Bacterial Foraging Algorithm With Potential Field Guidance Mechanism

Zhiyuan Li (2399808352@qq.com)
Liuzhou Railway Vocational Technical College
Zhicheng Wang
Liuzhou Railway Vocational Technical College

Research Article

Keywords: Bacterial foraging optimization, Potential field guidance, Local dimension update, Double Gaussian function, Function optimization

Posted Date: November 3rd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1024712/v1

Bacterial Foraging Algorithm with Potential field guidance Mechanism

Zhiyuan Li[1], Zhicheng Wang[1]
[1]Liuzhou Railway Vocational Technical College, Liuzhou, 545616, China

Abstract

To address the problems of weak quorum sensing ability and slow convergence speed in bacterial foraging algorithm, a bacterial foraging algorithm with potential field guidance mechanism is proposed. The algorithm combines the sampling guidance mechanism in the artificial potential field algorithm to provide the optimization direction for each bacterium; The original swimming operation of bacterial foraging algorithm is used to realize the local optimization strategy, and the local dimension update is added after swimming, so that the search range of bacteria in chemotaxis operation is wider; In the elimination and dispersal operation of bacterial foraging algorithm, double Gaussian function is introduced to re initialize the location of bacteria, so as to better avoid the algorithm falling into local extremum and improve the optimization ability of the algorithm. The experimental results show that the improved bacterial foraging algorithm has better optimization ability than the basic bacterial foraging algorithm.

Keywords: Bacterial foraging optimization; Potential field guidance; Local dimension update; Double Gaussian function; Function optimization

Introduction

Bacterial foraging optimization algorithm (BFO) is a bionic group algorithm [1] that mimics the drug-tending motility of E. coli and has been widely used in the fields of proportional-integral-differential controller (PID) parameter tuning, pattern recognition, image processing, etc. The algorithm has been widely used in the fields of proportional-integral-differential controller (PID), parameter tuning, pattern recognition, and image processing. The algorithm is simple to construct, easy to implement, parallel in search, and easy to jump out of local extremes, thus attracting the interest of many researchers. In the convergence operation of the bacterial foraging algorithm, individuals search blindly throughout the optimal search space without any orientation because the direction of each bacterial swim is randomly generated. Simulation experiments by Tang et al. [2]proved that the population sensing mechanism of the BFO interferes with the convergence of the bacterial foraging algorithm, and Chu Y [3] found through extensive experimental tests that when there are more individuals in the population distributed around the local extrema, the individuals who have already found the global optimal solution will also fall into the local optimum due to the effect of the group induction mechanism, which also causes the bacterial foraging algorithm to fail to converge quickly to obtain the global optimal solution of the function when solving high-dimensional problems or complex functions. The *elimination and dispersal* operation just helps the algorithm to jump out of the global optimum by re-initializing the position of bacterial individuals, and this approach may cause the re-initialized bacteria to remain in the vicinity of the local optimum solution, which cannot achieve the desired effect.

To address the above problem, many scholars have proposed solutions. Mishra et al. [4] have proposed a solution in a fuzzy bacterial foraging algorithm (Fuzzy bacterial foraging (FBF), and then, many scholars have studied the step size in bacterial foraging algorithm. Niu et al.[5] proposed nonlinearly decreasing and exponentially nonlinearly decreasing step sizes by verifying the effect of step size on global convergence. Tan et al. [6]

calculated the current bacterial convergence step size based on the current optimal position of bacteria, the historical optimal position and the average position of all bacteria in the population. Zhen L et al. [7] proposed in an improved bacterial foraging optimization algorithm based on nonlinearly decreasing cosine adaptive step size. To further improve the performance of the algorithm, scholars started to combine the basic bacterial foraging algorithm with other metaheuristics to propose a hybrid bacterial foraging algorithm. Zhou W H et al. [8] combined the bacterial foraging algorithm with particle swarm algorithm and proposed variable probability hybrid bacterial foraging optimization algorithm. Liu et al. [9] introduced the idea of clonal selection in immune algorithm to bacterial foraging algorithm, and Du et al. [10] designed a convergence method of employed bees based on artificial bee swarm algorithm. In order to improve the global optimization capability of the algorithm, Zhang et al. [11]introduced the quantum potential energy trap in quantum mechanics into the bacterial foraging algorithm to improve the optimization capability of the bacterial foraging algorithm to a certain extent; Liu et al. [12]designed a nonlinear dynamic adaptive rotation angle to continue to improve the convergence operation in the quantum bacterial foraging algorithm to further improve the convergence speed of the algorithm. From the existing literature on bacterial foraging algorithms, it can be seen that most of the improvement schemes are limited to the improvement of the step length of the algorithm, without considering the influence of the relationship between individual bacteria in the whole population on the optimization process. Although the quantum bacterial foraging algorithm has improved its solving performance compared with the basic bacterial foraging algorithm, the solving ability of the algorithm becomes weaker as the number of dimensions increases.

In this paper, we propose a bacterial foraging optimization algorithm with potential field guidance mechanism (PBFO), which calculates the acting force between bacteria with the help of the guidance mechanism in the artificial potential field algorithm, and gives the determined swimming direction of bacteria in the convergence operation according to the gravitational relationship between bacteria. After the bacteria finish swimming, the local dimension update operation is designed to update the bacteria after the swimming, so that the bacteria can maintain a better merit-seeking ability even after the dimension increase. A double Gaussian function is introduced in the *elimination and dispersal* operation to re-initialize individual positions to further strengthen the diversity of the population. Finally, the performance of PBFO is tested by the benchmark function, and the test results are compared with Particle swarm optimization (PSO), Gravitational search algorithm (GSA), PBFO derivative algorithm and other improved algorithms, and the experimental results The experimental results show that PBFO can obtain better optimal solutions and has a faster convergence speed.

2.Related works

2.1 Basic bacterial foraging optimization algorithm

BFO consists of three principal parts: 1) *chemotaxis*, 2) *reproduction*, and 3) *elimination and dispersal* [13], and a brief introduction of each part is given as follows

2.1.1 *Chemotaxis*

In biology, the *chemotaxis* is a process of bacteria movement for gaining the nutrient. One of the E-coli bacterium properties is that, it can move in two diverse ways, swimming and tumbling. In swimming, the bacterium

swims in the same direction to search for nutrient, and in tumbling, bacterium changes the direction to another direction. Assume $P_i(j,k,l)$ shows the current position of ith bacterium, jth chemotaxis step, kth reproduction step, and elimination and dispersal step, then the position of bacterium in the next chemotaxis step by tumbling will be as:

$$\phi(i) = \frac{\Delta}{\sqrt{\Delta \times \Delta^T}} \quad (1)$$

$$P_i(j+1,k,l) = P_i(j,k,l) + C(i) \times \phi(i) \quad (2)$$

where, $C(i)$ shows the number of steps, $\phi(i)$ represents the direction of bacterial chemotaxis direction that is specified by the tumbling of ith bacterium and Δ indicates a vector of random directions in population size with continuous values between [-1, 1].

