Applied Soft Computing 51 (2017) 294–313

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Optimal foraging algorithm for global optimization

Guang-Yu Zhu*, Wei-Bo Zhang

College of Mechanical Engineering and Automation, Fuzhou University, Fuzhou, Fujian 35002, PR China

ARTICLE INFO

Article history:
Received 15 November 2012
Received in revised form
29 November 2016
Accepted 29 November 2016
Available online 6 December 2016

Keywords:
Optimal foraging algorithm (OFA)
Optimal foraging theory
Stochastic search algorithm
Evolutionary algorithms
Behavioral ecology

ABSTRACT

An optimization algorithm, inspired by the animal Behavioral Ecology Theory—Optimal Foraging Theory, named the Optimal Foraging Algorithm (OFA) has been developed. As a new stochastic search algorithm, OFA is used to solve the global optimization problems following the animal foraging behavior. During foraging, animals know how to find the best pitch with abundant prey; in establishing OFA, the basic operator of OFA was constructed following this foraging strategy. During foraging, an individual of the foraging swarms obtained more opportunities to capture prey through recruitment; in OFA the recruitment was adopted to ensure the algorithm has a higher chance to receive the optimal solution. Meanwhile, the precise model of prey choices proposed by Krebs et al. was modified and adopted to establish the optimal solution choosing strategy of OFA. The OFA was tested on the benchmark functions that present difficulties common to many global optimization problems. The performance comparisons among the OFA, real coded genetic algorithms (RCGAs), Differential Evolution (DE), Particle Swarm Optimization (PSO) algorithm, Bees Algorithm (BA), Bacteria Foraging Optimization Algorithm (BFOA) and Shuffled Frog-leaping Algorithm (SFLA) are carried out through experiments. The parameter of OFA and the dimensions of the multi-functions are researched. The results obtained by experiments and Kruskal-Wallis test indicate that the performance of OFA is better than the other six algorithms in terms of the ability to converge to the optimal or the near-optimal solutions, and the performance of OFA is the second-best one from the view of the statistical analysis.

1. Introduction

Interest in solving the real-world optimization applications with global optimization technology is increasing. Over the last few years, examples of the role played by this technology in practical applications abound in physics, chemistry, mechanics, society, economics, management and biomedicine. In most cases of practical interest, the global optimization is very difficult, when the characteristics of complexity, nonlinear, multi-local optimization, modeling difficulties of practical engineering problems are concerned. Thus in recent years, researchers have sought to present a series of intelligent optimization algorithms by imitating the phenomena of nature [1–7]. These ideas expand the feasible area of constructing optimization algorithms and make it an attractive choice to propose new optimization algorithms from various research areas.

In nature, foraging is fundamental to the lives of organisms and different organisms have different foraging behaviors. Based on imitating foraging behavior of several specific organisms, several optimization algorithms were established, such as Ant Colony Optimization, Bacterial Foraging Optimization Algorithm and so on. It was known that animals' foraging efficiency is always being maximized by natural selection for more survival and reproductive opportunities. The strategy of animals' foraging behavior, which can ensure the maximum foraging efficiency, has been deeply studied and named Optimal Foraging Theory in Behavioral Ecology [8,9]. In our study, we try to build an optimization algorithm with the guide of Optimal Foraging Theory (OFT). This algorithm is different with other existed optimization algorithms that were built by imitating foraging behavior of several specific organisms.

A model, called Optimal Foraging Algorithm (OFA), has been developed based on OFT in our former works [10]. In this paper, OFA was tested with benchmark functions and compared with several other optimization algorithms. Moreover the performance of OFA was analyzed based on the experimental results and by the statistic method of the Kruskal-Wallis test. Two types of the dynamic self-adaptive parameter k are compared for analyzing the influence on the performance of OFA.

The remainder of the paper is organized as follows: Section 2 introduces animals' foraging behavior and several optimization

* Corresponding author.
E-mail address: zhugy@fzu.edu.cn (G.-Y. Zhu).

http://dx.doi.org/10.1016/j.asoc.2016.11.047
1568-4946/

algorithms established based on some specific foraging behavior; Section 3 discusses the optimal foraging theory; Section 4 presents the optimal foraging algorithm in detail; Section 5 outlines the algorithm process and corresponding Pseudocode; Section 6 compares the performance of OFA with Real Coded Genetic Algorithms (RCGAs), Differential Evolution (DE), Particle Swarm Optimization algorithm (PSO), Bees Algorithm (BA), Bacteria Foraging Optimization Algorithm (BFOA) and Shuffled Frog-leaping Algorithm (SFLA) through experiments. The performance of OFA was evaluated directly by analyzing the results of experiments and was also tested with the statistical significance test method. Section 7 discusses on the parameter of OFA and the dimensions of the multi-functions, and Section 8 concludes the paper. The Appendix A details twenty-six benchmark functions.

2. Foraging behavior and optimization algorithms based on foraging behavior

Foraging behavior is one of the most common, natural behaviors in the biological world that has drawn great attention from some researchers. Foraging behavior includes searching and consuming behaviors that have a close relationship with food. Foraging behavior of different organisms varies widely. The foraging behaviors of some different organisms have been studied, and several different optimization algorithms have been established based on the studies of the searching behavior, including the famous Ant Colony Optimization, Bacterial Foraging Optimization Algorithm, Shuffled Frog-leaping Algorithm and a set of optimal algorithms based on bee foraging model. The brief introduction of these algorithms is given in the following:

Targeted to solve the discrete optimization problems, Ant Colony Optimization (ACO) was inspired by the foraging behavior of ant colonies that can find the shortest path between their nest and a food source by marking the path they follow with a chemical called pheromone. This collective trail-laying and trail-following behavior whereby an ant is influenced by a chemical trail left by other ants is the inspiring source of ACO [3].

Proposed by Passino, Bacteria Foraging Optimization Algorithm (BFOA), a newcomer to the family of nature-inspired optimization algorithms, was inspired by the social foraging behavior of *Escherichia coli* [6]. The group foraging strategies of a swarm of *Escherichia coli* were followed in establishing BFOA for multi-optimal function optimization. During searching for nutrients, a bacterium takes foraging decisions by moving small steps to ensure the energy obtained per unit time is maximized, and the individual bacterium also communicates with others by sending signals. The searching process is called chemotaxis, and the key idea of BFOA is mimicking chemotactic movement of virtual bacteria in the problem search space [11]. There are four prime steps in BFOA: Chemotaxis, Swarming, Reproduction, and Elimination and Dispersal. Chemotaxis: This process simulates the movement of an *E. coli* cell through swimming and tumbling via flagella. Swarming: When stimulated by a high level of succinate, the *E. coli* cells release an attractant aspertate, which helps them to aggregate into groups and thus move as concentric patterns of swarms with high bacterial density. Reproduction: The least healthy bacteria eventually die while each of the healthier bacteria (those yielding lower value of the objective function) asexually split into two bacteria, which are then placed in the same location. Elimination and dispersal: Some bacteria are liquidated at random with a very small probability while the new replacements are randomly initialized over the search space.

There exists a set of intelligent optimization algorithms that were established based on the honey-searching behavior models of honey bees. Among them, the Bees Algorithm and the Artificial Bee Colony Algorithm are much more mature, and there are some similarities between them [12].

Proposed by Pham et al., the Bees Algorithm mimics the food foraging behavior of swarms of honey bees [7]. Exactly, it is an ingenious new mathematical procedure based on the behavior of honey bees. In its basic version, the algorithm performs neighborhood search combined with random search and can be used for both combinatorial optimization and functional optimization [7]. The main mathematical procedures of Bees Algorithm are: (i) Select sites for the neighborhood search; (ii) Recruit bees for the selected sites (more bees for best sites) and evaluate the fitnesses; (iii) Select the fittest bee from each patch; (iv) Assign remaining bees to search randomly and evaluate their fitnesses. In step (i), bees that have the highest fitness are chosen as "selected bees" and sites visited by them are chosen for the neighborhood search. Then, in steps (ii) and (iii), the algorithm conducts searches in the neighborhood of the selected sites, assigning more bees to search near to the best sites. In step (iv), the remaining bees in the population are assigned randomly around the search space scouting for the new potential solutions [7].

The Artificial Bee Colony (ABC) algorithm is a swarm based meta-heuristic algorithm that was introduced by Karaboga for optimizing the numerical problems [13]. In ABC, the colony of artificial bees contains three groups of bees: employed bees associated with specific food sources, onlooker bees watching the dance of employed bees within the hive to choose a food source, and scout bees searching for food sources randomly. Initially, all food source positions are discovered by scout bees. Thereafter, the nectar of food sources are exploited by employed bees and onlooker bees. Then, the employed bee which was exploiting the exhausted food source becomes a scout bee in search of further food sources once again. The general scheme of the ABC algorithm includes four phases: initialization phase, employed bees phase, onlooker bees phase, and scout bees phase [13].

Shuffled Frog-leaping Algorithm (SFLA) was initially proposed by Eusuff and Lansey [14,15]. The concept of the SFLA is based on observing, imitating, and modeling the social behavior of a group of frogs when they search for the location of a rich source of food [14]. The population in the SFLA, in general, consists of a set of frogs that is subdivided into different groups referred to as memeplexes which have different cultures of frogs. Each memeplex performs a local search. The individual frogs inside each memeplex hold ideas that are influenced by the ideas of other frogs and develop during a memtic evolution process. Meme was defined as a unit of intellectual or cultural information that can be passed with a non-genetic method, such as imitation. Meme was used to express the individual's thought, behavior, and other information. Thus, SFLA has the advantages of both the genetic evolution-based memetic algorithm (MA) and the social behavior-based PSO algorithm [15].

3. Optimal Foraging Theory

The original Optimal Foraging Theory was developed by the ecologists and the zoologists to explain resource use and dietary patterns of animals. Beginning with MacArthur, Pianka and Emlen, many researchers have attempted to explain and predict some aspect of animals' foraging behavior. Their studies articulated the ideas that would form the basis of Optimal Foraging Theory [16]. Optimal Foraging Theory claimed: natural selection favors individuals whose foraging behavior is as energy efficient as possible. In other words, animals tend to feed on the prey that maximizes their net energy intake per unit of foraging time [17].

For a foraging animal, three questions must be answered during foraging. They are: a.) Where would the animal look for food? b.) When would the animal transfer foraging patches? c.) What type

of food would the animal choose? Therefore, in Behavioral Ecology, Optimal Foraging Theory mainly is used in the following categories: (i) choice by an animal of which food types to eat (i.e., optimal diet); (ii) choice of patch type in which to feed (i.e., optimal patch choice); (iii) optimal allocation of time to different patches; and (iv) the optimal patterns and speed of movements [16]. Foraging patches, where the prey of a forager is concentrated in them, are the small areas with a significant travel time between them.

The mapping between concepts of OFT and the OFA algorithm can be shown in Fig. 1.

The animal will forage in the current forging position and its neighborhood after they determine the current position is the patch of the greatest food abundance. As show in Fig. 1(a1), some animal forage in patch I, and other animal forage in patch II, certainly, they also can forage in patch III or patch IV. As the food is consumed, the rate of food intake in the current patch drops to the average rate of the habitat, the animal should leave the current patch. as show in Fig. 1(b1). After entering the new patch, the animal will verify if the prey is the profitable prey or the unprofitable prey? If the prey is the profitable prey, more and more animals will be attracted, as show in Fig. 1(c1).

Different researchers have proposed different foraging models, which aimed to predict foraging behavior [16,18–22]. These mathematical models are comprised of four features: a goal, a currency, a set of constraints, and a set of options. Typically, the goal is to maximize foraging efficiency. The currency is usually a measure of the energy, such as calories. The set of constraints are all limiting factors, such as a set amount of time a forager is able to devote to foraging. The set of options are those choices available to the forager, including choices about how a forager will spend his/her time and choices regarding potential food resources. In these models, the fitness of a foraging animal is a function of efficiency of foraging measured in terms of usually energy. These models are called optimal foraging models and the theory to embody these models is called optimal foraging theory. Therefore, there is a resource aspect inherent in optimal foraging theory. The relationship with resource (or energy) lies in two aspects: 1) in a patch, the amount of the food can reflect the amount of the energy. For example, in Fig. 1(a1), the amount of the prey: a bigger value means a better resource. 2) The quality of the prey, namely the amount of the energy of the prey, like in Fig. 1(a1), some prey are big and some prey is small, the bigger prey has more energy. Each one of the derived three questions that the foraging animal needed to answer during foraging is related with resource (energy). "Where would the animal look for food?" aims to find the patch with the greatest food abundance, namely the greatest resource abundance. "When would the animal transfer foraging patches?" aims to figure out the best time to change the foraging patch to ensure the greatest probability of finding another rich patch with the greatest resource. "What type of food would the animal choose?" is related directly to the energy intake. The best food is the one with the maximum energy.

