ISSN : 0974 - 7435 Volume 10 Issue 23

2014

BioTechnology

An Indian Journal

FULL PAPER

BTAIJ, 10(23), 2014 [14215-14222]

A new approach to bacterial foraging optimization based on evolution strategies

Feng Xiaohua[1,2]*, He Yuyao[1], Yu Juan[1]

[1]School of Marine Science and Technology Northwestern Polytechnical University, Xi'an, Shaanxi,710068, (CHINA)

[2]School of Electronic information engineering Xi'an Technological University, Xi'an, Shaanxi, 710021, (CHINA)

E-mail: nxfxh@163.com

ABSTRACT

Bacterial Foraging Optimization Algorithm (BFOA) is inspired by the social foraging behaviour of Escherichia coli. Although the BFOA has successfully been applied to many kinds of optimization problems, experimentation with complex problems reports that the basic BFO algorithm possesses a poor performance mainly because of its constant chemotactic step. In this paper, a new self-adaptive approach to BFO based on ES (ES-ABFO) is proposed. In the proposed algorithm, each bacterial decides the step size C on the basis of the objective function value. When it is far away from the best objective, the step size C is large. Otherwise, the step size C is small. In this way, the step size C can be regarded as an evolution progress with self-adaptive adjusting. And it can keep right balance between an exploration of the whole search space and an exploitation of the promising areas. In order to prove the validity of the ES-ABFO, two experiments have been done for a set of benchmark functions and then they have been compared with basic BFOA. The performance comparisons indicated that the ES-ABFO is capable of alleviating the problems of premature convergence in BFO. And it is suitable to solve the complex optimization problems.

KEYWORDS

Bacterial foraging; Optimization; Chemotaxis; Evolution strategies; Self-adaptive.

INTRODUCTION

Recently natural swarm inspired algorithms like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) have found their way into this domain and proved their effectiveness. Following the same trend of swarm based algorithms, The bacterial foraging optimization Algorithm(BFOA)proposed by Passino in 2002[1],which inspired by the E. Coli foraging strategy, is a new comer to the family of nature inspired optimization algorithms. Since its biological motivation and graceful structure, BFOA has drawn attention of researchers from diverse fields of knowledge.

One major step in BFOA is the simulated chemotactic movement, Chemotaxis is a foraging strategy that implements a type of local optimization, where the bacteria try to climb up the nutrient concentration to avoid noxious substance and search for ways out of neutral media. If the step size of chemotaxis is lager, the area of nutrient concentration will be missed. Otherwise the area of nutrient concentration will be not discovered. However the step size C of Chemotaxis is constant in classical BFOA, which limit to explore its local and global search properties separately. Several researchers have concentrated on it, and proposed several adaptive methods. Dasgupta et al.[2-6] showed that it is necessary to modify its value on the run for the algorithm to converge. Mishra[7] suggested using a Fuzzy Logic Controller (FLC) to adapt this parameter. Nevertheless this requires the tuning of a complete FLC, which implies the selection of several more parameters. Hanning Chen[8] introduced the adaptive search strategy, which allows each bacterium strikes a good balance between exploration and exploitation during algorithmic execution by tuning its run-length unit self-adaptively. Ben Niu[9] raised a point of linear decreasing chemotaxis step to self-adaptive optimize the step size C of chemotaxis.

This paper presents a modification for the BFOA by means of introducing the evolution strategies[10] to improve its computation speed and convergence. The paper is organized as follows. In Section II we describe the original BFOA. In Section III we proposed modification to the algorithm. In Section IV several experimental studies were done to examine its performance. We carried out a simulation study using some common benchmark functions comparing the proposed algorithm with the original and adaptive bacteria algorithms. And the results for these tests were shown. Finally, we present some conclusions in Section VI.

BACTERIAL FORAGING OPTIMIZATION ALGORITHM

Bacterial foraging optimization algorithm (BFOA) process can be subdivided into four motile behaviours namely chemotaxis, swarming, reproduction, and elimination and dispersal[1].