2.1.2 *Reproduction*

In the *reproduction* part, those bacteria that have enough nutrient will be reproduced and others will be eliminated. To do so, a health status of each bacterium is calculated. The health status is the sum of step fitness during its life and can be defined as:

$$J^i_{health} = \sum_{j=1}^{N_c} J(i,j,k,l) \quad (3)$$

where, N_c is total number of chemotaxis steps, and J_{health} is calculated for ith bacterium, jth chemotaxis step, kth reproduction step, and lth elimination and dispersal step. Eventually, the health status of all bacteria will be sorted in ascending order and the first half of the bacteria reproduce and surrogate into second half of the bacteria based on their top health status. In other words, the smaller J_{health} values related to healthier bacteria. Thus, bacteria with smaller health values have more chance to survive. This process not only keeps the population constant, but also the healthier bacteria continue to next generation.

2.1.3. *Elimination and dispersal*

When the density of bacteria getting high in a small area, the temperature of this area getting high and may not be enough nutrient for all bacteria as well. Thus, in this part, the population of bacteria may randomly change their positions to avoid these flaws. Elimination and dispersal event relocates the bacteria in different environments to avoid the bacteria death and local optimum solution(s).

In this operation, a random number *rand* is generated for each bacterial individual. If the value of the bacteria is less than the *elimination and dispersal* threshold P_{ed}, the bacterium is migrated, and the random initialization position is re-randomized in the entire solution space.

The algorithm flow is shown in Figure 1.

2.2 Artificial potential field method

The artificial potential field method (APF) [14] was proposed by Oussama Khatib and was first applied to mobile robot path planning. The algorithm introduces the concept of a generalized potential field and constructs an artificial abstract potential field. The target point region generates a gravitational potential field and the obstacles generate a repulsive potential field. The artificial potential field is generally described by an electrostatic field model, which is illustrated in Figure2. Positive and negative charges produce an electric field, which produces repulsive forces on homogeneous charges and gravitational forces on heterogeneous charges. The charge is subjected to the superposition of the electric field, the motion occurs to produce current. In analogy to the electrostatic field, the artificial potential field is a virtual field generated and superimposed by the target and the obstacle respectively, and its potential field exerts a gravitational repulsive effect on the robot. After the robot is subjected to the potential

field, it starts to move to generate a planned path.

An electric charge is subjected to an electric field force in an electric field. If more than one charge acts at the same time, follow the rules of vector operation to calculate the magnitude and direction of the electric field force. Analogously to the electric field force, the target area and the obstacle will also generate potential field force in the potential field. The negative gradient direction is the direction where the function value decreases fastest. The gravitational force generated by the gravitational potential field on the robot is defined as the negative gradient of the gravitational potential field, and the repulsive force generated by the repulsive potential field on the robot is the negative gradient of the repulsive potential field. The robot follows the rules of vector operation to calculate the magnitude and direction of the force on the potential field. The target area is represented as the gravitational force on the robot, and the obstacle is represented as the repulsive force on the robot. The repulsive force generated by the obstacles and the gravitational force generated by the target location are superimposed, and their combined force is the total force on the robot in the artificial potential field. Under the force of the potential field, the robot moves to produce a planned path.

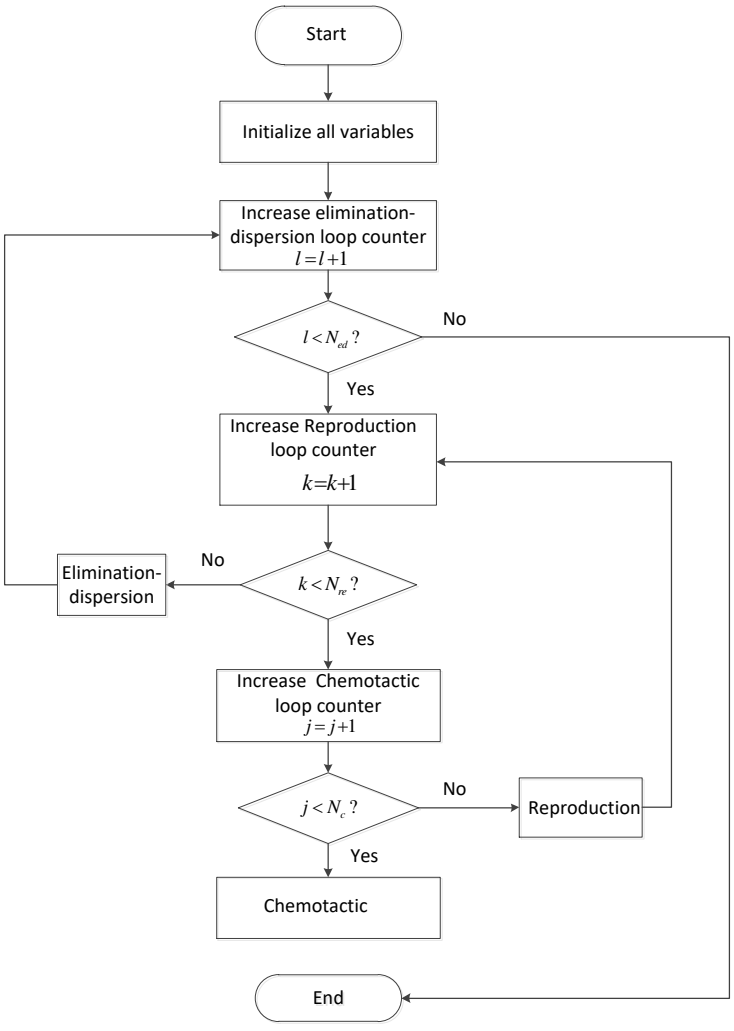

Fig. 1 Flow chart of bacterial foraging algorithm

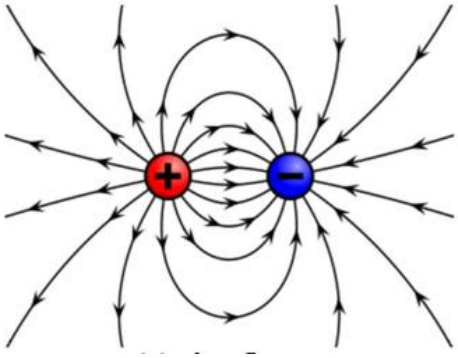

Fig.2 Electrostatic potential field model

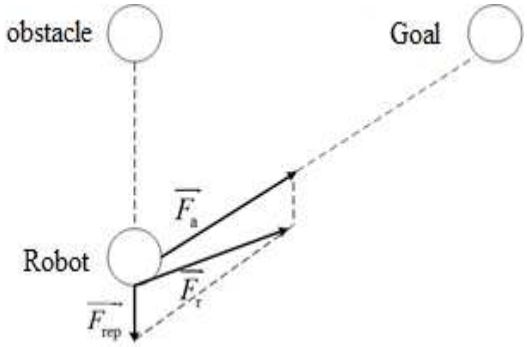

Fig.3 Artificial potential field force schematic

The force analysis of a robot in an artificial potential field is shown in Figure3. The artificial potential field method creates a potential field in the robot's motion space. The repulsive force of the target on the robot is monotonically decreasing as the distance between the robot and the obstacle increases, and the attractive force of the target on the robot is monotonically increasing as the distance from the target increases. At the location, the robot is subjected to the attractive potential field and moves toward the target area, while it is subjected to the repulsive potential field generated in the obstacle space and moves away from the obstacle. Finally, the robot moves in the direction of the combined force.

