Sensors and Materials, Vol. 32, No. 4 (2020) 1311–1338
MYU Tokyo

S & M 2571

Robust Bacterial Foraging Algorithms Based on Few-excellent-individuals Guidance Strategy

Hongwei Gao,[1] Jiahui Yu,[1,2] Dai Peng,[1] Zhaojie Ju,[2*] and Yanju Liu[1**]

[1]School of Automation and Electrical Engineering, Shenyang Ligong University,
Shenyang 110159, China
[2]School of Computing, University of Portsmouth,
Portsmouth PO1 3HE, UK

(Received August 29, 2019; accepted December 13, 2019)

Keywords: bacterial foraging optimization, 80/20 rule, constriction factor PSO, gradient migration probability, robustness

In recent years, the novel bacterial foraging optimization has been widely applied. However, in past studies, the process of bacterial foraging lacked guidance and the structure of the algorithm was inadequate, which resulted in a low convergence speed and a large number of parameters in the algorithm, thus reducing its search accuracy and speed. Additionally, researchers only improved the algorithm for complex situations, for which a comprehensive evaluation of its robustness could not be made. Here, to resolve these issues, two improved algorithms are proposed and compared comprehensively. Our algorithms are suitable for modeling the foraging process of organisms in nature: a small number of individuals with rich resources can attract other individuals to forage locally. First, we propose a decreasing composite function and gradient migration behavior and introduce the 80/20 rule. A few excellent individuals guide the population to migrate to the optimal solution and increase the convergence speed. Second, we introduce the renewal speed of particles and propose another composite function, and the biological characteristics of *Escherichia coli* are also introduced to achieve the screening of excellent individuals. Finally, we show the results of numerous experiments and comprehensively evaluate the applicability of the proposed organisms.

1. Introduction

Owing to the wide application of intelligent optimization algorithms in various research fields, researchers have studied bionic-based swarm intelligence optimization algorithms. The classical algorithms include the artificial fish swarm (AFS) algorithm, genetic algorithm (GA), ant colony optimization (ACO), bacterial foraging optimization (BFO), artificial bee colony (ABC) algorithm, and standard particle swarm optimization (PSO).[1–3] Until recently, most studies indicated that the BFO algorithm is a relatively good and representative swarm intelligence optimization algorithm. In 2002, Passino proposed a BFO algorithm based on the foraging behavior and regularity of *Escherichia coli* in the intestines of organisms.[4] The

*Corresponding author: e-mail: zhaojie.ju@port.ac.uk
**Corresponding author: e-mail: 6133292@qq.com
https://doi.org/10.18494/SAM.2020.2571

algorithm is based on the biopsy model of the large intestine, which was proposed by Passino and Berg and co-workers.[5–8] The BFO algorithm is mainly composed of four behavioral actions: chemotaxis, swarming, reproduction, and elimination and dispersal. The amount of research on this algorithm is still limited and it still needs to be more widely studied and developed. The current research is mainly focused on theoretical analysis, the multiobjective optimization problem, and application in optimization problems in engineering. There are four main areas of research on the BFO algorithm: the improvement of the operation of the algorithm, the improvement of other aspects of the algorithm itself, its combined use with other algorithms, and the research and application of algorithm theory. In this paper, we focus on the improvement of the algorithm itself and its introduction into other algorithms.

In 2002, Passino published a BFO algorithm and applied it to various fields, such as adaptive control of liquid level control systems, the types of tasks of decision systems, and the selection and lengths of task processes.[4] In 2003, Liu performed a preliminary study on the behavior of the algorithm and gave a proof of its convergence.[9] In 2006, the process by which a monome searches for food and the evolution of a small-scale bacterial population under dynamic conditions were successfully imitated by the Tang research team, and a dynamic BFO algorithm was proposed.[10] In 2011, Niu proposed a BFO algorithm based on time-varying chemotaxis steps.[11] In 2015, Meng *et al.* introduced a variation factor in the chemotaxis of BFO where in each iteration, individual bacteria are selected and mutated with a certain probability. By testing, it was found that the wavelet variability method is the most effective, and this improvement was used to solve the array synthesis problem.[12] In 2017, Tang used the improved BFO algorithm to solve multilevel threshold problems in image processing. They introduced the PSO operator in the process of chemotaxis to enhance the searchability, and the elite preservation strategy was adopted in the breeding process. The algorithm obtained an optimal solution by maximizing the Tsallis threshold function.[13] Then, Verma used a BFO algorithm and a minimum kernel similarity method to construct a fuzzy system to detect image edges. Here, the BFO algorithm was used to optimize the fuzzy membership function and the parameters of the enhanced fuzzy operator.[14] However, although these studies have improved the optimization ability of the BFO algorithm in practical applications, most of the recent studies had problems with a large number of parameters that were not easy to control.[15,16] In particular, fixed values were used for some important parameters, limiting the performance of the algorithm. Secondly, the parameter optimization problem has not been discussed because the design of most algorithms adopted a multilayer loop nesting structure.[17] Besides, few groups have compared or comprehensively evaluated the improved BFO algorithms on a variety of complex functions. To solve these problems, we have carried out a more detailed study.

Recently, researchers have attempted to combine the BFO algorithm with the ideas or operators of other excellent algorithms. To solve the problem of high computational complexity caused by multitarget and multisequence comparisons in different protein sequence analyses, Manikandan combined the BFO algorithm and the GA algorithm to propose a BFOGA algorithm to achieve efficient comparison.[18] Zhou used an enhanced BFO-gravity search algorithm to study model-solving problems. The algorithm used population reconstruction, an adaptive selection chemotaxis operator, and a local search strategy, and also used an external

archive to save the elite solution set, which solves the optimization problem more effectively.[19] These BFO algorithms are used in a variety of dynamic optimization problems, such as robot path planning in dynamic environments, robotic navigation along a wall, vehicle path planning, dynamic scheduling problems in manufacturing units, dynamic portfolio optimization, and information dynamic routing optimization.[20,21] However, in these applications, the above algorithms all have the problem of premature convergence, late convergence or even no convergence. This is because there are no specific principles in the selection of individuals that act as guiding individuals, and the problem of parameter simplification has seldom been considered. In this paper, we report a detailed comparison between different group intelligent algorithms to improve the convergence performance of the BFO algorithm, thereby improving the search accuracy and search speed of the algorithm.

In this work, we propose two improved algorithms, namely, convergent 80/20 rule bacteria foraging optimization (C28BFO) and convergent particle swarm optimization bacteria foraging optimization (CPSOBFO). A decreasing composite function, the idea of the gradient, the constriction factor PSO, and the 80/20 rule are combined in our improved algorithms so that different grades of bacteria carry out their particular elimination and dispersal activities.

The remainder of this paper is organized as follows. Section 2 is a review of the related work on BFO. In Sect. 3, we explain the problem statement and give an outline of our proposed approaches. Sections 4 and 5 introduce the process of the proposed method. In Sect. 6, we report various experimental results as well as a comparison with state-of-the-art methods. Section 7 summarizes the work of this paper and gives a further direction for extending the study.

2. Related Work

2.1 BFO algorithm

The BFO algorithm simulates four population behaviors of *E. coli* in a real environment: chemotaxis, swarming, reproduction, and elimination and dispersal. Foraging involves these four behavioral processes, that is, the search for the optimal solution. When using the BFO algorithm to solve optimization problems, we usually need to go through the process shown in Fig. 1. The specific parameters of the algorithm are shown in Table 1.

The initial values of parameters j, k, and l are all zero and refer to the parameters in Table 1. Firstly, $P(j, k, l) = \{\theta^i(j, k, l) \mid i = 1, 2, ..., S\}$ indicates the set of locations after j chemotaxis operations, k replication operations, and i migration operations for each individual in the

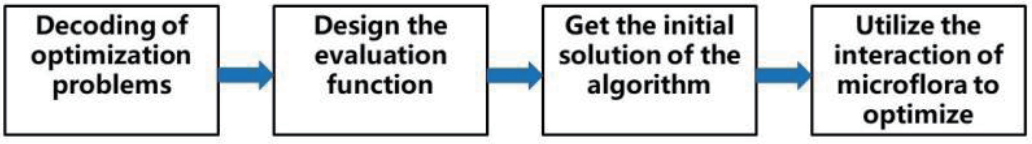


Fig. 1. General process of the BFO algorithm for solving optimization problems.

Table 1
Description of parameters and labels of algorithm.

Symbol	Meaning	Symbol	Meaning
j	Number of chemotaxis operations	k	Number of replications
l	Number of migration operations	P	Dimension of search space
S	Total number of individuals in population	$C(i)$	Step size in the chemotaxis
N_c	Number of trending operations	N_{re}	Number of replications of *E. coli*
N_{ed}	Number of migrations	P_{ed}	Migration probability of *E. coli*
N_s	Maximum number of steps in one direction per trending operation		

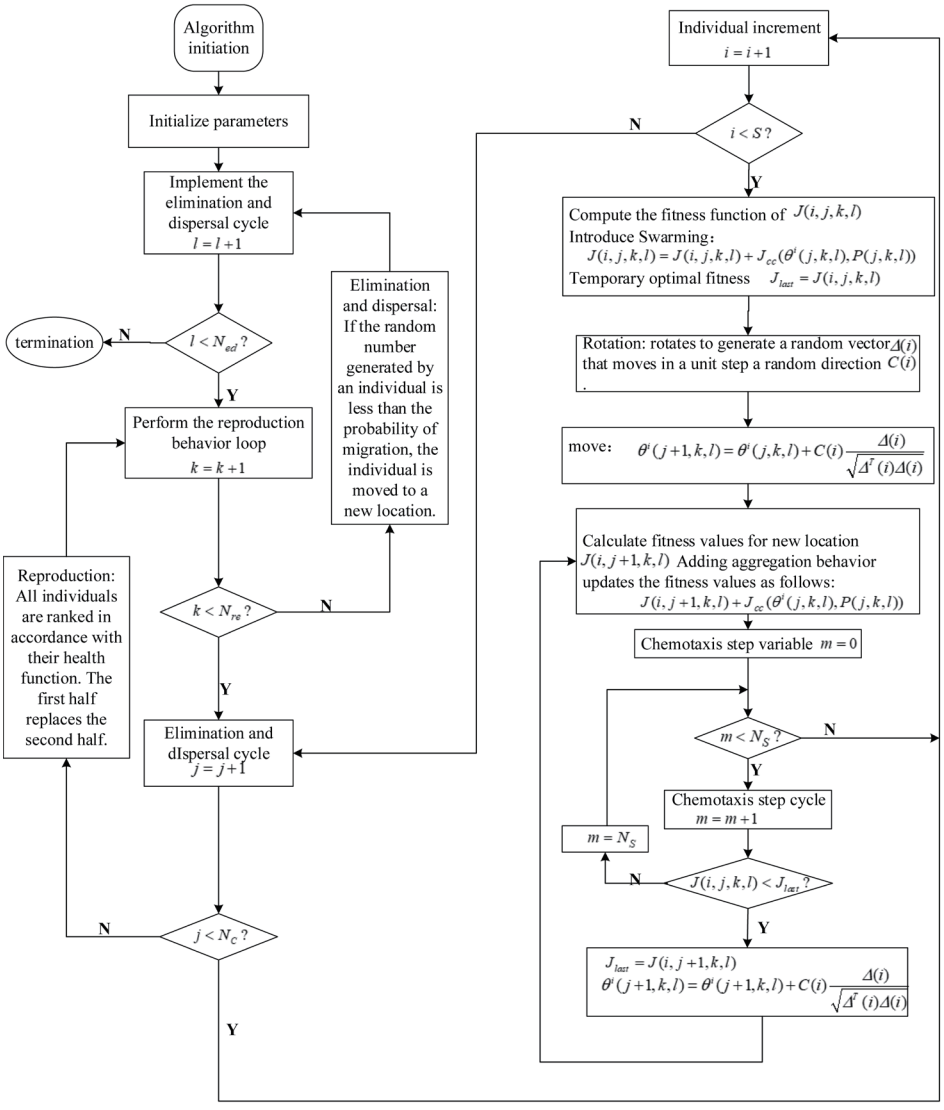


Fig. 2. Detailed process of BFO.

