IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 13, 2019, accepted July 1, 2019, date of publication July 8, 2019, date of current version August 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927327

Enhanced Bacterial Foraging Optimization Based on Progressive Exploitation Toward Local Optimum and Adaptive Raid

DONGXING WANG[1,2], XU QIAN[1], XIAOJUAN BAN[2], BOYUAN MA[2], YAN MA[1], AND ZIYI LV[1]

[1]School of Mechanical Electronic and Information Engineering, China University of Mining and Technology, Beijing 100083, China
[2]Beijing Advanced Innovation Center for Materials Genome Engineering, University of Science and Technology Beijing, Beijing 100083, China

Corresponding authors: Xu Qian (xuqiancumtb@163.com) and Xiaojuan Ban (banxj@ustb.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1001404, and in part by the National Natural Science Foundation of China under Grant 61873299 and Grant 51761135121.

ABSTRACT Since the bacterial foraging optimization algorithm (BFO) was proposed, many variants about it have been designed in order to improve the performance and applied in different fields. Even so, people are constantly probing new methods designed to enhance the performance of BFO, so as to form new variants with superior performance. As a new variant of original BFO, bacterial foraging optimization using strategies of progressive exploitation approximating local optimum and adaptive raid (BFO-DX) was proposed. On the one hand, the strategy of progressive exploration approximating local optimum (PELO) was introduced into BFO to enhance its ability of exploitation in a local space, which enables the algorithm to find the global optima better possibly. On the other hand, the strategy of the adaptive raid for the leader (ARL) was adopted to boost the speed of convergence by strengthening its exploration capacity. The numerical experiments indicates that the BFO-DX possesses better ability of finding global optima, better stability and other acceptable terms such as iteration and running time compared with classical genetic algorithm (GA), particle swarm algorithm (PSO), and conventional bacterial foraging optimization (BFO).

INDEX TERMS Bacterial foraging optimization, progressive exploitation, local optimum, adaptive raid, exploration.

I. INTRODUCTION

The metaheuristic search technology based on swarm intelligence has been increasing in popularity due to its ability to solve a variety of complex scientific and engineering problems [1]. Such technology models the social behavior of certain living creatures, where each individual is simple, has limited cognitive capability and communicates only locally, but as a whole the swarm can act in a coordinated way and yield intelligent behavior to obtain global optima, especially, information exchange among individuals and balance between local exploitation and global exploration are two crucial mechanisms in different swarm intelligence algorithms [2].

Inspired by the foraging behavior of human bacterium coli, in 2002, Passino proposed bacterial foraging optimization algorithm (BFO) [3], which is a new member in the family of bionic evolutionary algorithms. Due to its intuition and easy to understand the natural mechanism, in nearly two decades since it was proposed, much attention of researchers from different fields has been attracted, the original BFO is continually improved to be different variants in different ways and utilized in different fields, such as: In paper [4]–[6], Dasgupta and Das et al. analyzed the effect of the step-size of the adaptive mechanism on the convergence and stability of the algorithm, but their theoretical analysis was based on certain assumptions and only considered the directional operation of a single particle in one-dimensional continuous space. In paper [7], Zhou Ya-lan introduced research and application on bacteria foraging optimization algorithm. In paper [8], BFO is used to further optimize the edge weights to achieve the better localization accuracy in order to improve the performance of node localization in 3D space for wireless sensor network and the simulation results show the superiority of the new algorithm as compared to centroid method, weighted centroid and existing 3D localization algorithms.

The associate editor coordinating the review of this manuscript and approving it for publication was Huiping Li.

In paper [9], authors proposed two novel BFO algorithms to deal with feature selection problem by redefining data structure of each bacterium to establish the mapping relationship between the bacterium and the feature subset and designing an adaptive method for evaluating the importance of features. In paper [10], authors propose a solution to the menu planning problem adapting the bacterial foraging-based optimization algorithm and the results obtained were satisfactory from an expert's point of view. In paper [11], BFO was adopted to handle with the problem about high consumption of energy and energy crisis by shifting the load from on peak hours to off peak hours in demand side management, then the electric cost can be reduced and electric load can be better dispatched and managed, accordingly simulation results show that BFO is perform better than Crow Search Algorithm in reducing the total cost and peak average ratio and archived the objectives. In paper [12], a chaotic bacterial foraging optimization was integrated with Gauss mutation approach termed CBFO-FKNN, which efficiently resolved the parameter tuning issues of the FKNN(fuzzy k-nearest neighbor) and the simulation results indicated the proposed approach outperformed the other five FKNN models based on BFO, PSO(Particle Swarm Optimization), GA (Genetic Algorithm), fruit fly optimization and firefly algorithm, and will bring great convenience to the clinicians to make a better decision in the clinical diagnosis. In paper [13], BFO was utilized to evaluate the performance of Home Energy Management System in order to reduce the cost and Peak to Average Ratio and manage the power consumption. In paper [14], BFO is used to optimize the objective function about given walking and queue times in order to enable the crowd evacuation model effectively describe a real evacuation, and the results of real and simulation experiments show that it does. In paper [15], BFO was enhanced principally by relying upon breeding between PSO with BFO systems and used to optimize and set the MPF (Modulated power filters) parameters to enhance and tune the MG (micro grid) dynamic response. Taking the aviation equipment scheduled maintenance as a prototype, authors improves bacteria foraging optimization algorithm to solve the task-scheduling problem in paper [16]. On the one hand, inspired by gene mutation, the activity of bacteria is dynamically adjusted to make good bacteria more capable of action. On the other hand, a bacterial quorum sensing mechanism allows bacteria to guide their swimming routes by using their peer experience and enhance their global search capability. In paper [17], Bacteria Foraging Optimization Algorithm is adopted to help to shift the load from on-peak to off-peak hours in order to minimize electricity cost and Peak to Average Ratio. A new computational approach based on bacterial foraging optimization for functional module detection in PPI networks is presented in paper [18]. In paper [19], authors propose a hybrid approach integrating the bacterial foraging optimization algorithm in a radial basis function neural network, applied to image classification, in order to improve the classification rate and the objective function value. In paper [20], authors propose a novel bacterial algorithm based on control mechanisms and modified population updating strategies for feature selection in classification and comparison experiments indicate that the proposed BAFS outperforms other algorithms by achieving higher classification performance. In paper [21], Yang Cuicui et al. proposed a new bacterial foraging optimizer using new designed chemotaxis and conjugation strategies, and experimental results demonstrated excellent performance of the new algorithm in terms of solution quality and computational efficiency. However, there is no an omnipotent algorithm, and any of all the BFO algorithms has their own limitations and is fit be used in a certain case. Even so, people are constantly improving their algorithms in order to enhance the performance of solving the same question.

On the basis of summarizing previous studies, especially referring to paper [3] and paper [7], this paper proposes a bacterial foraging optimization using strategies of progressive exploitation approximating local optimum and adaptive raid (BFO-DX).

II. PRINCIPLE OF ORIGINAL BFO

The process about BFO mainly includes Chemotaxis, Reproduction and Elimination-Dispersal.

A. CHEMOTAXIS

Usually, bacteria tumble frequently in the poor areas or the areas where food is scarce while they swim forward in the rich areas or the areas where food is full. During the whole life cycle of bacteria, swimming and rotating alternate, so that the bacterial population looks for food rich areas as far as possible and survive in the complex living environment. In bacterial foraging optimization algorithms, this mechanism is called Chemotaxis.

Here, similar to most of swarm intelligence optimization algorithms, Eq. (1) is employed to get individual locations.

$$\mathbf{x_i} = (\mathbf{x_{i\_1}}, \mathbf{x_{i\_2}}, \ldots, \mathbf{x_{i\_d}}, \ldots, \mathbf{x_{i\_D}}) \times (\mathbf{i = 1, 2}, \ldots, \mathbf{Num}; \mathbf{d = 1, 2}, \ldots, \mathbf{D};) \quad (1)$$

where, x_i means the location of the bacterium i; x_{i_d} means the coordinate of the bacterium i on the d-th dimension, Num is the number of bacterial population; D is the number of dimension in the solution space.