$$U_{\mathrm{a}}=\begin{cases}\frac{1}{2}k_{\mathrm{p}}\|z-z_{\mathrm{g}}\|^2 & \text{if } \|z-z_{\mathrm{g}}\|>r_{\mathrm{g}}\\ \frac{1}{2}k_{\mathrm{p}}(r_{\mathrm{g}}\|z-z_{\mathrm{g}}\|^2-r_{\mathrm{g}}^2) & \text{if } \|z-z_{\mathrm{g}}\|\le r_{\mathrm{g}}\end{cases} \tag{4}$$

$$\overrightarrow{F_{\mathrm{a}}}=\begin{cases}-k_{\mathrm{p}}\|z-z_{\mathrm{g}}\| & \text{if } \|z-z_{\mathrm{g}}\|>r_{\mathrm{g}}\\ -k_{\mathrm{p}}r_{\mathrm{g}}\dfrac{\|z-z_{\mathrm{g}}\|}{d(z,z_{\mathrm{g}})} & \text{if } \|z-z_{\mathrm{g}}\|\le r_{\mathrm{g}}\end{cases} \tag{5}$$

where, U_{a} denotes the gravitational field, r_{g} is the radius of the target area, z_{g} is the target point, and k_{p} is the gravitational gain coefficient.

$$d^*_{\mathrm{nearest}}=\arg\min_{z'\in Z_{\mathrm{obs}}}\|z-z'\| \tag{6}$$

$$U_{\mathrm{rep}}=\begin{cases}\frac{1}{2}k_{\mathrm{r}}(\frac{1}{d^*_{\mathrm{nearest}}}-\frac{1}{d^*_{\mathrm{obs}}})^2 & \text{if } d^*_{\mathrm{nearest}}\le d^*_{\mathrm{obs}}\\ 0 & \text{if } d^*_{\mathrm{nearest}}>d^*_{\mathrm{obs}}\end{cases} \tag{7}$$

$$U_{\mathrm{r}}=U_{\mathrm{a}}+U_{\mathrm{rep}}\qquad \overrightarrow{F_{\mathrm{r}}}=\overrightarrow{F_{\mathrm{a}}}+\overrightarrow{F_{\mathrm{rep}}} \tag{8}$$

where d^*_{nearest} is the distance of the robot from the nearest obstacle, and d^*_{obs} is a set constant. U_{rep} denotes the repulsive field. k_{r} is the gain factor of the obstacle repulsion. U_{r} denotes the combined force field.

3. Bacterial foraging algorithm with potential field guidance mechanism

3.1 Improvement of Chemotaxis operation

The basic bacterial foraging algorithm updates the position by the swimming of bacteria, randomly initializes the swimming direction of bacteria to search the whole solution space, and if the adaptation is better in the swimming direction, the bacteria continue to swim in that direction until they swim to the maximum number of steps; otherwise the bacteria rotate in the same place. By this search strategy makes the bacterial foraging algorithm have a strong local search ability. However, randomly generating the direction of bacterial swimming will definitely lead to a higher number of swims for the whole algorithm to obtain a better solution,

which slows down the convergence speed of the algorithm.

In order to accelerate the convergence speed of the bacterial foraging algorithm, the potential field guidance mechanism in the artificial potential field algorithm is introduced in this paper. The attraction force between bacteria and the acceleration generated between them are calculated, and the direction of the generated acceleration is used to replace the random direction of the individual bacteria during the swimming process. Under the effect of the attractive force, bacteria move toward the individual with higher inertial mass, and the bacteria with higher inertial mass occupy the position with better fitness. The attraction between individuals is used to share the better adapted position of bacteria, and the local optimization is performed according to the original swimming operation of the bacterial foraging algorithm. This operation accelerates the convergence speed of the algorithm without destroying the local mining ability of the original algorithm.

The movement of individual bacteria in the search space in the bacterial foraging algorithm with a potential field guidance mechanism and the basic bacterial foraging algorithm is illustrated by the process of the bacterial foraging algorithm and the gravitational bacterial foraging algorithm for the function finding. From Fig.4, it can be seen that in the absence of the potential field guidance mechanism, bacteria B2 are in a random wandering state. In Fig.5, since the direction of better adaptation can be transmitted by the attraction between individuals, bacteria B2, after being influenced by the attraction of bacteria B4 and repulse force derived from B1 and B3 in the population (the red line in Fig.5 indicates the combined direction of attraction of bacteria B2 by other bacteria), has a tendency to move in the direction of the current position of the optimal individual B4 in the population. The specific improvement process of the convergence operation is given. The size of the bacterial population is N and each bacterium has a D-dimensional space for finding the optimum. The tendency is to move towards the position of the former population optimum individual B4. The specific improvement process of the convergence operation is given. The size of the bacterial population is N, and each bacterium has a D-dimensional optimization space. The initial position of the bacteria is $P_i^d(j,k,l)$, which is the position of the bacteria i after j[th] chemotaxis, k[th] reproduction and l[th] elimination and dispersal, and the current fitness of the bacteria can be calculated as

$$J_i(j,k,l) = fitness(P_i^1(j,k,l), P_i^2(j,k,l), \cdots, P_i^D(j,k,l))$$

$$d \in \{1,2,\ldots,D\} \quad (9)$$

The definition of acting force $F_{io}^d(j,k,l)$ between bacteria i and o is shown in formula (10):

$$F_{io}^d(j,k,l) = \pm\lambda \left\| P_i^d(j,k,l) - P_o^d(j,k,l) \right\| \quad (10)$$

λ is the gain coefficient, when it takes a positive sign, $F_{io}^d(j,k,l)$ represents gravitational force, when it takes a negative sign, $F_{io}^d(j,k,l)$ represents repulsive force.

Considering the effect of bacteria on each other, define the formula for calculating the inertial mass of bacteria in the current population

$$m_i(j,k,l) = \frac{J_i(j,k,l) - worst(j,k,l)}{best(j,k,l) - worst(j,k,l)} \quad (11)$$

$$M_i(j,k,l) = \frac{m_i(j,k,l)}{\sum_{i=1}^{N} m_i(j,k,l)} \quad (12)$$

Where: $best(j,k,l)$ is the individual with

the best fitness in the current population and $worst(j,k,l)$ is the individual with the worst fitness in the current population.

$$\begin{aligned} best(j,k,l) &= \min\{J_1(j,k,l), J_2(j,k,l), \cdots, J_N(j,k,l)\} \\ worst(j,k,l) &= \max\{J_1(j,k,l), J_2(j,k,l), \cdots, J_N(j,k,l)\} \end{aligned} \quad (13)$$

