JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 8, No. 1, March 2014, pp. 1-10

Analysis and Improvement of the Bacterial Foraging Optimization Algorithm

Jun Li
Lanzhou Jiaotong University, Lanzhou , China
Lijane@mail.lzjtu.cn

Jianwu Dang*, Feng Bu, and Jiansheng Wang
Key Laboratory of Opto-electronics Technology and Intelligent Control, Ministry of Education, Lanzhou , China
Dangjw@ mail.lzjtu.cn, 64927171@qq.com, Wangjsh@mail.lzjtu.cn

Abstract
The Bacterial Foraging Optimization Algorithm is a swarm intelligence optimization algorithm. This paper first analyzes the chemotaxis, as well as elimination and dispersal operation, based on the basic Bacterial Foraging Optimization Algorithm. The elimination and dispersal operation makes a bacterium which has found or nearly found an optimal position escape away from that position, which greatly affects the convergence speed of the algorithm. In order to avoid this escape, the sphere of action of the elimination and dispersal operation can be altered in accordance with the generations of evolution. Secondly, we put forward an algorithm of an adaptive adjustment of step length we called improved bacterial foraging optimization (IBFO) after making a detailed analysis of the impacts of the step length on the efficiency and accuracy of the algorithm, based on chemotaxis operation. The classic test functions show that the convergence speed and accuracy of the IBFO algorithm is much better than the original algorithm.

Category: Smart and intelligent computing

Keywords: Bacterial Foraging optimization algorithm; Chemotaxis; Step; Elimination and dispersal; Escape

I. INTRODUCTION

Swarm intelligence, as an emerging intelligent computing technology, has been the focus of attention of artificial intelligence researchers. In 2002, Passino [1] who was inspired by the social foraging behavior of *Escherichia coli,* proposed the Bacteria Foraging Optimization Algorithm (BFOA), which has become a new member in the coveted realm of swarm intelligence. Since its inception, BFOA has drawn the attention of researchers in different fields of knowledge, in terms of its biological motivation, and elegant structure. The algorithm has been instructed in optimal search by swarm intelligence, which is produced by cooperation and competition among individuals within groups. It has advantages, such as parallel distributed processing, insensitivity to initial value, and global optimization.

In recent years, the BFO algorithm has gradually aroused the wide attention of experts and scholars at home and abroad, and corresponding research about the theory and its application has been launched. Abraham et al. [2], a research team on the BFO algorithm carried out a series

Open Access http://dx.doi.org/10.5626/JCSE.2014.8.1.1 http://jcse.kiise.org

Received 8 May 2013; **Revised** 26 December 2013; **Accepted** 8 January 2014
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

of studies, and concluded that a reproduction operator in the BFO algorithm is conducive to improve the convergence speed of the algorithm. They used genetic algorithms and a differential evolution algorithm to improve the ability of searching for the global optimum of the BFO algorithm [3]. The improved algorithm has been successfully applied to the optimization of the PID parameter. Also, Das et al. [4], Indian scholars, analyzed the influence of adaptive step size on its convergence and stability, from a theoretical point of view. However, their analysis was based on certain assumptions, which only consider the chemotaxis operation of a single particle in one-dimensional continuous space. A fast bacterial swarming algorithm based on the improved quorum sensing mechanism was proposed by Ying et al. [5], and applied to image compression. In addition, Li and Yang [6] proposed a bacterial colony optimization algorithm based on the improved mechanism of BFOA, and proved that the algorithm is effective. Tripathy et al. [7] solved the problem of optimal power flow by the BFOA. The BFO and other intelligent algorithms [8-10] are also combined, in order to improve the algorithm applied to predictive control [11], image clustering [12], and multi-objective optimization [13].

II. THE CLASSICAL BFO ALGORITHM

The foraging strategy of *E. coli* bacteria is governed by four processes, which are chemotaxis, reproduction, elimination and dispersal, and swarming. Below, we briefly describe each of these processes.

A. The Chemotaxis

Chemotaxis is achieved by swimming and tumbling. When a bacterium meets a favorable environment (rich in nutrients, and noxious free), it will continue swimming in the same direction. When it meets an unfavorable environment, it will tumble, i.e., change direction. Let S be the total number of bacteria in the population, and a bacterium position represents a candidate solution of the problem and information of the i-th bacterium with a d-dimensional vector represented as $\theta^i = [\theta^i_1, \theta^i_2, \ldots \theta^i_D]$, $i = 1, 2, \ldots, S$. Suppose $\theta^i(j,k,l)$ represents the i-th bacterium at the j-th chemotactic, k-th reproductive, and l-th elimination and dispersal step. Then in computational chemotaxis, the movement of the bacterium may be represented by

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i)\Phi(j) \tag{1}$$

where $C(i)$ is the size of the step taken in the random direction specified by the tumble (run length unit), and $\Phi(j)$ is in the random direction specified by the tumble.