population. Secondly, $J(i, j, k, l)$ indicates the value of the fitness function possessed by individual i after j chemotaxis operations, k replication operations, and i migration operations. In this paper, we have given all the actions and steps of the BFO algorithm, and the details are shown in Fig. 2.

2.2 Chemotaxis

Such behavior includes swimming and flipping, which respectively correspond to changes in position and direction. Individual i performs each location update in accordance with

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

where $\theta^i(j, k, l)$ and $\theta^i(j+1, k, l)$ are the positions of individual i after j and $j+1$ chemotaxis operations in the case of k replications and l migrations, respectively. Here, $C(i)$ is the step size in chemotaxis, indicating the distance of a single migration. $\Phi(j)$ is a unit vector in the direction selected after the direction is adjusted and is given by

$$\Phi(j) = \frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}}. \tag{2}$$

It can be seen from Eq. (2) that $\Phi(j)$ is also a unit vector with a random direction, simulating the randomness of the position update. Therefore, Eq. (1) can also be written as

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}}. \tag{3}$$

2.3 Swarming

Swarming is the embodiment of the synergistic relationship between bacteria, that is, bacteria can effectively influence other bacteria in the group to move quickly to an area with a better food source that they have found and maintain the spacing between. The combined release of gravitational and repulsive signals can not only make the individual more profitable in foraging but also effectively avoid malignant competition caused by bacterial clumps. The mathematical expression of the aggregation behavior among individuals is

$$\begin{aligned} J_{cc}(\theta, P(j,k,l)) &= \sum_{i=1}^{S} J^i{}_{cc}(\theta, \theta^i(j,k,l)) \\ &= \sum_{i=1}^{S}\left[-d_{attract}\exp\left(-w_{attract}\sum_{m=1}^{p}(\theta_m - \theta^i{}_m)^2\right)\right] \\ &\quad + \sum_{i=1}^{S}\left[-h_{repellant}\exp\left(-w_{repellant}\sum_{m=1}^{p}(\theta_m - \theta^i{}_m)^2\right)\right]. \end{aligned} \tag{4}$$

In the formula, $J_{cc}(\theta, P(j, k, l))$ is the influence value of the signal transmitted between all individuals, and is expressed as the sum of the aggregation action values of each individual and other individuals. $\theta = [\theta_1, \theta_2, ..., \theta_p]^{\mathrm{T}}$ is a point in this multidimensional search space

and θ_m is the m component of the spatial location. $P(j, k, l)$ is the position of the individual after j chemotaxes, k replications, and l migrations, and satisfies the relationship $P(j, k, l) = \{\theta^i(j, k, l)|i = 1, 2, ..., S\}$. This is inserted into $J_{cc}(\theta, P(j, k, l))$ to get the first of Eq. (4), which is decomposed into the attraction and repulsive signals of aggregation as follows:

$$J_{attract} = \sum_{i=1}^{S} J^i{}_{attract} = \sum_{i=1}^{S} \left[-d_{attract} \exp\left(-w_{attract} \sum_{m=1}^{p} (\theta_m - \theta^i{}_m)^2 \right) \right]. \tag{5}$$

$$J_{repellant} = \sum_{i=1}^{S} J^i{}_{repellant} = \sum_{i=1}^{S} \left[-h_{repellant} \exp\left(-w_{repellant} \sum_{m=1}^{p} (\theta_m - \theta^i{}_m)^2 \right) \right]. \tag{6}$$

Aggregation behavior is introduced into the updating of fitness values of tendentious individuals, as

$$J(i, j + 1, k, l) = J(i, j, k, l) + J_{cc}(\theta^i(j + 1, k, l), P(j + 1, k, l)). \tag{7}$$

Here, $J(i, j + 1, k, l)$ and $J(i, j, k, l)$ are the fitness values of individuals after the kth replication, the lth migration, and the jth and $(j + 1)$th chemotaxes, respectively.

2.4 Replication

Some individuals who have failed in their foraging search strategy are eliminated, but the remaining individuals with strong foraging ability will replicate themselves by division to ensure that the population size remains unchanged. This behavior is called replication behavior.

In the algorithm, the fitness values of individuals after previous tendentious behaviors are counted and processed by summation. Assuming that there are S individuals in the flora, their fitness values are sorted from small to large, so that $Sr = S/2$ takes the first Sr fitness value. This corresponds to Sr bacteria with strong foraging ability, which are then replicated. Such fitness and its value are collectively called the fitness function, which measures the energy gain after foraging. For individual i, the fitness function is

$$J^i{}_{health} = \sum_{j=1}^{N_C+1} J(i, j, k, l). \tag{8}$$

Here, $J(i, j, k, l)$ is the fitness function value of i after performing j chemotaxes, k replications, and l migrations.

2.5 Reproduction

In the algorithm, *E. coli* migrates with a fixed probability P_{ed}. After a certain algebraic replication, the individual will be eliminated. Then, a new individual with different locations

and foraging abilities will be generated randomly in the search space. The number of individuals remains unchanged, which is conducive to an individual searching for better food sources.

3. Problem Description and Proposed Approaches

In the conventional BFO algorithm, the chemotaxis step $C(i)$ in the random direction generated by the bacterial chemotactic rotation is a fixed constant and is too small or too large to affect the algorithm search effect. In the process of copying, the original algorithm performs the sorting of individual bacteria, selects the individuals in the first half and replicates on them. The original purpose is to give only excellent bacteria the right to survive and multiply. However, when making a selection, the algorithm is very likely to retain the poor bacterial individuals that are within half of the sequence, which affects the algorithm. In this work, we attempt to solve the above-discussed problems. The main contributions are summarized as follows.

First, we propose the novel C28BFO algorithm, in which a chemotaxis step is designed on the basis of decreasing composite function, and the 80/20 rule is introduced to improve the screening method of excellent individuals. The gradient migration probability is also proposed to improve the migration behavior. The simulation results of the improved algorithm are given to prove its effectiveness.

Second, to make the algorithm convergence faster with higher convergence precision, in this paper, we not only propose a new chemotaxis step size expression but also introduce the memory and perceptual energy of the particle swarm algorithm and apply them to the BFO algorithm. At the same time, a phage's foraging process of *E. coli* is introduced to eliminate the bacteria with poor foraging ability to optimize the optimization ability of the algorithm. We name this method the CPSOBFO algorithm and verify its improvement.

Third, we test the BFO algorithm, C28BFO algorithm, CPSOBFO algorithm, and GA algorithm simultaneously on a variety of complex optimization functions, including the Sphere function, Rosenbrock function, Rastrigin function, and Griewank function. Then, using the test results, the convergence performance of each algorithm under different applications is comprehensively evaluated. In the following section, we introduce the readers to a basic mathematical understanding of these two algorithms for completeness.

4. C28BFO Algorithm

4.1 Improvement of step-size function

Compared with the previous original algorithm, the time-varying chemotactic step size is designed, in this study, in accordance with the number of iterations, current fitness values, and migration operations, so as to provide a better convergence for the algorithm. The step-size function consists of four parts: iteration number function, fitness function, migration behavior function, and parameter adjustment, which are shown in Fig. 3.

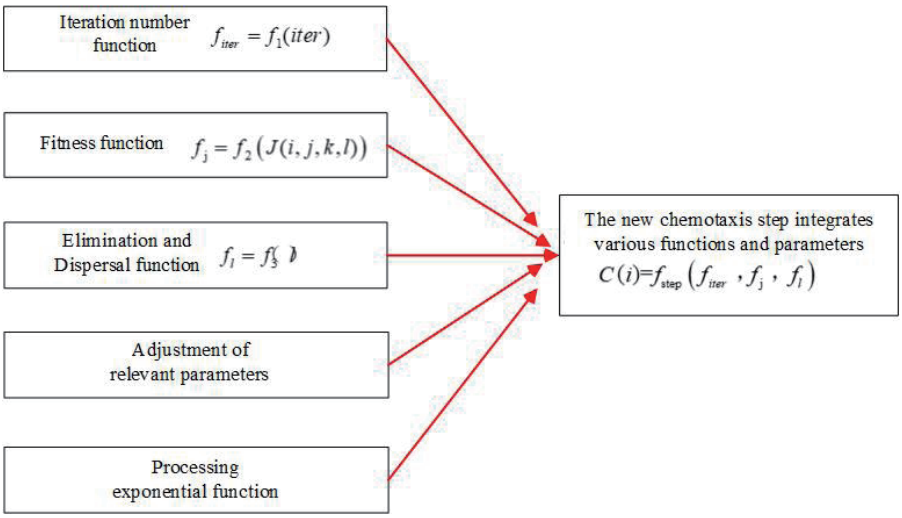


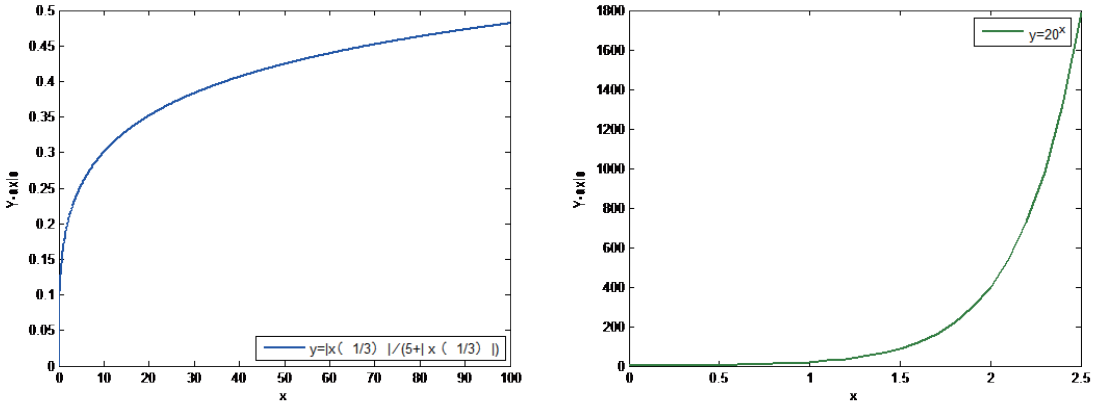
Fig. 3. (Color online) Composition diagram of new chemotactic step function.

