\theta(i,j+1,k,l), P(i,j+1,k,l))$
[g] Swim:
m = 0 (counter for swim length)
While *m* < *Ns*
m = *m* + 1;
if $(J(i,j+1,k,l) < J_{last})$
$J_{last} = J(i,j+1,k,l)$
$\theta(i,j+1,k,l) = \theta(i,j,k,l) + C(i) * \frac{\Delta(i)}{\sqrt{\Delta(i)\,\Delta(i)}}$
compute the new $J(i,j+1,k,l)$ as we did in [f]
else *m=Ns*.
End,
Hybridization
- *DE Mutation* according to (6)
$V(i,j+1,k,l) = g_{best} + F * (\theta(i_1,j,k,l) - \theta(i_2,j,k,l))$
- *DE Crossover* according to (7)
$U(i,j+1,k,l)$ is obtained using $V(i,j+1,k,l)$
- *PSO velocity* is updated according to (3)
$H(i,j+1,k,l) = U(i,j,k,l) + C(i) * \boldsymbol{velocity}$;
- *Adaptive BFOA*: chemotaxis step size *S(i)* is updated according to (5)
$K(i,j+1,k,l) = U(i,j,k,l) + \boldsymbol{S(i)} * \frac{\Delta(i)}{\sqrt{\Delta(i)\,\Delta(i)}}$;
- *Total hybridization*:
$L(i,j+1,k,l) = U(i,j,k,l) + \boldsymbol{S(i)} * \boldsymbol{velocity}$;
- *Adaptive DE selection*:
The original vector *θ*(*i*, *j*+1, *k*, *l*) is replaced by the new best position between (*U*(*i*, *j*+1, *k*, *l*) or *H*(*i*, *j*+1, *k*, *l*) or *K*(*i*, *j*+1, *k*, *l*) or *L*(*i*, *j*+1, *k*, *l*)) if value of objective function for it is smaller.
[h] Go to next bacterium
[Step 5]: Reproduction: Compute the health of bacterium *i*:
$J^i_{health} = \sum_{j=1}^{Nc+1} J(i,j,k,l)$
Bacteria with the highest values die, the remaining bacteria reproduce
[Step 6]: Elimination-dispersal: Eliminate and disperse bacteria with probability *Ped*.

IV. EXPERIMENTAL STUDIES

To fully evaluate the performance of our new technique and the other algorithms, a big number of well-known standard benchmark functions of varying complexity were employed [8]. All tested functions are two dimensional except the "Hartman" function which is 3-dimensional function. Also "Power Sum" and "Shekel" functions are 4-dimendional functions.

A. Experimental setting

In order to make the comparison fair enough, the populations for all the considered algorithms (for all problems tested) were initialized using the same random seeds. The best-suited sets of parameters employed in the simulation study are chosen after a series of tuning experiments. These parameters are given below:

TABLE II. PARAMETER SETUP FOR BFOA

N	100	***C***	0.1
Nc	100	$\boldsymbol{d_{attractant}}$	0.1
Nre	16	$\boldsymbol{W_{attractant}}$	0.2
Ned	2	$\boldsymbol{W_{repellant}}$	10
Ns	12	$\boldsymbol{h_{repellant}}$	0.1
ped	0.25		

TABLE III. COMMON PARAMETER SETUP FOR PSO AND BSO

C1	***C2***	***W***
1.2	0.5	0.9

TABLE IV. COMMON PARAMETER SETUP FOR DE AND CDE

F	***CR***
0.5	0.9

TABLE V. PARAMETER SETUP FOR ABFOA AND THE PROPOSED ABFOA

	ABFOA used in [14]	***The proposed ABFOA***
C	become adaptive as suggested in [14]	become adaptive as suggested in (5)
λ	400	5000

The same parameters values were used for ABFSO_CDE.

B. Results and discussions

For all tables of results obtained, 10 independent runs of each of the eight algorithms were executed for a given function of a given dimension. The tests results obtained are used to compare the effectiveness of all competitor algorithms with respect to two measures of performance which are solution quality and convergence speed (through the number of Function Evaluations FEs marked).

1) Table 6: it makes a comparison between the different algorithms on quality of the optimum solution over well known benchmarks functions. The mean, the best-of-run values and the standard deviation (within parenthesis) of 10 independent runs for each of the competitor algorithms BFOA, Proposed ABFOA and ACBSFO_DES were presented in Table 6. The best solution in each case has been marked in bold. From Table 6, it may be observed that the performance of the two ameliorated variants of BFOA remained consistently superior to that of the classical BFOA over all benchmark problems. Also, we can see that the ACBSFO_DES proved its superiority for all existing algorithms and showed great performances with all test benchmark functions.