Then, the dynamic locations of each bacterium in the iterations on chemotaxis, reproduction, elimination-dispersal can be got by Eq. (3).

$$\Phi(\mathbf{j}) = \mathbf{RandomSelect(1, D)} \quad (2)$$

$$\mathbf{x_i(j + 1, k, l)} = \mathbf{x_i(j, k, l)} + \mathbf{C(i)}\Phi(\mathbf{j}) \times (\mathbf{j = 1, 2}, \ldots, \mathbf{Nc}; \mathbf{k = 1, 2}, \ldots, \mathbf{Nre}; \mathbf{l = 1, 2}, \ldots, \mathbf{Ned}) \quad (3)$$

where, Φ(j) denotes the random forward direction selected by tumbling; RandomSelect(1,D) generates a D-dimensional vector whose elements are -1 or 1 randomly; x_i (j, k, l) means the location of the bacterium i in j-th chemotaxis,

k-th reproduction and l-th elimination and dispersal; C(i) denotes the step-size that bacterium i moves forward and it should be a fixed value bigger than zero(it is set at 0.001 in this paper); Num is the number of bacterial population; Nc is the maximum number of chemotaxis; Nre is the maximum number of reproduction; Ned is the maximum number of elimination-dispersal.

In the specific process of chemotaxis, firstly, a bacterium will tumble and a randomly generated direction $\Phi(j)$ can be got according to Eq. (2), and the bacterial swims in *j-th* chemotaxis according to Eq. (3); then if the fitness of the bacterium i in *j-th* chemotaxis is better than the current fitness(In this article, assume that the smaller fitness is, the better), the bacterium i moves to the new location and continues to swim according to Eq. (3) until the control parameter *n* is bigger than the argument *Ns*, otherwise, break out the loop and goes to the next chemotaxis just according to the Eq. (4).

$$\begin{cases} \textbf{go to the next chemotaxis}, \\ \qquad \textbf{fitness\_new} >= \textbf{fitness(i) or n} > \textbf{Ns} \\ \mathbf{x_i(j,k,l)} = \mathbf{x_i(j+1,k,l)} \textbf{ and swim according to Eq.(3)}, \\ \qquad \textbf{fitness\_new} < \textbf{fitness(i)} \end{cases} \tag{4}$$

where, fitness_new means the new fitness value about the current bacterium after previous swimming. n is a control parameter that indicates the current iteration number of swimming forwards. Ns is a argument that indicates the allowed maximum value of swimming forwards.

B. REPRODUCTION

During the process of reproduction, the whole bacterial population will be sorted in descending order with their fitness from bad to good, and then the bacteria with worse fitness should eliminate while the others should reproduce in order to keep the size of their population. The related rules about reproduction are included in Eq. (5).

$$\begin{cases} \textbf{index} = \textbf{sort(fitness)} \\ \textbf{pop} = \textbf{pop(index)} \\ \textbf{pop(1 : Sr)} = \textbf{pop((Num} - \textbf{Sr} + \textbf{1}) : \textbf{Num)} \\ \textbf{fitness} = \textbf{FunFit(pop)} \end{cases} \tag{5}$$

where, **fitness** is a Num *1 matrix which reflect the whole fitness record of bacterial population; sort() is a function that returns the ordered sequence about bacteria, which is arranged according to their fitness given as parameters from bad to good; the index is the descending ordered sequence of bacteria, which is arranged according to the fitness of bacteria from bad to good; pop(index) adjusts the order of the whole bacterial population according to the index; pop is the bacterial population; sr is a value that means the number of eliminated and regenerated bacteria. Pop(1:Sr) means the bacteria from number 1 to number Sr as well as pop((Num-Sr +1):Num) means ones from number (Num - Sr +1) to number Num. FunFit(pop) counts the whole fitness of bacterial population and returns a Num *1 matrix.

C. ELIMINATION-DISPERSAL

In bacterial foraging optimization algorithms, the mechanism of elimination-dispersal is just like the mutation operator in GA and it guarantees that a certain number of fresh individuals with new fitness is injected into bacterial populations, and makes the whole bacterial population have strong environmental adaptability and helps to jump out of the local optimal solution, so as to improve the probability of finding the global optima.

The process of elimination-dispersal can be conducted according to Eq. (6) following.

$$\begin{aligned} \textbf{pop(i, :)} = \textbf{Generation(1, Dim, range\_max, range\_min)}, \\ \textbf{if rand} < \textbf{Ped (i} = \textbf{1, 2}, \ldots\ldots, \textbf{Num)} \end{aligned} \tag{6}$$

where, pop(i,:) means the location of the i-th bacterium in all dimensions; Generation is a function that generates a certain number of bacteria by the given parameters, and Dim means the given number of dimension, range_max is the upper bound of the solution space and range_min is the lower bound of the solution space and to be simple, the ranges of solution space are set at the same value [range_min, range_max] in each dimension; rand is a random variable uniformly distributed in interval [0, 1]; Ped is the value of the probability for elimination-dispersal.

D. CLUSTERING

In addition to the three main operations mentioned above, BFOA has the characteristics of clustering, which is that each bacterium receives attraction signals from other individuals in the population to swim to the center of the population as well as rejection signals from circumjacent bacteria in order to keep safe distance between each other. The detail of this process can be got by Eq. (7).

$$\begin{cases} \mathbf{J_{cc}(P(j,k,l))} \\ = \sum_{i=1}^{\mathbf{Num}} \left(-\mathbf{d_{attractant}} * \exp\left(-\mathbf{w_{attractant}} \sum_{d=1}^{\mathbf{D}} \left(\mathbf{P_d} - \mathbf{P_d^i} \right)^2 \right) \right) \\ \qquad + \sum_{i=1}^{\mathbf{Num}} \left(\mathbf{h_{repellant}} \right. \\ \qquad \left. * \exp\left(-\mathbf{w_{repellant}} \sum_{d=1}^{\mathbf{D}} \left(\mathbf{P_d} - \mathbf{P_d^i} \right)^2 \right) \right) \\ \mathbf{J(i,j+1,k,l)} = \mathbf{J(i,j,k,l)} + \mathbf{J_{cc}(P(j,k,l))} \end{cases} \tag{7}$$

where, $J_{cc}(P(j, k, l))$ mean the fitness offset produced by environmental impact in j-th chemotaxis, k-th reproduction and l-th elimination-dispersal; $d_{attractant}$, $w_{attractant}$, $h_{repellant}$ and $w_{repellant}$ are values for the environment; Num is the number of bacterial population; D is the number of dimension; P_d means the whole bacterial population coordinates

on d-th dimension; P_d^i is the i-th bacterial coordinate on d-th dimension; J(i,j,k,l) means the fitness of i-th bacterium in j-th chemotaxis, k-th reproduction and l-th elimination-dispersal.

III. IMPROVEMENT

To improve the search optimization quality of the original bacterial foraging optimization algorithms, all of the variants of BFO mentioned above have their own strategies or mechanisms. However, on the one hand, each of them is limited to its specific application area and has own limitations; On the other hand, for the same kind of problems, people always keep exploring new solutions so that problems can be better solved or new methods can be obtained. Both of the above aspects encourage us to bring forth a new BFO named BFO-DX, in which the strategy named PELO is adopted as well as the strategy ARL, the details are followed.

A. STRATEGY OF PROGRESSIVE EXPLOITATION APPROXIMATING LOCAL OPTIMUM (PELO)

In the original BFO, the process of chemotaxis consists of tumbling and swimming, the former determines the direction of next step while the latter does the distance to go forward. For tumbling, a bacterium gets a random direction for the next step by Eq. (2) and goes forward according to Eq. (3) for swimming, and will keep swimming until the control parameter n is bigger than the argument Ns according to Eq. (4). The original mechanisms about chemotaxis enable the BFO possess ability of exploitation in a local neighborhood space, so as to search the potential global optimum.

Regretfully, the mechanisms of original chemotaxis are not enough to guarantee that the BFO can find the global optimum. From Fig.1 (a) and (b) shown about original chemotaxis, it is obvious that the number of forward swimming is limited and fixed, and that is absolutely insufficient in the solution space of complex optimization problems, especially in the space of continuous solutions.