B. The Reproduction

The health status (fitness) of each bacterium is calculated after each completed chemotaxis process. The sum of the cost function is

$$J^i_{health} = \sum_{j=1}^{Nc} P^{i,j,k,l} \tag{2}$$

where N_C is the total number of steps in a complete chemotaxis process. Locations of healthier bacteria represent better sets of optimization parameters. To further speed up and refine the search, a greater number of bacteria are required to be placed at these locations in the optimization domain. This is done in the reproduction step. The healthier half of bacteria (with minimum value of cost function) are allowed to survive, while the other half die. Each surviving bacterium splits up into two bacteria and they are placed at the same location. In this way, the population of bacteria remains constant.

C. The Elimination and Dispersal Operation

The chemotaxis provides a basis for local search, and the reproduction process speeds up the convergence, which has been simulated by the classical BFO. While to a large extent, chemotaxis and reproduction alone are not enough for global optima searching, since bacteria may get stuck around the initial positions or local optima, it is possible for the diversity of BFO to change either gradually or suddenly to eliminate the accident of being trapped into the local optima. In BFO, the dispersion event happens after a certain number of reproduction processes. Then, some bacteria are chosen to be killed according to a preset probability P_{ed} or moved to another position within the environment.

D. The 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 between each other. BFO simulates this social behavior by representing the combined cell-to-cell attraction and repelling effect as:

$$J_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{cc}(\theta, \theta^i(j,k,l)) =$$
$$\sum_{i=1}^{S}\left[-d_{attractant}\exp\left(-\omega_{attractant}\sum_{m=1}^{D}(\theta_m - \theta^i_m)^2\right)\right] +$$
$$\sum_{i=1}^{S}\left[h_{repellant}\exp\left(-\omega_{repellant}\sum_{m=1}^{D}(\theta_m - \theta^i_m)^2\right)\right] \tag{3}$$

where $J_{cc}(\theta^i,\theta)$ is the cost function value, which is added to the actual cost function. It is minimized to present a time varying cost function. S is the total number of bacteria and P is the number of parameters to be optimized in each bacterium. $d_{attractant}$, $\omega_{attractant}$, $h_{repellant}$, and $\omega_{repellant}$ are different coefficients that are properly chosen.

III. ANALYSIS OF BFOA

A. Analysis of the Elimination and Dispersal Step in BFOA

The elimination and dispersal operator is an indispensable link in BFOA. With the probability P_{ed}, each bacterium eliminates and disperses in order to keep the number of bacteria in the population constant. If a bacterium is eliminated, another bacterium is simply dispersed to a random location on the optimization domain. As per the optimization of multi-modal function, the bacterium is easily trapped into local optima and it is difficult to escape. Thus, the convergence speed and accuracy of the algorithm is affected. The elimination-dispersal operator helps bacteria that are trapped into local optima to escape. A greater probability of migration can provide more opportunity for the bacteria to escape from the local optimum. However, at the same time, the solution of the local optimum brings a new problem, called "escape". It results in reducing the convergence speed and accuracy of the algorithm. Such a situation is not expected to exist.

For example, the classical BFO algorithm is applied into solving a simple nonlinear function: $z = (x - 15)^2 + (y - 15)^2$, where $x \in [0,30]$, $y \in [0,30]$. Assuming that the total number of bacteria is 10 in the population, the number of evolution generations is 4, the chemotaxis operation is performed 10 times in each generation, and the probability of elimination and dispersal is 20%, the bacterial individual trajectories are shown in Fig. 1.

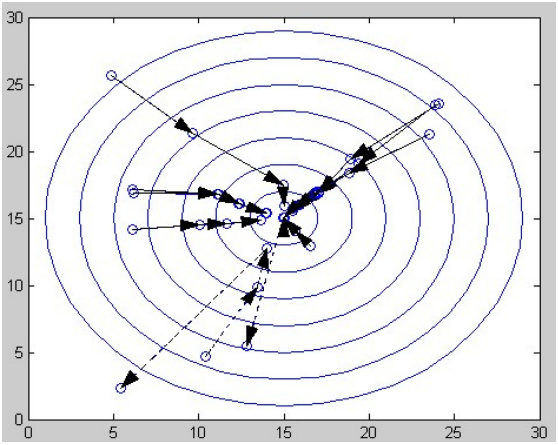

Fig. 1. Bacteria trajectories landscape.

This function has an optimal solution that z equals 0, where x = 15, y = 15. The origin shows the position of the bacterial individual in Fig. 1, and the arrow lines show the change in position of individual bacteria, after a generation optimization. As shown in the figure, most bacteria are constantly moving to the optimal position. However, some bacteria marked as dashed lines at or close to the optimal value have suddenly moved away from the optimum value, due to the operation of elimination-dispersal.