Therefore, λ can be calculated by the formula (14) and acting force $F_{io1}^d(j,k,l)$ between bacteria *i* and *o* is shown in formula (15):

$$\lambda = M_i(j,k,l)M_o(j,k,l) \quad (14)$$

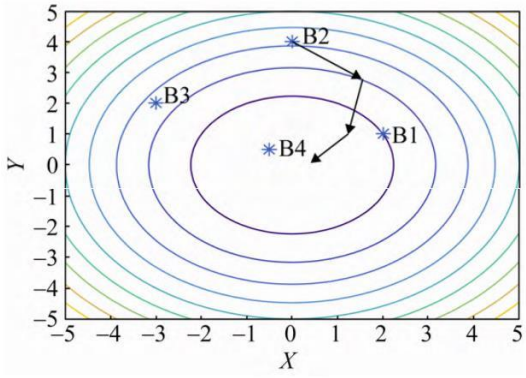

Fig. 4 BFO algorithm bacterial swimming

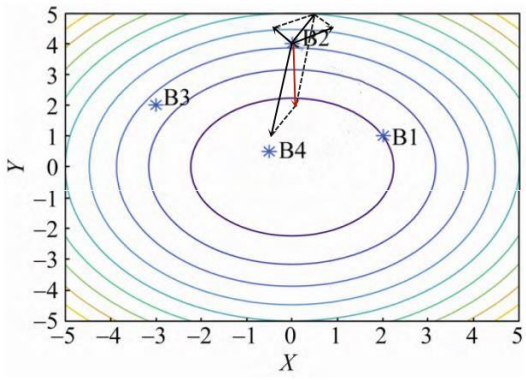

Fig. 5 PBFO algorithm bacterial swimming

$$F_{io}^d(j,k,l) = \frac{-M_i(j,k,l)M_o(j,k,l)(P_i^d(j,k,l) - P_o^d(j,k,l))}{d_{io}(j,k,l) + \mu} \quad (15)$$

where $d_{io}(j,k,l)$ is Euclidean distance between bacterium *i* and bacterium *o*; μ is a very small constant. With the formula for the acting force between two bacteria, the sum of the acting forces $F_i^d(j,k,l)$)of the d[th] dimension of bacteria *i* by other bacteria in the current population can be calculated as equation (16).

$$F_i^d(j,k,l) = \sum_{o\neq i, o=1}^{N} rand \times F_{io}^d(j,k,l) \quad (16)$$

Further the acceleration of bacteria *i* in the d[th] dimension can be obtained Equation.

$$a_i^d = \frac{F_i^d(j,k,l)}{M_i(j,k,l)} \quad (17)$$

A simpler acceleration can be obtained by combining Equations. (15) to (17)

$$a_i^d(j,k,l) = \sum_{o\neq i, o=1}^{N} \left(-\frac{M_o(j,k,l)}{R_{io}} \times [rand \times (P_i^d(j,k,l) - P_o^d(j,k,l)) \right) \quad (18)$$

$a_i(j,k,l) = (a_i^1(j,k,l), a_i^2(j,k,l), \cdots, a_i^D(j,k,l))$ is used instead of the original random direction $\phi(i)$,The location update formula of bacteria is shown in formula (19).

$$P_i^d(j+1,k,l) = P_i^d(j,k,l) + C(i) \times a_i^d \quad (19)$$

3.2 Local dimension update

By using potential field guidance in the bacterial foraging algorithm, the overall swimming trend of the bacterial population starts to move toward the optimal solution. However, as the number of iterations increases, the diversity of the bacterial population keeps decreasing, and the gravitational interaction between bacteria will only be influenced by the optimal bacterial individuals in the population, and thus the algorithm tends to fall into the local optimum. When solving the high-dimensional multi-peak function, some dimensions will move toward the optimal solution in the process of moving the population toward the optimal solution, but some dimensions will move in the opposite direction, which is called "two steps forward, one step back" [15], which also makes some high-dimensional multi-peak functions difficult to optimize. In order not to

completely destroy the optimal solution obtained by the potential field guidance mechanism search and to consider the phenomenon of "two steps forward, one step back" in the optimization of high-dimensional multi-peak functions, we perform a local dimensional update of the current bacteria after they have swum. The update process is as follows: (1) Randomly select the M^{th} dimension on the current bacteria as the starting position for the local dimension update, in order to ensure that the dimension length of the local update is L, the starting point P should satisfy M≤D-L+1. (2) Replace the original position vector P(j,k,l) of bacteria *i* from dimension P to P+L-1 with the dimension value of the corresponding position of the vector $P_{best}(j,k,l)$-$P_{worst}(j,k,l)$, where $P_{best}(j,k,l)$ is the position of the best adapted bacteria in the current population, and $P_{worst}(j,k,l)$ is the position of the worst adapted bacteria in the current population. $P_{new}(j,k,l)$ is the new position vector. (3) If the newly generated $P_{new}(j,k,l)$ has a better fitness value than P (j,k,l), then $P_{new}(j,k,l)$ is used to replace P(j,k,l), otherwise, the bacterial position P(j,k,l) of *i* is maintained unchanged.

3.3 Improvement of *elimination and dispersal operation*

In order to maintain the diversity of the population and help individuals jump out of the local optimal value, the bacterial foraging algorithm randomly assigns an *elimination and dispersal* probability Rand to each bacterium, and sets the *elimination and dispersal* threshold as P_{ed}. When the *elimination and dispersal* probability *rand* of a bacterium in the population is less than the *elimination and dispersal* threshold, it is considered that the bacterium has died, need to reinitialize the location of the currently dead bacteria in the optimization space. However, this *elimination and dispersal* method may not make the individual near the local optimal solution jump out of the local extreme value. Random initialization also has a certain probability, resulting in the initialization solution near its original position, which is not conducive to the increase of bacterial population diversity. In order to avoid this phenomenon, this paper introduces the double Gaussian function to make the newly generated bacterial position value better and set it away from the original position [16-17]. Take the bacterial individual P to undergo *elimination and dispersal* operation as an example to illustrate the difference between the initialization of double Gaussian function and random initialization of bacterial position. The original position of point P is p(1,1) and the Elimination and dispersal range is [– 8,8]. Observe the difference between the position after 100 times of random *elimination and dispersal*. In Figure 6, the red circle represents the point P to be migrated, the blue triangle represents the individual position of bacteria after random migration, and the green asterisk represents the individual position of bacteria after migration by using double Gaussian function. It can be seen that there are very few individuals near the original bacterial P position, and most bacterial positions are far away from bacterial P; With random initialization, bacteria are distributed in the whole solution space, and many individuals are initialized not far from the bacterial P position, which is not conducive to the increase of population diversity. Using double Gaussian function to complete the *elimination and dispersal* operation, the population has higher diversity and can enhance the global optimization ability of the algorithm.