Since the late 1960′s, numerous models designed to predict the foraging behavior have been developed in order to satisfy different criteria and/or hypotheses. There are some categories of foraging models, such as prey choice, patch choice [23]. These models include: the patch choice model, the diet-breadth model, Pianke and MacArthur model, Glasser's model, Krebs's model et al. [23]

Optimal Foraging Theory has been applied in anthropology and archaeology [24,25]. The last few decades have witnessed a rapid rise in the use of foraging models derived from behavioral ecology to explain and predict temporal and spatial differences in faunal assemblages. The optimal foraging theory appeals to archaeology focused initially on problems in prehistoric human subsistence, especially those posed by patterned variation in archaeological faunal and floral assemblages and related elements of technology. Early archaeological applications date to the 1970s. Much of this work has focused on five general issues: changes in diet breadth among foragers, the origin and diffusion of domesticated plants and animals, links between foraging and technology, constraints on resource transport imposed by central place foraging, and the processes of colonization and competitive exclusion among foragers [24]. Recent applications of foraging models to the zooarchaeological record reveal important variability in human-prey interactions across time and space [25].

4. Optimal Foraging Algorithm

In this paper, the global optimization problem is defined as follows: $\boldsymbol{X} = [x_1, \cdots, x_i, \cdots x_d]^T$ is wanted to satisfy $F(\boldsymbol{X}^*) = \min_{\boldsymbol{X} \in R} f(\boldsymbol{X})$, $R = \{\boldsymbol{X} | x_i^L \le x_i \le x_i^U\}$, $i = 1, 2, \cdots, d$, where $f(\boldsymbol{X})$ is the objective function, $\boldsymbol{X}$, a d-dimensional state vector, one solution of the objective function, x_i, the ith component of $\boldsymbol{X}$, $F(\boldsymbol{X})$, objective function value and $F(\boldsymbol{X}^*)$ is the optimal objective function value, $\boldsymbol{X}^*$ is the optimal vector, namely the optimal solution, R, the constraint space constructed by constraints, x_i^L, x_i^U denote, respectively, the lower and the upper bounds of the ith state vector, and the maximizing optimization problem $f(\boldsymbol{X})$ is equivalent to the minimizing optimization problem $-f(\boldsymbol{X})$.

The process of solving the global optimization problem is similar to that of animal foraging. The neighborhood of the global optimal solution, namely the local optimal solution of function $f(\boldsymbol{X})$ in the constraint space can be seen as the various patches in the animal foraging habitat. The optimization process can be seen as an animal, following Optimal Foraging Theory, forages in the different patches to find the optimal patch where the net rate of energy intake can be maximized. After the optimal patch is found, the animal will search the optimal position within the patch according to the model of best prey. The optimal solution of the global optimization problem is achieved when the final optimal position is found.

Inspired by optimal foraging theory, OFA is represented. In OFA, the individual is regarded as a foraging animal whose position in a patch denotes a solution of the objective function. Corresponding to the resource (or energy) in OFT, the resource (or energy) of OFA also lies in two aspects: 1) the number of the near-optimal solutions. In one area (or patch), the near-optimal solutions only exist in certain position. Therefore, we think the number of the near-optimal solutions in an area is related with resource, a bigger number means a better resource. For example, in Fig. 1(a2), there are many near-optimal solutions in patch I, this patch can be treated as a resource abundance area. 2) The function value of the solution is regarded as the energy of the solution, a bigger value means a higher energy of the solution, and the energy of this position is higher. It means the quality of the solution is much better.

4.1. Where would the algorithm search for the optimal solution?

In OFA, the individual is regarded as a certain foraging animal whose position in a patch denotes a solution of the objective function. This solution is assumed to be the current optimal solution or the optimal foraging position of the individual obtained through foraging. Firstly, the individual must determine where to continue the search. According to OFT, animals always forage in the area of the greatest food abundance and always allocate their most foraging time to the patch of the greatest food [16,26], since the current forging position and its neighborhood is assumed to be the patch with abundant food, the algorithm will search the optimal solution near the current position. The current position and its neighborhood are regarded as an area with many solutions, and these solutions might be the potential near-optimal solutions. Thus the current position and its neighborhood is an area with abundant resource.

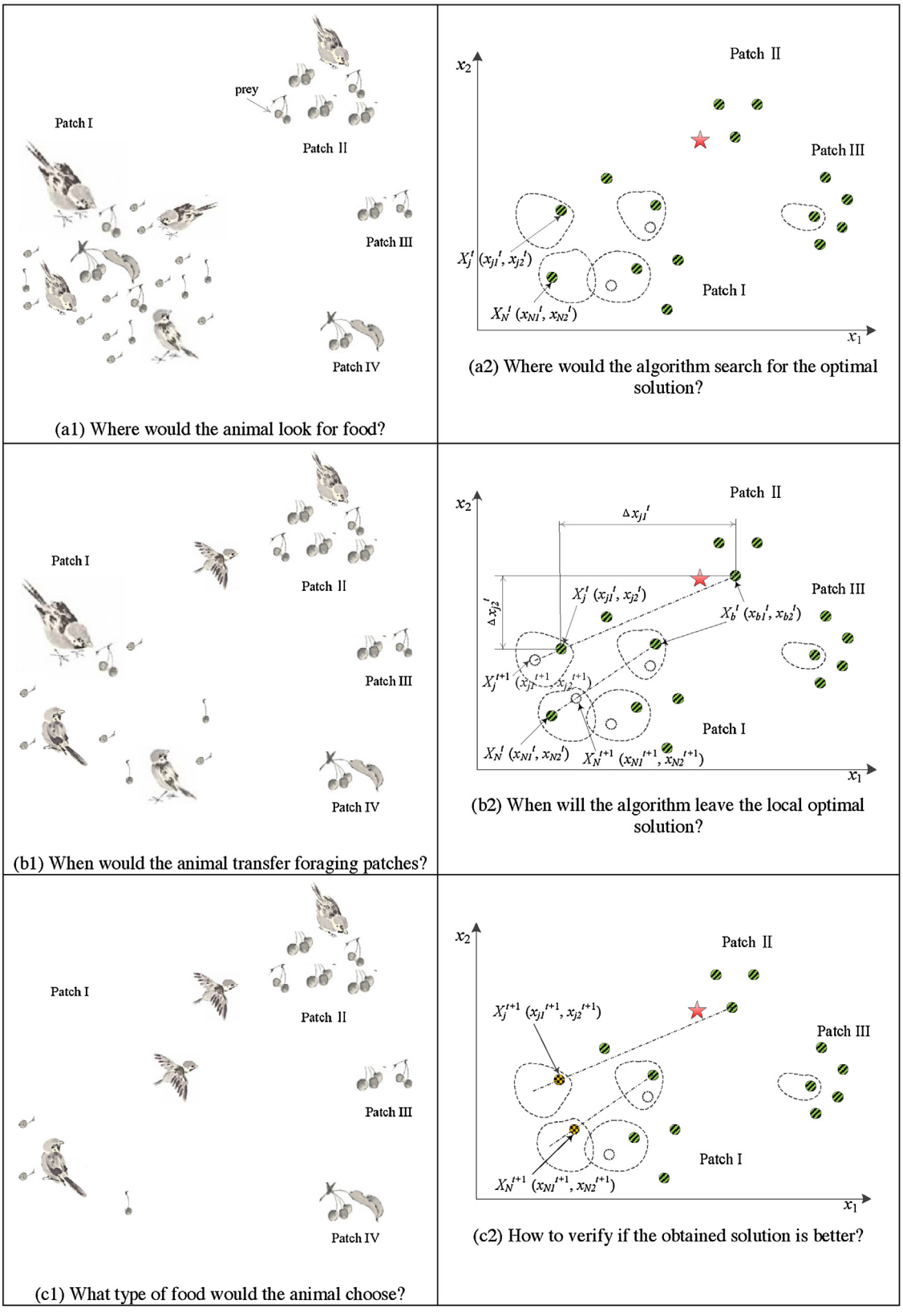

Fig. 1. Optimal Foraging Theory and Optimal Foraging Algorithm.

The animal foraging position is expressed by a two-dimensional variable of longitude and latitude and can move along the two directions respectively, as show in Fig. 1(a2), the cycles filled with oblique line indicate the animal foraging position, those cycles are divided three patches. To go a step further, the position in OFA is expressed by a d-dimensional variable. The method used to search each vector of the better position near the current position is as follows:

$$x_i^{t+1} = x_i^t - k \times r_{1i} \times \Delta x_i^t + k \times r_{2i} \times \Delta x_i^t \quad (t = 1, 2, \cdots; i = 1, 2, \cdots, d) \tag{1}$$

Where x_i^{t+1}, is the ith vector of position obtained after $t+1$ times foraging of an individual animal, and $\boldsymbol{X}^{t+1} = \left[x_1^{t+1}, \cdots, x_i^{t+1}, \cdots x_d^{t+1}\right]^T$, a new position (obtained solution), x_i^t is the ith vector of position obtained after t times foraging, and $\boldsymbol{X}^t = \left[x_1^t, \cdots, x_i^t, \cdots x_d^t\right]^T$, the current position of the individual, k is the scale factor, $r_{1i}, r_{2i} \in [0, 1]$ are the uniform random numbers, Δx_i^t, the ith vector increment of the t times foraging and $\Delta \boldsymbol{X}^t = \left[\Delta x_1^t, \cdots, \Delta x_i^t, \cdots, \Delta x_d^t\right]$ denotes the position increment of the t times foraging, Δx_i^t and $\Delta \boldsymbol{X}^t$ will be introduced in more detail in Section 4.2.

In order to express directly the following opinion: $\boldsymbol{X}^{t+1}$ could appear in any position around $\boldsymbol{X}^t$, which means that individual's new foraging position could be in any random position around the current position $\boldsymbol{X}^t$. Two different random numbers (r_{1i}, r_{2i}) are used in Formula (1), so the vector increment Δx_i^t can be added, subtracted or done nothing to x_i^t. There will be three conditions. (i) If $r_{1i} > r_{2i}$, then, x_i^{t+1} is equal to x_i^t minus $(r_{1i} - r_{2i}) \times k \times \Delta x_i^t$. For example with $x_1$axis, as show in Fig. 1(b2), for position X_j^t, $r_{1j1} > r_{2j1}$, then, x_{j1}^{t+1} is equal to x_{j1}^t minus $\left(r_{1j1} - r_{2j1}\right) \times k \times \Delta x_{j1}^t$, x_{j1}^{t+1} is less than x_{j1}^t. x_{j1}^{t+1} should locate at the left of x_{j1}^t. (ii) If $r_{1i} < r_{2i}$, then x_i^{t+1} is equal to x_i^t plus $(r_{2i} - r_{1i}) \times k \times \Delta x_i^t$. For example, in Fig. 1(b2), for position X_N^t, $r_{1N1} < r_{2N1}$, then x_{N1}^{t+1} is equal to x_{N1}^t plus $(r_{2N1} - r_{1N1}) \times k \times \Delta x_{N1}^t$, x_{N1}^{t+1} is more than x_{N1}^t. x_{N1}^{t+1} should locate at the right of x_{N1}^t. X_j^{t+1} and X_N^{t+1} are showed with the hollow dotted line circle in Fig. 1(b2). (iii) If $r_{1i} = r_{2i}$, then x_i^{t+1} is equal to x_i^t. Combining the three conditions, x_i^{t+1} could locate randomly beside x_i^t, which means a little bit before or after (left or right) x_i^t or sometimes could locate exactly in the x_i^t. New foraging position $\boldsymbol{X}^{t+1} = \left[x_1^{t+1}, \cdots, x_i^{t+1}, \cdots x_d^{t+1}\right]^T$ can be obtained after all d-dimensional x_i^{t+1} have been calculated according to Formula (1). By using Formula (1), the individual of OFA could forage in any arbitrary position besides the current position.

4.2. When will the algorithm leave the local optimal solution and continue the search?

Studies indicate that animals do not allocate all available time to the area of the greatest food abundance during foraging. According to optimal foraging theory, animals always allocate the greatest amount of time to the patch of the greatest food abundance and progressively less time to progressively worse areas [16]. Marginal value theorem of optimal foraging theory thought that an animal should leave a patch when its rate of food intake in the patch drops to the average rate of the habitat [19,21]. Research on gregarious species, such as birds (*Cranes*, *Gnamptogenys moelleri*), show that leaving rich patches earlier than predicted could benefit the birds in the long term, since it would allow them to visit more patches per day than a simple rate-maximizing rule [27], thus the probability of finding another rich patch will be high [27,28].