How to avoid escaping? In the optimization process of an individual bacterium, if the individual bacterium that is selected for the elimination-dispersal operator is found at, or close to, the global optimal solutions, escape will essentially happen. Thus, in order to avoid the escape, we should prevent such bacteria from being selected.

Therefore, the elimination-dispersal operator needs to be improved. The bacterial individuals are sorted in accordance with their current fitness values. The elimination-dispersal operator selects bacterial individuals, according to the probability of P_{ed}, only from the fitness value ranked behind some bacteria individuals; while the others will not be selected, because the bacterial individual standing in the front of the individual is found at, or very close to, the global optimal solution. Thus, escape can be effectively avoided. By increasing generations, bacterial individuals continue to move closer to the optimal position. The proportion of the elimination and dispersal steps of bacteria should be appropriately reduced. Let Q be the percentage of elimination and dispersal of bacteria, and its initial value is 1. The generation is the chemotactic generation counter, and its initial value is 0. An elimination and dispersal step is done after every ten generations in the algorithm, and the scope of the elimination and dispersal should be reduced in every 20 generations. This means that if generation MOD 10 = 0, then the elimination and dispersal step can be done. Let ged = generation DIV 20, Q = 1 – (2ged*L), where L is the percentage of initial value of bacteria that do not participate in the elimination and dispersal. As the generations increase, ged increases, and the percentage of elimination and dispersal decreases. Therefore, the occurrence of escape can be avoided, and the speed of convergence can be greatly improved.

B. Analysis of the Chemotaxis Step in BFOA

The chemotaxis operation is one of the most important steps in BFOA. During the chemotaxis operation, bacteria are continually swimming to find the optimal solution to the problem. At the initial location, a bacterium tumbles to take a random direction and then measures the food concentration. After that, it swims a fixed distance and then measures the concentration there. This action of tumble and swim constitutes one chemotactic step. If the concentration is superior at the next location, the bacteria will take another step toward that direction. If the con-

centration at the next location is lesser than that of the previous location, the bacteria will tumble to find another direction, and swim in this new direction. This process is carried out repeatedly, until the maximal number of steps, which is limited by the lifetime of the bacteria.

In the chemotactic steps, step size is an important parameter, when the bacteria select to swim forward in a certain direction [14-17]. How to set the step size? In the conventional BFOA, a simply fixed step size was selected based on experience. However, such treatment often makes the convergence speed of the algorithm slow, or falls into a local optimum. Thus, there should be better parameter values.

The selection of the step sizes is a critical issue, throughout the design process of the algorithm. If the step sizes are too small, the search will be trapped into local optima. On the other hand, if the steps are too long, the search will miss the global optimum. After taking this into consideration, equations for long tumble size (LT), short tumble size (ST), and swim size (SW) were defined. Almost every user intervention is needed, due to it being automatically updated during the process.

The BFOA was applied, to solve the Schaffer function.

$$f(x)=\sum_{i=1}^{n}\frac{(\sin\sqrt{x_i^2+x_{i+1}^2})-0.5}{(1+0.001(x_i^2+x_{i+1}^2))^2}+0.5\ ,\ x_i\in[-4,4].$$

This function has a global optimum, where $x_i = 0$ (i =1, 2, ..., n), and the optimal value is 0, but there are too many local optima (the function value is about -0.9903) surrounding the optimal point.

We assume that the total number of bacteria is 50 in the population, the dimension of the solution space is set to be 2, the maximum number of steps in the same direction is 4, the depth of the attractant and the height of the repellant are both 0.1, the width of the attractant is 0.2, the width of the repellant is 10, the initial probability of the elimination-dispersal is 30%, the evolution generation is 100, the step size is 0.1 and 0.001, respectively, and the mean minimum value H of function is plotted as shown in Fig. 2, after the algorithm is implemented 20 times. In addition, assume that the evolution generation is 50, and the step size is 0.5. The mean minimum value H of the function is plotted in Fig. 3, after the algorithm is implemented in one run of every 10 generations.

In Fig. 2, we can get a result that if the step size is set to be 0.1, the convergence speed of the algorithm is very high, but no longer converges after 60 generations. If the step size is set to be 0.001, the convergence speed of the algorithm is apparently reduced after 100 iterations. It has not yet met the expected precision. In order to reach the required accuracy, the iteration times must be increased, and its efficiency will be greatly reduced.

After 20 generations have evolved in Fig. 3, the curve decreases stepwise every 10 generations, and the optimal value converges once. The reason is that the algorithm is designed to reduce half of the step sizes every 10 generations. Obviously, the small step length can improve the accuracy of the algorithm.