The newly generated bacterial location formula is shown in Equation (20). up^d and low^d are the upper and lower bounds of the dimension of the solution space, respectively; e is the random number of the standard normal distribution; σ_1 and σ_2 are the variance

of the Gaussian function, between [0,1].

$$P_i^d(j,k,l+1) = (low^d + \sigma_1 \times e) \times H(r_1 - r) + (up^d + \sigma_2 \times e) \times H(r_1 - r) \quad (20)$$

$$r = \frac{P_i^d(j,k,l) - low^d}{up^d - low^d} \quad (21)$$

$H(x)$ is a step function, which is defined as shown in Equation (22).

$$H(x) = \begin{cases} 0 & x < 0 \\ 0.5 & x = 0 \\ 1 & x > 0 \end{cases} \quad (22)$$

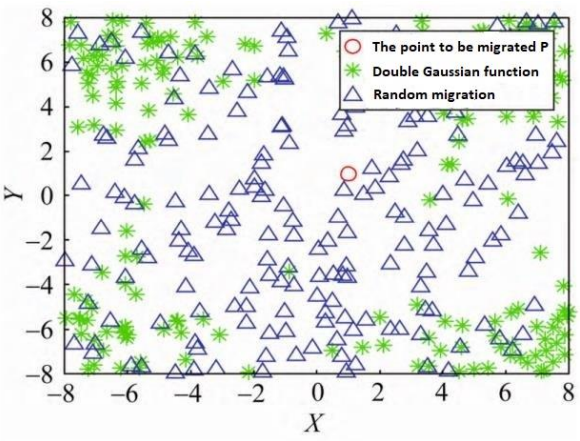

Fig.6 Random migration and double Gaussian migration

3.4 The process of proposed algorithm

The process of PBFO algorithm is as follows:

Step 1: Initialize the population and set the parameters: chemotaxis time N_c, chemotaxis swimming step N_s, reproduction time N_{re}, elimination and dispersal time N_{ed}, the probability of elimination P_{ed}, population size N, local dimension update length L;

Step 2: Adjust bacterial step size and choose a guided direction by formula (18);

Step 3: Move one step on the selected direction by formula (19) and calculate the new fitness value by benchmark function;

Step 4: Determine whether to continue chemotaxis. If the fitness value of new location is smaller that of initial location, turn back to Step 2 until the algorithm reaches the number of flipping or the fitness value cannot change. If the original value is smaller than the fitness value of new location, go to the next step;

Step 5: Local dimension update operations according to the description of Section 3.2;

Step 6: Choose half bacteria to reproduce and kill unselected bacteria;

Step 7: Adjust the possibility of elimination and dispersal P_{ed} by fitness value and produce a random number. If the random number is less than adjusted probability Ped, bacterium goes to Step 8. Otherwise, bacterium goes to Step 9;

Step 8: Bacterial *i* Re-initialize its position according to the formula (20) in Section 3.3.

Step 9: Turn back to Step 1 until the algorithm reaches the number of iteration or get the optimal value.

Therefore, the PBFO Algorithm flow chart is showed in Tab.1, which is depicted by the pseudo code.

Tab.1 The pseudo code of PBFO algorithm

PBFO Algorithm

```
Initialization parameters: N, N_r N_C, N_ed, N_s, P_ed, L
for l = 1: N_ed//Elimination and dispersal cycle start
  for k = 1: N_re // Reproduction cycle start
    for J = 1: N_c // Chemotaxis cycle start
      Calculate the adaptation of bacteria in the current population
      The acceleration of each bacterium is calculated from the formula (18) in 3.1.
        for i = 1: N
            Keep bacteria's adaptivity value J_last
            Update the location of bacteria i by formula (19) in 3.1
                m = 0; // Tour operation start
                While m <Ns
                  m = m + 1;
                  if J (i, j, k, l) <J_last
                    J_last = J (i, j, k, l);
              The formula (19) in 3.1 continues to update the location of bacteria
                  else
                    m = Ns;
                end
        end // Tour operation end
      Local dimension update operations according to the description of Section 3.2
      end
   end // End of Chemotaxis
   Copy the position of the first N/2 bacteria to the next N/2 bacteria ascending order
          according to the value of the objective function// Reproduction operation
  end // End of Reproduction operation
  for i = 1: n// Elimination and dispersal operation
   if rand <P_ed
   Bacterial i Re-initialize its position according to the formula (20) in Section 3.3.
   else
     Keep the position of the original bacteria constant
  end
end
end // End of Elimination and dispersal
```

4.Results and discussion

4.1 Experimental environment and parameter settings

In this paper, 10 standard functions are selected to test the PBFO algorithm, and the test results are compared with the standard bacterial foraging algorithm (BFO), the standard particle swarm algorithm (PSO), the gravitational search algorithm (GSA) and some recently proposed improved intelligent algorithms including LAQBFO, QBFO, NAQBFO, and APSO-IV. The experimental data of LAQBFO, QBFO and NAQBFO experimental data are from the literature [12] and APSO-IV experimental data are from the literature [8].

The dimensions of these 10 test functions are all set to 30 dimensions and all of the optimal values are 0, as shown in Table 2. All simulation experiments were performed on a machine with windows10, Intel Core i7 CPU, main frequency 1.6 GHZ, and the software used for the experiments was Matlab2016a.

For the standard BFO and PBFO algorithm, the main parameters are set as follows: number of bacteria N=50, Number of *elimination and dispersal* N_{ed}=4, number of *reproduction* N_{re}=3, number of *chemotaxis*

N_c=100, number of swims m=5, probability of *elimination and dispersal* P_{ed}=0.25, swim step C=0.01(ub-lb), where ub denotes the upper bound of the dimension of the optimization function, lb denotes the lower bound of the dimension of the optimization function, and for The PBFO algorithm also requires a separate local dimensional update parameter The value of L is set as follows: L=3. For the gravitational search algorithm, the parameters are set as follows: number of population particles N=50, parameter G=100, α =20, iter_max=1200. For the particle swarm algorithm, the parameters are set as follows: number of particles N=50, learning factor c1=2, c2=2, weight factor w1=0.9, w2=0.1, iter_max=1200.

Tab. 2 Test function

Function	Dim	Range	Optimal value
$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	[–100,100]	0
$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[–10,10]	0
$f_3(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	30	[–100,100]	0
$f_4(x)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	30	[–100,100]	0
$f_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	[–30,30]	0
$f_6(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	30	[–1.28,1.28]	0
$f_7(x)=1+\sum_{i=1}^{n}(\frac{x_i^2}{4000})-\prod_{i=1}^{n}(\cos(\frac{x_i}{\sqrt{i}}))$	30	[–600,600]	0
$f_8(x)=10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i))$	30	[–5.12,5.12]	0
$f_9(x)=-20\exp(0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	30	[–32,32]	0
$f_{10}(x)=418.9829n+\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	[–500,500]	0

4.2 Experimental results

The results in Tab.2 to Tab.4 are the best values obtained after the algorithm runs 30 times, the values in parentheses are the average of the results of 30 experiments for each benchmark function.