In establishing OFA, we need to find answers for: when is the best time to leave and how does the individual leave the current local solution? Along with the increasing of the iterations, more and more near-optimal solutions are found, the probability of finding better near-optimal solution is become less, thus it is time to leave to search for other areas with abundant potential optimal solutions, likely the patch with abundant resource (energy). We adopt "recruitment" behavior for this question. Recruitment is often used by some insects during foraging to achieve the best searching efficiency [29,30]. During foraging the cooperation among animals is often needed. When some animals find the prey, the animal will soon recruit other animals; this recruitment behavior ensures animals always tend to the patch of the greatest prey. Therefore, we introduce the concepts of "Group" and "Recruitment" in OFA. "Group" means a fauna composed of N (N = 2,3,. . .) individuals that dispersed in different patches with different quality, whose essence is a set of solutions in the constraint space. The meaning of "Recruitment" can be explained as follows: after searching, the objective function value of each individual's position of a group is calculated, and individuals are ordered based on the values from best to worse. Then the individual with better value recruits the individual with poorer value; thus the individuals will tend to move from the patches with low quality to the patches with high quality. In Fig. 1(b2), $\boldsymbol{X}_b^t$ is the position obtained by the recruit individual b, $\boldsymbol{X}_j^t$ is the position obtained by the recruited individual j.

For OFA with "Group" concept, $\Delta \boldsymbol{X}^t$ in Section 4.1 is defined as the recruitment increment of tth foraging. The recruitment increment is as follows: during tth foraging, after being sorted according to the function value to form a sequence, the individuals scattered in various patches are regarded as a recruited individual in turn. From this sequence an individual with a better value compared with another individual (the recruited individual) is selected randomly as the recruit individual. The recruited individual moves one position increment towards the recruit individual; thus the global information of the better individual is shared by other individuals. The best individual (located in the first place of the sequence) will move one position increment towards the worst one, which obviously is unnatural, but it will enlarge the searching space of the group for OFA and will avoid being entrapped in local optimal solution. Each vector increment Δx_{ji}^t of the recruitment increment $\Delta \boldsymbol{X}_j^t$, $\Delta \boldsymbol{X}_j^t = \left[\Delta x_{j1}^t, \cdots, \Delta x_{ji}^t, \cdots, \Delta x_{jd}^t\right]$, for the jth individual to move from current position X_j^t to the new position $\boldsymbol{X}_j^{t+1}$, needs to be calculated separately. The mathematical description of the ith vector increment Δx_{ji}^t is given:

$$\begin{cases} \Delta x_{ji}^t = x_{bi}^t - x_{ji}^t (F_b^t < F_j^t; F_j^t \neq \min\left(F_1^t, \cdots, F_N^t\right); b = 1, \cdots, N; j = 1, \cdots, N; i = 1, \cdots, d) \\ \Delta x_{ji}^t = x_{Ni}^t - x_{ji}^t \left(F_j^t = \min\left(F_1^t, \cdots, F_N^t\right); F_N^t = \max\left(F_1^t, \cdots, F_N^t\right); i = 1, \cdots, d\right) \end{cases} \tag{2}$$

Where Δx_{ji}^t denotes recruitment increment of jth individual's ith vector during tth search; x_{bi}^t, x_{ji}^t, x_{Ni}^t denote, respectively, during t times foraging, the ith vector of position obtained by the recruit individual b, the ith vector of the position obtained by the recruited individual j and the ith vector of the worst position of the group. F_b^t,F_j^t,F_N^t denote the corresponding objective function values of each position respectively. In Fig. 1(b2), the vector increment Δx_{j1}^t and Δx_{j2}^t are drawn.

Join Formula (2) in Formula (1), and Formula (3) is obtained as follows:

$$\begin{cases} x_{ji}^{t+1} = x_{ji}^{t} - k \times r_{1ji} \times \left(x_{bi}^{t} - x_{ji}^{t}\right) + k \times r_{2ji} \times \left(x_{bi}^{t} - x_{ji}^{t}\right) & \left(F_b^t < F_j^t; F_j^t \neq \min\left(F_1^t, \cdots, F_N^t\right);\right. \\ & \left. b = 1, \cdots, N; j = 1, \cdots, N; i = 1, \cdots, d\right) \\ x_{ji}^{t+1} = x_{ji}^{t} - k \times r_{1ji} \times \left(x_{Ni}^{t} - x_{ji}^{t}\right) + k \times r_{2ji} \times \left(x_{Ni}^{t} - x_{ji}^{t}\right) & \left(F_j^t = \min\left(F_1^t, \cdots, F_N^t\right); i = 1, \cdots, d;\right. \\ & \left. F_N^t = \max\left(F_1^t, \cdots, F_N^t\right)\right) \end{cases} \quad (3)$$

In OFA, each vector x_{ji}^{t+1} of the new position $\boldsymbol{X}_j^{t+1}$ of any individual j can be obtained with Formula (3), which is used to determine when and how the individual will leave the current position. After better positions were found by some individuals, the current individual will search much better positions near these better positions by moving towards one of these individuals. If the current individual is the best individual, then it will search a better position near the current position by moving towards the worst individual.

Each individual of OFA can forage near his own current position by using Formula (3). Moreover, each individual, except the best individual, tends to move to the position of a better individual.

4.3. How to verify if the obtained solution is better?

After all the individuals in the group updated their positions according to Formula (3), one group's foraging behavior has completed. Thus the group's foraging number t is plus 1. Next, the following question must be answered: "Is the obtained new position better or worse and can this new position be used in the following search?" The precise model of prey choices developed by Krebs et al. [18] is adopted in OFA to answer this question whose mathematic expression is given in Formula (4). Weather the solution is better or not depends on the function value of the optimal solution, this value is regarded as the measure of the resource (energy). The better solution with bigger function value means the one with much more energy intake. The function value is used to replace a variable—the net energy gain.

In Krebs's model, it was assumed that only two types of prey existed during animal foraging. They are profitable prey type 1 and unprofitable prey type 2. According to optimal foraging theory, animals always choose the prey with a better energy intake. If the Formula (4) is satisfied, the unprofitable prey type 2 will be ignored by the animal:

$$\frac{\lambda_1 E_1}{1 + \lambda_1 h_1} > \frac{E_2}{h_2} \quad (4)$$

where E_1 denotes the net energy gain of profitable prey type 1; E_2, the net energy gain of unprofitable prey type 2; h_1, the handling time for the profitable prey type 1; h_2, the handling time for the unprofitable prey type 2; and λ_1, the rate of encounter the profitable prey type 1.

In OFA, during judging whether the better position is obtained or not with the precise model of prey choices, the position is seen as the prey in the model; the corresponding function value is seen as the energy of the prey. Whether the position obtained after the $t+1$ times search by individual j is better than that which was obtained after t times search is determined as follows: we hypothesize that a better position can be obtained with the increasing of the searching times, which means the profitable prey is obtained. Thus, the position obtained after $t+1$ times search is seen as the profitable prey. The corresponding function value is seen as the energy of the profitable prey, namely F_j^{t+1} is E_1. For example, the function value F_j^{t+1} corresponding to position $\boldsymbol{X}_j^{t+1}$ is seen as E_1 in Fig. 1(c2). The position obtained after t times search is seen as the unprofitable prey. The corresponding function value is seen as the energy of the unprofitable prey type, namely F_j^t is E_2. For example, the function value F_j^t corresponding to position X_j^t is seen as E_2 in Fig. 1(b2). Formula (4) is deduced from the total gained energy; the time cost for the energy gain is made up of time spent searching for the prey and time spent capturing, killing and eating the prey. In OFA, the time cost for searching for the optimal position is similar to the time spent searching for the prey, which can be expressed by the group's foraging number t, for the essence of the group's foraging number is also a time measured for searching out better positions. Thus the group's foraging number t is seen as the handling time of Krebs's model. For the maximizing optimization problem, in OFA, Formula (4) can be described as:

$$\frac{\lambda_j^{t+1} F_j^{t+1}}{1 + \lambda_j^{t+1}(t+1)} > \frac{F_j^t}{t} \quad (5)$$

For the minimizing optimization problem, the maximizing F is equivalent to minimizing $-F$, then the Formula (5) is redefined as:

$$\frac{\lambda_j^{t+1} F_j^{t+1}}{1 + \lambda_j^{t+1}(t+1)} < \frac{F_j^t}{t} \quad (6)$$

If Formula (6) is satisfied, the position obtained after $t+1$ times search can be used by the individual for the next search. Otherwise, the position obtained after $t+1$ times search will be ignored, and the position obtained after t times search is kept for the next search. $\lambda_j^{t+1} \in [0, 1]$ is a random number. For example, in Fig. 1(c2), the individual N obtains a new position $\boldsymbol{X}_N^{t+1}$ by judging with Formula (6), the individual j continue to stay the old position by judging with Formula (6), $\boldsymbol{X}_j^{t+1}$ is the same position with $\boldsymbol{X}_j^t$.

Uniting Formula (3) and Formula (6), we finally define optimal foraging algorithm for the minimizing optimization problems.

5. Algorithm flowchart

OFA is implemented as follows: first the initial foraging position of each individual is generated randomly in the constraint space; then the objective function value of each individual is calculated and ordered. Next the new foraging position of each individual is obtained with Formula (3); after that the solution represented by the new position is checked with Formula (6) to find out if the solution is better on not. If the solution is better, the new position is preserved for the next foraging, otherwise the new position is ignored and the former position is used for the next foraging. Repeat the above mentioned process; in other words, search positions repeatedly with OFA. The best position obtained by each time of foraging is recorded during the search; thus when the algorithm is terminated, the recorded position is known as the final optimal solution. Corresponding pseudocode of OFA is given in Fig. 2.

From Fig. 2, the number of operations performed by the algorithm is analyzed. The results are in Table 1. The number of operations performed by the OFA is $N \times d + N + NlogN + ((N \times d + N) + N + NlogN) \times Max\_t$, so the time complexity of OFA is $O(N \times d \times Max\_t)$.

For OFA, the storage spaces needed are $O(N \times d)$ in initial phase and the storage spaces needed are $O(N \times d \times Max\_t)$ in the algorithm running phase. The total storage spaces is $O(N \times d) + O(N \times d \times Max\_t) = O(N \times d \times (Max\_t + 1))$.

```
                /* Optimal Foraging Algorithm   */
  /* Initialization. Generate uniformly distributed random group P^l composed N individuals*/
1.  for (j=1; j<=N; j++)
2.    {
3.       for (i=1; i <= d; i++)
4.          {x_ji^1 = x_i^L + rand(0,1) × (x_i^U − x_i^L);    }
5.     }
6.  Compute the objective function value F_j^1 of each individual X_j^1 in the original group P^1;
7.  Order F_j^1 from the best to the worse to obtain F_j^1 (j = 1,2,⋯,N), and the corresponding sequence of
       X_j^1;
8.  t = 1, F_best = F_1^t = min(F_1^t,⋯,F_N^t); F_N^t = max(F_1^t,⋯,F_N^t)
9.  while (t <= Max_t)
10.{
11.     for (j=1; j<=N; j++)
12.       {    if (F_j^t = F_best)
13.                  for (i=1; i <= d; i++)
14.                  {   r_1ji = rand(0,1), r_2ji = rand(0,1);
15.                      x_ji^{t+1} = x_ji^t − k × r_1ji × (x_Ni^t − x_ji^t) + k × r_2ji × (x_Ni^t − x_ji^t)
16.                       if (x_ji^{t+1} > x_i^U)    x_ji^{t+1} = 2 × x_i^U − x_ji^{t+1};
17.                       if (x_ji^{t+1} < x_i^U)    x_ji^{t+1} = 2 × x_i^L − x_ji^{t+1};
18.                       }
19.              else
20.                  {  According to the ordered F_j^t, choose a  value from 1,2,⋯,j−1 randomly
                          as variable b , then get X_b^t and F_b^t < F_j^t.
21.                  for (i=1; i <= d; i++)
22.                   {   r_1ji = rand(0,1), r_2ji = rand(0,1);
23.                       x_ji^{t+1} = x_ji^t − k × r_1ji × (x_bi^t − x_ji^t) + k × r_2ji × (x_bi^t − x_ji^t)
24.                       if (x_ji^{t+1} > x_i^U)    x_ji^{t+1} = 2 × x_i^U − x_ji^{t+1};
25.                       if (x_ji^{t+1} < x_i^U)    x_ji^{t+1} = 2 × x_i^L − x_ji^{t+1};
26.                   }
27.                  }
28.            Compute the objective function value F_j^{t+1} of X_j^{t+1}    ;
29.       }
30.       for (j=1; j<=N; j++)
31.     {    λ_j^{t+1} = rand(0,1) ;
32.            if ((λ_j^{t+1} F_j^{t+1}) / (1+λ_j^{t+1}(t+1)) < F_j^t / t)
33.                  {X_j^{t+1} = X_j^{t+1};   F_j^{t+1} = F_j^{t+1}; }
34.            else
35.                  {  X_j^{t+1} = X_j^t;   F_j^{t+1} = F_j^t; }
36.       }
37.       Order F_j^{t+1} from the best to the worse to obtain F_j^{t+1} (j=1,2,...,N) and the corresponding
            sequence of  X_j^{t+1}.
38.     if (F_1^{t+1} < F_best)
39.       {F_best = F_1^{t+1}; X_best = X_1^{t+1}; }
40.     t = t+1;
41. }
42. Output X_best, F_best;
```

Fig. 2. Pseudocode for Optimal Foraging Algorithm. (*Max_t* is the maximum group's foraging number and ε is the function error value.).