From the above discussion, it is clear that the selection of the step sizes is a critical issue, throughout the design process of the algorithm. If the step sizes are too small, the search can be trapped into local optima. If step sizes are too long, the search will miss the global optimum. Bacteria with larger step sizes will move in the entire search space, while the bacteria with smaller step sizes can do fine only in search around local optimal solutions. Hence, the chemotactic operator (i.e., the step size) should be chosen, in order to allow the bacteria to explore the entire search space, and search effectively around the potential solutions. Therefore, the fixed step size cannot meet the requirements of accuracy and convergence

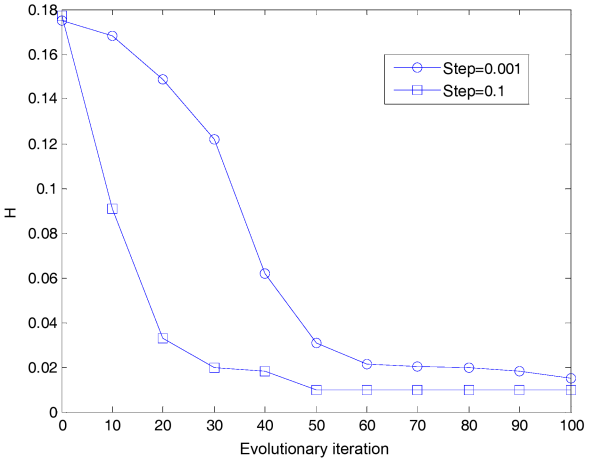

Fig. 2. The mean minimum value H of the Schaffer function (the step size is 0.1 and 0.001, respectively).

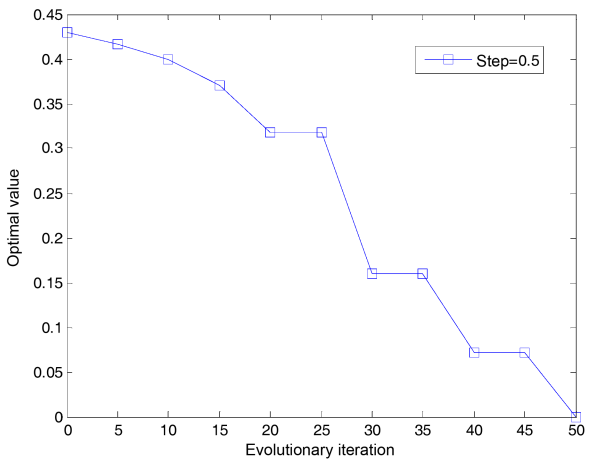

Fig. 3. The mean minimum value H of the Schaffer function (the step size is 0.5).

speed at the same time.

It is convincing that variable step sizes can not only meet the high convergence speed, but also satisfy the requirement of high precision. By means of the variable step size, we let the initial value of the step size, STEP = 0.002R, where R is the optimal interval width. It is gradually reduced, along with an increasing number of iterations. In an early iterative algorithm, the step size decreases slowly, but it shrinks faster and faster, along with an increasing number of iterations. Thus, in the early algorithm execution, bacteria rush to the optimal solution space, and accelerate the convergence speed. This improves the convergence precision, with the step size decreasing rapidly.

C. Improved BFO Algorithm

We briefly outline the novel BFO algorithm step-by-step, as follows.

[Step 1] Initialize parameters S, try_number, STEP, $d_{attractant}$, $\omega_{attractant}$, $h_{repellant}$, $\omega_{repellant}$, P_{ed}, L.

where,

S: the number of bacteria in the population,

try_number: the maximum number of steps in the same direction,

STEP: the step size,

$d_{attractant}$: the depth of the attractant,

$\omega_{attractant}$: the width of the attractant,

$h_{repellant}$: the height of the repellant,

$\omega_{repellant}$: the width of the repellant,

P_{ed}: elimination-dispersal with probability,

L: the percentage of the initial elimination-dispersal.

[Step 2] Compute the initial fitness values of each bacterium.

[Step 3] Let i = 0, generation = 0, t = 0, Q = 1.

[Step 4] Chemotactic operator and quorum sensing mechanism:

[SubStep 4.1] If generation MOD 10 == 0, {g= generation DIV 10, STEP=STEP/2g},

[SubStep 4.2] let J_{last} be represented as the current fitness value,

[SubStep 4.3] get the new fitness value J_{next}, when a bacterium runs length unit STEP in the random direction,

[SubStep 4.4] J_{cc}, the influence value of other bacteria on an bacterium, is calculated according to Step 2, let $J_{next} = J_{next} + J_{cc}$,

[SubStep 4.5] if t < try_number, then t = t +1, else t = 0, go to SubStep 4.8,

[SubStep 4.6] if J_{next} is superior to J_{last}, then $J_{last} = J_{next}$, else t = 0, go to SubStep 4.8,

[SubStep 4.7] to walk step length STEP in the same direction, J_{next} = fitness value of the new position, go to SubStep 4.4,

[SubStep 4.8] let i = i + 1, if i < S, then go to SubStep 4.1, else i = 0, generation= generation + 1, go to Step 5.