TABLE VI. average, the best-of-run values and the standard deviation (in parenthesis) for 10 independent runs tested on benchmark functions

	BFOA	Proposed ABFOA	ACBSFO _DES		BFOA	Proposed ABFOA	ACBSFO _DES
Function	***Mean Best (Std)***	***Mean Best (Std)***	***Mean Best (Std)***	**Function**	***Mean Best (Std)***	***Mean Best (Std)***	***Mean Best (Std)***
Sphere, f(x*) = 0	0.0488 0.0098 (0.0311)	7.9909e-008 2.3373e-008 (5.0017e-008)	**0 0 (0)**	**Easom f(x*) = - 1**	-0.9707 -0.9873 (0.0195)	**-1 -1 (0)**	**-1 -1 (0)**
Ackley f(x*) = 0	1.9305 0.5129 (1.2987)	1.5745 1.5745 (0)	**8.8818e-016 8.8818e-016 (0)**	**Hump f(x*) = 0**	0.5767 0.0368 (0.5183)	5.6027e-007 1.0229e-007 (4.9352e-007)	**4.6510e-008 4.6510e-008 (0)**
Griewank f(x*) = 0	0.0121 0.0020 (0.0131)	0.0296 0.0296 (0)	**0 0 (0)**	**Levy f(x*) = 0**	0.0197 0.0045 (0.0272)	3.8351e-008 1.0309e-008 (3.9783e-008)	**1.9617e-016 0 (3.9233e-016)**
Rosenbrock f(x*) = 0	1.1594 0.0054 (1.4644)	2.0488e-005 0 (2.4446e-005)	**1.1964e-015 0 (1.6878e-015)**	**Matyas f(x*) = 0**	0.0056 1.1508e-004 (0.0082)	0.0120 0.0111 (6.3351e-004)	**0 0 (0)**
Rastrigin f(x*) = 0	4.1984 3.1998 (0.7072)	5.8274e-005 1.8165e-005 (4.3740e-005)	**0 0 (0)**	**Perm f(x*) = 0**	1.5999 0.2558 (1.5067)	3.9659e-006 5.3258e-007 (2.8724e-006)	**7.4996e-019 1.5956e-025 (1.4933e-018)**
Goldstein & Price f(x*) = 3	22.2849 3.8004 (30.4038)	**3 3 (0)**	**3 3 (0)**	**Shubert f(x*) = -186.7309**	-60.1688 -148.8145 (60.1906)	-186.7307 -186.7307 (0)	**-186.7309 -186.7309 (0)**
Beale f(x*) = 0	0.2238 0.0289 (0.1790)	5.0420e-007 0 (4.4215e-007)	**3.9413e-014 0 (7.0272e-014)**	**Sum Squares f(x*) = 0**	0.0528 0.0119 (0.0488)	2.4323e-004 0 (4.8450e-004)	**0 0 (0)**
Bohachevsky f(x*) = 0	0.9250 0.0061 (0.7657)	0.4699 0.4699 (0)	**0 0 (0)**	**Hartmann (p=3) f(x*)= -3.86278**	-2.0813 -3.5316 (1.2537)	-0.0009 0 (0.0018)	**-3.8628 -3.8628 (0)**
Booth f(x*) = 0	0.9477 0.0154 (1.7541)	1.9717e-006 4.2053e-009 (3.8078e-006)	**9.1195e-014 4.6377e-021 (1.8181e-013)**	**Power Sum (p=4) f(x*) = 0**	42.9096 13.5984 (36.3247)	5.9055e-004 3.7908e-004 (2.3553e-004)	**4.9531e-006 4.4685e-007 (7.0543e-006)**
Dixon & Price f(x*) = 0	0.3752 0.0120 (0.4874)	7.7393e-007 7.8057e-008 (1.0538e-006)	**1.6958e-020 0 (1.5254e-020)**				

2) Table 7: In order to compare the speed of the algorithms, different numbers of Function Evaluations (FEs) were noted to express the complexity of the different algorithms. In table 7, we fixed the cut-of value (the error) equal to 10.00e-010, then we evaluate the number of Function Evaluations FEs (for 10 independent runs) required by each of the competitor algorithms to reach the prescribed given cut-of value. The entries marked "NA" (Not Applicable) in this table, imply that no run of the corresponding algorithm converged to the cut-of value within the maximum number of FEs allowed (the maximum number of FEs used is 3,000,000). Table 7 shows, for all test functions and all algorithms, the number of FEs necessary to find the optimum solution previously fixed within a given tolerance or cut-of value. From simulations and results, it's remarkable that our proposed method generally reduces the number of FEs to reach the given cut-of fitness value as compared with the other BFOA variants and as compared with PSO and DE algorithms.