Accordingly, PELO is adopted to improve this disadvantage, and in PELO each bacterium firstly obtains a random direction for swimming forward according to Eq. (8). Then, a new location can be got according to Eq.(9), and its fitness can be calculated by Eq.(10), if the new fitness is better than the current fitness, the current bacterium should swim to the new position. Otherwise, step-size will be reduced, thereby the algorithm will search the potential global optimum with a shorter step-size, which is believed to be able to enhance exploration capacity in a small local neighborhood space, and the value of controlling parameter n will be increased by 1. Finally, if the value of n is bigger than mul_2 which means the allowed maximum number of reducing the step-size, break out of the loop and go to the next step. Otherwise, the new location will be calculated again with new parameters and continue the loop until the condition for breaking out of the

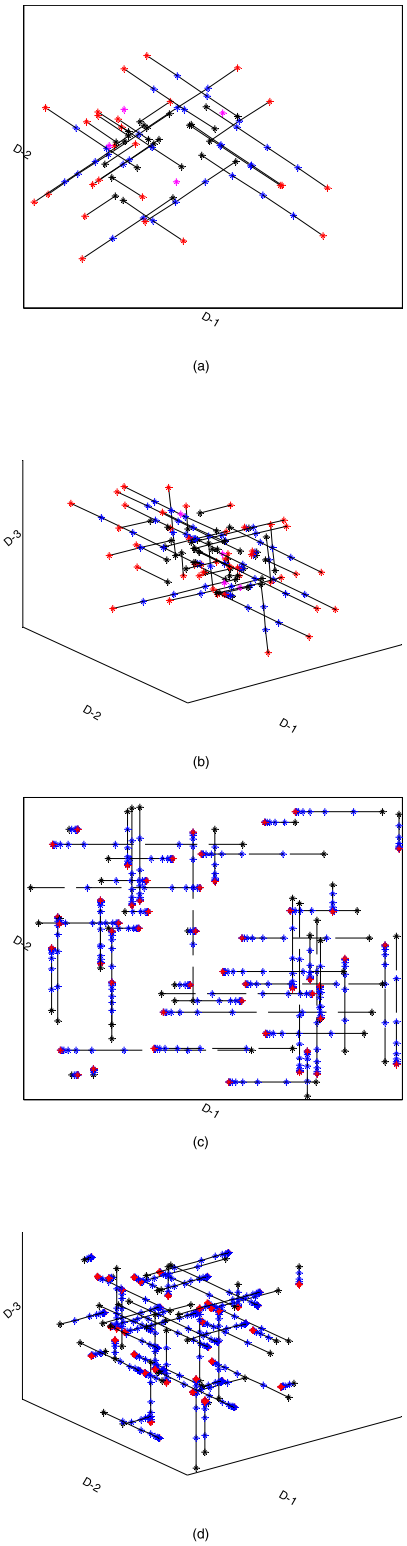

FIGURE 1. Black points mean the current locations of bacteria; Blue points mean the temporary locations of bacteria on the way of swimming; Red points mean the final locations in the process of a chemotaxis; Pink point means that the current location is same with the final one, in other words, the bacterium did not move; Black dot lines mean the swimming routes of bacteria. (a) and (b) show the two and three dimensional sketch maps of traditional chemotaxis about BFO respectively while (c) and (d) show the ones of PELO. (In accordance with the experimental parameters in this paper, 50 bacteria were drawn in the schematic diagram).

loop is satisfied, which is shown in Eq. (11).

$$\begin{cases} \mathbf{forward\_or\_backward} = \mathbf{2} * \mathbf{round(rand)} - \mathbf{1} \\ \mathbf{Temp\_dim} = \mathbf{unidrnd(Dim)} \\ \mathbf{Direction} = (\mathbf{0}, \mathbf{0}, \ldots, \mathbf{d_{Temp\_dim}}, \ldots, \mathbf{0}) \\ \quad {}^{*}\mathbf{forward\_or\_backward(Temp\_dim = 1, 2, \ldots, Dim)} \end{cases} \tag{8}$$

where, forward_or_backward means the direction that a bacterium swims along with a dimension. rand is a random variable uniformly distributed in interval [0, 1], round() rounds the elements of X to the nearest integers, so (2* round (rand) – 1) returns -1 or 1 randomly; Temp_dim means a randomly selected dimension from all Dim ones; unidrnd() generates random numbers for the discrete uniform distribution with the given parameters; Dim means the given number of dimension; Direction means the direction that a bacterium swims along with all dimension; d_{Temp_dim} means the value of some a bacterium on its Temp_dim-th dimension and it is 1.

$$\begin{cases} \mathbf{ti} = \mathbf{j} + (\mathbf{k} - \mathbf{1}) {}^{*}\mathbf{Nc} + (\mathbf{l} - \mathbf{1}) {}^{*}\mathbf{Nre} {}^{*}\mathbf{Nc} \\ (\mathbf{j} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{Nc}; \mathbf{k} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{Nre}; \mathbf{l} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{Ned}) \\ \mathbf{T} = \mathbf{Nc} {}^{*}\mathbf{Nre} {}^{*}\mathbf{Ned} \\ \mathbf{step\_size} = (\mathbf{1} - ((\mathbf{ti} - \mathbf{1})/\mathbf{T})^{\mathbf{5}}) {}^{*}\mathbf{step\_size\_original} \\ \mathbf{new\_location} = \mathbf{x_{current}} + \mathbf{step\_size} {}^{*}\mathbf{Direction} \end{cases} \tag{9}$$

where, ti is the number of chemotaxis during the whole optimization process; T is the maximum number of iterations about chemotaxis; step_size means the step-size of swimming; step_size_original means the original preset value about step-size; new_location is the new location after swimming forward; $x_{current}$ is the current bacterium and its detail location can be got by Eq. (1); Direction can be got by Eq. (8).

$$\mathbf{new\_fitness} = \mathbf{FunFit(new\_location)} \tag{10}$$

where, new_fitness is the fitness of new location.

$$\begin{cases} \mathbf{Loop}: \\ \mathbf{x_{current}} = \mathbf{new\_location}, \ \mathbf{and\ fitness_{current}} \\ \qquad = \mathbf{new\_fitness}, \mathbf{if\ new\_fitness\ is\ better\ than} \\ \qquad \mathbf{fitness_{current}} \\ \mathbf{step\_size} = \mathbf{step\_size/mul\_1}, \ \mathbf{and\ n} = \mathbf{n} + \mathbf{1}, \\ \qquad \mathbf{if\ new\_fitness\ is\ not\ better\ than} \\ \qquad \mathbf{fitness_{current}} \\ \mathbf{break\ out\ of\ the\ loop}, \quad \mathbf{if\ n} > \mathbf{mul\_2} \\ \mathbf{new\_location} = \mathbf{x_{current}} + \mathbf{step\_size} {}^{*}\mathbf{Direction}, \ \mathbf{and} \\ \mathbf{continue\ the\ loop}, \quad \mathbf{if\ n} <= \mathbf{mul\_2} \end{cases} \tag{11}$$

where, Loop means the start of a loop; $fitness_{current}$ is the fitness of current bacterium; mul_1 is a preset parameter to reduce the step-size during the period of swimming(in this paper, it is set at 10); n is a parameter that controls the number of the step-size to be reduced, and its starting value is set to 1. mul_2 is a preset parameter, which means the allowed maximum number of reducing the step-size.

By comparing Fig.1(a) and (b) with Fig.1 (c) and (d), it can be seen that the bacteria constantly adjust their steps to swim forward more times than traditional chemotaxis. In fact, in the traditional chemotaxis, a bacterium stops as soon as it encounter a little difficulty (the new fitness is not better than its current fitness) and the preset number of swimming is a fixed small value in general. Both of them make it difficult for a bacterium to perform well during a chemotaxis. Contrarily, PELO enable a bacterium swim forward for many million times until the stopping condition is satisfied and the number of times depends mainly on the distance between the current location and the potential optimal location. PELO can be imagined as a pile driver., and it keeps hitting the ''optimization algorithm'' to go deeper gradually, that means approximating the potential optimal solution constantly.

Thus, PELO can improve the performance of BFO.

B. STRATEGY OF ADAPTIVE RAID FOR LEADER (ARL)

In the original BFO, to search for food in their own way, each of bacteria receives attraction signals from other individuals in the population, that means individuals will swim to the center of the population, and at the same time, each individual gets repellent signals from nearby individuals in order to be away from each other, this mechanism helps the bacterium population to find the best food resource, however, practice has proved that it is not enough for BFO as an intelligent optimization algorithm. Actually, the conventional BFO is lack of the efficient information exchanging ways between individuals, each bacterium tumbles and swims along its direction generated randomly during each chemotaxis process, so macroscopically, bacterial population shows a chaotic phenomenon. Especially, any of bacteria has no direct relationship with the current best bacterium or temporary optima, that's a pity.