Table 1
The number of operations performed by the OFA.

				Line number in Fig. 2	the number of operations performed by the algorithm T(n)
initializing the group stage				line 1–5	$N \times d$
computing the objective function value of each individual in the original group				line 6	N
Quicksort for initializing the group				line 7	$NlogN$
obtaining the new position of foraging's group		Line number	T(n)		
	obtaining a new foraging position	line 13–18 or line 21–27	d	line 11–29	$N \times d + N$
	computing the objective function value	Line 28	N		
selecting the better solution				line 30–36	N
Quicksort for new the group				line 37	$NlogN$
Algorithm runs with judging algorithm to terminate				Line 9–41	$((N \times d + N) + N + NlogN) \times Max\_t$

6. Compare the performance of seven algorithms

The objective of this experiment is to compare the performance of OFA with several well-known stochastic search algorithms and two optimization algorithms established by imitating the foraging behavior of certain specific organisms. The well-known stochastic search algorithms selected in this paper are: RCGAs [31,32], DE [33], and PSO [4]. The algorithms established by imitating the foraging behavior of certain specific organisms include ACO, BFOA, SFLA and algorithms based on foraging behavior of honey bees, such as Bees Algorithm and Artificial Bee Colony algorithm. ACO is mainly used to solve the discrete optimization problems [3]. And just as mentioned above, Bees Algorithm (BA) and Artificial Bee Colony algorithm are similar [12]. Thus in this paper, BA, BFOA and SFLA are selected together with RCGAs, DE and PSO in performance comparison with OFA. BA, BFOA and SFLA have been introduced in Section 2.

The performance comparisons of OFA with RCGAs, DE, PSO, BA, BFOA and SFLA were carried out with some base benchmark functions and the large-scale global optimization (LSGO) benchmark functions described in the Appendix. The base benchmark function consists of 20 unconstrained single-objective benchmark functions with different characteristics chosen from the literature [33–38]. These functions are divided into three categories of different complexities. Functions f_1-f_6 are unimodal functions that are relatively easy to optimize, but when the problem dimension increases, these functions will become difficult to optimize. Functions f_7-f_{12} are multimodal functions with many local optima, and they represent the most difficult class of problems for many optimization algorithms. Functions f_{13}-f_{20} are likewise multimodal functions, but they only contain a few local optima [33,35,36,39]. The LSGO benchmark functions are chosen from the literature [40]. These functions are divided into four categories of different complexities. Functions f_{21}, f_{22} are the full-separable functions, functions f_{23}, f_{24} are the partially additively separable functions, and function f_{25} is the overlapping function, function f_{26} is the non-separable function. More detailed description and codes for these functions can be found in literature [40].

6.1. Performance evaluation criteria

The performance evaluation criterion must be determined to complete the testing experiments. The criterions are as follows:

(1) *Optimization Error*: The function error for a solution X is $|F(\boldsymbol{X}) - F(\boldsymbol{X}^*)|$, where $\boldsymbol{X}^*$ is the global optimal solution of the function. The maximum group's foraging number is set to be 20,000 for each function to minimize the error. The minimum function error is recorded in each run that is obtained when the group's foraging number goes up to a maximum of 20,000. The average minimum function error are calculated and called *Optimization Error*; meanwhile the standard deviation of *Optimization Error* is calculated. *Optimization Error* embodies the computational accuracy achieved by the algorithm.

In this paper, we decide that the optimal solution is obtained when *Optimization Error* is less than 1.0e-6; the near-optimal solution is obtained when *Optimization Error* is less than 1.0e-1.

(2) The success ration. Two success ration are defined basing on the minimum function error is less than 1.0e-6 or the minimum function error is less than 1.0e-1. The success ration of the optimal solution and the success ration of the near-optimal solution are defined as fellow:

$$SR_{optimal} = \frac{the\ total\ number\ of\ the\ minimum\ function\ error\ less\ than\ 1.0e-6}{The\ total\ run\ times} \times 100\% \quad (7)$$

$$SR_{near-optimal} = \frac{the\ total\ number\ of\ the\ minimum\ function\ error\ more\ than\ 1.0e-6\ and\ less\ than\ 1.0e-1}{The\ total\ run\ times} \times 100\% \quad (8)$$

(3) Based on the statistic method of Kruskal-Wallis test, the experiments data have been analyze to find out if there exist significant differences among these algorithms during solving these benchmark functions.

(4) The evaluation curves of the algorithms: These curves show respectively the average minimum function error of multiple runs of each function in the experiment. Based on the evaluation curves of the algorithms, the converging speed of OFA has been analyzed.

6.2. Algorithm parameters setting

Evolutionary Algorithms (EAs), including genetic algorithms, evolution strategies, evolutionary programming and genetic programming [2], are simple and are the very popular forms of searching and optimization technique [41]. EAs can be seen as the special kinds of generate-and-test algorithms [41]. In this paper, genetic algorithms are adopted as the test algorithms. According to the coded form of solution, there exist two forms of genetic algorithms: binary coded genetic algorithms and real coded genetic algorithms (RCGAs) [32,42]. RCGAs have been shown to outperform binary coded genetic algorithms [43–45]. Thus, RCGAs is selected for the performance comparison with OFA. Differential Evolution (DE) is one of the most recent evolutionary algorithms for solving real-parameter optimization. Like other EAs, DE is a population-based, stochastic global optimizer capable of working reliably in a nonlinear and multimodal environment [31]. The chosen type of DE is DE/*rand*/1/*bin*. Particle Swarm Optimization (PSO) algorithm was developed by Kennedy and Eberhart [4], who were inspired by the social behavior of a flock of migrating birds trying to reach an unknown destination.

In the experiments, the same initial random population is used for the evaluation of RCGA, DE, PSO, BA, BFOA, SFLA and OFA. The dimensions of multi-functions (the function f_1-f_{12}) are set $d = 30$. Each experiment was repeated 50 times with different random seeds for the base benchmark functions. The dimensions of the LSGO benchmark functions (the function f_{21}-f_{26}) are $d = 1000$, and run 25 times with different random seeds for these functions [40].

During the performance comparison, the parameter settings for OFA are as follows: the group size $N = 20$, scale factor $k = t/Max\_t$ and $Max\_t = 20000$ is the maximum group's foraging number.

For RCGA, the *linear crossover* operator and the *non-uniform mutation* operator are adopted. These operators are the most suitable ones for building RCGAs [42]. The selection operator is *normgeomselect,* which is a ranking selection function based on the normalized geometric distribution. The *elitist selection* was also adopted, ensuring that the best performing chromosome always survives intact from one generation to the next [32]. The parameter settings for RCGAs are as follows: similar to OFA, the population size is 20 individuals and the iteration number is 20000. The crossover probability $p_c = 0.85$, the probability of mutation $p_m = 0.3$, the parameter b used by the non-uniform mutation is 5.

The classic DE uses only three control parameters, namely, population size P, scaling factor F, and crossover rate C_r. We chose $F = 1.0$ and $C_r = 0.5$ for all the functions in the experiment without

Table 2
Comparison of computational accuracy for the base benchmark functions. (The best performing algorithm(s) that converged to the near-optimal solution is emphasized in boldface together with underline.).

base benchmark	*Optimization Error* Std dev						
	RCGAs	DE	PSO	BA	BFOA	SFLA	OFA
f_1	6.305	8.568e-17	1.115e-02	2.07e-03	9.609e-02	3.006e-06	3.592e-139
	3.917	6.935e-17	4.547e-03	1.753e-04	6.185e-02	1.965e-05	8.531e-139
f_2	1.158e02	4.865e-11	1.585	2.006e-01	5.596	7.066e-01	1.112e-89
	2.348e01	2.586e-11	2.74	1.058e-02	1.981e01	1.527	2.128e-89
f_3	3.5e05	9.053e-13	4.126e03	4.355	7.581e01	5.301e-03	**1.635e-134**
	2.827e05	9.954e-13	1.409e04	5.0	2.38e02	3.441e-02	5.394e-134
f_4	1.772e01	4.281	5.787	2.995	3.816	2.114e01	**4.239e-05**
	3.838	7.494e-01	1.548	3.944	1.214e01	5.811	1.002e-04
f_5	1.665e02	1.892e01	5.569e01	**9.321e-01**	3.792e01	2.254e01	2.123e01
	7.822e01	8.018e-01	1.52e01	1.197e-01	5.165	6.867	1.668
f_6	2.884	3.676e-02	5.925e-03	**1.16e-03**	1.105	3.944e-01	3.533e-03
	1.628	8.806e-03	5.656e-03	5.738e-04	6.474e-01	2.655e-01	9.251e-04
f_7	4.19e02	**2.369**	1.028e04	8.291e01	4.511e03	6.422e03	5.246e03
	1.795e-06	1.675e01	5.955e02	5.862e01	7.0e02	7.921e02	4.601e02
f_8	7.504e01	**6.062e-01**	3.658e02	1.64	4.569e01	5.645e01	1.314e02
	2.161e01	9.204e-01	3.989e01	5.906	3.504e01	1.585e01	2.054e01
f_9	1.51	2.344e-09	1.023e-01	3.691e-02	3.511e-01	7.07e-01	**4.796e-15**
	5.234e-01	1.205e-09	2.48e-01	2.612e-03	2.367e-01	7.178e-01	1.077e-15
f_{10}	1.478e01	2.958e-04	1.022	**2.733e-04**	2.512e01	6.437e-01	5.577e-04
	8.69	1.464e-03	4.449e-02	1.043e-03	1.021e02	1.747	1.635e-03
f_{11}	2.566e05	**2.356e-08**	1.321e01	9.805e-01	2.181	2.474	2.429e-02
	8.867e05	1.635e-07	6.417	2.032	7.706	3.62	7.092e-02
f_{12}	2.538e06	**3.61e-11**	2.382e01	1.006e01	7.225e-01	9.815	1.247e-03
	3.739e06	5.558e-11	9.261	1.807e01	4.491	1.997e01	3.17e-03
f_{13}	**1.622e-07**	**1.622e-07**	7.743	**1.622e-07**	1.347	4.268	9.989e-02
	2.08e-16	0	1.336	3.833e-15	1.463	4.313	4.131e-01
f_{14}	5.811e-03	**1.401e-08**	1.284e-02	1.134e-07	5.538e-04	2.216e-03	3.958e-05
	8.333e-03	2.297e-19	1.492e-02	7.156e-08	2.903e-04	5.819e-03	3.684e-05
f_{15}	4.535e-07	4.535e-07	1.834e-03	**4.366e-07**	2.775e-06	1.632e-02	1.915e-06
	6.513e-15	2.309e-16	1.297e-02	1.491e-08	2.222e-06	1.154e-01	1.982e-06
f_{16}	5.717e-06	1.752e-03	6.194e-02	3.626e-07	1.637e-06	4.371e-01	**3.577e-07**
	2.671e-05	6.069e-13	3.564e-01	4.29e-09	1.097e-06	8.141e-01	3.763e-11
f_{17}	**3.237e-07**	2.505e-01	4.218	1.71e-04	1.779e-02	4.948	2.564e-03
	1.152e-08	1.263	3.447	1.138e-04	1.044e-03	3.476	6.775e-03
f_{18}	**4.056e-05**	4.057e-05	5.236	1.353e-04	2.131e-02	3.497	1.345e-01
	4.849e-08	8.94e-15	3.237	8.342e-05	9.536e-03	3.714	9.444e-01
f_{19}	**9.773e-06**	9.817e-06	5.625	1.908e-04	1.284e-01	4.007	6.572e-04
	2.734e-07	8.972e-15	3.131	1.25e-04	7.56e-01	3.77	1.357e-03
f_{20}	6.084e-14	**2.975e-14**	3.24	8.452e-07	2.166e-04	5.4e-01	4.384e-10
	2.252e-14	2.328e-15	8.863	9.191e-07	2.289e-04	3.818	1.475e-09

adjusting them to their optimal values for different problems. Similar to OFA, the population size is $P=20$ and the iteration number is 20000.

The PSO uses four control parameters, namely, population Size P, the acceleration constant c_1 and c_2, and the inertia weight factor ω. We chose $c_1 = c_2 = 2.05$ and $\omega = 0.9 - 0.5 \times \left(t/\max\_Iter\right)$ [46] for all the functions in experiment, and the population size is $P=20$ and the iteration number is $Max\_Iter = 20000$.

The parameters of BA that needed to be set are the number of scout bees (n), the number of sites selected out of n visited sites (m), the number of best sites out of m selected sites (e),the number of bees recruited for best e sites (nep), the number of bees recruited for the other (m-e) selected sites (nsp), and the initial size of patches (ngh) that is expressed with the patch radius for neighborhood search [7]. Refer to reference [47], the matlab code was rewritten and the parameters of BA were set. Because every individual of BA has local searches (local iterations) in each iteration, this is obviously not fair for the other comparison algorithms. Therefore we choose smaller nep, nsp compared with the values in reference [47] because nep, nsp have effects on local search. In this paper, parameter settings of BA are as follow: $n=20$, $m=13$, $e=7$, $nep=nsp=5$, $ngh=0.0234$ and the iteration number is 20000.