TABLE VII. number of FEs required to converge to the cut-of fitness value tested on benchmark functions over the successful runs (the cut-of value used is 10.00e-010)

Function	PSO	DE	BFOA	ABFOA used in [15]	Proposed ABFOA	CDE	BSO	ACBSFO_DES
Sphere, f(x*) = 0	204 000	333 000	NA	NA	1 847 455	263	36 432	**254**
Ackley, f(x*) = 0	903 000	729 000	NA	NA	NA	516	95 535	**253**
Griewank, f(x*) = 0	192 000	423 000	NA	NA	NA	290	46 814	**250**
Rastrigin, f(x*) = 0	471 000	495 000	NA	NA	1 677 164	260	32 017	**253**
Bohachevsky, f(x*) = 0	377 000	405 000	NA	NA	NA	510	38 748	**252**
Levy, f(x*) = 0	340 000	324 000	NA	NA	1 522 054	NA	31 151	**25 799**
Matyas, f(x*) = 0	310 000	297 000	NA	NA	2 342 258	270	21 306	**260**
Sum Squares, f(x*) = 0	237 000	351 000	NA	NA	2 153 353	458	38 190	**252**
Zakharov, f (x*) = 0	270 000	315 000	NA	NA	1 481 063	506	25 732	**246**

C. Comparisons with other contributions

For a real fair experimentation study, the algorithm ABFSO_CDE was compared with three other contributions.

The first one which is described in [15] proposed two adaptive BFOA schemes ($ABFOA_1$ and $ABFOA_2$. The second contribution is described in [14]. It presented the hybridization of BFOA with the DE algorithm. This leads to the Chemotactic Differential Evolution algorithm (CDE). The

third contribution is described in [17] and it proposed two other adaptive BFOA schemes ($ABFOA_0$ and $ABFOA_1$). Results of some of those algorithms (mentioned in [14], [15] and [17]) are integrated in table 8 to be compared with the results of ACBSFO_DES algorithm. Our tests suite includes 5 well-known benchmark functions [8] of varying complexity and varying dimensions. In Table 8, *Dim* represents the dimension of the search space. In order to make the comparison fair enough, the same parameters of BFOA in [14], [15] and [17] are used in the ACBSFO_DES algorithm. The experimentation results obtained affirm the discussion already made. From table 8, it is remarkable that always and in all cases, the new hybrid ACBSFO_DES algorithm possesses the best results as compared with the other variants of BFOA of other contributions. And although functions are high dimensional and multi-model, our new technique has proved its height efficiency.

TABLE VIII. Mean and Standard Deviation (in parentheses) over five benchmarks

		Contributions of [15]			**Contributions of [14]**		**Contributions of [17]**		**Our system**
Function	***Dim***	***BSO***	***$ABFOA_1$***	***$ABFOA_2$***	***BFOA_GA***	***CDE***	***$ABFO_0$***	***$ABFO_1$***	***ACBSFO_DES***
Rastrigin f(x*) = 0	2	Not treated	Not treated	Not treated	Not treated	Not treated	**0 (0)**	1.2655e–09 (7.4950e–10)	**0 (0)**
	15	0.2632 (0.2348)	0.3044 (0.6784)	2.9823 (0.5719)	1.00e-05 (0)	1.00e-05 (0)	Not treated	Not treated	**0 (0)**
	30	13.7731 (3.9453)	2.5372 (0.3820)	8.1121 (4.3625)	6.7827e-05 (1.364e-08)	1.00e-05 (0)	Not treated	Not treated	**0 (0)**
Goldstein&Price f(x*) = 3	2	3.443712 (0.007326)	3.572012 (0.00093)	**3 (0)**	Not treated	Not treated	Not treated	Not treated	**3 (0)**
Griewank f(x*) = 0	2	Not treated	Not treated	Not treated	Not treated	Not treated	5.2282e–10 (4.4791e–10)	5.1542e–06 (1.5892e–05)	**0 (0)**
	15	0.1741 (0.097)	0.0321 (0.02264)	0.05113 (0.02351)	1.00e-05 (0)	1.00e-05 (0)	Not treated	Not treated	**0 (0)**
	30	0.2565 (0.1431)	0.1914 (0.0117)	0.2028 (0.1532)	2.1214e-04 (6.232e-05)	1.00e-05 (0)	Not treated	Not treated	**0 (0)**
Ackley f(x*) = 0	15	0.1025 (0.00347)	0.7613 (0.0542)	0.6757 (0.2741)	4.0936e-04 (5.992e-03)	1.00e-05 (0)	Not treated	Not treated	**8.8818e-016 (0)**
	30	0.5954 (0.1246)	0.5038 (0.5512)	0.7316 (0.6745)	7.2582e-03 (5.002e-04)	1.00e-05 (0)	Not treated	Not treated	**8.8818e-016 (0)**
Sphere f(x*) = 0	2	Not treated	Not treated	Not treated	Not treated	Not treated	1.3068e–112 (7.034e–112)	**0 (0)**	**0 (0)**
	15	0.001 (0)	0.001 (0)	0.001 (0)	1.00e-05 (0)	1.00e-05 (0)	Not treated	Not treated	**0 (0)**
	30	0.056 (0.0112)	0.022 (0.00625)	0.044 (0.0721)	1.00e-05 (0)	1.00e-05 (0)	Not treated	Not treated	**0 (0)**