Chemotaxis

This process simulates the movement of an E.coli cell through swimming and tumbling via flagella. 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 models of operation for a run lifetime. Supposed $\theta^i(i,k,l)$ represents the ith bacterium at jth chemotactic, kth reproductive and lth elimination and dispersal step. *C(i)* is the size of the step taken in the random direction specified by the tumble (run length unit). Then in computation chemotaxis the movement of the bacterium may be represented by

$$\theta^i(j+1,k,l)=\theta^i(i,k,l)+c(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad \textbf{(1)}$$

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

Swarming

E.coli bacterium has a specific sensing, actuation and decision-making mechanism. As each bacterium moves, it releases attractant to signal other bacteria to swarm towards it. Meanwhile, each bacterium releases repellent to warn other bacteria to keep a safe distance from it. BFOA simulates this social behaviour by representing the combined cell-to-cell attraction and repelling effect as

$$\begin{aligned} J_{cc}(\theta,P(j,k,l)) &= \sum_{i=1}^{S} J_{cc}(\theta,\theta^i(j,k,l)) \\ &= \sum_{i=1}^{S}[-d_{attrac\tan t}\exp(-w_{attrac\tan t}\sum_{m=1}^{p}(\theta_m-\theta_m^i)^2)] \\ &= \sum_{i=1}^{S}[h_{repellant}\exp(-w_{repellant}\sum_{m=1}^{p}(\theta_m-\theta_m^i)^2)] \end{aligned} \quad \textbf{(2)}$$

where $J_{cc}(\theta,P(j,k,l))$ is the objective function value to be added to the actual objective function (to be minimized) to present a time varying objective function, S is the total number of bacteria, p is the number of variables to be optimized, which are present in each bacterium and $\theta=[\vartheta_1,\theta_2,\theta_3,...\theta_p]^T$ is a point in the p-dimensional search domain. $d_{attrac\tan t}$, $w_{attrac\tan t}$, $h_{repellant}$, $w_{repellant}$ are different coefficients that should be chosen properly.

Reproduction

The health status of each bacterium is calculated as the sum of the step fitness during its life. All bacteria are sorted in reverse order according to health status. Only the first half of population survives and a surviving bacterium split into two bacteria, which are placed in the same location. Thus the population of bacteria keeps constant.

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 of 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 BFOA some bacteria are liquidated at random with a very small probability while the new replacements are randomly initialized over the search space.

The BFOA Algorithm

[Step 1] Initialize parameters *p, S, Nc, Ns, Nre, Ned, Ped, C(i)(i=1,2…S),* θ^i

[Step 2] Elimination-dispersal loop: *l=l+1*

[Step 3] Reproduction loop: *k=k+1*

[Step 4] Chemotaxis loop: *j=j+1*

[a] For *i* =1,2…S take a chemotactic step for bacterium *i* as follows.

[b] Compute fitness function, *J (i, j, k, l)*.

Let, $J(i,j,k,l) = J(i,j,k,l) + J_{cc}(\theta^i(j,k,l), P(j,k,l))$ (i.e. add on the cell-to-cell attractant–repellent profile to simulate the swarming behaviour)where, Jcc is defined in(2).

[c] Let $J_{last} = J(i,j,k,l)$ to save this value, since we may find a better cost via a run.

[d] Tumble: generate a random vector $\Delta(i) \in \Re^p$ with each element $\Delta_m(i), m = 1,2,...,p,$ a random number on[−1, 1].

[e] Move: Let $\theta^i(j+1,k,l) = \theta^i(i,k,l) + c(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$

This results in a step of size C(i) in the direction of the tumble for bacterium i.

[f] Compute $J(i,j+1,k,l)$ and let

$$J(i,j,k,l) = J(i,j,k,l) + J_{cc}(\theta^i(j,k,l), P(j,k,l))$$

[g] Swim

1)let *m*=0 (counter for swim length)

2)while *m*<*Ns*(if have not climbed down too long)

3)let *m-m+1*

4)if $J(i,j+1,k,l) < J_{last}$

Let $J_{last} = J(i,j+1,k,l)$ and let

$$\theta^i(j+1,k,l) = \theta^i(i,k,l) + c(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

and use this $\theta^i(j+1,k,l)$ to compute the new $J(i,j+1,k,l)$ as we did in (f)

5)Else let *m*=*Ns*, this is the end of the while statement

[h] Go to next bacterium (*i*+ 1) if *i*≠*S* [i.e., go to (b) to process the next bacterium

[Step 5] If *j* < *Nc*, go to Step 4. In this case, continue chemotaxis since the life of the bacteria is not over

[Step 6] Reproduction

For the given k and l, and for each *i* =1, 2,..., *S,* let

$$J_{health}^i = \sum_{j=1}^{Nc+1} J(i,j,k,l)$$

the health of the bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic parameters *C*(i)in order of ascending cost Jhealth (higher cost means lower health).The *Sr* bacteria with the highest Jhealth values die and the remaining *Sr* bacteria with the best values split (this process is performed by placing the copies that are made at the same location as their parent)

[Step 7] If k < Nre, go to Step 3. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop

[Step 8] Elimination-dispersal. For *i = 1, 2..., S* with probability *Ped*, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if a bacterium is eliminated, simply disperse another one to a random location on the optimization domain. If l < *Ned*, then go to Step2; otherwise end.