As shown in Fig. 3, the function modules are combined to improve the step function to improve the optimization characteristics of the algorithm. Specifically, the design expression of the new chemotaxis step function for individual bacteria i is shown in

$$C(i) = \frac{3\left(\dfrac{(iter_{max} - iter)\left|\sqrt[3]{J(i,\ j,\ k,\ l)}\right|}{iter_{max}\left(\mu + \left|\sqrt[3]{J(i,\ j,\ k,\ l)}\right|\right)\beta^{l}} + \zeta\right)}{n} + C_{min}\ . \tag{9}$$

In Eq. (9), $C(i)$ is the chemotaxis step size of *E. coli*, $iter_{max}$ is the maximum number of iterations, $iter$ is the current iteration number, $J(i, j, k, l)$ is the fitness value corresponding to the current position of bacterial individual i, μ is a large positive adjustment constant, β is a positive value greater than 1, l is the number of migrations of the current bacteria, ζ is the guarantee factor of the step and is between 0 and 1, and n is a positive value of the control convergence speed and accuracy .

In the step function of Eq. (9), $(iter_{max} - iter)/iter_{max}$ is a function that linearly decreases as the number of iterations increases. The function $|J(i, j, k, l)^{(1/3)}| / (\mu + |J(i, j, k, l)^{(1/3)}|)$ is a decrease in the bacterial fitness value and the reduced function. For example, for $x = J(i, j, k, l)$, $\mu = 5$, $y = |x^{(1/3)}| / (5 + |x^{(1/3)}|)$, and drawing with 0.1 as the smallest unit, the argument x takes values from 0 to 100, and the function image is as shown in Fig. 4(a). From this image, it can be seen that as x decreases, the function decreases from steep to urgent. As the iterative process of the algorithm continues, the bacteria are continuously close to the optimal point because of positioning by random distribution in the search solution space. At these positions, the bacteria pay a relatively small price and obtain relatively rich benefits, and the fitness value decreases. It can then be predicted that as the bacterial fitness value decreases, the function consisting of this fitness value will show a decreasing trend. This is in line with the step size design intent of



(a) (b)

Fig. 4. (Color online) Plots of different functions. (a) $y = |x^{(1/3)}|/(5 + |x^{(1/3)}|)$. (b) $y = 20^x$.

performing a larger step search at the beginning of the algorithm and a smaller step search later in the algorithm.

The β^l in the step function is an exponential function with the number of migrations of bacteria as an independent variable, and its function value increases as l increases. For example, the function $y = 20^x$ takes β of 20 and l of x. When drawing, 0.1 is the smallest unit, and the argument x takes a value from 0 to 2.5. The function image is shown in Fig. 4(b). By observing the function image, it can be found that the logarithmic function exhibits an increasing growth rate as the independent variable x increases. Since the function β^l is located in the denominator of the step function, when the number of migrations increases, the reciprocal of the function exhibits a downward trend that is slowed down by the rate of decline. The function as a parameter of the step size also makes the step size appear from a rush to a slow situation, and this is also in line with the intent of the step size design.

In summary, as the number of iterations increases, ($iter_{max} - iter$) becomes smaller, and $iter_{max}$ is a fixed value, which makes $C(i)$ smaller. At the same time, as the number of bacterial searches increases, the bacteria move away from the dangerous area and continue to approach the ideal food source area, then the bacterial fitness value decreases accordingly. The above fraction is also fed back to the algorithm with a smaller step size $C(i)$, which, in turn, increases the search accuracy of the algorithm. New individuals resulting from migratory behavior may have different spatial locations and foraging capabilities, enabling individuals to more easily search for better food source areas. Therefore, it is effective to feed back the migration behavior to the chemotaxis step, which can reflect the search accuracy. When the number of migration operations l increases, β^l will increase and the step size $C(i)$ will become smaller. On the basis of the above ideas, we propose the chemotaxis step function.

4.2 Improvement of replication

In the process of individual retention and replication, the 80/20 rule is introduced to improve the process. The 80/20 rule has been applied in hardware design and intelligent algorithm

research and has achieved excellent results.[22] In accordance with this rule, approximately 20% of bacterial individuals in the entire flora account for nearly 80% of the high-quality food source energy, that is, 20% of the bacterial individuals consume 80% of the energy. From the perspective of biological cell division, its specific split replication process will be as shown in Fig. 5.

In the traditional BFO algorithm, the total number of individuals in the flora is S, and the number of individuals retained is $Sr = S/2$. In the improved algorithm, S_f is taken as the number of excellent individuals retained in accordance with 80/20 rule, so that $S_f = S/5$. The newly generated individual retains the location information and other basic information of the parent individual. $\theta(i, j, k, l)$ is used to indicate the position of the ith bacterium in the search space after performing the jth chemotaxis operation, the kth copy operation, and the lth migration operation. For the copying process, the location replication process of bacterial individuals in the search space involved is described below.

$$\theta(i + S_f, j, k, l) = \theta(i, j, k, l) \tag{10}$$

$$\theta(i + 2S_f, j, k, l) = \theta(i, j, k, l) \tag{11}$$

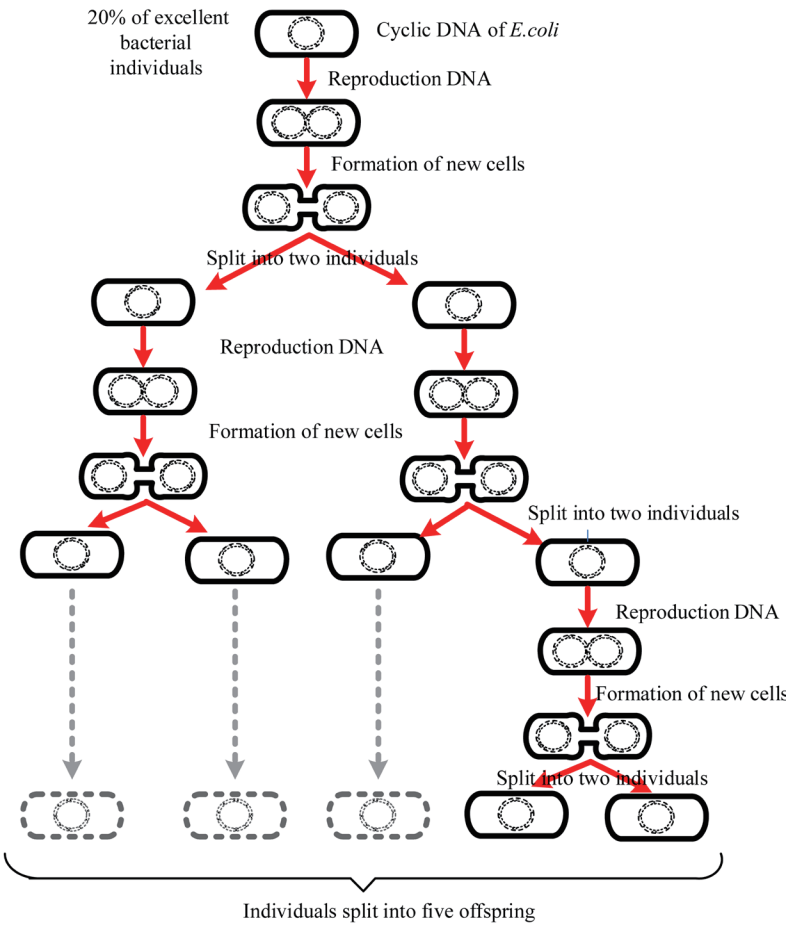


Fig. 5. (Color online) The five-times binary fission behavior of the excellent individual.

$$\theta(i + 3S_f, j, k, l) = \theta(i, j, k, l) \tag{12}$$

$$\theta(i + 4S_f, j, k, l) = \theta(i, j, k, l) \tag{13}$$

Equation (10) denotes the first replication of the remaining 25% of *E. coli* in the flora and the location information in the solution space. Equation (11) denotes the replication of half the $2S_f$ bacteria and the retention of their location information. Equation (12) denotes the replication and cloning of the location information of the other half of the $2S_f$ bacteria in the flora. At present, the number of bacteria in the flora is $4S_f$. Equation (13) represents the replication of 25% of the current flora and the assignment of identical location information to new individuals. When the above-mentioned replication process is completed sequentially, a new population with good foraging ability is born, and the number of the population is still S.

For the individual in the solution space of the algorithm, the information contained in the algorithm itself is not only the location information but also other information, such as the chemotaxis step information, which is very important information for determining the performance of the algorithm. Other information about individuals is represented here as $f(i, j, k, l)$, and the replication process is described as

$$f(i + S_f, j, k, l) = f(i, j, k, l), \tag{14}$$

$$f(i + 2S_f, j, k, l) = f(i, j, k, l), \tag{15}$$

$$f(i + 3S_f, j, k, l) = f(i, j, k, l), \tag{16}$$

$$f(i + 4S_f, j, k, l) = f(i, j, k, l). \tag{17}$$

Equation (14) indicates that one-quarter of the number of bacteria retained in the flora is replicated, and information related to the individual bacterium, such as the size of the chemotaxis step, must be replicated during the replication process. Similarly, Eqs. (15) and (16) sequentially indicate that the relevant information of the first half and the latter half of the newly generated $2S_f$ bacteria in the flora are separately copied. Equation (17) represents the last copy of one-quarter of the current $4S_f$ bacteria, which replicates the relevant information carried by the individual. When the above two types of replication behavior processes are completed, it can be considered that *E. coli* completed the improved replication behavior based on the 80/20 rule. The specific operation process of the BFO algorithm for simulating the elimination and replication behavior of individual bacteria is shown in Fig. 6.