The BFOA also requires a number of parameters to be set, they are the total number of bacteria in the population (S), the number of chemotactic steps (N_c), the swimming length (Ns), the number of reproduction steps (Nre), the number of elimination-dispersal events (Ned), elimination-dispersal probability (Ped), and the size of the step taken in the random direction specified by the tumble ($C(i)$) [11]. Refer to reference [48], the matlab code was rewritten and the parameters of BFOA were set. Similarly, every individual of BFOA has local search (local iterations) in each iteration, and the parameters (Ns) have effects on local search, thus we choose smaller Ns value than that in reference [48]. Thus the parameters are set as reference [48]. For BFOA, we chose $S=20$, $Ns=Nre=4$, $Ned=2$, $Ped=0.25$, and $C(i)=0.05$, and the iteration number is $Max\_Iter=20000$, then $N_c = Max\_Iter/(Ned \times Nre) = 2500$.

The parameters of SFLA include the frog number in the swarm (n), the number of memeplex (m) or the frogs number in a memeplex (p), number of offsprings ($alpha$), maximum number of local iterations ($beta$), step size ($sigma$). Refer to reference [49], the matlab code was rewritten and the parameters of SFLA were set. Similarly, the individual of SFLA has local search (local iterations) in each iteration, we choose smaller $alpha$, $beta$, $sigma$ compared with the values in reference [49] because $alpha$, $beta$, $sigma$ have effects on local search. For SFLA, we chose $n=20$, $p=4$, $m=n/p$, $alpha=beta=sigma=2$ and the iteration number is 20000.

6.3. Experimental results of the base benchmark functions

The optimization results of RCGAs, DE, PSO, BA, BFOA, SFLA and OFA obtained in the experiments are given in Tables 2 and 3.

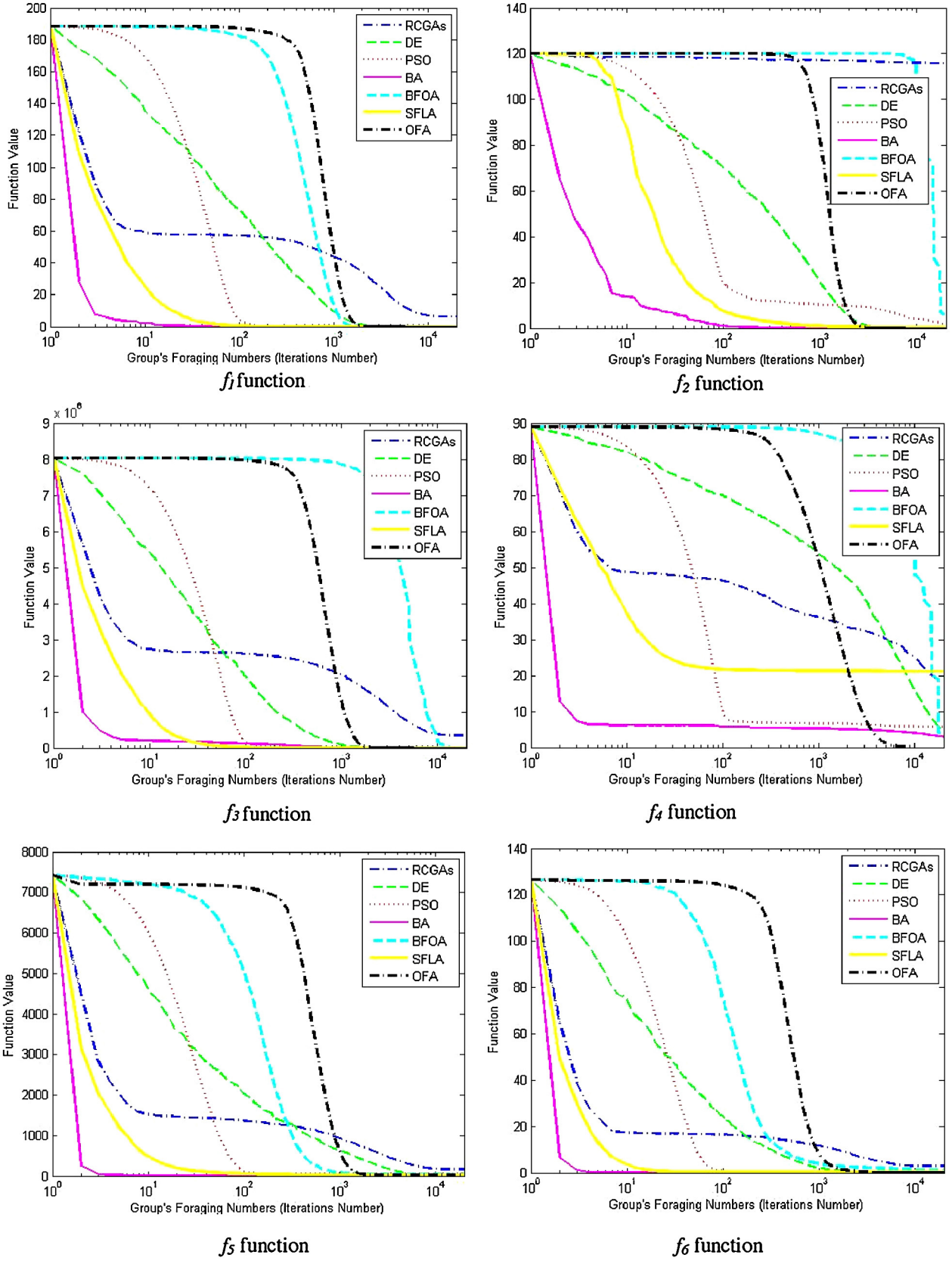

Fig. 3. The evolution curves of benchmark functions f_1-f_6.
(In Fig. 3, the Semi-log scale plot is used in drawing the figure, a base 10 logarithmic scale is used for the horizontal axis and a linear scale for the vertical axis.)

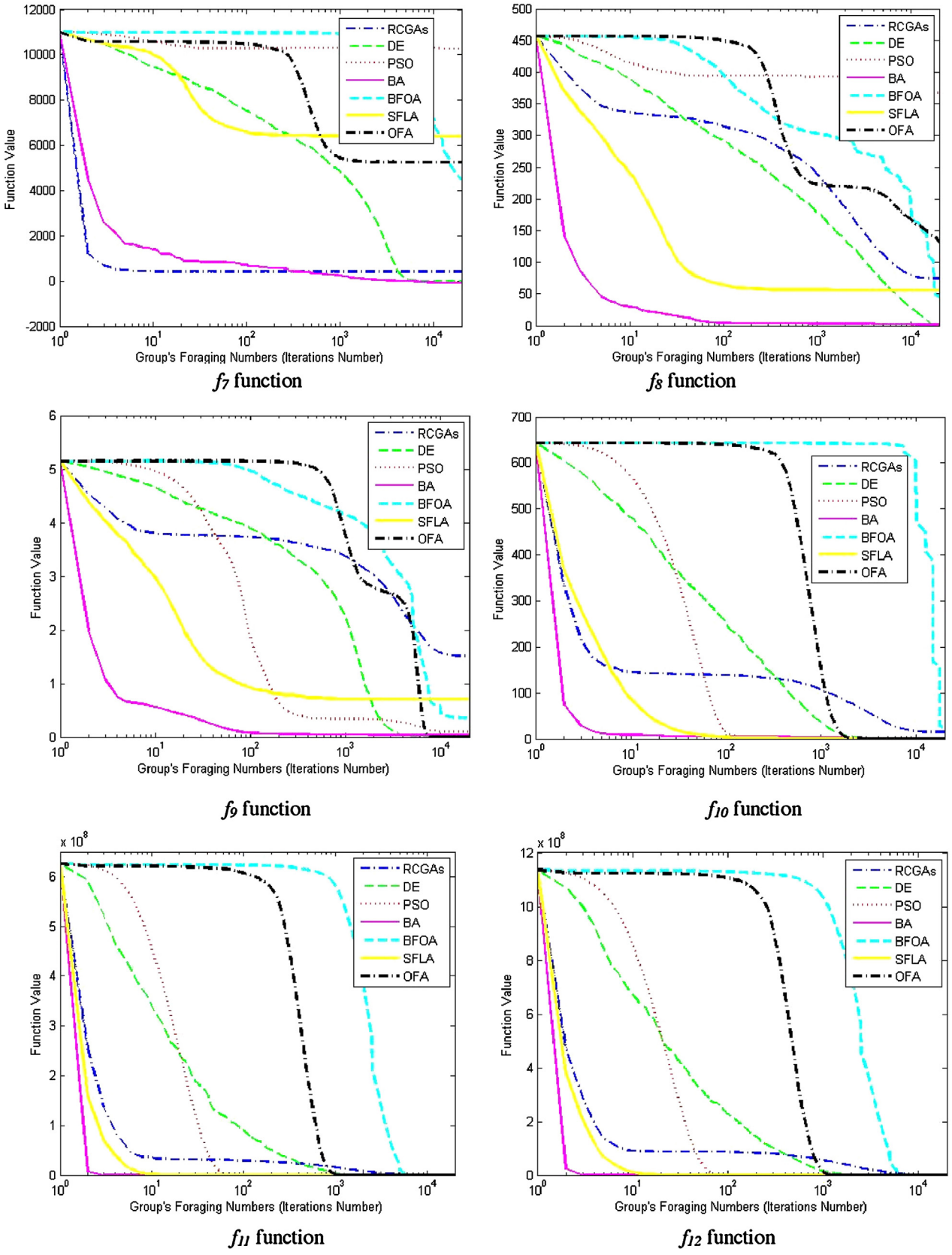

Fig. 4. The evolution curves of benchmark functions *f7-f12*.
(In Fig. 4, the Semi-log scale plot is used in drawing the figure, a base 10 logarithmic scale is used for the horizontal axis and a linear scale for the vertical axis.)

Figs. 3–5 depict the evolution curves of function $f_1 - f_6, f_7 - f_{12}$, and $f_{13} - f_{20}$ respectively.

In general, more optimal solutions or more near-optimal solutions can be obtained with OFA for these twenty base benchmark functions. Combined with the consideration of the computational accuracy, the success ration and the results of statistical significance test, we conclude that OFA is more suitable for solving the unimodal functions, such as functions $f_1 - f_6$. The better optimization result can still be obtained even if the problem dimension increases. DE is suitable for solving the multimodal functions with many local optima, such as functions $f_7 - f_{12}$. BA and RCGAs is suitable for the multimodal functions with fewer numbers of local optima, such as functions $f_{13} - f_{20}$. The analyses are described in the following sections.

From Table 2, the OFA is able to locate the optimal solution or the near-optimal solution for sixteen benchmark functions with relatively small variance, and has difficulty with benchmark functions f_5, f_7, f_8, f_{18}. The DE is able to locate the optimal solution or near-optimal solution for the fifteen benchmark functions except $f_4, f_5, f_7, f_8, f_{17}$. The RCGAs is able to locate the optimal solution or near-optimal solution for the eight benchmark functions $f_{13} - f_{20}$. The PSO is able to converge to the optimal solution or near-optimal solution for the five benchmark functions $f_1, f_6, f_{14} - f_{16}$. The BA is able to locate the optimal solution or near-optimal solution for the twelve benchmark functions except $f_2 - f_5, f_7, f_8, f_{11}, f_{12}$. The BFOA is able to converge to the optimal solution or near-optimal solution for the seven benchmark functions $f_1, f_{14} - f_{18}, f_{20}$. The SFLA is able to converge to the optimal solution or near-optimal solution for the four benchmark functions f_1, f_3, f_{14}, f_{15}. From the view of the computational accuracy, the OFA is the best one.

From Table 3, the OFA is able to locate the optimal solution or the near-optimal solution for sixteen benchmark functions, the success rations of OFA are higher, the sum of success ration, the success ration of the optimal solution plus the success ration of the near-optimal solution, are 100% for these functions except f_{11}, f_{13}. Either the success rations of the optimal solution are 100%, or the success rations of the near-optimal solution are 100%. Even for functions f_{11}, f_{13}, the success ration of the near-optimal solution for f_{11} is 94%, the sum of the success rations of f_{13} is 94%. DE was found having the same conclusion as OFA.