[Step 5] Reproduction steps:

[SubStep 5.1] Reproduction operator will be done every 5 generations,

[SubStep 5.2] sort descending order by the current fitness values of each bacterium,

[SubStep 5.3] locations of one half of bacteria, with inferior fitness values, are replaced by the other half, with superior fitness values.

[Step 6] Elimination and dispersal operator:

[SubStep 6.1] Elimination-dispersal operator will be done every 20 generations,

[SubStep 6.2] update range of the current elimination and dispersal Q = 1 – (2ged*L), if Q < 0, then let Q = 0,

[SubStep 6.3] sort descending order by the current fitness values of each bacterium,

[SubStep 6.4] let S_{ed} = S*Q, select bacterial from queue rear sorted S_{ed}, in accordance with the probability of the elimination and dispersal P_{ed}, to take the elimination-dispersal operator,

[Step 7] Check the terminal condition (usually reaches a predetermined evolution generations or good enough fitness values), and if the terminal condition is satisfied, output optimal solution and algorithm terminates; otherwise, go to Step 4.

IV. EXPERIMENTAL RESULTS

To illustrate the performance of the improved BFO algorithm, we present the results of the improved BFOA, using a test-suite of five well-known benchmark functions:

1) Rosenbrock function

$$f_1(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

The local minimum of the Rosenbrock function which is two-dimensional, odd behaving, and hard to minimize, is calculated, where $x \in [-2.048, 2.048]$. The function has the global optimum, which is 0, when $x_i = 1$ ($i = 1, 2, ..., n–1$).

2) Rotated hyper-ellipsoid function

$$f_2(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$$

The rotated hyper-ellipsoid function that is unimodal is calculated, and its global minimum is $x_i = 0$ ($i = 1, 2, ..., n$).

Its value is 0, where x∈[-65.536, 65.536].

3) Ackley function

$$f_3(x) = -a \times e^{-b\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}} - e^{\frac{\sum_{i=1}^{n} \cos(c \times x_i)}{n}} + a + e^1$$

This function that is multimodal is calculated. Its global minimum is at x_i = 0 (i = 1, 2, ..., n), and the value is 0, where x∈[-32.768, 32.768], a = 20, b = 0.2, c = 2*pi.

4) Rastrigin function

$$f_4(x) = 10 \times n + \sum_{i=1}^{n} (x_i^2 - 10 \times \cos(2\pi x_i))$$

This function that is a multimodal is calculated. Its global minimum is at x_i = 0 (i = 1, 2, ..., n), and the value is 0, where x∈[-5.12, 5.12].

5) Griewank function

$$f_5(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

This function that is multimodal is calculated. Its global minimum is at x_i = 0 (i = 1, 2, ..., n), and the value is 0, where x∈[-600, 600].

Using a computer PC (Pentium 4/3.0 G, Memory 4 G) with OS platform for Windows 7, C programming language, the performance of the algorithm is evaluated according to the following methods: 1) with fixed evolutionary iteration, the convergence speed and precision of the algorithm is evaluated; 2) with fixed target value of convergence precision, the number of iterations of the algorithm that reaches the required accuracy is evaluated; and 3) the performance of the algorithm is compared to other algorithms.

A. Convergence Speed and Accuracy of the Algorithm under the Fixed Evolutionary Iterations

Suppose that the rest of the parameter settings that were kept in the algorithms are as follows. The total number of bacterial population is 50. The dimension of solution space for the four testing functions is set to be 2. The maximum number of steps in the same direction is 4. The number of fixed evolutionary iterations is 200. The depth of the attractant and the height of the repellant are both 0.1. The width of the attractant is 0.2 and the width of the repellant is 10. Let the initial probability of the elimination-dispersal P_{ed} be 15%, and the step sizes STEP = 0.001R, based on the BFO algorithm. In addition, let the initial probability of the elimination-dispersal P_{ed} be 30%, the step sizes STEP=0.002R, and L=3% in the improved IBFO. Fig. 4 presents the convergence characteristics using the basic bacterial foraging algorithm and the improved algorithm (IBFO) to run 20 times in terms of the best fitness value of the median run of each algorithm for each test function.

The figures above depict optimal results, for different test functions. We can see that the classical BFO algorithm with a fixed step size almost stops converging, when it optimizes to a certain value, or even jumps, due to the escape of bacteria; whereas, the improved BFO (IBFO) algorithm overcomes these shortcomings. With an increasing number of iterations, and decreasing step sizes, the accuracy of the algorithm is greatly improved, and the convergence speed of the algorithm is significantly increased. Therefore, the improved algorithm performance is much better than that of the basic algorithm.