Each bacterium has an independent label from 1 to 5 S_f. In the four blocks in the middle of Fig. 6, the white area indicates the space where the individual population of bacteria is copied, in which the black dots represent the mother bacteria, the green areas represent the space of the newly generated individual groups after replication, among which gray dots indicate progeny bacteria and red arrows indicate the correspondence between the mother and the offspring. The six blocks on the right side of Fig. 6 represent the serial number relationship of individuals.

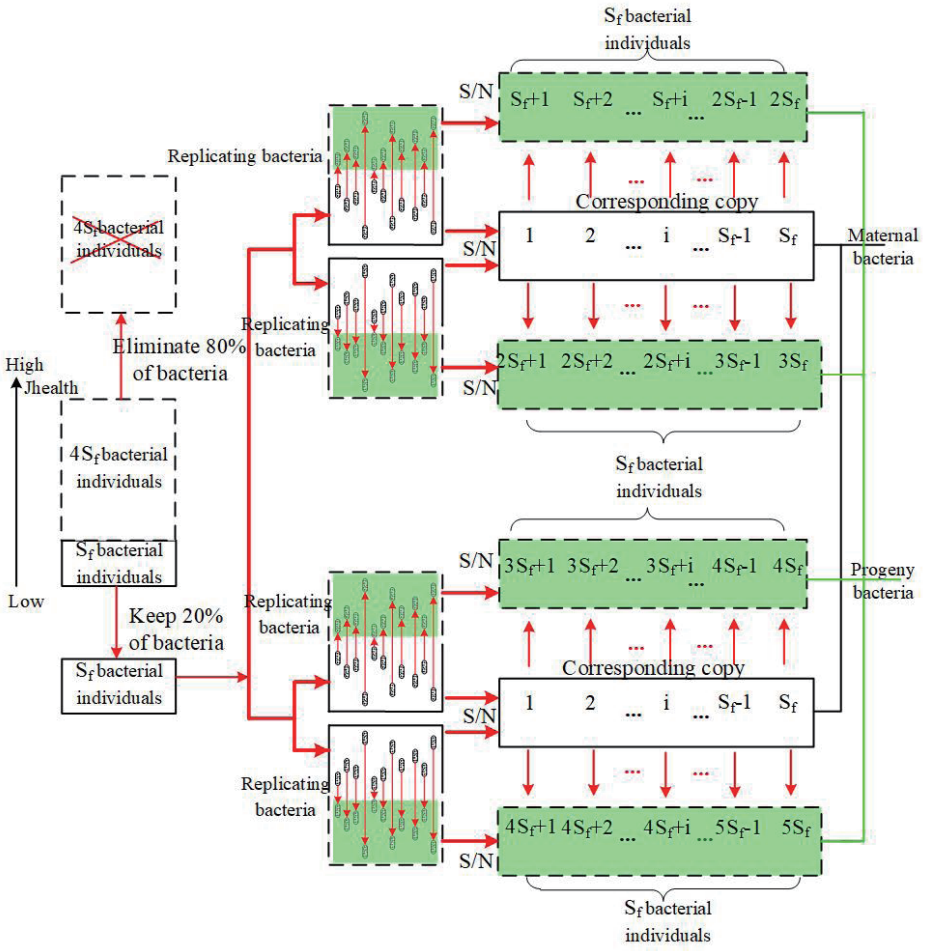


Fig. 6. (Color online) Reproduction of all the *E. coli* individuals in the improved algorithm.

From the quantitative relationship between the serial numbers, the corresponding relationship between the two individuals can be accurately determined, for example, whether it is a relationship between a parent and a child, or whether it is a neighbor relationship.

4.3 Improvement of elimination and dispersal

In the original BFO algorithm, the bacteria were all redistributed into the solution space with a certain probability P_{ed}. However, this approach did not consider the problem that the bacteria around the global optimum also migrate with P_{ed}. The original intention of the originally designed migration behavior was to help individuals who were in a local best area and its surrounding areas to jump out of their current position through such a redistribution. However, this generalized approach has led to the erroneous result of the forced migration of many elite individuals, causing a deterioration in understanding. If this situation can be reduced, the migration will be done well.

We propose a gradient migration behavior based on individual fitness values as the basis for migration. The fitness value of each bacterial individual is statistically summed to determine the health function value of the bacterium J^i_{health} (taking individual i as an example). As mentioned above, the health function of a bacterium reflects the ability of the bacterium to forage in the living space. The health function values of all bacterial individuals in the population are sorted, and bacteria with low health function values are identified. Individuals with high health function values have poor foraging ability and they are the target of priority migration. This clarifies that a more targeted migration strategy should be adopted for bacteria with different foraging abilities.

P_4, P_3, P_2, and P_1 are four important parameters, and their meanings are as follows. P_4: the migration probability (P_{ed}) of 0–25% of bacterial individuals is 0.75. P_3: P_{ed} of 25–50% of bacterial individuals is 0.1855. P_2: P_{ed} of 50–75% of bacterial individuals is 0.48757. P_1: P_{ed} of 75–100% of bacterial individuals is 0.15625. The relationship between these four parameters is as below:

$$P_4 = 75\% \times P_{ed} = 75\%P_{ed}, \tag{18}$$

$$P_3 = 25\% \times 75\% \times P_{ed} = 18.75\%P_{ed}, \tag{19}$$

$$P_2 = 25\% \times 25\% \times 75\% \times P_{ed} = 4.6875\%P_{ed}, \tag{20}$$

$$P_1 = 25\% \times 25\% \times 25\% \times 75\% \times P_{ed} = 1.5625\%P_{ed}. \tag{21}$$

This design ensures that more outstanding individuals are unaffected by forced migration and that these individuals with poor foraging ability and tendency to fall into local optima have a large migration probability. The goal of the design is to keep the bacteria with good foraging ability close to the best of the world as much as possible and to help those bacteria that are trapped near local extremes to migrate with suitable probability. The specific process of the C28BFO algorithm is shown in Fig. 7.

5. CPSOBFO Algorithm

5.1 Improvement of particle swarm optimization

The individual bacteria in the original BFO algorithm did not obtain any historical empirical data. The search for individuals placed more interest on nonhistorical search without comparison of historical values and only the individual interaction behavior was used for individual orientation. The aggregate operation at the current optimal position of the population is close to such a nonhistorical operation and is insufficient to fully exploit the maximum performance of the algorithm search. For the BFO algorithm, the introduction of the constriction factor PSO enables the full use of the ability of individuals in the PSO algorithm to perceive themselves and the historical optimal position of the group.[23,24]

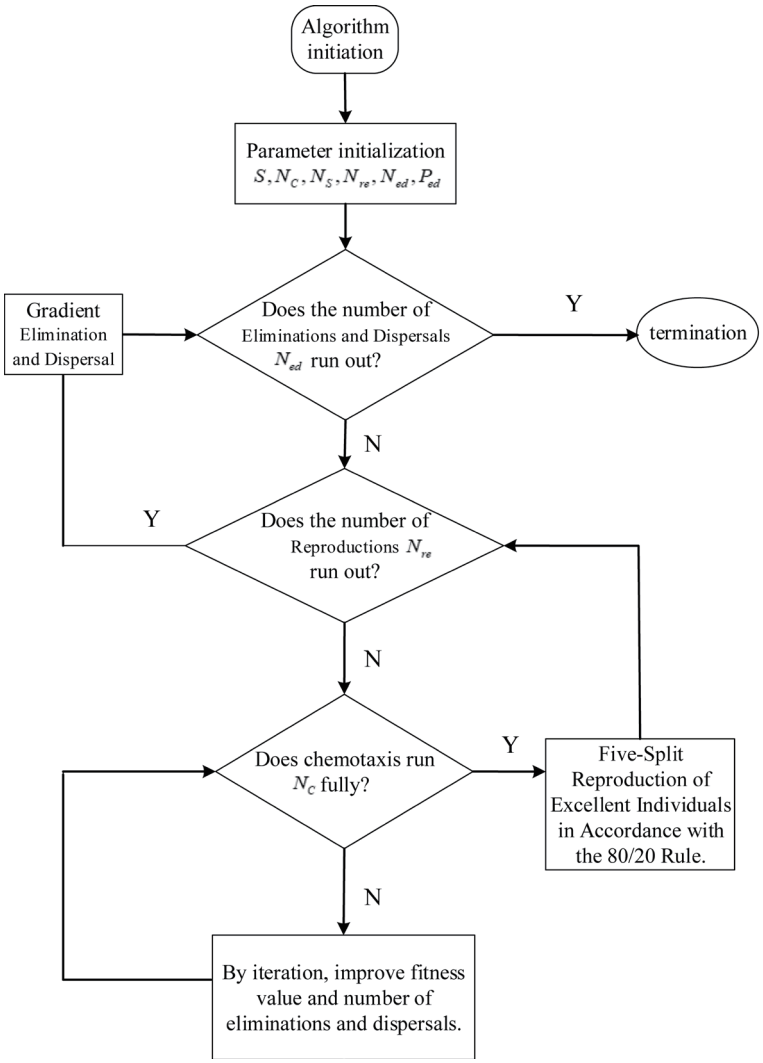


Fig. 7. Overall flow diagram of C28BFO algorithm.

In the BFO algorithm, the forward direction adjustment vector of the *E. coli* individual is $\Phi(j)$, its length is a unit value, and the direction is random. In the improved algorithm proposed in this paper, the particle update rate V_{id}^{k+1} of the constriction-factor-based PSO algorithm is regarded as the adjustment vector of the individual direction of the bacteria instead of $\Phi(j)$, as shown in

$$\Phi(j) = \chi\left[V_{id}^{k} + \phi_1(Pbest_{id}^{k} - X_{id}^{k}) + \phi_2(Gbest_{d}^{k} - X_{id}^{k})\right], \tag{22}$$

$$\chi = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}. \tag{23}$$

Equation (23) is derived from the case where n is taken as 1 in Eq. (22). Other parameters involved in the two equations, such as ϕ_1, ϕ_2, etc., are determined by the values of parameters c_1, r_1 and c_2, r_2. The basis for selecting the local optimum $Pbest_{id}^k$ and the global optimal position $Gbest_d^k$ is the level of bacterial fitness.