Kruskal-Wallis test is a nonparametric version of one-way analysis of variance (one-way ANOVA). Kruskal-Wallis test is used to

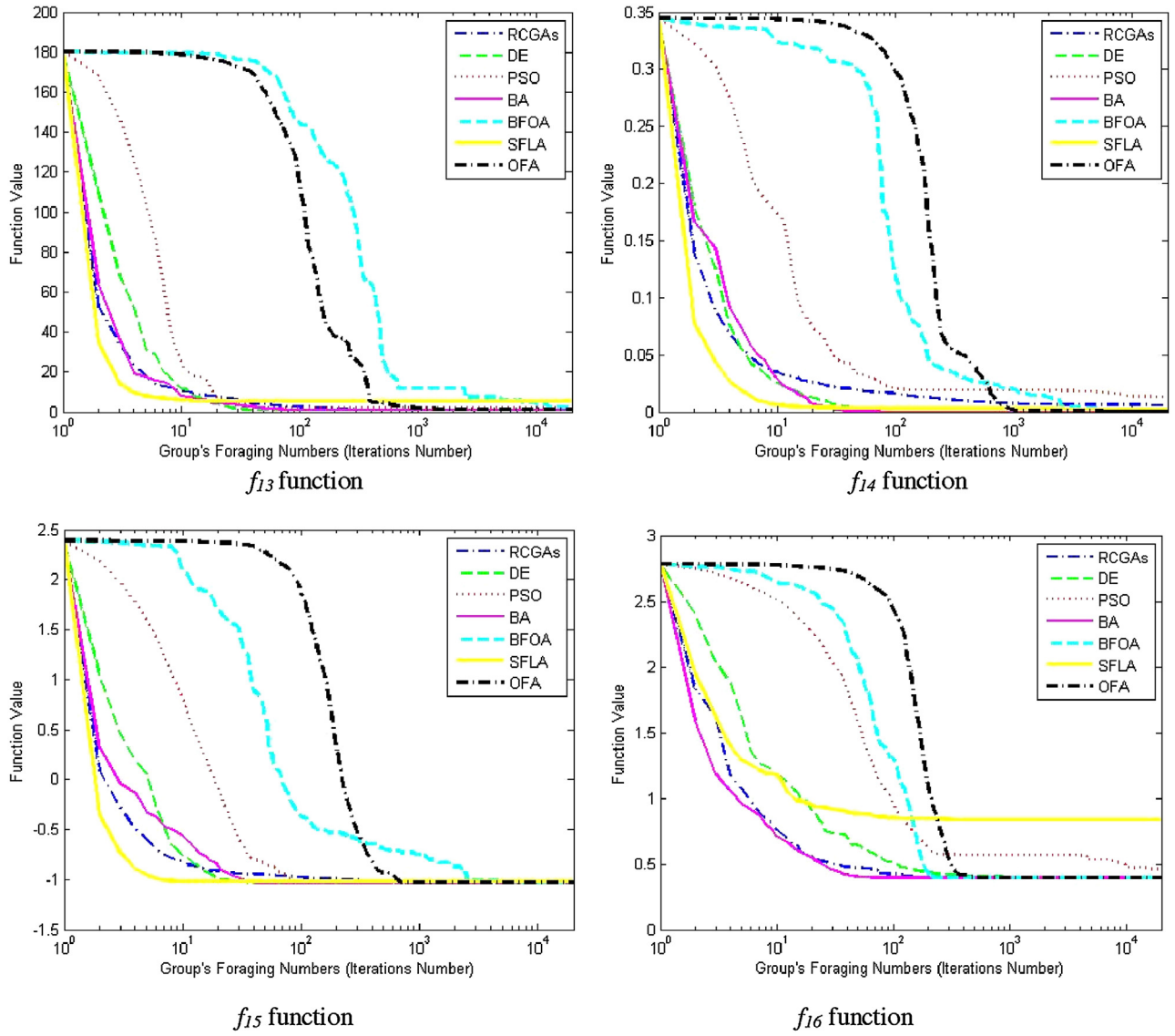

Fig. 5. The evolution curves of benchmark functions $f_{13} - f_{20}$.
(In Fig. 5, the Semi-log scale plot is used in drawing the figure, a base 10 logarithmic scale is used for the horizontal axis and a linear scale for the vertical axis.)

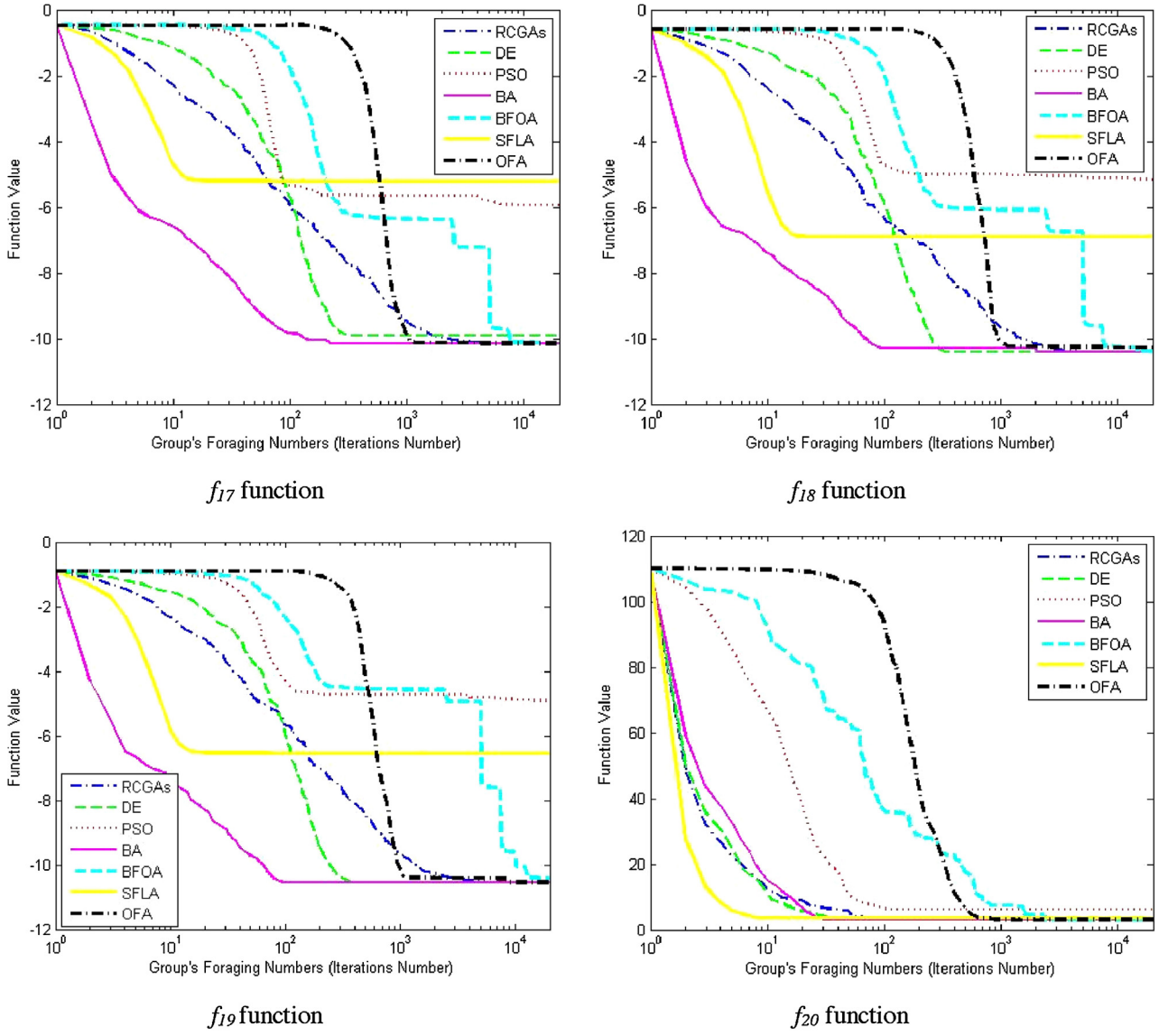

Fig. 5. (*Continued*)

analyze the data of Table 2 to validate the performances of these algorithms and to find out if there are significant differences among these algorithms. For each algorithm, the data obtained by $f_1 - f_{20}$ are handled as one group with Kruskal-Wallis test. The Kruskal-Wallis ANOVA table obtained is shown below (Table 4). This table is a standard ANOVA table, calculated using the ranks of the data rather than their numeric values.

In Table 4, SS denotes the sum of squares, df denotes the degrees of the freedom, MS denotes the mean square. From the results, we can see that p-value (p = 7.68e-06) is less then 0.05, and the null hypothesis is false, so at least one algorithm is significantly different from the others.

As shown in Table 4, there exist significant differences among the experimental data of these algorithms. Thus multiple comparison procedure is carried out to analyze the results of Kruskal-Wallis test, and the estimated value of the mean ranks for each group (algorithm) is achieved. The results are shown in Table 5.

In Table 5, the second row contains the estimated values of the mean ranks for each group, and the third row contains their standard errors. As shown in Table 5, the estimated value of the mean ranks of DE is the smallest one, which means the performance of DE is the best one. The estimated value of the mean ranks of OFA is the second-best one, which means the performance of OFA is the second-best one among these 7 algorithms when solving these twenty base benchmark functions.

6.3.1. *Results analyzing of functions $f_1 - f_6$*

From Table 2, it has been found that OFA outperforms RCGAs, DE, PSO, BA, BFOA and SFLA in terms of computational accuracy for function $f_1 - f_6$. Through analyzing the data of these six unimodal functions with Kruskal-Wallis test, the Kruskal-Wallis ANOVA table obtained is given in Table 6:

From the results, we can see that p-value (p = 0.0117) is less then 0.05. Thus, when handling these unimodal functions, at least one algorithm is significantly different from others. Then, multiple comparison procedure is carried out to analyze the results of Kruskal-Wallis test, and the estimated values of the mean ranks are achieved. The results are shown in Table 7.

Table 7 indicates that the estimated value of the mean ranks of OFA is the smallest one, which means the performance of OFA is the best when handling the unimodal functions.

Table 3
The success ration of the optimal solution and the success ration of the near-optimal solution for the base benchmark functions.

	$SR_{optimal}$ (%) $SR_{near-optimal}$ (%)						
	RCGAs	DE	PSO	BA	BFOA	SFLA	OFA
f_1	0	100	0	0	28	94	100
	0	0	100	100	2	6	0
f_2	0	100	0	0	28	4	100
	0	0	0	0	0	56	0
f_3	0	100	0	0	28	92	100
	0	0	0	0	0	6	0
f_4	0	0	0	0	20	0	6
	0	0	0	38	0	0	94
f_5	0	0	0	0	0	0	0
	2	0	0	0	0	0	0
f_6	0	0	0	0	0	0	0
	0	100	100	100	22	0	100
f_7	0	0	0	0	0	0	0
	0	98	0	98	0	0	0
f_8	0	0	0	0	18	0	0
	0	60	0	0	0	0	0
f_9	0	100	0	0	30	26	100
	0	0	90	100	0	22	0
f_{10}	0	96	0	0	18	6	78
	0	4	0	100	76	34	22
f_{11}	0	98	0	0	0	22	0
	0	2	0	72	92	0	94
f_{12}	0	100	0	0	0	0	0
	0	0	0	36	78	2	100
f_{13}	100	100	30	100	40	14	84
	0	0	32	0	0	0	10
f_{14}	0	100	0	100	0	36	6
	100	0	100	0	100	64	94
f_{15}	100	100	96	100	22	98	38
	0	0	4	0	78	0	62
f_{16}	96	34	90	100	38	0	100
	4	66	4	0	62	56	0
f_{17}	100	96	36	0	0	32	0
	0	0	2	100	100	0	100
f_{18}	0	0	0	0	0	0	2
	100	100	26	100	100	52	96
f_{19}	0	0	0	0	0	0	0
	100	100	22	100	98	46	100
f_{20}	100	100	88	72	0	98	100
	0	0	0	28	100	0	0

Table 4
Kruskal-Wallis ANOVA table.

Source	SS	df	MS	Chi-sq	Prob > Chi-sq
Groups	55436.9	6	9239.49	33.7	7.68e-06
Error	173215.6	133	1302.37		
Total	228652.5	139			

Table 5
The estimated values of the mean ranks.

	RCGAs	DE	PSO	BA	BFOA	SFLA	OFA
Estimated value of the mean ranks	83.175	39.625	95	58.7	81.65	87.4	47.95
Standard errors	9.069	9.069	9.069	9.069	9.069	9.069	9.069

Table 6
Kruskal-Wallis ANOVA Table.

Source	SS	df	MS	Chi-sq	Prob > Chi-sq
Groups	2470	6	411.67	16.41	0.0117
Error	3700.5	35	105.73		
Total	6170.5	41			

Table 7
The estimated values of the mean ranks.

	RCGAs	DE	PSO	BA	BFOA	SFLA	OFA
Estimated value of the mean ranks	34.167	14.667	26	17.833	27.833	20.667	9.833
Standard errors	5.008	5.008	5.008	5.008	5.008	5.008	5.008

Table 8
Kruskal-Wallis ANOVA Table.