BFO BASED ON EVOLUTION STRATEGIES

Evolution strategies

Evolutionary Strategies (ES) is a technique that makes part of the set of techniques of the Evolutionary Computation. This technique can be defined as an algorithm where individuals (potential solutions) are codified by a real value variable set, the "genome"[10].

The ES were initially developed with the purpose of parameter optimization. The first ES algorithm proposed by Rechenberg, 1965, used a mutation-selection schema, known as two-membered ES or (1 + 1) - ES algorithm. It works with an individual, which creates an offspring through mutation. The best of these both individuals is deterministically selected to integrate the next generation. In 1971,Rechenberg proposed the multimembered ES, (μ+ 1) -ES, where, μ parents, (μ> 1), participate in the generation of one descendant. In this method all parents have the same likelihood of matching. After, the (μ+λ) –ES proposed by Schwefel in 1975, specifies that μ parents produce λ descendants, where (λ>μ). The descendants compete with their parents in the selection of the best, μ individuals to the creation of the next generation. This procedure presents local optimal. To solve it, the (μ,λ) -ES was proposed, where the lifetime of an individual is just during a generation. Recent evidences points that the last algorithm is as good as the first one in practical applications[11].

The notation of an ES algorithm is the following (μ+λ) –ES or (μ,λ) –ES. Where: μ is the size of the population. The "+" operator indicates that μ andλwill compete to the next generation. The μ best individuals will be selected. The "," operator indicates that the μ best individuals will be selected just only between the descendants. λ is the number of descendants created at each generation.

The definite question expression way, in this kind of expression the individual is composed of the goal variable X and σ the standard deviation,each part may has a component:

$$(X,\sigma)=((x_1,x_2,...,x_{i,},...,x_n),(\sigma_1,\sigma_2,...,\sigma_{i,},...,\sigma_n))$$

$$\mathbf{\sigma_i^{'}=\sigma_i\bullet exp(r'\cdot N(0,1)+r\cdot N_i(0,1))} \quad \textbf{(3)}$$

$$\mathbf{x_i^{'}=x_i+\sigma_i\bullet N_i(0,1)} \quad \textbf{(4)}$$

Where:(x_i,σ_i)-father generation of individual *i*-th component, $(x_i^{'},\sigma_i^{'})$ -son generation the new individual *i*-th component, $N(0,1)$ -obedient standard normal distribution random number, $N_i(0,1)$ -produces in view of the component one time conforms to the standard normal distribution random number, $r^{'}=(\sqrt{2\sqrt{n}})^{-1}$ -global coefficient, is took to 1, $r=(\sqrt{2n})^{-1}$ -local coefficient, is took to 1, $\sigma(0)=3.0$. the above equation indicated the new individual is the random mutation from in the old individual foundation.

BFO based on evolution strategies

It is acknowledged that the most critical parameter is the step size *C* because of its strong influence in the algorithm stability and convergence. The each bacterial decide the every step size *C* mainly on the basis of the objective function value. When it is far away from the best objective, The step size *C* is large, otherwise, the step size *C* is short. In this way, the step size *C* can be regarded as an evolution progress with self-adaptive adjusting. One new self-adaptive approach to BFO based on ES is proposed in this paper.

When the *i*-th bacterial swims alone the direction in the *j*-th Chemotaxis, the step size *C* is described as:

$$C_i^j(m+1)=C_i^j(m)+\sigma_i(m)\bullet N_m(0,1) \quad \textbf{(5)}$$

Where: $C_i^j(m+1)$ -son generation of the new step size *C*, $C_i^j(m)$ -father generation of the old step size *C*, $N_m(0,1)$ -produces in view of the component one time conforms to the standard normal distribution random number, $\sigma_i(m)$ -mutation strength.