In paper [22], a leader strategy was used to lead the wolf population to find the best food resource and it works excellent. In paper [23], a strategy of jumping for raid (based on leader strategy) was used to strengthen the link between other bacteria and the best one, so as to accelerate the speed of the BFO convergence and it works excellent, too. All these aspects promote us to adopt the strategy of ARL in BFO aimed to speed up its convergence and improve its performance in the same or similar way, and the details are shown in the Figure 2 following.

According to ARL, bacterial population should be arranged according to the fitness of the bacterial population from bad to good, and the half of bacteria with worse fitness should be airdropped directly around the best bacterium with best fitness by Eq. 12 and Eq. 13 while the other half of bacteria rush towards the location of the current best bacterium and try to search foods at the same time on the raid according to Eq. 14 and Eq.15. The sketch maps of ARL about

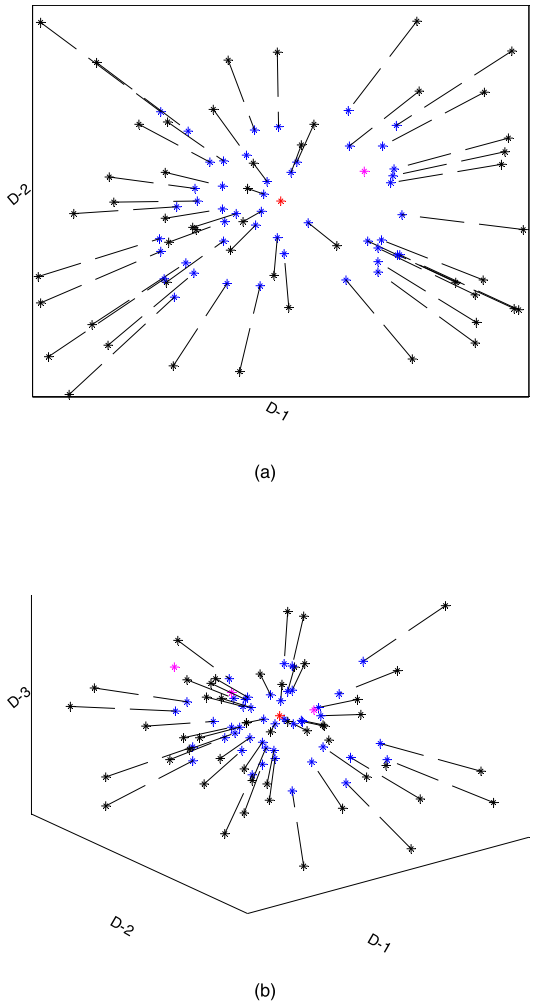

FIGURE 2. Black points mean the current locations of bacteria except the leader bacterium; Blue points mean the possible locations of bacteria after conducting an ARL operation; Pink points are the symmetric points of bacteria centered on the best bacterium; Red point means the current best bacterium; Black dot lines mean the raid routes of bacteria while pink ones mean the jumping routes of bacteria (considering the experiment in this paper, the figure shows 50 bacteria including the best one).

two-dimension and three- dimension are shown in Fig.2 (a) and (b) respectively.

$$\mathbf{step\_raid} = (\mathbf{1} - ((\mathbf{ti} - \mathbf{1})/\mathbf{T})\text{\^{}}\mathbf{5})^{*}\mathbf{step\_raid\_original} \tag{12}$$

$$\begin{cases} \mathbf{pop} = \mathbf{sort}(\mathbf{pop}) \\ \mathbf{pop}(\mathbf{1} : \mathbf{half\_num}) = \mathbf{Generation}(\mathbf{half\_num}, \mathbf{Dim}, \\ \qquad \mathbf{step\_raid}, \mathbf{step\_raid}^{*}(-\mathbf{1})) + \mathbf{bestone} \end{cases} \tag{13}$$

where, step_raid is the step-size during the period of ARL; step_raid_original is a preset parameter and its value is setted at 0.9 in this paper; pop is the bacterial population; sort() is a function that returns the ordered sequence about bacteria, which is arranged according to their fitness given as parameters from bad to good; half_num means the half

TABLE 1. Algorithm arguments.

Algorithm	Main parameters
GA	the crossover probability is 0.7, the mutation probability is 0.01, the maximum number of generations allowed is 6000, the size of population is 50.
PSO	P(1)-Epochs between updating display:0, P(2)-maximum number of iterations (epochs) to train is 6000, P(3)-population size is 50, P(4)-acceleration const 1 (local best influence) is 2, P(5)-acceleration const 2 (global best influence) is 2, P(6)-initial inertia weight is 0.9, P(7)-final inertia weight is 0.4, P(8)-epoch when inertial weight at final value is 1500, P(9)-minimum global error gradient: 1e-25, P(10)-epochs before error gradient criterion terminates run: 250, P(11)-error goal: NaN, P(12)-type flag (which kind of PSO to use): 0, P(13)- PSOseed: 0.
BFO	Number of bacteria: s=50, Number of chemotaxis: Nc=100, Number of swimming: Ns=4, Number of reproduction: Nre=4, Number of elimination-dispersal: Ned=15, Number of bacteria for reproduction: Sr = S/2, Probability about elimination-dispersal: Ped=0.25,the swimming step-size: C=0.001, parameter about attracting : d_attract = 0.05, speed about attracting material releases : ommiga_attract = 0.05, parameter about repellant : h_repellant = 0.05, speed about repellant material releases: ommiga_repellant = 0.05.
BFO-DX	Number of bacteria: s=50, Number of chemotaxis: Nc=100, Number of new swimming: Ns=1000, Number of reproduction: Nre=4, Number of elimination-dispersal: Ned=15, Number of bacteria for reproduction: Sr = S/2, Probability about elimination-dispersal: Ped=0.25, step_swimming_original = 1.5, step_raid_original = 0.9.

TABLE 2. Test functions.

Order	Function	Expression	Dimension	Range	Theoretical Optima
1	Sphere	$f_1 = \sum_{i=1}^{p} x_i^2$	10 30 50	[-100,100]	min f = 0
2	Schwefel' Problem 2.22	$f_2 = \sum_{i=1}^{p} \lvert x_i \rvert + \prod_{i=1}^{p} \lvert x_i \rvert$	10 30 50	[-500,500]	min f = 0
3	HappyCat	$f_3 = \mid \sum_{i=1}^{p} x_i^2 - \mathbf{p} \mid^{\frac{1}{4}} + \frac{0.5 * \sum_{i=1}^{p} x_i^2}{p} - \mathbf{0.5}$	10 30 50	[-100, 100]	min f = 0
4	Griewank	$f_4 = \frac{1}{4000} * (\sum_{i=1}^{p} x_i^2) - (\prod_{i=1}^{p} \cos(\frac{x_i}{\sqrt{i}})) + 1$	10 30 50	[-600,600]	min f = 0
5	Rastrigin	$f_5 = \sum_{i=1}^{p} (x_i^2 - 10 * \cos(2\pi x_i) + 10)$	10 30 50	[-5.12,5.12]	min f = 0
6	Ackley	$f_6 = -20*\exp(-0.2* \sqrt{\frac{1}{p}\sum_{i=1}^{p} x_i^2}) - \exp(\frac{1}{p}\sum_{i=1}^{p} \cos(2\pi x_i)) + 20 + e$	10 30 50	[-32,32]	min f = 0

number of bacterial population; ''1:half_num means'' a range from 1 to half_num; Generation() is a function that generates required bacteria according to the given parameters; bestone means the bacterium with the best fitness currently.

$$\mathbf{x_{id-opposition} = 2^{*}x_{bd} - x_{id} (i = 1, 2 \ldots N; d = 1, 2 \ldots D)} \tag{14}$$

where, x_{id} is the coordinate of the wolf i at the d-th dimension. xbd is the coordinate of the best bacterium at the d-th dimension.

If the fitness of the opposite position is better than the current position, change the current position to the opposite position. Otherwise, each of wolves except the lead wolf changes its position by the Eq. (15).

$$\begin{cases} \mathbf{x_{i-m\_d} = (x_{i\_d} + x_{l\_d})/2} \\ \mathbf{x_i = Bestfitness([x_{i\_1}, x_{i\_2}, \ldots x_{i\_(D-1)}, x_{i\_D}],} \\ \mathbf{[x_{i\_1}, x_{i\_2}, \ldots x_{i\_(D-1)}, x_{i-m\_D}],} \\ \mathbf{\ldots [x_{i-m\_1}, x_{i-m\_2}, \ldots x_{i-m\_(D-1)}, x_{i-m\_D}])} \\ \mathbf{(i = 1, 2 \ldots Q; d = 1, 2 \ldots D)} \end{cases} \tag{15}$$

where, x_{i-m_d} is the position of the middle point at the d-th dimension between the i-th bacterium and the best bacterium. x_{i_d} is the position of the wolf i at the d-th dimension. x_{l_d} is the position of the best bacterium at the d-th dimension. Bestfitness returns the bacterium with the best fitness from the given bacteria. D is the max dimension number of the solution space.