Table 1. The number of iterations under the specified convergence accuracy for benchmark functions f_1, f_2, f_3, f_4, and f_5

Algorithm	Function	Success rate (%)	Mean minimum	Minimum	Maximum
BFO	f_1	46	310	240	440
	f_2	38	340	236	450
	f_3	22	410	339	580
	f_4	26	325	248	466
	f_5	8	529	442	592
IBFO	f_1	100	63	56	101
	f_2	100	70	62	113
	f_3	100	91	88	190
	f_4	100	89	78	138
	f_5	100	169	93	468

BFO: bacteria foraging optimization, IBFO: improved BFO.

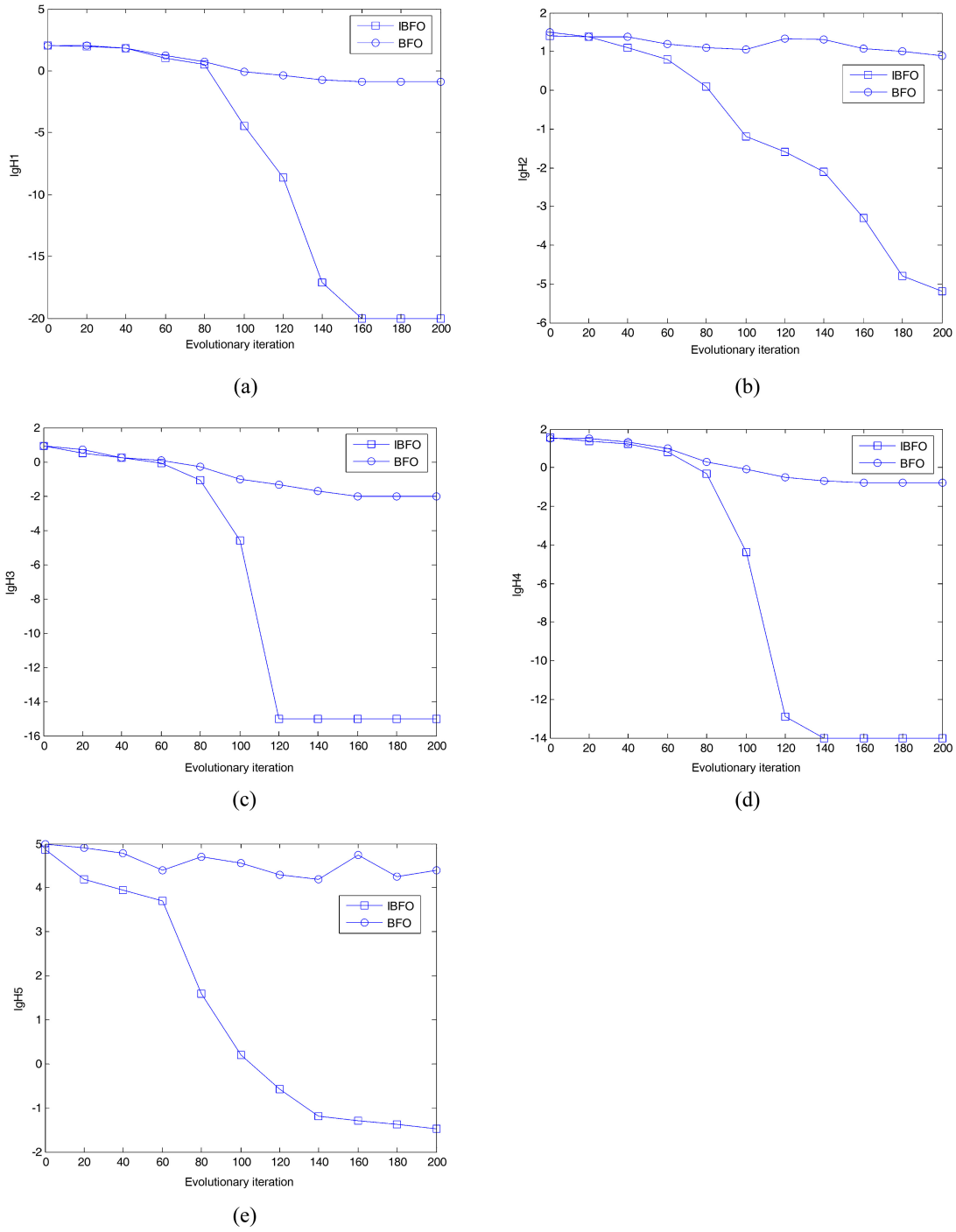

Fig. 4. Convergence results of the classical bacteria foraging optimization (BFO) and the improved BFO (IBFO) algorithm on the benchmark functions. (a) Rosenbrock, (b) rotated hyper-ellipsoid, (c) Ackley, (d) Rastrigin, and (e) Griewank functions.