For the adaptation of the step size, we use a modification the 1/5-th rule extracted from the Evolution Strategies (ES)[10]. A rule to control the size of $\sigma_i(m)$ depending on the cost value found by the bacteria is included. If the cost value has decreased or has sustained, $\sigma_i(m+1)=0.85\sigma_i(m)$,otherwise $\sigma_i(m+1)=1.18\cdot\sigma_i(m)$. The chosen values for the adaptation are based on the works of ES in the change of mutation strength[10], since we can compare the movement of the bacteria to the mutation in ES. The results show that they work in an acceptable way.

The flowchart of the EA-BFO algorithm can be illustrated by Figure 1.

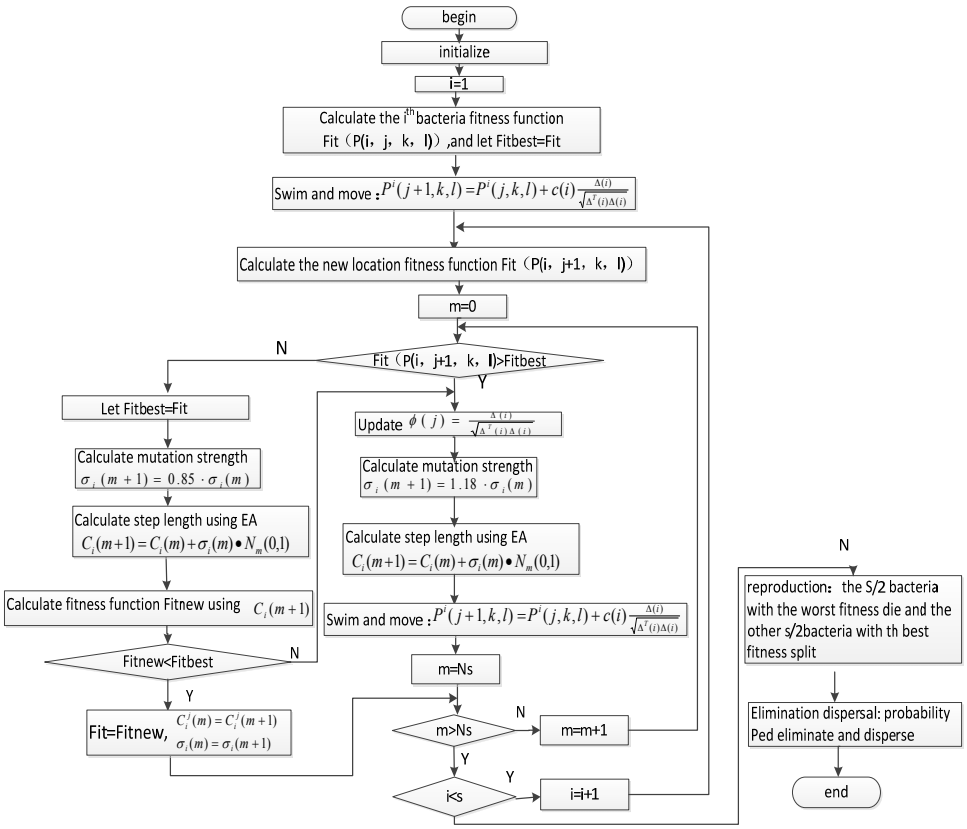

Figure 1: The flowchart of the ES-ABFO algorithm

EXPERIMENTAL STUDY

This section presents two experiments. One is an extensive comparison among the performances of ES-ABFO, the classical BFO, a standard real-coded GA, and the standard PSO algorithm. The other is an intensive comparison among the parameters of ES-ABFO influencing on performance of ES-ABFO algorithm.