Add an inertia weight coefficient w in front of Eq. (22), which constitutes a new particle velocity update formula, as shown in Eq. (24); the particle position update formula is retained, as shown in Eq. (25).

$$V_{id}^{k+1} = wV_{id}^k + c_1 r_1 (Pbest_{id}^k - X_{id}^k) + c_2 r_2 (Gbest_d^k - X_{id}^k) \tag{24}$$

$$X_{id}^{k+1} = X_{id}^k + V_{id}^{k+1} \tag{25}$$

In the SPSO algorithm, the local and global optimizations can be balanced by adjusting w. Larger w values can enhance the algorithm's global search capabilities, while smaller w values can enhance the algorithm's local search capabilities. To achieve a search balance, speed exceeding V_{dmax} can be removed by artificially reducing the value of w. The number of iterations can also be reduced by decreasing the value of w. Regarding the SPSO algorithm, the moving direction of the particle consists of three parts, namely, the original particle velocity V_{id}^k, the distance $(Pbest_{id}^k - X_{id}^k)$ in the space relative to the local optimal position of the particle itself, and the distance $(Gbest_d^k - X_{id}^k)$ in the space relative to the global optimal position of the group. The importance of each part is determined by the respective weight coefficients w, $c_1 r_1$, and $c_2 r_2$.

Ebethart proposed a strategy for reducing the inertia weight coefficient w on the basis of the number of incremental iterations.[25] The algorithm achieves a large w value with a small number of iterations and thus has a strong ability to search for new regions. As the value of w decreases, the convergence ability of the algorithm is gradually enhanced to allow it to search for potential optimal solutions more finely. Ebethart experimentally found that w from 0.9 to 0.4 results in a better performance of the algorithm for both search range and search accuracy and also gives the following linear decrement equation, where k_{max} and k are the maximum number of iterations and the current number of iterations, and w_{start} and w_{end} are the starting and ending values of the inertia weight coefficient, respectively.

$$w_k = w_{start} - \frac{w_{start} - w_{end}}{k_{max}} \times k \tag{26}$$

5.2 Improvement of chemotaxis step size

We propose to use a composite function to control the change in the step size. The function includes an e-exponential function, a logarithmic function, and a trigonometric function. It exhibits a decreasing trend, so that the algorithm has a relatively large step size at the beginning, which is beneficial for obtaining a stronger global search capability. In the later

stage, the algorithm has a relatively small step size, which provides the algorithm with a better local searchability. Therefore, the improved algorithm gradually evolves from the initial wide search range to the later finer search accuracy, which helps to acquire the optimal solution as soon as possible. We analyze the improved step function in detail.

For the cosine function, the slope of the function curve changes periodically. For example, the slope of the curve of the function cos(x) in the interval (0, π) exhibits three states, which are followed by a slow decline, sharp decline, and gentle decline. Within the interval (0, π/2), the cos(x) function exhibits a slow to rapid decline as x increases. In the composite function, $x = (iter_{max} + iter) \times \pi/24$ is included in the interval (0, π/2) and increases as k increases. Figure 8(a) shows the image of the cosine function in the interval [−π/2, 3π/2] to facilitate the understanding of the characteristics of the function (Note: $\pi \approx 3.1415926$).

Since the composite function that controls the step size requires a logarithmic function, we briefly introduce the nature of the logarithmic function to be used. First, the base of the logarithmic function should be a value greater than 0 but not 1. For example, when $0 < a < 1$, the logarithmic function exhibits a decreasing trend; when $a > 1$, the logarithmic function exhibits an increasing trend. The plot of this function is shown in Fig. 8(b).

The value of the function $\cos((iter_{max} + iter)\pi/24)$ must be between 0 and 1. For this function being the base of the logarithmic function, the plot is shown in Fig. 9(a) with $0 < a < 1$. The base of the exponential function is replaced by $\cos((iter_{max} + iter)\pi/24)$. Here, we assume that the cosine function is a constant, then the function $y = \log_{\cos((iter_{max} + iter)\pi/24)}x$ decreases with increasing x, and when $x > 1$, the function value is less than 0.

In Fig. 9(b), when $0 < a < 1$, if the absolute value of the original logarithm function is $y = |\log_a x|$, then the part of the original logarithm function less than 0 is on the x-axis. The $y = \log_a x$ function image with $a > 1$ is presented on the plot. By observing the image of the segment, it can be found that the function gradually increases as x increases, and conversely, the function gradually decreases as x decreases.

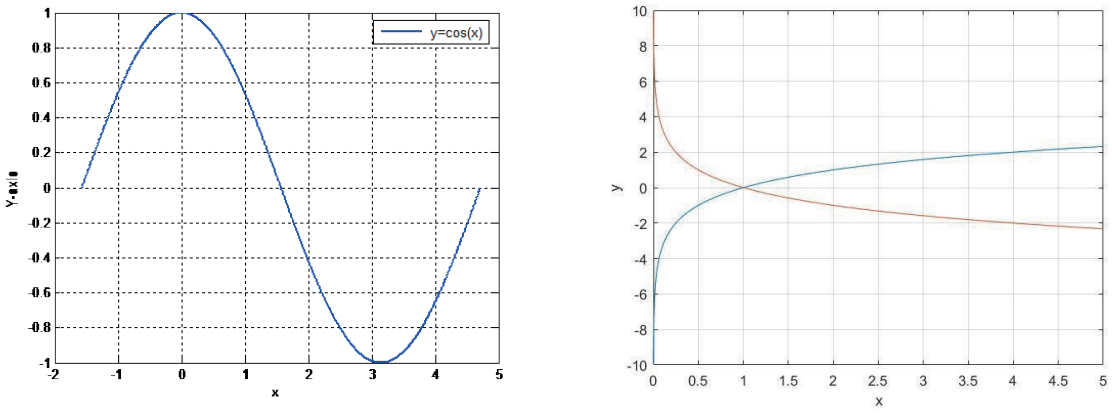
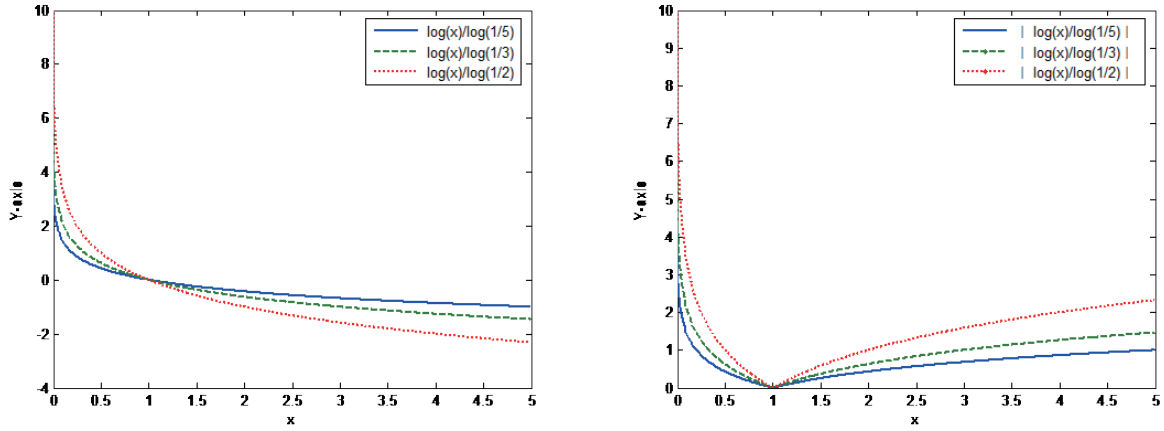


Fig. 8. (Color online) Plots of basic functions. (a) Cosine function in the interval [−π/2, 3π/2]. (b) Plot of logarithmic function.



(a) (b)

Fig. 9. (Color online) Plots of different functions when bases are 1/5, 1/3, and 1/2. (a) Logarithmic function. (b) Logarithmic absolute function value.

Now, we analyze the influence of the logarithm when the base a of the logarithm function $y = \log_a x$ takes a different value. Since only the case where the bottom value is greater than 0 and less than 1 is analyzed, the results obtained with base numbers 1/5, 1/3, and 1/2 are compared, and the corresponding logarithmic function size relationship is shown in Fig. 9(a). In this figure, these logarithmic functions with base numbers 1/5, 1/3, and 1/2 are all from 0.001 to 5, and the minimum unit of advance is 0.01. The 100 on the *x*-axis represents 100 minimum advance units, that is, 0.01 × 100 = 1. Similarly, 150 on the *x*-axis represents the value 1.5, 300 represents the value 3, and 500 represents the value 5. Intercepting the arguments of <1<5<5 in Fig. 9(a), we find that the different bottom functions with the same *x* value have the relationship $\log_{1/5} x > \log_{1/3} x > \log_{1/2} x$ when the base is at (0, 1). In the meantime, the function value decreases as the base increases. Under the same conditions, the relationship $|\log_{1/5} x| < |\log_{1/3} x| < |\log_{1/2} x|$ can be derived, as shown in Fig. 9(b). In addition, the logarithmic absolute function value involved in the step function numerator is a function of $\cos((iter_{max} + iter)\pi/24)$ with $(iter_{max} - iter + 1)$ being an independent variable. Within the effective range, the base function gradually decreases as the *iter* increases. The independent variable function gradually decreases as the *iter* increases. Therefore, it can be seen that the function numerator gradually becomes smaller as *iter* increases, and the process of decrement is also in line to design a new step.

Combining the previously introduced cosine functions and logarithmic functions, we propose a new chemotaxis step size expression based on the above analysis:

$$C(i) = \left(C_{max} - C_{min}\right) \exp\left(\frac{\left| \log_{\cos\left(\frac{(iter_{max}+iter)\pi}{24}\right)} (iter_{max} - iter + 1) \right|}{\left| \log_{\left(\frac{\sqrt{2}}{2}\right)} 7 \right|} \right) + C_{min}, \tag{27}$$

where C_{max} and C_{min} are the maximum and minimum step sizes, and $iter_{max}$ and $iter$ are the maximum number of iterations and the current number of iterations, respectively. The function value of the e-index first increases gradually and then increases sharply as the independent variable becomes larger.

Since the position of the bacteria in the solution space is random, only a small number of bacterial individuals gather near the initial global advantage at the beginning of the algorithm. The new step size design allows bacteria to initially have a relatively large-varying step size and a wide range of search capabilities, i.e., a strong global search capability. As the number of iterations increases, the chemotaxis step size of the bacteria is gradually reduced, and the accuracy of the search is gradually improved, that is, the local search ability becomes strong. The new chemotaxis step strategy proposed in this paper takes into account locality and globality and thus has a higher convergence speed.

5.3 Improvement of reproduction

In the original BFO algorithm, the replication process plays a role in screening elite individuals, sorting all bacteria in accordance with their health function, and retaining the top 50% of the individuals and copying them once, so that all individuals in the new population have excellent characteristics. In this paper, the biological characteristics are introduced, that is, they have the characteristics of preying on other bacteria. Here, the biological predation habit of the T4 phage infecting *E. coli* is introduced to achieve the effect of eliminating undesirable individuals in the flora.