Source	SS	df	MS	Chi-sq	Prob > Chi-sq
Groups	2272.67	6	378.78	15.1	0.0195
Error	3897.83	35	111.37		
Total	6170.5	41			

Also, known from the evolution curves in Fig. 3, OFA can quickly converge to the optimal solution with a relatively small group's foraging number. The convergence speed of OFA in the initial 500 iterations (group's foraging numbers) is slow, but after that, OFA converged rapidly. This is because we assume the foraging individual of the algorithm is searching directly at the optimal or near-optimal positions, but since the position of each foraging individual is generated randomly in the beginning, those positions cannot be guaranteed to be the optimal or near-optimal positions. Therefore, our assumption cannot be guaranteed in the initial phase, thus the converging speed of the algorithm is slow. At same time, the scale factor k is set: $k = t/Max\_t$, the parameter k is too small in the initial 500 iterations (group's foraging numbers), k varies from 0.00005 (1/20000) to 0.025 (500/20000), this condition make that the scope of the dotted lines in Fig. 1(a2) is small, the chance that OFA can obtain the optimal or the near-optimal solution is lower. Along with the running of the algorithm, several individuals find the optimal or the near-optimal positions. Together with the help of recruitment, most individuals can search at the optimal or near-optimal positions now. Thereafter, the algorithm converges rapidly.

From above analysis, it has been found that OFA outperforms RCGAs, DE, PSO, BA, BFOA and SFLA when solving the unimodal functions.

6.3.2. *Results analyzing of functions $f_7 - f_{12}$*

Fig. 4 shows typical evolution curves of functions $f_7 - f_{12}$. Here, the convergence speed of OFA is also slow in the initial stage, but becomes rapid after that. For functions f_7, f_8, OFA is entrapped in local optimal solutions.

Through analyzing the data of functions $f_7 - f_{12}$ with Kruskal-Wallis test, the Kruskal-Wallis ANOVA table obtained is given in Table 8.

From the results of Table 8, we can see that p-value (p = 0.0195) is less then 0.05, which means when solving these multimodal functions, at least one algorithm is significantly different from others. Then, the multiple comparison procedure is carried out to analyze the results of Kruskal-Wallis test, and the estimated values of the mean ranks are achieved. The results are shown in Table 9.

Table 9 indicates that the estimated value of the mean ranks of DE is the smallest one, which means the performance of DE is the best when solving the multimodal functions $f_7 - f_{12}$, and OFA is the second-best one.

From above analysis, it has been found that DE outperforms other algorithms and OFA is a strong competitor when solving the multimodal functions $f_7 - f_{12}$.

Table 9
The estimated values of the mean ranks.

	RCGAs	DE	PSO	BA	BFOA	SFLA	OFA
Estimated value of the mean ranks	32.333	8.333	26.333	18.333	24.167	24.333	16.167
Standard errors	5.008	5.008	5.008	5.008	5.008	5.008	5.008

Table 10
Kruskal-Wallis ANOVA Table.

Source	SS	df	MS	Chi-sq	Prob > Chi-sq
Groups	8479.6	6	1413.26	31.88	1.72e-05
Error	6147.9	49	125.47		
Total	14627.5	55			

Table 11
The estimated values of the mean ranks.

	RCGAs	DE	PSO	BA	BFOA	SFLA	OFA
Estimated value of the mean ranks	16.063	18.188	46.5	16	32.125	45.625	25
Standard errors	5.766	5.766	5.766	5.766	5.766	5.766	5.766

6.3.3. Results analyzing of functions $f_{13} - f_{20}$

Fig. 5 depicts the evolution curves of function $f_{13} - f_{20}$. Still, OFA converges slowly at first, then turns to be very fast, except function f_{16}.

Through analyzing the data of functions $f_{13} - f_{20}$ with Kruskal-Wallis test, the Kruskal-Wallis ANOVA table obtained is given in Table 10:

From the results, we can see that p-value (p = 1.72e-05) is less then 0.05, which means when solving these multimodal functions, at least one algorithm is significantly different from others. Then multiple comparison procedure is carried out to analyze the results of Kruskal-Wallis test, and the estimated values of the mean ranks are achieved. The results are shown in Table 11.

Table 11 indicates that the estimated value of the mean ranks of BA is the smallest one, which means the performance of BA is the best one when solving the unimodal functions $f_{13} - f_{20}$, and RCGAs is the second-best one and BA, RCGAs are relatively close, DE and OFA are also strong competitors.

6.4. Experimental results of the LSGO benchmark functions

The optimization results of RCGAs, DE, PSO, BA, BFOA, SFLA and OFA obtained from the experiments of LSGO functions are given in Tables 12 and 13. Fig. 6 depicts the evolution curves of functions$f_{21} - f_{26}$.

From Tables 12 and 13, it can be seen that the LSGO benchmark functions are really a challenge for seven algorithms. No one can get an optimal solution or a near-optimal solution for these six functions. OFA can get the best solutions in five functions $(f_{21} - f_{23}, f_{25}, f_{26})$ comparing with other six algorithms. From Fig. 6, there are five OFA's evolution curves have the similar shape to the OFA's evolution curve of the base benchmark functions.

Table 13
The success ration of the optimal solution and the success ration of the near-optimal solution for LSGO functions.

	$SR_{optimal}$ (%) $SR_{near-optimal}$ (%)						
	RCGAs	DE	PSO	BA	BFOA	SFLA	OFA
f_{21}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
f_{22}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
f_{23}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
f_{24}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
f_{25}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
f_{26}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0

Table 14
Kruskal-Wallis ANOVA Table.

Source	SS	df	MS	Chi-sq	Prob > Chi-sq
Groups	892.08	6	148.681	5.93	0.431
Error	5276.92	35	150.769		
Total	6169	41			

Through analyzing the data of functions $f_{21} - f_{26}$ with Kruskal-Wallis test, the Kruskal-Wallis ANOVA table obtained is given in Table 14.

From the results of Table 14, we can see that p-value (p = 0.431) is more then 0.05, which means when solving these LSGO functions, there is no significantly different among these seven algorithms.

7. Discussion on the parameter of OFA and the dimensions of the multi-functions

The scale factor k is a parameter of OFA. Through the observation for the evaluation curves of OFA and analysis in section 6.3.1, we know that the parameter k has obvious influence on the performance of OFA. In the research of PSO, the dynamic self-adaptive strategy allows for a more efficient search of the solution space and improving algorithm performance near the optima [45]. Two types of the dynamic self-adaptive parameter k are compared for analyzing the influence on the performance of OFA. One type is the utilization of an increasing scale factor $k = t/Max\_t$, this type has been utilized during the performance comparison of several algorithms. The other type is the utilization of a decreasing scale factor $k = 0.9 - 0.5 \times (t/\max\_Iter)$.

The base benchmark functions that belong to three categories are adopted, and the dimensions of multi-functions $d = 30$. the maximum group's foraging number is $Max\_Iter = 20000$, the group size

Table 12
Comparison of computational accuracy for LSGO function.

base benchmark	*Optimization Error* Std dev						
	RCGAs	DE	PSO	BA	BFOA	SFLA	OFA
f_{21}	2.279e11	2.098e11	2.098e11	1.007e11	2.06e11	5.255e10	2.379e07
	1.679e10	0	8.81e-06	8.69e08	4.629e10	8.443e09	5.841e06
f_{22}	5.956e04	4.327e04	4.323e04	3.452e04	5.056e04	4.079e04	1.712e04
	3.608e03	1.26e-11	4.487e01	6.153e02	3.305e04	2.182e03	1.707e03
f_{23}	4.29e07	4.767e07	4.882e07	3.637e07	4.341e07	2.59e07	1.148e07
	5.57e06	1.694e-08	1.36e06	6.451e05	1.606e07	3.625e06	5.351e05
f_{24}	9.426e07	9.579e07	9.61e07	9.211e07	9.513e07	9.317e07	9.432e07
	2.045e05	3.625e05	2.537e05	4.314e05	5.062e05	5.939e05	3.314e05
f_{25}	2.747e12	1.711e12	1.711e12	9.858e11	1.884e12	8.746e11	6.739e10
	2.263e11	6.343e-04	8.324e-04	3.404e10	1.398e12	7.506e10	3.912e10
f_{26}	1.011e16	2.776e11	2.776e11	3.648e10	1.254e15	1.781e14	1.368e08
	3.986e15	1.801e-04	1.643e-04	6.78e10	1.338e13	6.562e13	2.375e07

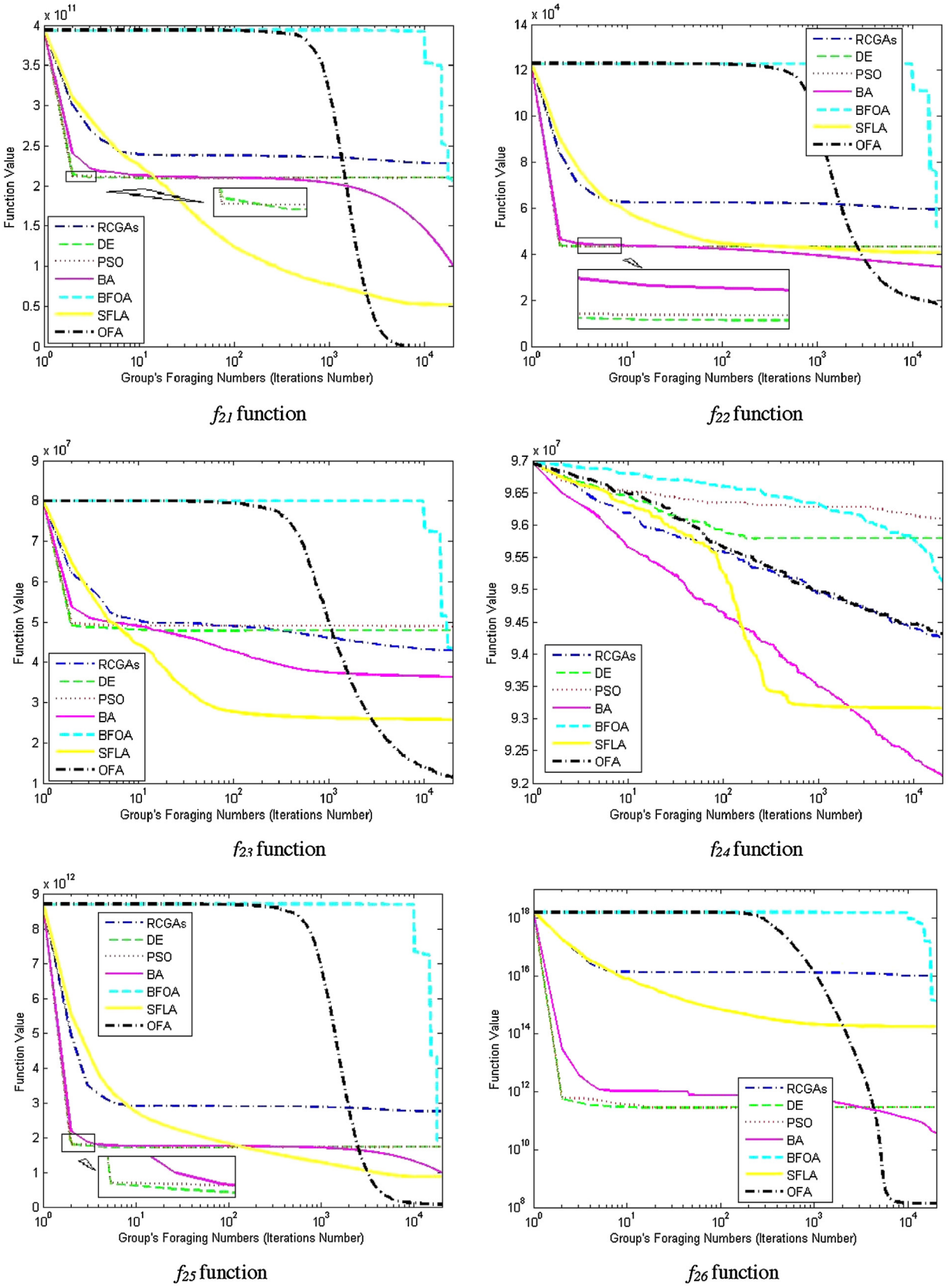

Fig. 6. The evolution curves of benchmark functions $f_{21} - f_{26}$.
In Fig. 6, the Semi-log scale plot is used in drawing the figure, a base 10 logarithmic scale is used for the horizontal axis and a linear scale for the vertical axis. The log-log scale plot is used for the function f_{26}.)

Table 15
The performance of OFA with two types of the dynamic self-adaptive k..