TABLE 1 : Description of the benchmark function used

Function	Mathematical representation	Range of search	Theoretical optima
Sphere function (f_1)	$f_1(\vec{x})=\sum_{i=1}^{p}x_i^2$	$(-100,100)^p$	$f_1(\vec{0})=0$
Rosenbrock (f_2)	$f_2(\vec{x})=\sum_{i=1}^{p-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$(-100,100)^p$	$f_2(\vec{1})=0$
Rastrigin (f_3)	$f_3(\vec{x})=\sum_{i=1}^{p}[x_i^2-10\cos(2\pi x_i)+10]$	$(-10,10)^p$	$f_3(\vec{0})=0$
Griewank (f_4)	$f_4(\vec{x})=\frac{1}{4000}\sum_{i=1}^{p}x_i^2-\prod_{i=1}^{p}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$(-600,600)^p$	$f_4(\vec{0})=0$
Ackley (f_5)	$f_5(\vec{X})=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)+20+e$	$(-32,32)^p$	$f_5(\vec{0})=0$
Step (f_6)	$f_6(\vec{X})=\sum_{i=1}^{p}(\lfloor x_i+0.5\rfloor)^2$	$(-100,100)^p$	$f_6(\vec{p})=0,$ $-\frac{1}{2}\le p_i<\frac{1}{2}$
Schwefel's Problem 2.22 (f_7)	$f_7(\vec{X})=\sum_{i=1}^{p}\lvert x_i\rvert+\prod_{i=1}^{p}\lvert x_i\rvert$	$(-500,500)^p$	$f_7(\vec{0})=0$
Shekel's Fox-holes (f_8)	$f_8(\vec{x})=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	$(-65.536,65.536)^2$	$f_8(-32,-32)=0.998$
Six-Hump Camel-Back function (f_9)	$f_9(\vec{X})=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^6$	$(-5,5)$	$f_9(0.08983,-0.7126)=f_9(-0.08983,0.7126)=-1.0316285$
Goldstein–Price function (f_{10})	$f_{10}(\vec{x})=\{1+(x_0+x_1+1)^2(19-14x_0+3x_0^2-14x_1-6x_0x_1+3x_1^2)\}\{30+(2x_0-3x_1)^2\times(18-32x_0+12x_0^2+48x_1-36x_0x_1+27x_1^2)\}$	$(-2,2)^2$	$f_{10}(0,-1)=3$

Benchmark functions

To test the modifications to the algorithm, we carried on a benchmark study using 10 well known test functions[12]. In TABLE 1. *p* represents the number of dimensions and we used *p*=15,30,50,while functions f8 to f10 are 2-D. the first function is unimodal function with only one global minimum. It can be used to test convergence speed and precision. The others are multimodal with a considerable number of local minima in the region of interest. They are capable of finding the best result and precision in general searching. TABLE 1 summarizes the initialization and search ranges used for all the functions.

The extensive comparison

(1)Parameter settings for algorithms

Experiment was conducted to compare four algorithms including the original BFO, the real-coded Genetic Algorithm (GA), the standard Particle Swarm Optimization (PSO) and the proposed ES-ABFO on ten benchmark functions. The experiments run 50 times respectively for each algorithm on each benchmark function and the maximum generation is set at 1000. The mean values and standard deviation of the best-of-run values were recorded.

The original BFO and the ES-ABFO employ the same parameter setup, except with the difference that the chemotactic step sizes in ES-ABFO has been made adaptive according to (3) and (4). After performing a series of hand-tuning experiments, we found $\sigma_i(0)=0.001$ gives good result. The parameters settings for ES-ABFO are summarized in TABLE 2.

TABLE 2 : Common parameters setup for BFO and ES-ABFO

S	*Nc*	*Ns*	*Ned*	*Nre*	*Ped*
100	50	6	4	8	0.25
Ci(0)	*σi(0)*	*dattr*	*wattr*	*wrepel*	*hrepel*
0.1	0.001	0.1	0.2	10	0.1

The PSO algorithm we used is the standard one and the parameters were given by the default setting[13]. The GA algorithm we executed is a real-coded GA with intermediate crossover and Gaussian mutation (the parameters were to be the same of)[14]. The population size of all the algorithms was set at 100.

(2)Simulation result and discussion

The algorithms on the quality of the best solutions obtained are compared in TABLE 3. It presents the evolution process for all algorithms according to the reported result in TABLE 3.

From the results, we observe that ES-ABFO achieved significantly better performance on all benchmark functions than the original BFO algorithm. It is because that the chemotactic step size is constant in the original BFO, which makes the original BFO algorithm get into local minimum. And the ES-ABFO remains the super competitive edge to GA and PSO in the most of cases. The function 1 is adopted to assess convergence rates of optimization algorithm, but in the follow phase, the performance course of the ES-ABFO is accelerated. A self-adaption chemotatic step size is the primary action. From TABLE 3, we find that the number of dimension will effect on the convergence speed and precision of optimization algorithms. When increase of dimension is from 15-dim to 50-dim, the precision of convergence falls and the convergence rate becomes slow.