V. CONCLUSION

In this work, to overcome the delay in optimization and to further enhance the performance of BFOA, a new improved hybrid BFOA named ABFSO_CDE is introduced which coupled the PSO and DE operators with the ameliorated step size of the chemotaxic movement. From the simulation results of the experimentation step, we can affirm that this new hybrid method proves its superiority over the other compared methods and greatly improved the optimization performance of BFOA. The future research effort should focus on the application of the BFOA and its variants not only on benchmark functions but also on complex problems such as learning of artificial neural networks. Also, we can focus on improving the convergence speed of BFOA.

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program. This work was also supported in the framework of the IT4 Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 by operational programme 'Research and Development for Innovations' funded by the Structural Funds of the European Union and state budget of the Czech Republic, EU.

REFERENCES

[1] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," IEEE Control Syst. Mag., vol. 22, no. 3, pp. 52–67, Jun. 2002.

[2] Y. Liu and K. M. Passino, "Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviors," J. Optimization Theory Applicat., vol. 115, no. 3, pp. 603–628, 2002.

[3] S. Mishra, "A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation", IEEE Trans. E.C, vol. 9, no. 1, pp. 61– 73, 2005.

[4] M. Tripathy, S. Mishra, L. L. Lai, and Q. P. Zhang, "Transmission loss reduction based on FACTS and bacteria foraging algorithm," in Proc. PPSN, pp. 222–231, 2006.

[5] S. Mishra, and C. N. Bhende, "Bacterial foraging technique-based optimized active power filter for load compensation," IEEE Trans. Power Delivery, vol. 22, no. 1, pp. 457–465, 2007.

[6] J.S. Raj and V. Vasudevan, "Smart Bacterial Foraging Optimization Algorithm for Scheduling in Grid", European Journal of Scientific Research, vol. 94, no. 2, pp.253-260, 2013.

[7] D.H. Kim and C.H. Cho, "Bacterial foraging based neural network fuzzy learning," in Proc. IICAI 2005, pp. 2030–2036, 2005.

[8] C. Li, S. Yang, T.T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan "Benchmark generator for CEC 2009 competition on dynamic optimization", University of Leicester, University of Birmingham, Nanyang Technological University, Tech. Rep, 2008.

[9] J.H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Harbor, 1992.

[10] R. Storn and K.V. Price, "Differential evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", Journal of Global Optimization, vol. 11, no. 4, pp. 341–359, 1997.

[11] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", in proceedings, ICNN, vol.4, pp.1942-1948, Nov 1995.

[12] 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.

[13] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "Synergy of PSO and Bacterial Foraging Optimization: A Comparative Study on Numerical Benchmarks", Inter. Symp. on Hybrid AIS, vol. 44, pp. 255-263, 2007.

[14] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "A synergy of differential evolution and bacterial foraging optimization for global optimization", N. N World, vol. 17, no. 6, pp. 607-626, 2007.

[15] S. Dasgupta, S. Das, A. Abraham, S. Member, and A. Biswas, "Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis " IEEE Trans. On Evol. Comput., vol. 13, no. 4, 2009.

[16] R. Panda and M.K. Naik, "A Crossover Bacterial Foraging Optimization Algorithm", Applied Comput. Inte. and SC, vol. 2012, 7 pages, 2012.

[17] H. Chen, Y. Zhu, K. H, "Adaptive Bacterial Foraging Optimization", Abstract and Applied Analysis, vol. 2011, 27 pages, 2011.