Seen from Eq.(15), there are 2^D points in the function Bestfitness(). At the end of ARL, the updated location is better than the current position or not worse than the current position at least. And Seen from Fig.2 (a) and (b), macroscopically, the bacterial population is approaching the current optimal individual and the investigation on the opposite points increases the possibility of finding the optimal solution, which is why ARL can accelerate the speed of convergence and improve the performance of BFO.

IV. STEPS OF BFO-DX

Based on the above strategies of PELO and ARL, BFO is improved and BFO-DX is proposed. The following is the steps about how to implement the new proposed algorithm.

A. INITIALIZATION

The following parameters should be initialized: number of bacteria: S, number of chemotaxis: Nc, number of swimming: Ns, number of reproduction: Nre, number of elimination-dispersal: Ned, number of bacteria for reproduction: Sr, probability about elimination-dispersal: Ped, number of dimension: Dim, upper boundary : bound_max, lower boundary: bound_min, objective function: FitnessFunction, population of bacteria: P by Eq. (6), current best bacterium: bestone, current best fitness: besthealth, maximum number of iterations: T, original step-size for PELO: step_size_original and original step-size for ARL- step_raid_original.

TABLE 3. Data of experiments.

Function	Algorithm	Optimal value	Worst value	Average value	Standard deviation	Average iteration	Average time
1 Sphere min f= 0	GA	4.39E-05	0.00038852	0.00016437	7.10E-09	933.36	1.6694
	PSO	1.43E-31	1.02E-29	1.66E-30	7.23E-60	2266.52	0.26204
	BFO	5.78E-05	7.76E-05	6.53E-05	4.47E-11	6000	98.368
	BFO-DX	**0**	**0**	**0**	**0**	2248.6	504.4696
2 Schwefel' Problem 2.22 min f= 0	GA	0.012806	1.3536	0.40619	0.14316	2085.4	3.6487
	PSO	1265.8058	1.48E+58	5.94E+56	8.46E+114	2619.72	0.30751
	BFO	0.033047	0.041327	0.038417	9.90E-06	6.00E+03	1.02E+02
	BFO-DX	**0**	**0**	**0**	**0**	3920.2	1133.0607
3 HappyCat min f= 0	GA	0.0056192	0.027566	0.01764	3.54E-05	140.48	0.27246
	PSO	0	0.033731	0.0017941	4.39E-05	1278.68	0.15791
	BFO	1.8229	1.8236	1.8232	5.45E-08	6000	104.3437
	BFO-DX	**0**	**0**	**0**	**0**	**35.8**	5.9189
4 Griewank min f= 0	GA	5.25E-05	0.024683	0.0014935	2.45E-05	665.36	1.2301
	PSO	6.2122	14.1392	9.3269	5.1433	4073	0.78765
	BFO	3.47E-06	3.84E-06	3.64E-06	1.49E-14	6000	105.9878
	BFO-DX	**0**	0.066124	0.033794	0.00079019	664.4	148.7515
5 Rastrigin min f= 0	GA	2.9854	40.7985	14.0503	62.3658	973.04	1.7639
	PSO	15.9193	45.7681	28.8936	75.5902	2062	0.34203
	BFO	8.62E-03	0.011999	0.01038	1.27E-06	6000	104.4283
	BFO-DX	6.9647	12.9345	9.5516	3.8014	1000	223.2299
6 Ackley min f= 0	GA	0.00012983	2.3164	1.0387	0.71758	1124.2	2.0677
	PSO	3.0939	7.8981	5.0224	1.1841	1986.84	0.29056
	BFO	0.0046151	0.0054824	0.0051359	8.43E-08	6000	108.4898
	BFO-DX	**2.13E-14**	**2.80E-06**	**5.60E-07**	**1.25E-12**	1000	271.2922

(a) Dimension = 30

TABLE 3. ***(Continued.)*** **Data of experiments.**

Function	Algorithm	Optimal value	Worst value	Average value	Standard deviation	Average iteration	Average time
1 Sphere min f= 0	GA	0.00039564	0.0040238	0.0012047	6.71E-07	2801.36	6.3393
	PSO	4.22E-29	1.18E-26	1.76E-27	4.93E-54	3299.44	0.51836
	BFO	9.68E-05	0.00014548	0.0001122	3.05E-10	6000	150.5383
	BFO-DX	**0**	**0**	**0**	**0**	3855.4	918.4212
2 Schwefel' Problem 2.22 min f= 0	GA	0.45512	7.4595	2.5459	2.4016	5254.04	11.8383
	PSO	2932.5992	7.28E+102	2.91E+101	2.03E+204	2239.08	0.35377
	BFO	0.061591	0.073414	0.068249	2.62E-05	6.00E+03	1.54E+02
	BFO-DX	**1.44E-291**	**1.12E-290**	**4.74E-291**	**0**	6000	1856.0944
3 HappyCat min f= 0	GA	0.0083691	0.028642	0.019549	3.05E-05	129.88	0.31774
	PSO	0	0.014543	0.0011778	9.14E-06	1310.76	0.21522
	BFO	2.1493	2.1498	2.1496	1.99E-08	6000	156.1046
	BFO-DX	**0**	**0**	**0**	**0**	**68.2**	12.2902
4 Griewank min f= 0	GA	0.00077787	0.017992	0.0044978	1.72E-05	1522.2	3.5731
	PSO	14.2107	28.7889	20.406	16.5573	4334	1.2402
	BFO	4.56E-06	5.17E-06	4.89E-06	3.98E-14	6000	161.1881
	BFO-DX	**0**	0.012316	0.0073931	1.70E-05	858	250.3504
5 Rastrigin min f= 0	GA	23.8899	68.731	37.1264	146.1874	2355.24	5.4789
	PSO	29.8488	76.6118	46.6039	113.3823	2744.12	0.65791
	BFO	2.03E-02	0.026189	0.022618	4.54E-06	6000	159.0328
	BFO-DX	12.9345	17.9093	16.3173	4.1974	1000	292.8324
6 Ackley min f= 0	GA	0.005149	2.3746	1.739	0.24249	1896.4	4.5661
	PSO	3.7342	7.5642	5.6012	0.75633	2635.52	0.50418
	BFO	0.0058573	0.0066561	0.0063526	9.30E-08	6000	160.9803
	BFO-DX	**8.09E-08**	19.9622	11.9753	95.6047	1000	329.5027