The T4 phage can infect *E. coli*. In general, a T4 phage can release 100 to 200 progeny individuals after a certain time of infection of a bacterium, which means that an equal number of *E. coli* individuals are eliminated. In BFO and its improved algorithm, the number of flora is not set to such a large number, because although high populations help to improve the ability to find the optimal solution in the solution space, it will take a long time. At the same time, the BFO algorithm and its improved algorithm are used to obtain the global optimal solution. For example, in fields such as PID control systems and image optimization, these applications often have strict requirements for task processing time, and the calculation time cannot be too slow. Therefore, in consideration of the limitations of the actual situation, it is necessary to impose certain conditions on the generation rate of the progeny of the T4 phage. Let $R(N)$ denote the number of progeny individuals generated after the phage individual infects the *E. coli* individual,

$$R(N) = (3N - 2) \times 2^N + N \text{ and } 2S > 2NS - R(N) > S. \tag{28}$$

The number of *E. coli* individuals in the flora is S, N is the number of progeny of the phage infecting the *E. coli*, and $2NS$ is the number of all bacterial individuals in the population of all bacterial individuals of the original population after N divisions and reproductions. $R(N)$ is defined by the above inequalities for S and N.

As a condition of the biological behavior of T4 phage, it is stipulated that *E. coli* cells need not retain half of the population and then replicate as specified by the original algorithm, but

instead, N bacteria of the entire population are replicated N times. At the same time, a phage is introduced to infect an *E. coli* individual, and N-proliferation is performed in the body in accordance with the $R(N)$ rule to generate progeny phage. In this paper, it is stipulated that when these phages reinvade the bacteria, they will perish with the host.

From the specific value of S, the N value can be determined and the number of *E. coli* $[2NS - R(N)]$ in the new population can be obtained. Then these individuals are sorted in accordance with the health value, and the former S bacteria are retained as all the individuals of the new population, thus achieving the screening of excellent individuals.

6. Results and Discussion

6.1 System specifications

All the algorithms are implemented using MATLAB 7.0, version 7.0.0.19920 (R14), as shown in Fig. 10. The main frequency of the processor is 2.0 GHz and the memory size is 3 GB.

6.2 Test functions

In this work, we partitioned each of the functions below into simulating and testing functions so that the performance of various algorithms can be tested and evaluated comprehensively. The images of all functions simulated by MATLAB are shown as Fig. 11.

Sphere function: The formula of the sphere function is shown as Eq. (29). From the image of the sphere function, we can see that its global optimum point is {0, ..., 0}, then the global optimum value corresponding to the function is $f(x) = 0$.

$$f(x) = \sum_{i=1}^{n} X_i^2 \tag{29}$$

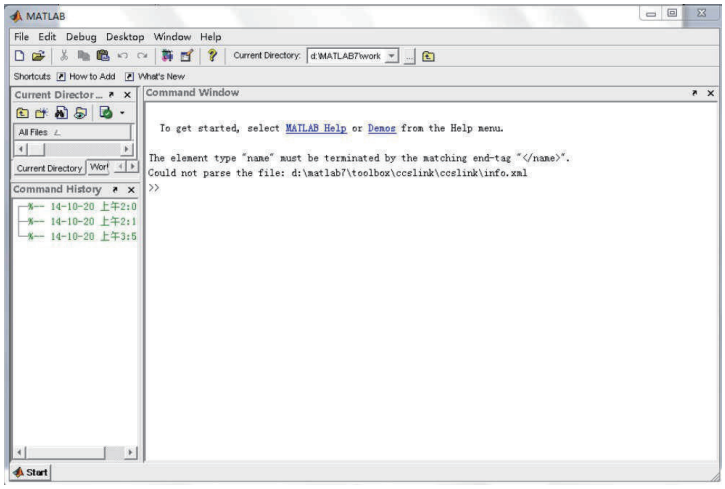


Fig. 10. (Color online) Operation interface of MATLAB development environment.

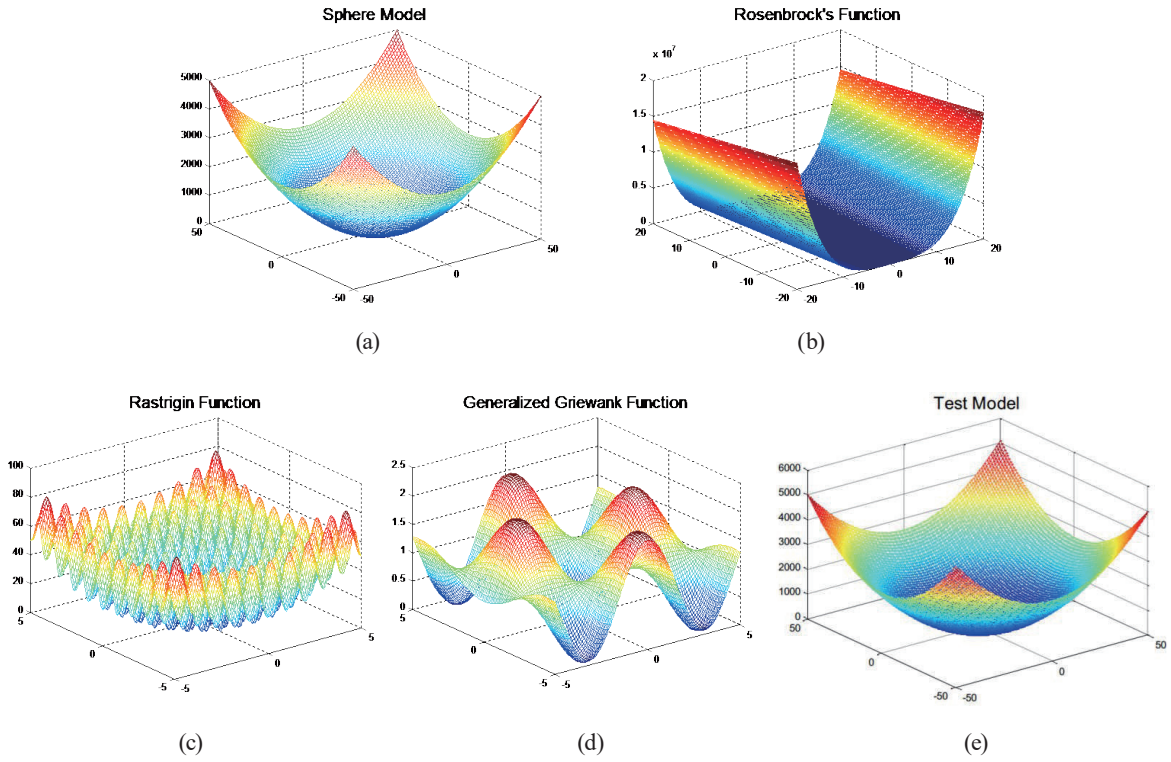


Fig. 11. (Color online) MATLAB simulation images of different test functions. (a) Sphere function. (b) Rosenbrock function. (c) Rastrigin function. (d) Griewank function. (e) Standard test model.

Rosenbrock function: The formula is shown as Eq. (30). Although the valley of this function is easily found, the algorithm is easily misled by the gradient information of the function, and a small change in the value in the valley makes it difficult to find the global minimum. From the image of the function, we can see that its global optimum point is {1, ..., 1}, where the global optimal value $f(x) = 0$ is obtained.

$$f(x) = \sum_{i=1}^{n} X_i^2 + X_i \tag{30}$$

Rastrigin function: It is difficult to find the minimum value of the function, which mainly depends on a larger search space of the function and a large number of deep local minimum values of the function arranged in accordance with the sinusoidal inflection points. The formula is shown as Eq. (31). The global minimum is the global optimal point at {0, ..., 0}. The function value corresponding to this point is $f(x) = 0$. It should be noted that the optimization algorithm easily falls into the local optimum.

$$f(x) = \sum_{i=1}^{n} (X_i^2 - 10\cos(2\pi X_i) + 10)^2 \tag{31}$$

Griewank function: Its mathematical description is shown as Eq. (32). This function is a complex multimodal function with local minimum points and large obstacles. The high correlation of variables makes the algorithm fall into a local optimum easily. The global optimal point of the function is at {0, ..., 0} with the corresponding function value of $f(x) = 0$.

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n}(X_i - 100)^2 - \prod_{i=1}^{n}\cos\left(\frac{X_i - 100}{\sqrt{i}}\right) + 1 \tag{32}$$

Standard test function: The selection of standard test functions is shown as Eq. (33). We discuss the simulation results of the traditional BFO algorithm, C28BFO algorithm, and CPSOBFO algorithm running on standard test functions.

$$f(x) = \sum_{i=1}^{n} X_i^2 + X_i \tag{33}$$

6.3 Simulation results

We optimize the parameters of the two improved algorithms, and then run the optimized algorithm on the standard test function to obtain the simulation results. Additionally, we compare the simulation results of the improved algorithm and the traditional algorithm and discuss the performance of the improved algorithm.

The parameters of the C28BFO algorithm are optimized as follows: the bacterial population is set to 50, the dimension of the search space is 15, the migration probability P_{ed} is 0.25, the minimum chemotaxis step C_{min} is 0.05, the number of chemotaxis executions N_c is 1000, the number of replications N_{re} is 5, the number of executions N_{ed} is 2, the maximum number of steps forward in the same direction during chemotaxis N_s is 4, the number of excellent individuals retained in the replication behavior is $S_f = S/5$, and the maximum number of iterations is 10000.

The parameters of the CPSOBFO algorithm are also optimized as follows: bacterial population size of 52, search space dimension of 15, migration probability P_{ed} of 0.25, maximum and minimum chemotaxis step sizes of $C_{max} = 0.1$ and 0.05, and $C_{min} = 0.1$ and 0.05, number of chemotaxis executions N_C of 1000, replication number N_{re} of 5, number of migration executions N_{ed} of 2, maximum number of steps forward in the same direction during chemotaxis of 4, and c_1 and c_2 of 2.05.

After setting the parameters, simulations with both algorithms are performed 10000 times. Several indicators for evaluating the performance of these algorithms are obtained: min value, max value, average value, and standard deviation. The standard data obtained from eight simulations with C28BFO and CPSOBFO algorithms are shown in Tables 2 and 3.

As shown, the simulated data are close to each other after a large number of iterations. At the same time, the simulated images for each group are similar. Therefore, the corresponding images of the first group of data in the table are given as an example.