The intensive comparison

In this section, two parameters mainly effecting on performance of ES-ABFO are considered, which include S-the scale of bacterial swarm and Ped elimination-dispersal probability. Other parameters are set as TABLE 2. The number of dimension of objective functions is 30.

(1)Setting S-the scale of bacterial swarm

In the TABLE 4, the scale of bacterial swarm *S* parameter is set. From the result, we can find the performance of ES-ABFO is inferior to as the decrease of the scale of bacterial swarm. Whereas the performance of ES-ABFO is improved obviously as the increase of the scale of bacterial swarm. We suggest that *S* parameter is set as about double dim.

(2)Setting eliminate and disperse probability Ped

In order to improve the capacity of searching the general best result in BFO algorithm, eliminate and disperse probability Ped is introduced. When BFO algorithm is run into local optimization, eliminate and disperse can help BFO algorithm to dap from the local optimization by means of dispersing the bacteria of bad fitness. In this section, we argument that Setting eliminate and disperse probability Ped how to effect the performance of ES-ABFO. In TABLE 5, Function *f2-f5* are selected and Ped is set 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, D=30. From the result, we can find eliminate and disperse probability *Ped* will effect the performance of ES-ABFO. In f2 function, performance of ES-ABFO is better as *Ped* decrease. In *f2-f5* function, when *Ped* is nearby 0.25, the result is better.

TABLE 3 : The best solutions

Func	Dim	Max no. of FEs	Mean best value/standard deviation			
			GA	PSO	BFO	ES-BFO
F1	15	$5x10^4$	0.001	0.001	0.001	0.001
	30	$1x10^5$	0.039/0.0015	0.065/0.0116	0.085/0.0025	0.024/0.00626
	50	$5x10^5$	0.261/0.0325	0.382/0.2281	0.771/0.1603	0.211/0.0666
F2	15	$5x10^4$	14.716/9.9889	0.523/0.1265	32.295/26.2341	12.532/2.586
	30	$1x10^5$	31.712/3.2424	12.813/2.2658	18.215/3.2154	6.217/2.0854
	50	$5x10^5$	57.473/14.651	135.260/45.7610	87.745/25.671	29..436/11.844
F3	15	$5x10^4$	0.4889/0.0353	0.2943/0.2509	11.042/5.7661	0.2864/0.5931
	30	$1x10^5$	3.628/0.8014	17.873/4.2712	17.531/9.8904	2.576/0.3798
	50	$5x10^5$	8.143/2.5813	16.332/7.3556	32.850/9.9687	6.238/1.4876
F4	15	$5x10^4$	0.0605/0.0047	0.1573/0.08779	0.09858/0.01874	0.0364/0.02331
	30	$1x10^5$	0.3474/0.3812	0.2684/0.2031	0.3937/0.0463	0.2303/0.1034
	50	$5x10^5$	0.4101/0.1103	0.5293/0.1946	0.5528/0.0503	0.3751/0.0678
F5	15	$5x10^4$	0.0985/0.0100	0.0206/0.00947	0.8278/0.0268	0.6241/0.0302
	30	$1x10^5$	0.2731/0.3687	0.2496/0.1871	0.3217/0.03142	0.1984/0.0128
	50	$5x10^5$	0.3863/0.0951	0.4764/0.1875	0.4821/0.0558	0.3419/0.0511
F6	15	$5x10^4$	0.001	0.0109/0.00102	0.00983/0.00094	0.001
	30	$1x10^5$	0.001	0.0489/0.0052	0.2088/0.0385	0.001
	50	$5x10^5$	6.6583/0.4879	4.5712/0.6812	10.5721/2.8943	3.2784/0.8321
F7	15	$5x10^4$	0.0090/0.0047	0.00651/0.0041	0.005901/0.0026	0.00305/0.0018
	30	$1x10^5$	0.6016/0.8124	0.0681/0.2463	0.0961/0.2710	0.01004/0.0093
	50	$5x10^5$	5.4865/2.1742	1.3628/0.4476	4.9211/5.1347	0.5489/0.3117
F8	2	$1x10^5$	0.9998/0.0039	0.9998121/0.00832	1.010453/0.0096	0.9995102/0.00378
F9	2	$1x10^5$	-1.0312/2.047	-1.031264/0.00549	-0.9528/0.00056	-1.031527/0.00041
F10	2	$1x10^5$	3.2104/0.0627	3.41925/0.00724	3.58793/0.09876	3.12047/0.00081