(b) Dimension = 50

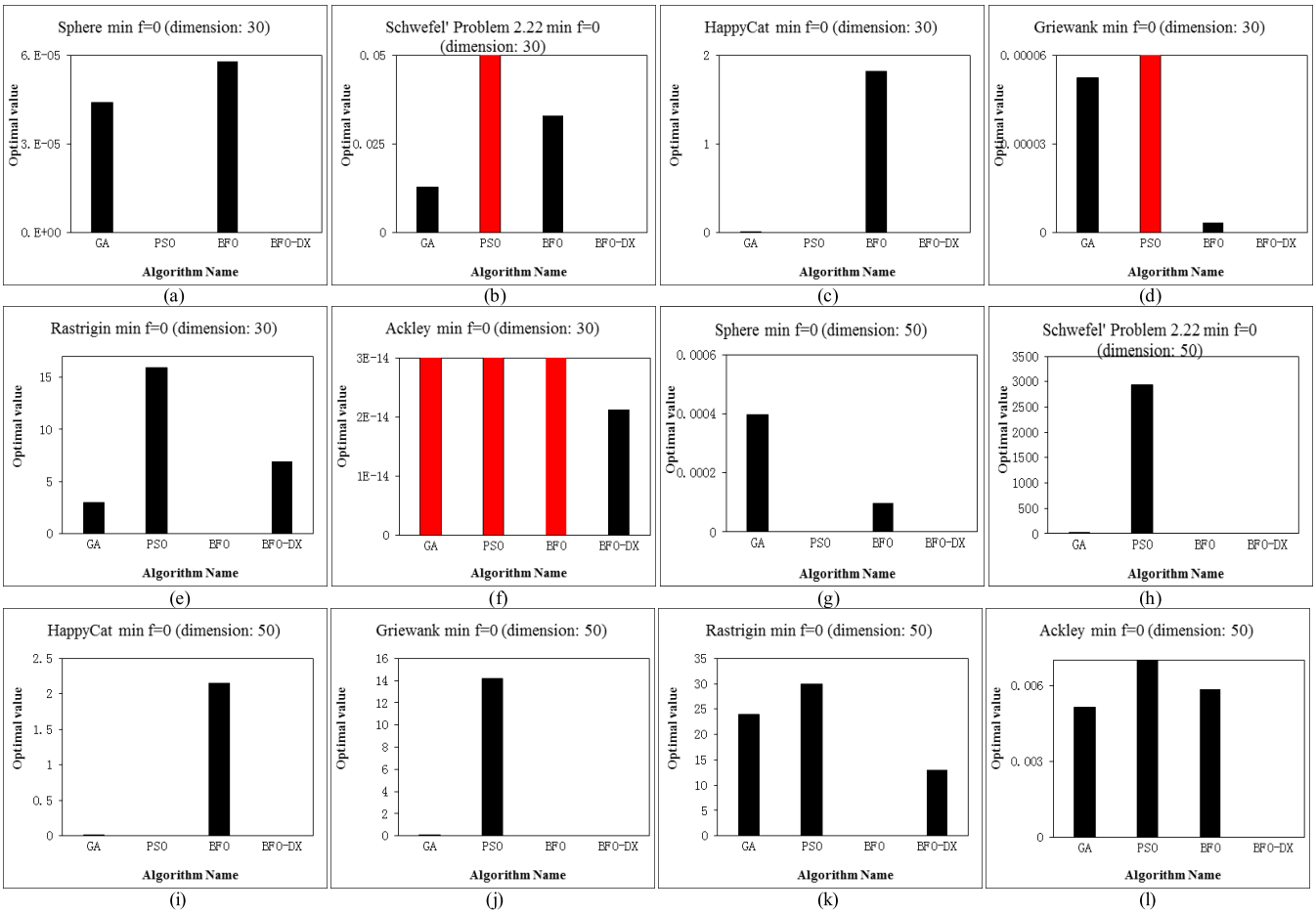

FIGURE 3. Histogram about the optimal values of the six test functions in 30 and 50-dimensional space respectively for GA, PSO, BFO and BFO-DX. About 30 dimensional space, (a) Sphere (b) Schwefel'Problem2.22 (c) HappyCat (d) Griewank (e) Rastrigin (f) Ackley. About 50-dimensional space, (g) Sphere (h) Schwefel'Problem2.22 (i) HappyCat (j) Griewank (k) Rastrigin (l) Ackley (Red means the value is beyond of the upper limit of the chart).

B. CHEMOTAXIS

In this step, any bacterium from the bacterial population will conduct the PELO and ARL. Firstly, a random direction for swimming forward should be obtained according to Eq. (8). Then, a new location can be got according to Eq. (9) as well as its fitness can be calculated by Eq. (10), if the new fitness is better than the current fitness, the current bacterium should swim to the new position. Otherwise, step-size will be reduced so that new algorithm can take a shorter step-size to search for the potential global optimum. And the loop about swimming will be continued until the controlling parameter n is bigger than mul_2 according to Eq. (11).

At the end of the chemotaxis, this bacterium will be compared with the current best bacterium, if the fitness of this bacterium is better than the fitness of current best bacterium, the best bacterium should be updated to this bacterium and the best fitness should be changed accordingly, otherwise, do nothing about updating.

Judge whether the number of chemotaxis is bigger than Nc or not, if not, continue in this loop about chemotaxis, otherwise, goes to 4.3.

C. REPRODUCTION

In the real living environment, the biological law of survival of the fittest is implemented. After the period of conducting new chemotaxis to search food, some weaker bacteria will starve to death while the stronger ones should reproduce themselves so as to keep the size of their population in the same number. And in bacterial foraging optimization algorithms, the related rules about reproduction are included in Eq. (5).

Finally, judge whether the number of reproduction is bigger than Nre or not, if not, continue the loop about reproduction, otherwise, goes to 4.4.

D. ELIMINATION-DISPERSAL

In this step, any bacterium in the population has the chance to eliminate and disperse, and the probability of elimination-dispersal is Ped that is a preset value, detailed rules follow the Eq. (6).

Judge whether the number of elimination-dispersal is bigger than Ned or not, if not, continue in this loop, otherwise, go to 4.5. Determine whether the number of

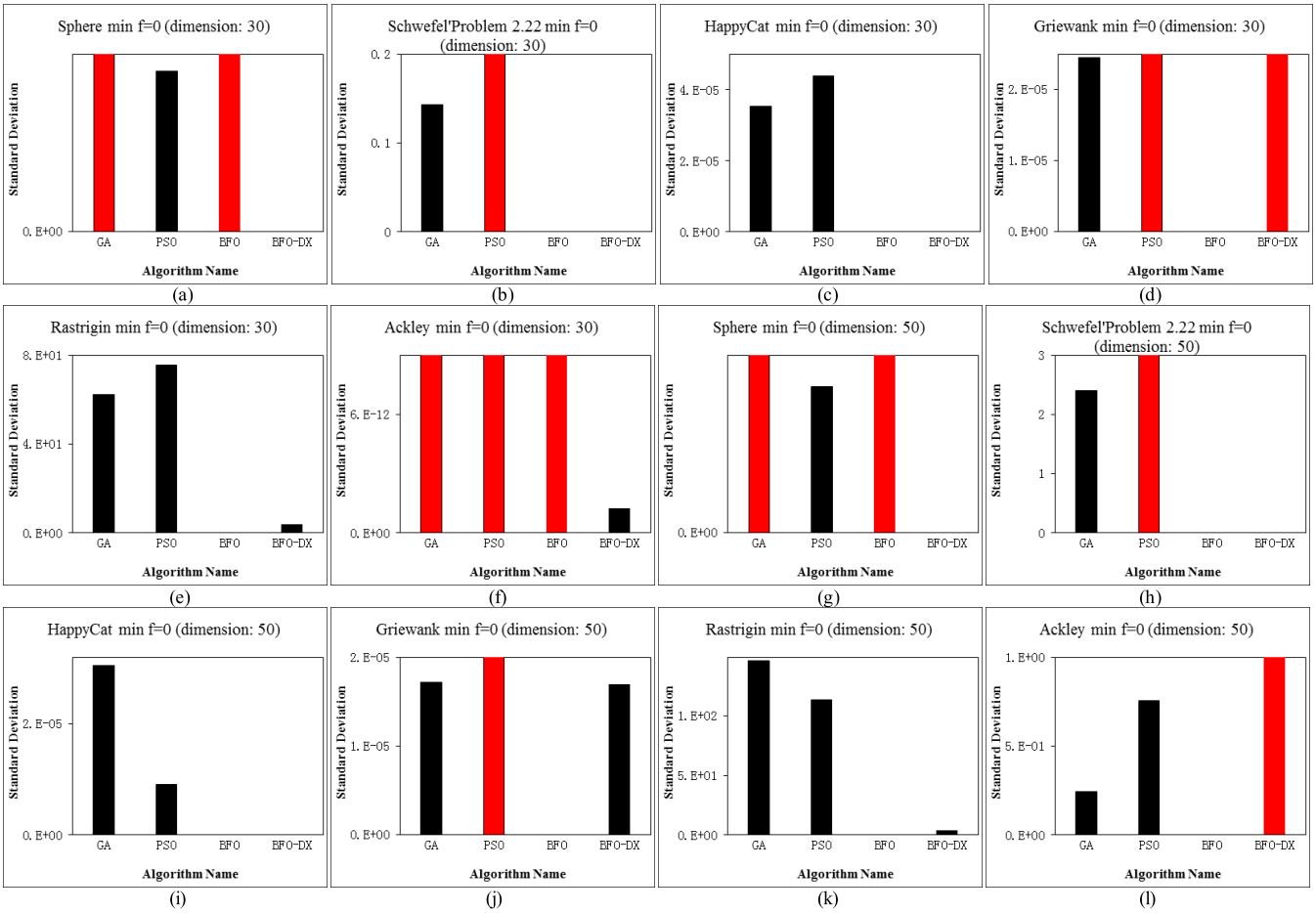

FIGURE 4. Histograms of the standard deviation about the six test functions in 30 and 50-dimensional space respectively for GA, PSO, BFO and BFO-DX. About 30 dimensional space: (a) Sphere (b) Schwefel' Problem2.22 (c) HappyCat (d) Griewank (e) Rastrigin (f) Ackley. About 50-dimensional space: (g) Sphere (h) Schwefel' Problem2.22 (i) HappyCat (j) Griewank (k) Rastrigin (l) Ackley. (Red means the value is beyond of the upper limit of the chart).

elimination-dispersal is greater than Ned or not, if not, then continue the loop, otherwise go to 4.5.