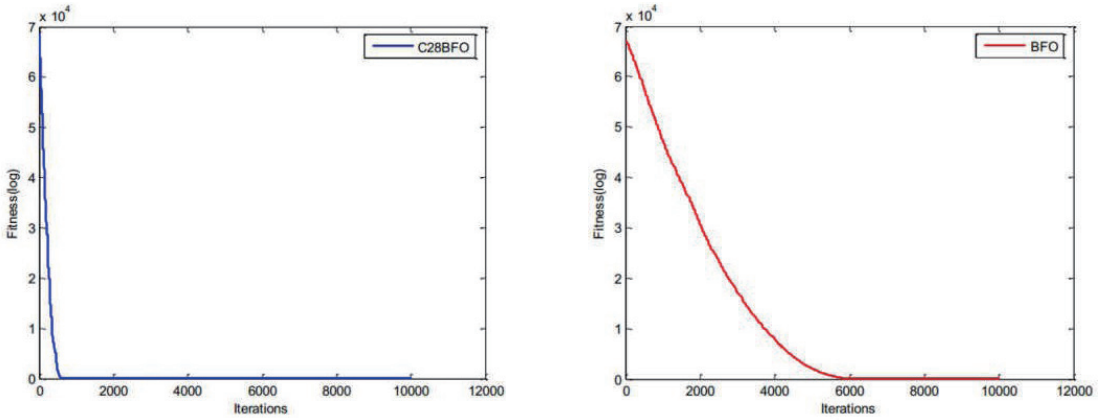
The results of simulations using the C28BFO algorithm shown in Fig. 12(a) demonstrate that the C28BFO algorithm finds the minimum value and shows good global convergence in

Table 2
Results of simulation with C28BFO algorithm.

S/N	Minimum	Maximum	Average value	Standard deviation
1	9.8535	6.8647e+004	1.3817e+003	7.4929e+003
2	9.0499	6.9979e+004	1.3145e+003	7.1791e+003
3	10.2501	6.5632e+004	1.3877e+003	7.1940e+003
4	12.8886	6.4345e+004	1.3878e+003	7.1507e+003
5	9.7737	6.6766e+004	1.2079e+003	6.8080e+003
6	7.8143	7.3082e+004	1.3745e+003	7.6817e+003
7	9.6852	7.0552e+004	1.3794e+003	7.5952e+003
8	10.3507	7.3194e+004	1.4279e+003	7.5933e+003

Table 3
Results of simulation with CPSOBFO algorithm.

S/N	Minimum	Maximum	Average value	Standard deviation
1	0.1012	6.3004e+004	9.1633e+003	1.5509e+004
2	0.1241	6.7967e+004	9.2164e+003	1.7492e+004
3	0.1016	6.6854e+004	1.0333e+004	1.6819e+004
4	0.1059	6.6021e+004	9.2164e+003	1.5768e+004
5	0.0971	6.4306e+004	9.6864e+003	1.5955e+004
6	0.0920	7.5300e+004	1.2640e+004	1.9540e+004
7	0.0965	6.8206e+004	1.0797e+004	1.7417e+004
8	0.0909	7.1584e+004	1.0884e+004	1.7797e+004

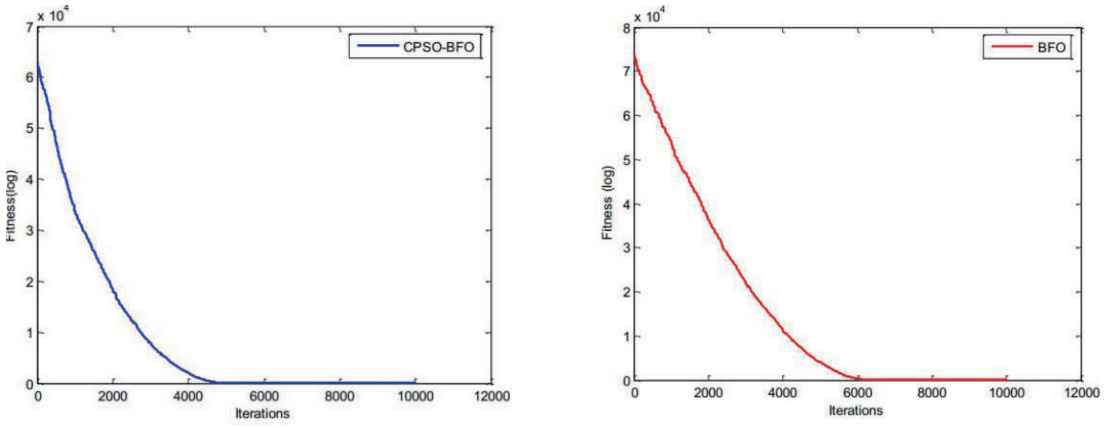


(a) (b)

Fig. 12. (Color online) MATLAB simulation image using the C28BFO algorithm. (a) Improved algorithm. (b) BFO algorithm.

the process of dealing with test function problems when the number of iterations is about 600. Figure 12(b) shows that the original BFO algorithm usually converges only after about 5800 iterations. Through the comparison, we can see that the C28BFO algorithm has a better overall performance.

Similarly, as shown in Fig. 13(a). the CPSOBFO algorithm finds the minimum value and shows good global convergence when the number of iterations is about 4500 times. Figure 13(b) shows that the original BFO algorithm usually converges only after about 6000 iterations. Compared with the BFO algorithm, the CPSOBFO algorithm has better overall performance.



(a) (b)

Fig. 13. (Color online) MATLAB simulation image using the CPSOBFO algorithms. (a) Improved algorithm. (b) BFO algorithm.

6.4 Test results

In this section, four kinds of algorithms, that is, the GA algorithm, the BFO algorithm, the C28BFO algorithm, and the CPSOBFO algorithm, are applied to complex test functions, including the Sphere function, Rosenbrock function, Rastrigin function, and Griewank function. For complex functions with different characteristics, we discuss the different performances of various algorithms referring to the test results and evaluate the robustness of the improved algorithms.

The results of tests using the four algorithms for the Sphere function are given in Table 4, and Fig. 14(a) shows the images for the four algorithms from 0 to 1000 iterations. In addition, n = 12 and C_{min} = 0.02 were selected in the test.

As shown, the GA algorithm drops faster, but quickly falls into a local optimum at about 100 iterations. Before 200 iterations, the C28BFO algorithm has a better fitness value; after 200 iterations, the CPSOBFO algorithm drops faster, and the convergence is the best among the four functions. The C28BFO algorithm is also slightly better than the BFO algorithm. Therefore, the two improved algorithms proposed in this paper show superior convergence properties for the Sphere test function over the traditional algorithms.

The results of tests using the four algorithms for the Rosenbrock function are given in Table 5, and Fig. 14(b) shows the images of the four algorithms from 0 to 1000 iterations. In addition, n = 12 and C_{min} = 0.05 were selected in the test.

As shown, the GA drops the fastest at the beginning and falls into a local optimum at about 50 iterations. The C28BFO algorithm dropped faster before 900 iterations, and the CPSOBFO algorithm slowed slightly faster than the C28BFO algorithm after 900 iterations. Both the C28BFO algorithm and the CPSOBFO algorithm are far superior to the GA algorithm and the BFO algorithm, and the convergence of the two improved algorithms for the Rosenbrock test function is similar.

Table 4
Results of simulation with the four algorithms for the Sphere function.

Test function	Minimum	Maximum	Average value	Standard deviation
BFO	0.1165	6.9690e+004	1.4812e+004	2.0093e+004
GA	4.3713e+004	6.2961e+004	4.6172e+004	1.3527e+003
C28BFO	0.0853	6.3004e+004	1.2031e+004	1.7856e+004
CPSOBFO	0.08090	7.3580e+004	1.19888e+004	1.8727e+004

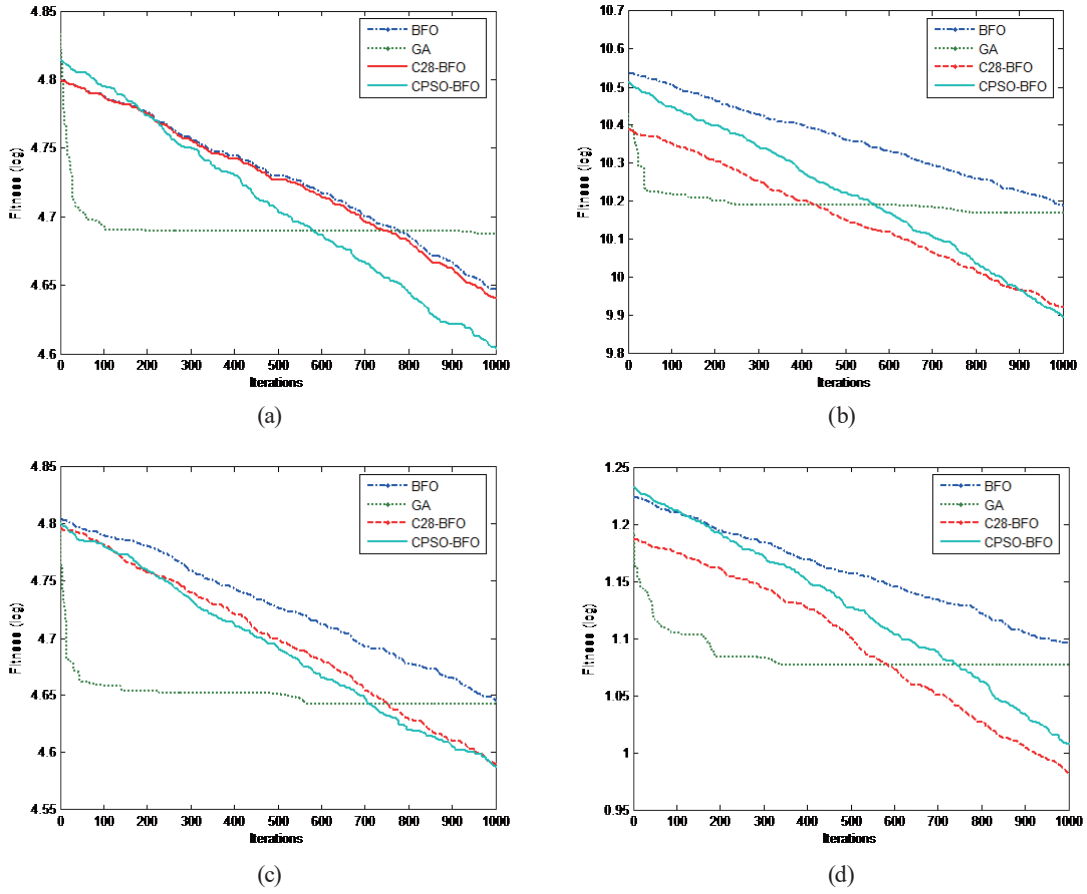


Fig. 14. (Color online) Results of algorithms for various test functions. (a) Sphere function. (b) Rosenbrock function. (c) Rastrigin function. (d) Griewank function.

Table 5
Results of simulation with the four algorithms for the Rosenbrock function.