From the experimental results in Table 3, we can see that the PBFO algorithm on the 10 benchmark functions is basically close to the global optimal solution of the objective function. In terms of convergence accuracy, the convergence accuracy of the PBFO algorithm is significantly higher than that of the base BFO for all benchmark functions. Contrast the PSO algorithm, although the convergence accuracy is weak in the functionality F1, F2, F7, F9, the PBFO algorithm is solved on other functions, especially in F3, F8, F10 function, which is significantly better than PSO. Compared with the gravitational search algorithm (GSA), which only solves the F1, F8, and F10 functions, the accuracy of PBFO is significantly better than that of PSO. Compared with gravity search algorithm (GSA), GSA is only better than PBFO in the optimization results of F1, F2, F4 and F9 functions, and PBFO algorithm is ahead of GSA in solving other functions; Moreover, GSA algorithm can't optimize F3 and F10 functions, but PBFO algorithm obtains better

results in optimizing the above two functions. Compared with LAQBFO, QBFO, NAQBFO and APSO-IV algorithms, PBFO is lower than APSO-IV only in the solution accuracy of F4 function. In other functions, PBFO algorithm has higher optimization accuracy

Tab. 3 Comparison of PBFO algorithm, basic algorithm and LAQBFO, QBFO, NAQBFO, APSO-IV algorithm

Function	BFO	GSA	PSO	LAQBFO	QBFO	NAQBFO	APSO-IV	PBFO
F1	2.37E+01 (2.82E+0)	1.25E-16 (2.33E-16)	3.19E-13 (8.80E-12)	--	--	--	1.3116E-02	2.38E-06 (1.99E-05)
F2	4.51E+00 (9.61E+00)	3.53E-08 (4.37E-08)	5.21E-07 (1.16E-05)	--	--	--	2.1745E-02	5.83E-04 (1.09E-03)
F3	2.46E+02 (2.84E+02)	3.01E+02 (4.47E+02)	4.78E+00 (8.32E+00)	--	--	--	9.8422E-02	6.58E-02 (1.13E-01)
F4	5.32E+00 (6.00E+00)	7.87E-02 (6.18E-01)	2.96E-01 (3.95E-01)	--	--	--	9.0489E-03	1.12E-01 (6.11E-01)
F5	4.27E+02 (4.37E+02)	2.73E+01 (2.75E+01)	1.28E+01 (5.46E+01)	2.64E+01	1.21E+02	2.61E+01	--	1.15E+00 (6.15E+00)
F6	4.47E-01 (5.61E-01)	2.32E-03 (8.10E-03)	3.44E-02 (4.48E-02)	--	--	--	8.03E-01	2.17E-03 (2.83E-03)
F7	1.27E+00 (1.29E+00)	1.62E+01 (1.99E+01)	3.21E-14 (1.23E-02)	1.17E-01	1.13E+00	1.03E-01	9.30E+00	2.49E-06 (2.60E-02)
F8	1.21E+02 (1.35E+02)	5.97E+00 (1.05E+01)	3.58E+01 (4.69E+01)	3.54E+01	1.02E+02	2.10E+01	--	4.59E-02 (2.49E-01)
F9	1.66E+01 (1.73E+01)	7.05E-09 (8.05E-09)	9.55E-07 (2.72E-06)	1.60E+00	1.11E+01	1.59E-01	6.61E+00	7.38E-04 (4.58E-01)
F10	4.06E+03 (4.25E+03)	8.96E+03 (9.93E+03)	5.73E+03 (6.25E+03)	9.27E+03	1.33E+04	1.74E+03	--	1.11E-03 (1.01E-02)

Tab. 4 Comparison of PBFO and derivative algorithm

Function	BFO	PBFO_1	PBFO_2
F1	2.37E+01 (2.82E+01)	7.62E-05 (1.17E-04)	1.57E+01 (1.80E+01)
F2	4.51E+00 (9.61E+00)	3.48E-02 (9.52E+04)	1.81E+00 (1.98E+00)
F3	2.46E+02 (2.84E+02)	8.97E+02 (2.09E+03)	1.04E+02 (1.33E+02)
F4	5.32E+00 (6.00E+00)	2.93E+00 (3.36E+00)	3.69E+00 (3.87E+00)
F5	4.27E+02 (4.37E+02)	2.34E-01 (2.52E+01)	2.25E+02 (2.63E+02)
F6	4.47E-01 (5.61E-01)	1.73E-02 (2.03E-02)	2.43E-01 (3.35E-01)
F7	1.27E+00 (1.29E+00)	2.71E+02 (3.31E+02)	1.15E+00 (1.17E+00)
F8	1.21E+02 (1.35E+02)	4.99E+01 (1.01E+02)	2.02E+01 (2.19E+01)
F9	1.66E+01 (1.73E+01)	1.88E+01 (1.91E+01)	2.65E+00 (2.84E+00)
F10	4.06E+03 (4.25E+03)	4.58E+03 (5.42E+03)	5.47E+01 (5.83E+01)

than these four improved intelligent algorithms.

To further validate the performance of the improved algorithm, the dimensionality of the test function in Table 1 is increased to 100 dimensions to test the performance of PBFO algorithm on high-dimensional functions and compare the experimental results with the performance of the basic BFO, ABC, CSO, HJCSO, BAABC, and SGHS algorithms. The experimental results of the ABC algorithm were obtained from the literature [18], CSO and HJCSO algorithms are from the literature [19], BAABC algorithm is from the literature [20], SGHS algorithm is from the literature [21]. The basic BFO algorithm and the parameter settings of PBFO algorithm are the same as those in this paper when solving 30-dimensional functions except the number of elimination and dispersal operations is increased to Ned=10.

The experimental results are shown in Tab.5. From the experimental results, it can be seen that PBFO solves 100-dimensional functions, tested on 9 functions. Only for the F7 function, the results are worse than HJCSO and BAABC, and the results of the other functions are better than the remaining 6 comparison algorithms. In order to test the effectiveness of different improvement strategies in combination with the basic BFO and its derived 2 algorithms, PBFO_1 and PBFO_2, are compared in this paper. PBFO_1 is the basic BFO that introduces only the improved strategy in Section 3.1, and PBFO_2 introduces only the improved strategy in Section 3.2. The parameters of both algorithms are set as above.

Comparing the basic BFO with PBFO_1, tested in function F1, F2 and F4, the fitness value of the PBFO_1 algorithm is ahead of that of the BFO, which is PBFO_1 transmits the information about the optimal position of bacteria in the population through the potential field guidance mechanism, which makes the whole population maintain the tendency to move toward the current optimal solution. In contrast, there is no potential field guidance mechanism in BFO, and in its Chemotaxis operation, random swimming is performed, making many swimming steps are invalid. In the test of functions F5 to F10, the optimal solution in F7, F9, and F10 is superior to PBFO_1, which is due to the local extrema in the high-dimensional multi-peak function. There are more value points, resulting in the dispersion of the solution, and the joint force derived in a gravitational way. The direction of attractive force deviates too much from the direction of the global optimal solution, which makes the attractive direction interferes with the movement of the bacteria toward the global optimal solution instead. The data of standard BFO were compared with PBFO_2, and the gravitational direction of the high-dimensional single-peak function F1~F4 was found to be too far from the global optimal solution. The solution of the high-dimensional single-peak functions F1 to F4 shows that the bacterial foraging algorithm, which only use local dimensional updates, has an improved ability to find the optimal solution of the function, but the effect is not very obvious. However, in the high dimensional multi-peaked functions, it can be seen from Table 3 that solving the functions F8~F10, the improvement effect is very obvious, and all of them have at least one order of magnitude in solution accuracy. The PBFO algorithm is obtained by combining the improvement strategies in Sections 3.1 and 3.2, and it can be seen from Tab.4 that PBFO obtains significantly better results than BFO.