E. GLOBAL oPTIMA

At the end of all loops, the bacterium with the best fitness will be the global optimum that the algorithm is committed to finding.

V. NUMERICAL EXPERIMENTS

In order to show the optimization performance of BFO-DX, comparisons between BFO-DX and some other state-of-the-art algorithms including classical genetic algorithm(GA), particle swarm algorithm(PSO) and conventional BFO shown in TABLE 1 were made on six classical benchmarks, which are present as well as the order, function, expression, dimension, feature, range and the theoretical optimum about each of the test functions in TABLE 2, and in the two-dimensional form, Ackley [24] function is characterized by a nearly flat outer region and a large hole at the center, and it poses a risk for optimization algorithms, particularly hill-climbing algorithms, to be trapped into one of its many local minima. Griewank [24] function has many widespread local minima, which are regularly distributed. Rastrigin [24] function has several local minima, and it is highly multimodal, but locations of the minima are regularly distributed. Schwefel [24] function is complex, with many local minima. Sphere [24] function has d local minima except for the global one. It is continuous, convex and unimodal. Happycat [25] is about a new class of simple and scalable test functions for unconstrained real-parameter optimization, which appear difficult to be optimized even if they have only one minimizer, and while the step-sizes drops down exponentially fast, the EA is still far away from the minimizer giving rise to premature convergence. TABLE 3 provides the results of the numerical experiments about the four algorithms on the six test functions to us to do comparative analysis and research next. Details are shown in tables following.

All numerical experiments are implemented on a computer equipped with Ubuntu 16.04.4 operating system, Intel(R) Core(TM) i7-5930K processor and 64G memory while the integrated development environment is MATLAB-2017a. Toolbox of MATLAB 2017a about genetic algorithm is used to carry out experiments for GA;

a toolbox named ''PSOt'' for MATLAB is used to carry out experiments for PSO[9]; BFO is achieved based on the ideas provided by reference [7]; To verify the excellent performance of the new algorithm, optimization calculation is running for 25 times on each test function. Then, all of the algorithms are evaluated from the optimal value, the worst value, the average value, the standard deviation, the average iteration and average time for getting the global optimum.

As shown in TABLE 3, seen from the optimal value, the BFO-DX has best performance on all the test functions except function 5 in 30-dimensional and 50-dimensional spaces shown in Fig.3 (e) and (k) respectively, and on many functions the optimal values found by the new algorithm are their theoretical optimal values including function 1,2,3,4 in 30-dimensional space and function 1,3,4 in 50-dimensional space shown in Fig.3 (a), (b), (c), (d), (g), (i) and (j) respectively. Regrettably, new algorithm does not have best performances on function 5 in 30-dimensional and 50-dimensional spaces, even so, its optimal values are not worst and acceptable compared with the other three algorithms shown in Fig.3 (e) and (k) respectively.

In addition, from the terms of ''Worst value'' and ''Average value'', on all the functions the new algorithm has best performances except function 4, 5 in 30-dimensional space and function4, 5, 6 in 50-dimensional space, especially on function 1, 2, 3 in 30-dimensional space and function 1, 3 in 50-dimensional space the ''Worst value'' and ''Average value'' are zero that means the theoretical optimal values. Therefore, BFO-DX has better optimization quality.

Moreover, for 25 times, ''Standard Deviations'' (SD) about new algorithm on nearly all the test functions except function 4, 5 in 30-dimensional space and function 4, 5, 6 in 50-dimensional space are best shown in Fig.4 (d), (e), (j), (k) and (l) respectively, and on function 1, 2, 3 in both 30-dimensianl and 50-dimensional space SD are zero shown in Fig.4 (a), (b), (c), (g), (h) and (i) respectively, and even on function 4, 5 in 30-dimensional and 50-dimensional spaces SD is not worst, that means stability of new algorithm is better.

Finally, BFO-DX has the smallest number of ''Average Iteration'' on function 3, 4, 6 in 30-dimensional space and function 3, 4, 5, 6 in 50-dimensional space, and on other test functions the numbers of ''Average Iteration'' spent by BFO-DX are not biggest compared with the other three algorithms, so to some extent, it can be concluded that BFO-DX has better speed of convergence than the other three algorithms. Unfortunately, on the other functions except function 3 in 30-dimensional and 50-dimensional spaces, the average times spent by BFO-DX are most, shown in TABLE 3.

In general, BFO-DX has good optimization quality and better stability.

VI. CONCLUSION & DISCUSSION

To improve the performance of original bacterial foraging optimization algorithm (BFO), in this paper, we propose a new bacterial foraging optimization algorithm, called BFO-DX, which makes use of strategies of progressive exploitation approximating local optimum and adaptive raid. The use of PELO is an important strategy for BFO-DX to enhance its ability of exploitation in a local space, which is like a pile driver and keeps hitting the BFO-DX and makes it gradually go deeper, that means approximating the potential optimal solution constantly, so PELO helps the algorithm to find the global optima better possibly. With the strategy of ARL, BFO-DX can strengthen the global exploration capacity in the early part of the optimization process and fine-tune the solutions at the latter part of the search. To evaluate the performance of the BFO-DX algorithm, a series of comparative experiments were carried out on six classical functions in 30-dimensional and 50 dimensional spaces as well as GA, PSO and original BFO were used as the comparative algorithms. Under the same condition, the numerical experiments indicates that BFO-DX possesses better ability of finding global optima, better stability and other acceptable terms such as iteration and running time.

As the data shown in TABLE 3, Fig.3 and Fig.4 above, compared with the three other algorithms, though BFO-DX has better performances in most of aspects on the test functions, it has poor capability in some aspects on some test functions, such as on the other functions except function 3 in 30-dimensional and 50-dimensional spaces the new algorithm spent most ''Average Time''. Consequently, on the whole, BFO-DX is a very competitive algorithm and is a new efficient method for addressing optimization problem.

Our future work is to further study the optimization mechanism of BFO-DX and to extend our research to more complex optimization problems and real different engineering applications.

ACKNOWLEDGMENT

The authors are indebted to appreciate their full supports of this pioneering work. The authors would also like to thank the reviewers for their valuable suggestions and comments.

REFERENCES

[1] M. G. Hinchey, R. Sterritt, and C. Rouff, ''Swarms and swarm intelligence,'' *Computer*, vol. 40, no. 4, pp. 111–113, 2007.
[2] P. J. Angeline, ''Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences,'' in *Proc. Int. Conf. Evol. Program.* Berlin, Germany: Springer, vol. 1447, 1998, pp. 601–610.
[3] K. M. Passino, ''Biomimicry of bacterial foraging for distributed optimization and control,'' *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52–67, Mar. 2002.
[4] S. Das, A. Biswas, S. Dasgupta, and A. Abraham, ''Bacterial foraging optimization algorithm: Theoretical foundations, analysis, and applications,'' in *Foundations of Computational Intelligence*, vol. 3. Springer, 2009.
[5] S. Dasgupta, S. Das, A. Abraham, and A. Biswas, ''Adaptive computational chemotaxis in bacterial foraging optimization: An analysis,'' *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 919–941, Aug. 2009.
[6] S. Das, S. Dasgupta, A. Biswas, A. Abraham, and A. Konar, ''On stability of the chemotactic dynamics in bacterial-foraging optimization algorithm,'' in *Proc. 5th Int. Conf. Soft Comput. Transdisciplinary Sci. Technol. (CSTST)*, vol. 39, 2008, pp. 245–251.
[7] Z. Yalan, ''Research and application on bacteria foraging optimization algorithm,'' *Comput. Eng. Appl.*, vol. 46, no. 20, pp. 16–21, 2010.