Test function	Minimum	Maximum	Average value	Standard deviation
BFO	225.3188	3.4477e+010	3.7777e+009	7.5218e+009
GA	1.1800e+010	2.6945e+010	1.3573e+010	1.3098e+009
C28-BFO	243.6926	2.4465e+010	2.0026e+009	4.8327e+009
CPSO-BFO	238.6558	3.2732e+010	2.3334e+009	5.7894e+009

The results of tests using the four algorithms for the Rastrigin function are given in Table 6, and Fig. 14(c) shows the images of the four algorithms from 0 to 1000 iterations. In addition, n = 12 and C_{min} = 0.05 were selected in the test.

Table 6
Results of simulation with the four algorithms for the Rastrigin function.

Test function	Minimum	Maximum	Average value	Standard deviation
BFO	240.3213	6.3737e+004	1.4217e+004	1.7534e+004
GA	4.0809e+004	6.2357e+004	4.2855e+004	1.3951e+003
C28BFO	66.7099	6.2688e+004	1.0420e+004	1.6313e+004
CPSOBFO	113.0388	6.312e+004	1.1199e+004	1.5986e+004

Table 7
Results of simulation with the four algorithms for the Griewank function.

Test function	Minimum	Maximum	Average value	Standard deviation
BFO	0.0210	16.7172	3.7408	4.8635
GA	11.7086	15.6413	11.7574	0.2118
C28BFO	0.4281	15.3644	2.6954	3.9745
CPSOBFO	0.3650	17.0981	3.0520	4.2946

As shown, the convergence performances of the BFO algorithm, the C28BFO algorithm, and the CPSOBFO algorithm are relatively good. However, compared with the BFO algorithm, the C28BFO algorithm and the CPSOBFO algorithm have better convergence, and the two perform similarly. Although the GA algorithm has a very fast convergence rate at the beginning, it starts to enter a slow decline period at about 50 iterations, and falls into a local optimum at about the 580th iteration. Therefore, the two improved algorithms proposed in this paper show better convergence.

The results of tests using the four algorithms for the Griewank function are given in Table 7, and Fig. 14(c) shows the images of the four algorithms from 0 to 1000 iterations. It should be noted that the algorithm function does not stop convergence when it is performed only 1000 times, but continues to calculate until the algorithm function converges or reaches the maximum number of iterations. In addition, $n = 12$ and $C_{min} = 0.05$ were selected in the test.

As shown, in the initial stage of the iteration, the GA algorithm has a good convergence speed, and it falls into the local optimum with 330 iterations. The BFO algorithm, the CPSOBFO algorithm, and the C28BFO algorithm show better convergence. It can be seen that the C28BFO algorithm has the best convergence, followed by the CPSOBFO algorithm and the BFO algorithm.

7. Conclusion and Future Directions

We studied and improved the traditional BFO algorithm, and proposed two novel algorithms. Both algorithms achieved remarkable results in solving the convergence problem. The optimization of the algorithm parameters greatly improved the search speed and accuracy of the algorithm. The results of the overall performance assessment in this work are summarized as follows.

First, the C28BFO algorithm was proposed to improve the chemotaxis, replication, and migration behaviors of the algorithm. The chemotaxis step was improved to a decreasing composite function, and the management behavior of the 80/20 rule was applied to improve the screening of elites. Then, the fixed probability migration behavior was improved to a migration

behavior with a gradient migration probability. Readers can explore more management ideas with the algorithm to achieve control of the group, such as The Mirror Theory and The Role Model effect.

Second, the related technology of particle swarm optimization based on a constriction factor was introduced to improve the chemotaxis and replication behaviors. Then, the CPSOBFO algorithm was proposed. For the chemotaxis behavior, the particle update rate was used instead of the *E. coli* individual's forward direction adjustment vector, and a composite function of several special functions was introduced as the chemotaxis steps. For the replication behavior, the biological characteristics of a phage foraging *E. coli* were introduced to achieve the screening of excellent individuals. Readers can extensively explore other optimization algorithms combined with BFO algorithms to improve the performance of the algorithm for specific issues such as the formation problem and optimal path.

Third, the results of simulation using the two improved algorithms for the standard test function were given and compared with those of using the GA algorithm. The results of tests using the four algorithms for various complex test functions were given, and the performance of each algorithm was comprehensively evaluated. Then, the performance of each algorithm in different situations was analyzed in turn; few studies have included such a comprehensive comparison. At present, the bionic algorithm, including the ant colony algorithm and combinatorial optimization, has very strict hardware requirements, and the research of bionic intelligent hardware is at the forefront of international research, so readers can explore this field.

Collectively, the study reported in this paper revealed the shortcomings of the traditional BFO algorithm and detailed improvement methods were proposed. The simulation results showed that the two improved algorithms have excellent performance in many complex test functions.

Acknowledgments

The authors would like to acknowledge support from the following projects: (1) Liaoning Provincial Science and Technology Department Natural Fund Guidance Project, project name: weighted joint target recognition based on spatial relationship of contour segments (No. 2019-ZD-0252); (2) Liaoning Province Higher Education Innovative Talents Program Support Project (No. LR2019058); (3) Liaoning Provincial Department of Education Project (No. LG201917); and (4) National Natural Science Foundation of China (No. 51575412).

References

1 H. Gao, D. Peng, B. Niu, and B. Li: Int.Conf. Intelligent Computing (2013) 641. https://doi.org/10.1007/978-3-642-39482-9_74
2 V. Gazi and KM. Passino: IEEE Trans. Automat. Contr. **48** (2003) 4. https://doi.org/10.1109/TAC.2003.809765
3 K. Dervis: Technical Report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department. https://pdfs.semanticscholar.org/015d/f4d97ed1f541752842c49d12e429a785460b.pdf
4 K. M. Passino: IEEE Control Syst. [serial online] **22** (2002) 3. https://doi.org/10.1109/MCS.2002.1004010
5 Y. Liu, K. M. Passino, and M. Polycarpou: Proc. 2002 American Control Conf. (IEEE 2002) 1278. https://doi.org/10.1109/ACC.2002.1023196

6 H. C. Berg and N. Brown: Nature **239** (1972) 5374. https://doi.org/10.1038/239500a0
7 G. Lowe, M. Meister, and N. Berg: Nature **325** (1987) 6105. https://doi.org/10.1038/325637a0
8 S. Mishra and C. N. Bhende: IEEE Trans. Powor Deli. **22** (2006) 457. https://doi.org/10.1109/TPWRD.2006.876651
9 Y. Liu and K. M. Passino: J. Optimiz. Theory Appl. **115** (2002) 3. https://doi.org/10.1109/ACC.2002.1023196
10 W. Tang, Q. Wu, and J. Saunders: Int. Conf. Computational Science and Its Applications. (2006) 556. https://doi.org/10.1007/11751540_59
11 B. Niu, Y. Fan, H. Wang, L. Li, and I. Wang: Int. J. Artif. Intell. **7** (2011) 11. https://doi.org/10.1155/2012/698057
12 Y. Meng, S. Zhao, and S. Hu: Int. J. Comput. Sci. Math. **6** (2015) 471. https://doi.org/10.1504/IJCSM.2015.072969
13 K. Tang, X. Xiao, J. Wu, J. Yang, and I. Luo: Appl. Intell. **46** (2017) 1. https://doi.org/10.1007/s10489-016-08329
14 O. P. Verma and S. Parihar: IEEE Trans. Fuzzy Syst. **25** (2017) 1. https://doi.org/10.1109/TFUZZ.2016.2551289
15 L. Chen: 19th Int. Conf. Network-Based Information Systems (2016). https://doi.org/10.1109/NBiS.2016.48
16 M. Tripathy and P. Mishra: Int. J. Elec. Power Energy Syst. **66** (2015). https://doi.org/10.1016/j.ijepes.2014.10.022
17 X. Yan, Z. Zhang, J. Guo, S. Li, and S. Zhao: Int. Conf. Intelligent Computing (Springer 2016) 627. https://doi.org/10.1007/978-3-319-42291-6_62
18 P. Manikandan and J. Ramyachitra: Sci. Rep. **7** (2017) 1. https://www.nature.com/articles/s41598-017-09499-1
19 J. Zhou, Y. Xu, Y. Zheng, and Y. Zhang: Energies **10** (2017) 7. https://doi.org/10.3390/en10070911
20 J. Lin and S. Lin: Int. J. Adv. Rob. Syst. **14** (2017) 4. https://doi.org/10.1177/1729881417720872
21 H. Hossein: Appl. Math. Modell. **40** (2016) 2. https://doi.org/10.1016/j.apm.2015.09.004
22 W. Steven and D. Sudhir: IBM J. Res. Dev. **38** (1994) 5. https://doi.org/10.1147/rd.385.0493
23 M. Clerc and K. James: IEEE Trans. Evol. Comput. **6** (2002) 1. https://doi.org/10.1109/4235.985692
24 E. Ozcan and C. K. Mohan: Proc. 1999 Congr. Evolutionary Computation-CEC99 **3** (1999). https://doi.org/10.1109/CEC.1999.785510
25 Y. Shi and R. Eberhart: Proc. Conf. Evolutionary Computation (IEEE, 1998) 696. https://doi.org/10.1109/ICEC.1998.699146

About the Authors

Hongwei Gao received his Ph.D. degree in pattern recognition and intelligent system from the Shenyang Institute of Automation (SIA), Chinese Academy of Sciences (CAS), in 2007. Since September 2015, he has been a Professor of the School of Automation and Electrical Engineering, Shenyang Ligong University. He is currently the leader of academic direction for optical and electrical measuring technology and systems. His research interests include digital image processing and analysis, stereo vision, and intelligent computation. He has published more than 60 technical articles in these areas as first author or co-author. (ghw1978@sohu.com)

Jiahui Yu received his B.S. and M.S. degrees from Shenyang Ligong University, China, in 2017 and 2019, respectively. Since 2019, he has been a Ph.D. candidate at the University of Portsmouth, U.K. His research interests are in deep learning, machine vision, and human-robot interaction and collaboration. (yujiahui77@163.com)

Dai Peng received his B.S. and M.S. degrees from Shenyang Ligong University, China, in 2012 and 2015, respectively. His research interests are in particle swarm optimization algorithm and machine vision. (657332929@qq.com)

Zhaojie Ju received his Ph.D. degree in intelligent robotics from the University of Portsmouth, U.K., in 2010. He held a research appointment at the University College London, U.K., before he started his independent academic position at the University of Portsmouth, U.K., in 2012. His research interests include machine intelligence, pattern recognition, and their applications on human motion analysis, multifingered robotic hand control, human-robot interaction and collaboration, and robot skill learning. (zhaojie.ju@port.ac.uk)

Yanju Liu received her Ph.D. degree from Shenyang University of Technology, China, in 2011 She has been a professor at Shenyang Ligong University. Her research interests are in intelligent instrumentation, networked measurement, and control technology. (6133292@qq.com)