Tab. 5 Optimization results of test function in 100 dimensions

Function	BFO	ABC	CSO	HJCSO	BAABC	SGHS	PBFO
F1	1.60E+02	--	8.34E+02	2.03E+00	2.90E-04	1.30E+00	3.35E-05
F2	2.92E+02	--	--	--	--	6.33E+00	3.35E-03
F3	1.64E+04	--	--	--	--	2.84E+04	4.82E+00
F5	2.59E+03	6.80E+09	3.47E+00	9.42E-01	2.20E+00	--	3.23E-01
F6	3.68E+00	--	--	--	4.20E-02	--	2.27E-02
F7	2.52E+00	3.40E+01	4.21E-01	8.17E-05	1.30E-05	5.14E-01	5.26E-02
F8	5.60E+02	2.24E+02	4.23E+02	5.93E-01		4.84E+01	2.32E-03
F9	1.95E+01	1.63E+01	6.56E+00	8.82E-02	6.10E-03	1.71E-01	4.91E-03
F10	1.60E+04	2.55E+00	--	--	--	--	1.32E-03

4.3 Convergence speed analysis

To demonstrate that the PBFO algorithm converges faster than that of the classical bacterial foraging algorithm, the convergence plots of BFO and PABFO on all functions are made in this paper, as shown in Fig.7 to Fig.16.