B. Evolutionary Iterations under the Fixed Convergence Accuracy of the Algorithm

Experimental parameters of the evolution iterations in the fixed convergence precision are set as follows: the target convergence precision is 0.0001; the maximum number of iterations is 600; and the other parameters are as shown above.

It is shown in Table 1 that five test functions independently operate 60 times, to acquire the number of iterations under the specified convergence accuracy, where the success rate is the number of runs to the target accu-

Table 2. Average and standard deviation (in parenthesis) of the best-of-run for 50 independent runs tested on five benchmark functions

Function	Dim.	Maximum no. of FEs	Mean best value (standard deviation)		
			IBFO	BFO	PSO
f_1	15	5×10^4	0.0416 (0.0046)	0.5950 (0.5623)	0.0721 (0.0276)
	30	1×10^5	0.8841 (0.3221)	1.2160 (0.9254)	1.0630 (0.0533)
f_2	15	5×10^4	1.3552 (0.7145)	4.8372 (3.3287)	0.8341 (0.6386)
	30	1×10^5	8.4228 (1.1683)	12.3243 (10.8654)	5.5988 (1.2147)
f_3	15	5×10^4	0.3552 (0.3259)	1.0332 (0.0287)	0.2341 (0.0186)
	30	1×10^5	0.4228 (0.1683)	2.3243 (1.8833)	1.3984 (0.8217)
f_4	15	5×10^4	1.9625 (0.2853)	3.4561 (2.6632)	10.4170 (3.7260)
	30	1×10^5	2.6447 (1.6559)	17.5248 (9.8962)	34.8370 (10.1280)
f_5	15	5×10^4	0.0010 (0)	0.2812 (0.0216)	0.1153 (0.0208)
	30	1×10^5	0.1927 (0.0252)	0.3729 (0.0346)	0.2035 (0.0953)

FE: function evaluation, BFO: bacteria foraging optimization, IBFO: improved BFO, PSO: particle swarm optimization.

Table 3. Particle swarm optimization algorithm relevant parameter

Number of bacteria in the population	50
C1	2
C2	2
ω	1.05

racy divided by the total number of experiments.

As shown in Table 1, the success rate of the basic BFO algorithm is small. The maximal success rate is only 46% in the four test functions, whereas the success rate in the improved algorithm (IBFO) is 100%. The improved algorithm (IBFO) in the case of successful convergence is better than the basic algorithm, in the required minimum number of iterations, and the maximum number of iterations. The above results indicate that the improved BFO (IBFO) algorithm is superior to the basic BFO algorithm.

C. Comparison with Other Algorithms

We have compared IBFO with BFO and particle swarm optimization (PSO). The comparison ends up with the condition that the number of iterations achieves 10^5, or the optimal value reaches the target value, i.e., 0.001. The solutions of 100 times operations of IBFO, BFO, and PSO on f_1, f_2, f_3, f_4, and f_5 are shown in Table 2. The parameters are the same as mentioned before. The parameters of PSO [18] are shown in Table 3.

V. CONCLUSIONS

The algorithm is improved in the elimination-dispersal and chemotaxis operations, based on the basic BFO. By limiting the range of the elimination-dispersal of bacteria, the escape phenomenon can be avoided and the convergence speed of the algorithm is effectively improved. Furthermore, the influence of the step size in the chemotaxis operations on the algorithm has been analyzed; the convergence speed and the precision of the algorithm with variable step size have been improved. Experiments show that the IBFO greatly improves its convergence speed and precision, and it is suitable for both unimodal and multimodal functions.

ACKNOWLEDGMENTS

This work was supported by the Plan of the Gansu Provincial Department of Education, 2012, under Grant 1204-13.

REFERENCES

1. K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems*, vol. 22, no. 3, pp. 52-67, 2002.
2. A. Abraham, A. Biswas, S. Dasgupta, and S. Das, "Analysis of reproduction operator in bacterial foraging optimization algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Hong Kong, 2008, pp. 1476-1483.
3. 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.
4. S. Das, A. Biswas, S. Dasgupta, and A. Abraham, "Bacte-