TABLE 4 : Setting S-the scale of bacterial swarm

S-the scale	10	15	20	30	50	80
F1 Mean best value	62.271	12.734	0.378	0.0258	0.0247	0.0244
F2 Mean best value	234.521	106.783	24.672	8.137	6.2224	6.2181

TABLE 5 : Setting eliminate and disperse probability Ped

Ped		0.1	0.15	0.2	0.25	0.3	0.35
Mean best value	F2	6.203	6.187	6.215	6.217	6.221	6.782
	F3	3.983	3.025	2.604	2.576	2.613	2.726
	F4	0.3215	0.2441	0.2358	0.2303	0.2301	0.3101
	F5	0.2548	0.2681	0.1973	0.1984	0.2001	0.2985

CONCLUSIONS

This paper has presented a modification for the Bacteria Foraging Algorithm named ES-ABFO, which employs Evolutionary Strategies to get a clear adaptation rule for the step size *C(i)*. It can dynamically adjust the chemotaxis step size to keep right balance between an exploration of the whole search space and an exploitation of the promising areas. In order to verify the feasibility and efficiency of ES-ABFO, two experiments have been done for a set of benchmark functions and then they have been compared with basic BFO algorithm. The performance comparisons indicated that this proposed method is capable of alleviating the problems of premature convergence in BFO. Nevertheless, there are still more work to be carried on. Such as the algorithm suffers of premature convergence in several tests and did not acquire the global minimum in the

function evaluation limit previously set, and it suffers slower convergence speed than basic BFO. The next issue to solve is how to improve the reproduction and elimination/dispersal mechanisms, so as to increase the convergence speed and improve convergence precision.

ACKNOWLEDGEMENT

This study was financially supported by the National Natural Science Foundation of China (Grant No.6127114360871080).

REFERENCES

[1] K.M.Passino; Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Syst.Mag., **22(3)**, 52–67 **(2002)**.

[2] A.Abraham, A.Biswas, S.Dasgupta, S.Das; Analysis of reproduction operator in bacterial foraging optimization algorithm, Evolutionary Computation CEC IEEE, 1476–1483 **(2008)**

[3] A.Biswas, S.Das, S.Dasgupta, A.Abraham; Stability analysis of the reproduction operator in bacterial foraging optimization, Proceedings of the 5th international conference on Soft computing as trans disciplinary science and technology, New York,NY, USA: ACM, 564–571 **(2008)**

[4] S.Das, S.Dasgupta, A.Biswas, A.Abraham, A.Konar; On stability of the chemotactic dynamics in bacterial-foraging optimization algorithm, Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, **39(3)**, 670–679 **(2009)**.

[5] S.Dasgupta, A.Biswas, A.Abraham, S.Das; Adaptive computational chemotaxis in bacterial foraging algorithm, in Complex, Intelligent and Software Intensive Systems, CISIS, 64–71 **(2008)**.

[6] S.Dasgupta, S.Das, A.Abraham, A.Biswas; Adaptive computational chemotaxis in bacterial foraging optimization: An analysis, *IEEE Transactions on Evolutionary Computation,* **13(4)**, 919–941 **(2009)**.

[7] S.Mishra; A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation Evolutionary Computation, IEEE Transactions on, **9(1)**, 61–73 **(2005)**.

[8] H.Chen, Y.Zhu, K.Hu; Self-adaptation in bacterial foraging optimization algorithm, 3rd Int.Conf.on ISKE, 1026-1031 **(2008)**

[9] N.Ben; A novel bacterial foraging optimizer with linear decreasing chemotaxis step, Intelligent Systems and Applications (ISA), 22-23 **(2010)**.

[10] H.Beyer; The theory of evolution strategies, Springer **(2001)**.

[11] H.Beyer, H.P.Schwefel; Evolution Strategies, Natural computing, **1**, 3-52 **(2002)**.

[12] X.Yao, Y.Liu, G.Lin; Evolutionary programming made faster, IEEE Trans Evolution Compute, **3(2)**, 82–102 **(1999)**.

[13] A.Biswas, S.Dasgupta, S.Das, A.Abraham; Synergy of PSO and bacterial foraging optimization-a comparative study on numerical benchmarks, Advances in Soft Computing Springer, **(44)**, 255–263 **(2007)**

[14] F.Herrera, M.Lozano; Gradual distributed real-coded genetic algorithms, IEEE Trans.Evol.Compu, **4(1)**, 43–62 **(2000)**