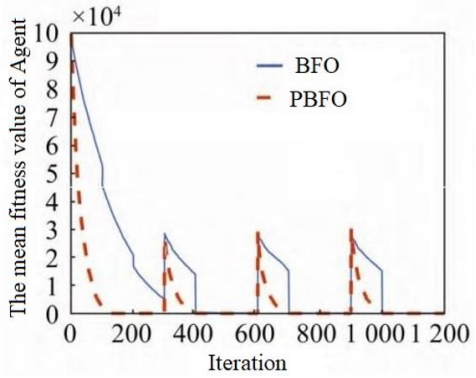

Fig. 7 PBFO and BFO convergence graph for F1

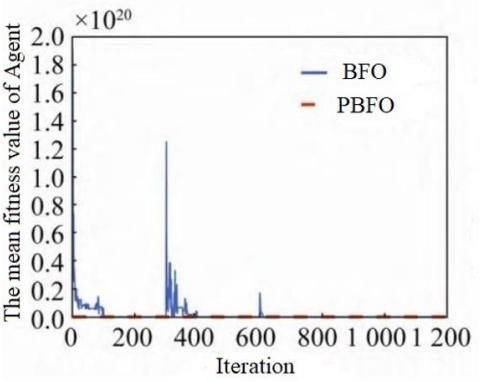

Fig. 8 PBFO and BFO convergence graph for F2

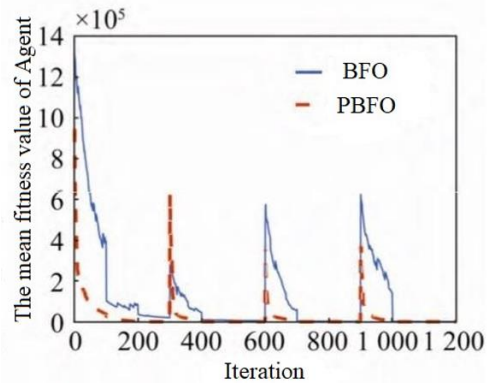

Fig. 9 PBFO and BFO convergence graph for F3

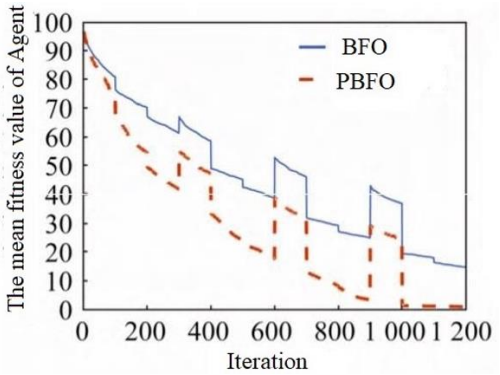

Fig. 10 PBFO and BFO convergence graph for F4

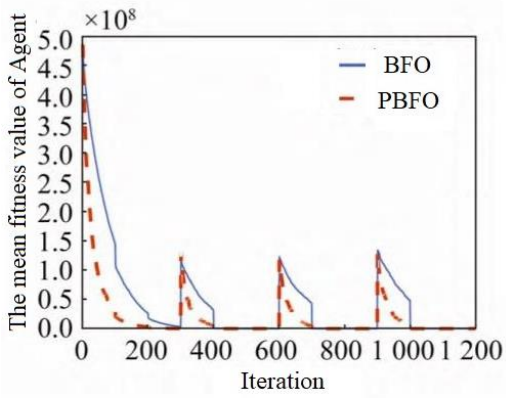

Fig. 11 PBFO and BFO convergence graph for F5

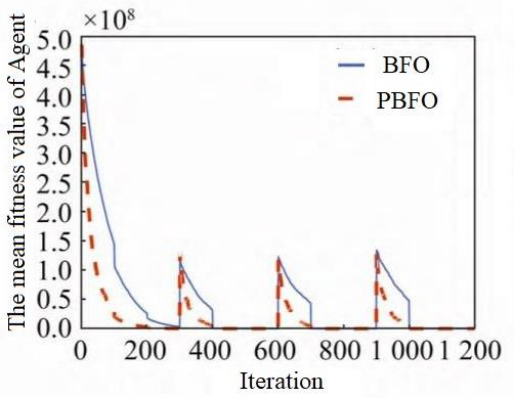

Fig. 12 PBFO and BFO convergence graph for F6

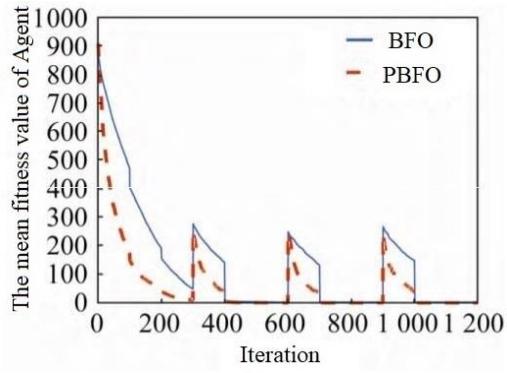

Fig. 13 PBFO and BFO convergence graph for F7

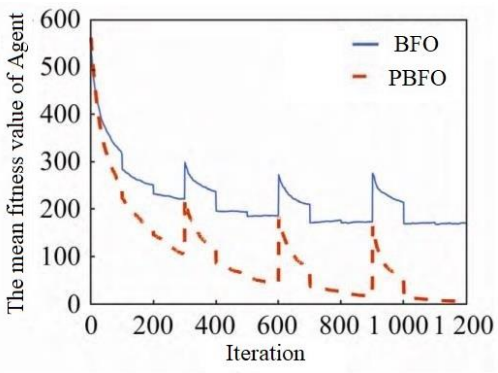

Fig. 14 PBFO and BFO convergence graph for F8

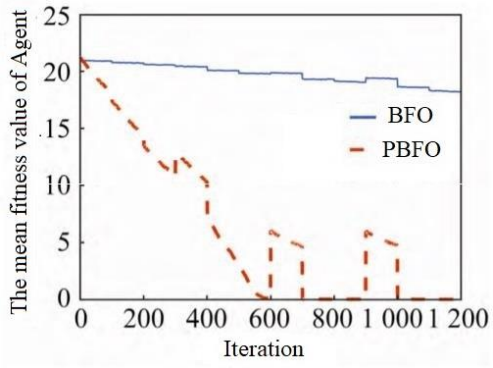

Fig. 15 PBFO and BFO convergence graph for F9

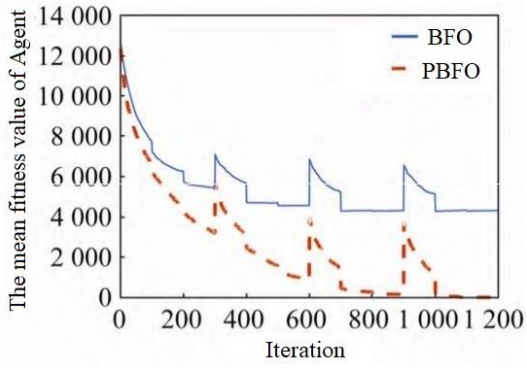

Fig. 16 PBFO and BFO convergence graph for F10

Comparing the convergence effect of the two algorithms for solving the benchmark functions, it is obvious that the convergence speed of PBFO is significantly faster than that of the standard BFO, which is mainly due to the potential field guidance mechanism in the PBFO algorithm that conveys the information of the optimal solution in the population. The convergence plots of functions F2, F4, F8, F9, F10 show that PBFO has a much higher convergence accuracy than the basic BFO, and is able to continue to explore the global optimal solution when the BFO has already converged, and the algorithm has an extremely strong mining ability. In all convergence plots, there are spikes appearing, which are due to the fact that when migrating, some individuals due to migrating and resulting in a decrease in the average fitness value of the population. The average fitness of the population was able to continue to decrease and outperform the population after the spikes were generated by using the double Gaussian function to complete the *elimination and dispersal*

operation. After the spikes were generated, the average fitness of the population was able to decrease consistently and better than that before the *elimination and dispersal* operation, indicating that the double Gaussian function approach effectively increased the population diversity and enabled the population to perform better for global search.

5.Conclusion

In this paper, by improving the basic BFO algorithm, we propose the PBFO algorithm. These improvements accelerate the convergence speed and improve the convergence accuracy of the bacterial foraging algorithm to a certain extent, and achieve better results in some multi-peak functions where optimization is difficult. The PBFO algorithm has better generalizability, but the accuracy of solving some special benchmark functions still needs to be improved. The main research point in the future is to further improve the accuracy of the algorithm on the benchmark functions and to apply the PBFO algorithm to more scenarios. The main research point is to further improve the accuracy of the algorithm on the benchmark functions and to apply the PBFO algorithm to more scenarios.

6. Acknowledgements

This work was financially supported by Gansu Province SME Innovation Fund, Development and Industrialization of Electrified Railway Contact Network Insulator Cleaning Robot (Project No.18CX6JA022)

Reference

[1] Ying C, Hua M, Liao H, et al. A Fast Bacterial Swarming Algorithm for high-dimensional function optimization[C]// Evolutionary Computation. IEEE, 2008.

[2] Tang W J, Wu Q H, Saunders J R. Bacterial foraging algorithm for dynamic environment[C]. Proceedings of IEEE Conference on Evolutionary Computation. Canada: IEEE, 2006: 4467-4473.

[3] Chu ying, Mi Hua, Ji Zhen, et al. Fast Bacterial Swarming Algorithm Based on Particle Swarm Optimization[J]. Journal of Data Acquisition & Processing, 2010, 25(4): 442-448.

[4]Mishra S. A hybrid least square-fuzzy bacteria foraging strategy for harmonic estimation[J]. IEEE Transactions on Evolutionary Computation (S1089-778X), 2005, 9(1): 61-73.

[5] Niu B, Wang J, Wang H. Bacterial-inspired algorithms for solving constrained optimization problems[J]. Neurocomputing (S0925-2312), 2015, 148: 54-62.

[6] Tan L J, Yi W J, Yang C, et al. Adaptive Structure-Redesigned-Based Bacterial Foraging Optimization[M]. Lanzhou China: Intelligent Computing Theories and Application, 2016.

[7] Liu Zhen, Sun Jinggao. An Improved Bacterial Foraging Optimization Algorithm[J]. Journal of East China University of Science and Technology, 2016, 42(2): 225-232.

[8] Zhou Wenhong, Lei Xin, Jiang Weiguo, et al. Variable probability And hybrid bacterial foraging Optimization algorithm[J]. Systems Engineering and Electronics, 2016, 38(4): 960-964.

[9] Liu Xiaolong, Zhao Kuiying. Bacteria foraging optimization algorithm based on immune algorithm[J]. Journal of Computer Applications, 2012, 32(3): 634-637.

[10] Du Pengzhen, Tang Zhenmin, Sun Yan. An Adaptive Bacterial Foraging Optimization Algorithm Mixed with Bee Colony Algorithm[J]. Computer Engineering, 2014, 40(7): 138-142.

[11] Zhang Guoyong, Wu Yonggang, Tan Yuxiang. Bacterial Foraging Optimization Algorithm with Quantum Behavior[J]. Journal of Electronics & Information Technology,

2013, 35(3): 614-621.
[12] Liu Lu, Shan Liang, Dai Yuewei, et al. Nonlinear notation angle for dynamic adaptation in quantum bacterial foraging optimization algorithm[J]. Control and Decision, 2017, 32(12): 2137-2144.
[13] Parvandeh S , Soltani P , Boroumand M , et al. A modified single and multi-objective bacteria foraging optimization for the solution of quadratic assignment problem[J]. 2016.
[14] A. H. Qureshi, Y. Ayaz, Potential functions based sampling heuristic for optimal path planning, Autonomous Robots 40 (6) (2016) 1079–1093.
[15] Jiang Jianguo, Zhou Jiawei, Zheng Yingchun, et al. Adaptive bacterial foraging Optimization algorithm[J]. Journal of Xidian University, 2015, 42(1): 75-81.
[16] Peng Junjun, Liu Yongjin. An efficient bird swarm optimization algorithm based on double Gaussian function[J]. Modern Electronics Technique, 2018, 41(23): 106-112.
[17] Xue Junjie, Wang Ying, Li Hao, et al. A smart wolf pack algorithm and its convergence analysis[J]. Control and Decision, 2016, 31(12): 2131-2139.
[18] Shi Xudong, Gao Yuelin. Hybrid algorithm based on chicken swarm optimization and artificial bee colony[J]. Journal of Hefei University of Technology (Natural Science), 2018, 41(5): 589-594.
[19] Zhang Muxue, Zhang Damin, He Ruiliang. A Chicken Swarm Algorithm Based on Hooke-Jeeves[J]. Microelectronics & Computer, 2018, 35(4): 46-52.
[20] He Guijiao, Zhou Shuliang, Feng Dongqing. Improved artificial bee algorithm for high dimensional complex optimization problems[J]. Computer Engineering and Applications, 2018, 54(12): 131-137.
[21] Pan Q K, Suganthan P N, Tasgetiren M F, et al. A self-adaptive global best harmony search algorithm for continuous optimization problems[J]. Applied Mathematics and Computation (S0096-3003), 2010, 216(3): 830-848.