rial foraging optimization algorithm: theoretical foundations, analysis, and applications," in *Foundations of Computational Intelligence Volume 3*, Heidelberg, Germany: Springer-Verlag, pp. 23-55, 2007.
5. C. Ying, M. Hua, J. Zhen, and W. Qinghua, "Fast bacterial swarming algorithm based on particle swarm optimization," *Journal of Data Acquisition and Processing*, no. 4, pp. 442-448, 2010.
6. M. Li and C. W. Yang, "Bacterial colony optimization algorithm," *Control Theory & Applications*, vol. 28, no. 2, pp. 223-228, 2011.
7. M. Tripathy, S. Mishra, L. L. Lai, and Q. P. Zhang, "Transmission loss reduction based on FACTS and bacteria foraging algorithm," in *Parallel Problem Solving from Nature-PPSN IX*, Heidelberg, Germany: Springer-Verlag, pp. 222-231, 2006.
8. P. Yang, Y. M. Sun, X. L. Xiao, and L. X. Che, "Particle swarm optimization based on chemotaxis operation of bacterial foraging algorithm," *Application Research of Computers*, no. 10, pp. 3640-3642, 2011.
9. W. L. Wang, "Research of hybrid optimization algorithms based on swarm intelligence," dissertation, Harbin Institute of Technology, Harbin, China, 2010.
10. X. L. Liu and K. L. Zhao, "Bacteria foraging optimization algorithm based on immune algorithm," *Journal of Computer Applications*, vol. 32, no. 3, pp. 634-637, 2012.
11. X. S. Wang, Y. H. Cheng, and M. L. Hao, "Estimation of distribution algorithm based on bacterial foraging and its application in predictive control," *Acta Electronica Sinica*, vol. 38, no. 2, pp. 333-339, 2010.
12. F. Feng, B. K. Wang, and S. Y. Yang, "Research on image cluster based on bacterial foraging optimization algorithm," *Journal of Tianjin Normal University*, no. 2, pp. 56-58, 2012.
13. S. J. Yang, S. W. Wang, J. Tao, and X. Liu, "Multi-objective optimization method based on hybrid swarm intelligence algorithm," *Computer Simulation*, vol. 29, no. 6, pp. 218-222, 2012.
14. D. Yang, X. Li, and L. Jiang, "Improved algorithm of bacterium foraging and its application," *Computer Engineering and Applications*, vol. 48, no. 13, pp. 31-34, 2012.
15. S. Mishra, "A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 61-73, 2005.
16. R. Majhi, G. Panda, B. Majhi, and G. Sahoo, "Efficient prediction of stock market indices using adaptive bacterial foraging optimization (ABFO) and BFO based techniques," *Expert Systems with Applications*, vol. 36, no. 6, pp. 10097-10104, 2009.
17. T. Datta, I. S. Misra, B. B. Mangaraj, and S. Imtiaj, "Improved adaptive bacteria foraging algorithm in optimization of antenna array for faster convergence," *Progress in Electromagnetics Research C*, vol. 1, pp. 143-157, 2008.
18. Y. Shen, B. Guo, and T. X. Gu, "Particle swarm optimization algorithm and comparison with genetic algorithm," *Journal of University of Electronic Science and Technology of China*, vol. 34, no. 5, pp. 696-699, 2005.
19. Yang Shang-jun, Wang She-wei, Tao Jun, and Liu Xue. "Multi-objective optimization method based on hybird swarm intelligence algorithm," *Computer Simulation*, vol. 29, no. 6, pp. 218-222, 2012.
20. Shen Yan, Guo Bing, and Gu Tian-xiang. "Particle swarm optimization algorithm andcomparison with genetic algorithm [J]," *Journal of UEST of China*, vol. 34, no.5, pp. 696-699, 2005.

Li Jun

Li Jun is an associate professor and doctoral candidate. She received her B.Sc. and M.Sc. degrees in Computer Science from Lanzhou Jiaotong University, Lanzhou, China, in 1996 and 2004, respectively. After graduation, she worked in Lanzhou Jiaotong University. She has been studying for a doctor's degree in the department of electronics and information of Lanzhou Jiaotong University, since 2009. Her main research direction is computational intelligence and intelligent information processing. In recent years, she has participated in a number of scientific research projects, winning the Science and Technology Progress Award of Gansu Province five times, and publishing many articles.

Jianwu Dang

Jianwu Dang received his M.Sc. degree in 1992, and Ph.D. degree in 1996, in Southwest Jiaotong University. He is currently a professor and Ph.D. supervisor in the School of Electronic & Information Engineering, Lanzhou Jiaotong University. His main research interests include intelligent information processing and neural networks. He has published more than 70 research articles in peer-reviewed journals and international conferences.

Feng Bu

Feng Bu received his B.Sc. degree in Computer Engineering from Lanzhou Jiaotong University, Lanzhou, China, in 2008. He is studying for his master's degree at Lanzhou Jiaotong University. His research interests include artificial intelligence, optimization, and computer vision.

Jiansheng Wang

Jiansheng Wang received his B.Sc. degree in Computer Science from Jilin University, Changchun, China in 1986. He joined the faculty of the Department of Computer Science at LZJTU, Lanzhou, China in 1996, where currently he is an associate professor. His research interests include Web services, information security, and distributed processing.