[8] G. Sharma and A. Kumar, "Fuzzy logic based 3D localization in wireless sensor networks using invasive weed and bacterial foraging optimization," *Telecommun. Syst.*, vol. 67, no. 2, pp. 149–162, 2018.
[9] Y.-P. Chen, Y. Li, G. Wang, Y.-F. Zheng, Q. Xu, J.-H. Fan, and X.-T. Cui, "A novel bacterial foraging optimization algorithm for feature selection," *Expert Syst. Appl.*, vol. 83, pp. 1–17, Oct. 2017.
[10] B. Hernández-Ocaña, O. Chávez-Bosquez, J. Hernández-Torruco, J. Canul-Reich, and P. Pozos-Parra, "Bacterial foraging optimization algorithm for menu planning," *IEEE Access*, vol. 6, pp. 8619–8629, 2018.
[11] A. Tabssam, K. Pervaz, A. Saba, Z. ul Abdeen, M. Farooqi, and N. Javaid, "Demand side management using bacterial foraging and crow search algorithm optimization techniques," in *Proc. Int. Conf. Intell. Netw. Collaborative Syst.* Cham, Switzerland: Springer, 2017.
[12] Z. Cai, J. Gu, C. Wen, D. Zhao, C. Huang, H. Huang, C. Tong, J. Li, and H. Chen, "An intelligent parkinson's disease diagnostic system based on a chaotic bacterial foraging optimization enhanced fuzzy KNN approach," *Comput. Math. Methods Med.*, vol. 2018, no. 3, Jun. 2018, Art. no. 2396952.
[13] C. H. A. ul Hassan, M. S. Khan, A. Ghafar, S. Aimal, S. Asif, and N. Javaid, "Energy optimization in smart grid using grey wolf optimization algorithm and bacterial foraging algorithm," in *Proc. Int. Conf. Intell. Netw. Collaborative Syst.*, Cham, Switzerland: Springer, 2017, pp. 166–177.
[14] M. Shibiao and C. Zhijun, "Crowd evacuation model based on bacterial foraging algorithm," *Int. J. Modern Phys. C*, vol. 29, no. 3, 2018, Art. no. 1850027.
[15] A. M. Othman and H. A. Gabbar, "Enhanced microgrid dynamic performance using a modulated power filter based on enhanced bacterial foraging optimization," *Energies*, vol. 10, no. 6, p. 776, 2018.
[16] J. Cui, "The application of improved bacteria foraging algorithm to the optimization of aviation equipment maintenance scheduling," *Tehnicki Vjesnik-Tehnical Gazette*, vol. 25, no. 4, pp. 1103–1109, 2018.
[17] W. Ali, A. U. Rehman, M. Junaid, S. A. A. Shaukat, Z. Faiz, and N. Javaid, "Home energy management using social spider and bacterial foraging algorithm," in *Proc. 20th Int. Conf. Netw.-Based Inf. Syst. (NBiS)*. Cham, Switzerland: Springer, 2017.
[18] C. Yang, J. Ji, and A. Zhang, "BFO-FMD: Bacterial foraging optimization for functional module detection in protein–protein interaction networks," *Soft Comput.*, vol. 28, pp. 3395–3416, May 2017.
[19] Y. T. Amghar, and H. Fizazi, "A hybrid bacterial foraging optimization algorithm and a radial basic function network for image classification," *J. Inf. Process. Syst.*, vol. 13, no. 2, pp. 215–235, 2017.
[20] H. Wang and B. Niu, *A Novel Bacterial Algorithm With Randomness Control for Feature Selection in Classification*. Amsterdam, The Netherlands: Elsevier, 2017.
[21] C. Yang, J. Ji, J. Liu, and B. Yin, "Bacterial foraging optimization using novel chemotaxis and conjugation strategies," *Inf. Sci.*, vol. 363, pp. 72–95, Oct. 2016.
[22] Z. Qiang and Y. Q. Zhou, "Wolf colony search algorithm based on leader strategy," *Appl. Res. Comput.*, vol. 30, no. 9, pp. 2629–2632, 2013.
[23] D. Wang, X. Qian, K. Liu, X. Ban, and X. Guan, "An adaptive distributed size wolf pack optimization algorithm using strategy of jumping for raid (September 2018)," *IEEE Access*, vol. 6, pp. 65260–65274, 2018.
[24] *Virtual Library of Simulation Experiments: Test Functions and Datasets, Optimization Test Problems*. [Online]. Available: http://www.sfu.ca/~ssurjano/optimization.html
[25] H.-G. Beyer and S. Finck, "HappyCat—A simple function class where well-known direct search algorithms do fail," in *Proc. PPSN*, 2012, pp. 367–376.

DONGXING WANG was born in Shangcai, Henan, China, in 1985. He received the B.S. degree in computer science and technology from Putian University, in 2008, and the M.S. degree in computer science and technology from the University of Science and Technology Beijing, in 2011. He is currently pursuing the Ph.D. degree in computer science and technology with the China University of Mining and Technology, Beijing.

From 2011 to 2014, he was an Engineer and a Senior Engineer with RuiQi Electronics Technology Company Ltd., and LiLin Technology Company Ltd., respectively. From 2014 to 2017, he has served as a college Teacher with the Zhangzhou City College, and is a member of the China Computer Federation. His research interests include computing intelligence and swarm intelligent optimization.

XU QIAN was born in 1962. He is currently a Ph.D. Professor and a Doctoral Supervisor.

He is also the Dean of the School of Mechanical Electrical and Information Engineering. He is a Top-notch talent in coal system, in 1998. He is the Leader of the reserve subject, in 2001, and a committee member of textbook research and compilation organized by the National Association of Higher Education. He has been studying in Japan for many times as a Senior Visiting Scholar. His research areas include databases, information fusion technology, software engineering theory and technology, computer supported cooperative work (CSCW) technology, and knowledge engineering in computer science at home and abroad core journals published more than 60 academic papers. Of these, six were EI searches, eight were cited by ISTP. He has hosted three provincial and ministerial level vertical fund projects (responsible persons). As the main researcher and developer, he has participated in one of the "863" national projects, and three provincial and ministerial level vertical fund projects.

Dr. Qian was awarded the First Prize of the Coal System Science and Technology Progress Award (2004), the Second Prize of the Henan Science and Technology Progress Award (1998), the Second Prize of the Coal System Science and Technology Progress Award (1999), the Third Prize of the Coal System Science and Technology Progress Award (1997 and 1999), the First Prize of the School-level Science and Technology Progress Award (1999), and the Second Prize of the School-level Science and Technology Progress Awards (1997, 1997, 1999, and 2001), and three prize of science and technology progress award at school level (1997).

XIAOJUAN BAN was born in 1970.

She is a Ph.D. Professor and a Doctoral Supervisor for artificial intelligence, human–computer interaction, and three-dimensional visualization with the University of Science and Technology Beijing. She is an outstanding talent of the Ministry of Education in the new century. She is a Party representative at the 11th and 12th Party Congresses in Beijing and has presided over a number of national and provincial scientific research projects, such as the National Natural Science Foundation, 863, and the Exploration Project of the General Equipment Department. She has published three monographs in Science Publishing House, National Defense Industry Publishing House, and one translation in Tsinghua University Publishing House. She has five national invention patents and eight software copyrights. She has published and related academic papers in famous Chinese and English journals such as the *Journal of Software*, the *Journal of Computer*, the *Journal of Automation*, the *Expert Systems with Applications*, the *Journal of Visualization*, the *Mathematical Problems in Engineering*, and other important international academic conferences such as EG, CDC, IFAC, IEEE-FUZZ, and so on. She has published more than 200 pieces, and over 100 are included in SCI and EI. Statistics were retrieved from SCI and CNKI, and the total number of papers cited at home and abroad was more than 300 times.

BOYUAN MA was born in Taiyuan, Shanxi, China, in 1992. He received the bachelor's degree in communication engineering from the Beijing Technology and Business University, in 2015, and the M.S. degree in computer science and technology from the University of Science and Technology Beijing, in 2017, where he is currently pursuing the Ph.D. degree in computer science and technology.

His research interests include computer vision and machine learning.

YAN MA was born in Hebei, China, in 1992. He received the B.S. degree in electrical engineering and automation from Liren College, Yanshan University. He is currently pursuing the Ph.D. degree in computer science and technology with the China University of Mining and Technology, Beijing.

ZIYI LV was born in Xinji, Hebei, China, in 1994. She received the B.S. degree in information management and information systems from the Institute of Disaster Prevention, in 2017. She is currently pursuing the M.S. degree in computer science and technology with the China University of Mining and Technology, Beijing.

Her research interests include computing intelligence and swarm intelligent optimization.