	Optimization Error			*Optimization Error*			*Optimization Error*	
	Increasing $k = t/Max\_t$	Decreasing $k = 0.9 - 0.5 \times \left(t/\max\_Iter\right)$		Increasing $k = t/Max\_t$	Decreasing $k = 0.9 - 0.5 \times \left(t/\max\_Iter\right)$		Increasing $k = t/Max\_t$	Decreasing $k = 0.9 - 0.5 \times \left(t/\max\_Iter\right)$
f_1	9.243e-139	7.578e-180	f_7	5.272e03	4.812e03	f_{13}	7.967e-02	7.947e-01
f_2	1.292e-89	1.16e-117	f_8	1.304e02	5.815e01	f_{14}	4.054e-05	2.38e-04
f_3	4.331e-135	1.575e-175	f_9	4.725e-15	4.441e-15	f_{15}	2.54e-06	4.927e-06
f_4	4.735e-05	1.281e-07	f_{10}	1.189e-03	8.379e-04	f_{16}	3.581e-07	3.578e-07
f_5	2.16e01	1.953e01	f_{11}	6.638e-03	3.169e-04	f_{17}	3.248e-03	3.746e-02
f_6	3.668e-03	2.357e-03	f_{12}	2.803e-03	1.102e-03	f_{18}	1.344e-01	4.346e-02
						f_{19}	1.351e-03	2.477e-02
						f_{20}	8.268e-10	1.88e-09

Table 16
The performance of OFA with the different dimensions of the multi-function. (The best value that converged to the near-optimal solution is emphasized in boldface together with underline.).

base benchmark	*Optimization Error* Std dev			base benchmark	*Optimization Error* Std dev		
	$d = 30$	$d = 50$	$d = 100$		$d = 30$	$d = 50$	$d = 100$
f_1	3.592e-139	2.796e-86	**1.085e-40**	f_7	5.246e03	1.025e04	2.565e04
	8.531e-139	7.422e-86	1.37e-40		4.601e02	6.023e02	7.707e02
f_2	**1.112e-89**	**7.084e-56**	**9.748e-26**	f_8	1.314e02	3.081e02	8.315e02
	2.128e-89	6.582e-56	4.946e-26		2.054e01	4.109e01	7.188e01
f_3	**1.635e-134**	**1.092e-81**	**2.469e-35**	f_9	**4.796e-15**	**7.923e-15**	**1.51e-14**
	5.394e-134	1.428e-81	1.993e-35		1.077e-15	5.024e-16	0
f_4	**4.239e-05**	2.038e-01	1.408e01	f_{10}	**5.577e-04**	**6.221e-04**	**5.011e-04**
	1.002e-04	1.693e-01	1.646		1.635e-03	2.077e-03	2.342e-03
f_5	2.123e01	4.215e01	9.421e01	f_{11}	**2.429e-02**	**9.962e-02**	1.708
	1.668	7.637e-01	8.76e-01		7.092e-02	4.361e-01	3.131
f_6	**3.533e-03**	**1.083e-02**	**5.454e-02**	f_{12}	**1.247e-03**	**2.53e-03**	**2.436e-02**
	9.251e-04	2.11e-03	9.775e-03		3.17e-03	5.114e-03	3.299e-02

$N = 20$. Each experiment was repeated 50 times with different random seeds. The *Optimization Error*s are given in Table 15.

From Table 12, OFA can keep the original characteristics when the parameter utilizes the decreasing type, and the computational accuracy has been improved among 13 functions. The performances of OFA with the decreasing k were improved for the unimodal functions $f_1 - f_6$, especially for $f_1 - f_4$, the obvious improvements are obtained. There are some small improvements for the multimodal functions $f_7 - f_{12}$ when the decreasing k is utilized. The performances of OFA with the decreasing k become poor for functions $f_{13} - f_{20}$ except f_{16}, f_{18}.

The dimensions of the multi-functions is set $d = 30$ in Section 6. In this section, the influence on the performance of OFA is tested by different dimensions of the multi-function. The dimensions of multi-functions is set $d = 50$ and $d = 100$ respectively. The other parameters are the same with section 6. The *Optimization Error*s are given in Table 16.

From Table 16, when $d = 30$, the OFA is able to locate the optimal solution or the near-optimal solution for nine base benchmark functions with relatively small variance; when $d = 50$, the OFA is able to locate the optimal solution or the near-optimal solution for eight base benchmark functions; when $d = 100$, the OFA is able to locate the optimal solution or the near-optimal solution for seven base benchmark functions. Although the *Optimization Error*s became worse with the increasing of the dimensions of the multi-functions, the OFA still demonstrates better optimization ability.

8. Conclusion

Some experiments with benchmark function were carried out to compare the performance of OFA with RCGAs, DE, PSO, BA, BFOA and SFLA. Clearly known from the optimization results obtained in experiments, OFA is better than the other six algorithms in terms of the ability to converge to the optimal or the near-optimal solutions, and the performance of OFA is the second-best one from the view of the statistical analysis. Through comparing the experimental results of the parameters analyzing of OFA and the dimension analyzing of the multi-functions respectively, it can be concluded that the performance of OFA is superior. From the view of the statistical analysis, there is no significantly different among these seven algorithms during solving the LSGO benchmark functions. Although OFA can get the best solutions in the most LSGO functions, but the optimal solution or the near-optimal solution still can't be obtained. When considering the converging ability and the computational accuracy, we find that OFA is more suitable for solving the unimodal functions, such as function $f_1 - f_6$. Upon analysis, it has been found that better performance of OFA owes to the directly improvement of the position of foraging individuals according to the optimal or the near-optimal positions and the higher searching efficiency resulted. With the recruitment increment, the foraging individual can be improved according to other better individuals; thus the global information can be used more comprehensively and extensively. By using the better individual selection model (Krebs's model), the better objective function value, the group's foraging number and the random number are linked together. Thus the superior individual selection of OFA can balanced between the better objective function value and the group's foraging number.

Rooted from Behavioral Ecology, the Optimal Foraging Theory has been successfully applied in anthropology and archaeology [24,25]. Based on the establishing work of OFA with the Optimal Theory, the further research has been done in this paper. We believe that it may become an efficient tool to deal with real-world problems.

On the other hand, some related aspects deserve a deeper analysis in the future. This is the case of the other different models to be tried in OFA, such as the prey choice model and the patch choice model, or the convergence of OFA to be studied basing on a reasonable mathematical model.

Acknowledgments

This work is financed by the natural science foundation of Fujian province (2014J01183), the program of department of science and technology of Fujian province (2016H0015). The authors acknowledge Marci Felbush for correcting the language of this paper.

Appendix A.

The detailed descriptions of twenty base benchmark functions are as follows:

1) f_1 function (Sphere function)

$$f_1(\boldsymbol{X}) = \sum_{i=1}^{d} x_i^2; \text{ with } -5.12 \le x_i \le 5.12; \boldsymbol{X}^* = (0, \cdots, 0); F(\boldsymbol{X}^*) = f_1(\boldsymbol{X}^*) = 0$$

2) f_2 function (Schwefel's problem 2.22)

$$f_2(\boldsymbol{X}) = \sum_{i=1}^{d} |x_i| + \prod_{i=1}^{d} |x_i|; \text{ with } -10 \le x_i \le 10; \boldsymbol{X}^* = (0, \cdots, 0); F(\boldsymbol{X}^*) = f_2(\boldsymbol{X}^*) = 0$$

3) f_3 function (Schwefel's problem1.2)

$$f_3(\boldsymbol{X}) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j \right)^2; \text{ with } -65 \le x_j \le 65; \boldsymbol{X}^* = (0, \cdots, 0); F(\boldsymbol{X}^*) = f_3(\boldsymbol{X}^*) = 0$$

4) f_4 function (Schwefel's problem 2.21)

$$f_4(\boldsymbol{X}) = \max_i \left\{ |x_i|, 1 \le i \le d \right\}; \text{ with } -100 \le x_i \le 100; \text{i} = 1, \cdots, d; \boldsymbol{X}^* = (0, \cdots, 0); F(\boldsymbol{X}^*) = f_4(\boldsymbol{X}^*) = 0$$

5) f_5 function (Rosenbrock's function)

$$f_5(\boldsymbol{X}) = \sum_{i=1}^{d-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right]; \text{ with } -2 \le x_j \le 2; \boldsymbol{X}^* = (1, \cdots, 1); F(\boldsymbol{X}^*) = f_5(\boldsymbol{X}^*) = 0$$

6) f_6 function (Quartic function)

$$f_6(X) = \sum_{i=1}^{d} i x_i^4 + \text{rand}(0, 1); \text{ with } -1.28 \le x_i \le 1.28; \boldsymbol{X}^* = (0, \cdots, 0); F(\boldsymbol{X}^*) = f_6(\boldsymbol{X}^*) = 0$$

7) f_7 function (Schwefel function)

$$f_7(\boldsymbol{X}) = 418.9829n - \sum_{i=1}^{d} \left(x_i \sin \sqrt{|x_i|} \right); \text{ with } -500 \le x_i \le 500; \boldsymbol{X}^* = (420.9687, \cdots, 420.9687); F(\boldsymbol{X}^*) = f_7(\boldsymbol{X}^*) = 0$$

8) f_8 (Rastrigin's function)

$$f_8(\boldsymbol{X}) = \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]; \text{ with } -5.12 \le x_j \le 5.12; \boldsymbol{X}^* = (0, \cdots, 0); F(\boldsymbol{X}^*) = f_8(\boldsymbol{X}^*) = 0$$

9) f_9 (Ackley's function)

$$f_9(\boldsymbol{X}) = 20 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2} \right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d} \cos(2\pi x_i) \right); \quad \text{with} \quad -2 \le x_i \le 2; \quad \boldsymbol{X}^* = (0, \cdots, 0); F(\boldsymbol{X}^*) = f_9(\boldsymbol{X}^*) = 0$$

10) f_{10} (Griewangk's function)

$$f_{10}(\boldsymbol{X}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} + \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1; \quad \text{with} \quad -600 \le x_i \le 600; \boldsymbol{X}^* = (0, \cdots, 0); F(\boldsymbol{X}^*) = f_{10}(\boldsymbol{X}^*) = 0$$

11,12) f_{11} and f_{12} (Generalized Penalized function)

$$f_{11}(\boldsymbol{X}) = \frac{\pi}{d} \left\{ 10sin^2(\pi y_1) + \sum_{i=1}^{d-1} (y_i - 1)^2 \left[1 + 10sin^2(\pi y_{i+1}) \right] + (y_d - 1)^2 \right\} + \sum_{i=1}^{d} u(x_i, 10, 100, 4); \text{ with } -50 \le x_i \le 50; \boldsymbol{X}^* = (-1, \cdots, -1); F(\boldsymbol{X}^*) = f_{11}(\boldsymbol{X}^*) = 0$$

$$f_{12}(\boldsymbol{X}) = 0.1 \left\{ 10sin^3(3\pi x_1) + \sum_{i=1}^{d-1} (x_i - 1)^2 \left[1 + sin^2(3\pi x_{i+1}) \right] + (x_d - 1)^2 \left[1 + sin^2(2\pi x_d) \right] \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4);$$

with $-50 \le x_i \le 50$; $\boldsymbol{X}^* = (1, \cdots, 1)$; $F(\boldsymbol{X}^*) = f_{12}(\boldsymbol{X}^*) = 0$

where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$

13) f_{13} (Foxholes function)

$$f_{13}(\boldsymbol{X}) = \left[0.02 + \sum_{i=0}^{24} \left(i + \sum_{j=1}^{2} \left(x_j - a_{ij} \right)^6 \right)^{-1} \right]^{-1}, \text{ with } a_{i1} = \left\{ -32, -16, 0, 16, 32 \right\} \text{ for } i = 0, 1, 2, 3, 4 \text{ and } a_{i1} = a_{imod5,1} \text{ as well as } a_{i2} = \left\{ -32, -16, 0, 16, 32 \right\} \text{ for } i = 0, 5, 10, 15, 20 \text{ and } a_{i2} = a_{i+k,2}, \text{ with } -65.536 \le x_j \le 65.536; \boldsymbol{X}^* = (-32, 32); F(\boldsymbol{X}^*) = f_{13}(\boldsymbol{X}^*) \cong 0.998004$$

14) f_{14} (Kowalik function)

$$f_{14}(\boldsymbol{X}) = \sum_{i=1}^{11} \left[a_i - \frac{x_i\left(b_i^2 + b_i x_2\right)}{b_i^2} + b_i x_3 + x_4 \right]^2; \quad \text{with} \quad -5 \le x_i \le 5; \text{where}$$

$$a_i = \left\{ 0.1957, 0.1947, 0.1735, 0.16, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.235, 0.0246 \right\}$$

$$b_i^{-1} = \left\{ 0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16 \right\}$$

$\boldsymbol{X}^* = (0.1928, 0.1908, 0.1231, 0.1358)$; $F(\boldsymbol{X}^*) = f_{14}(\boldsymbol{X}^*) = 3.075e-04$

15) f_{15} (Six-Hump Camel Back function)

$$f_{15}(\boldsymbol{X}) = 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4; \quad \text{with} \quad -5 < x_1, x_2 < 5$$

$\boldsymbol{X}^* = (0.0898, -0.7126) = (-0.0898, 0.7126)$; $F(\boldsymbol{X}^*) = f_{15}(\boldsymbol{X}^*) = -1.031628$

16) f_{16} (Branin's function)

$$f_{16}(\boldsymbol{X}) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6 \right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10; \text{ with } -5 < x_1 < 10; 0 < x_2 < 15$$

$\boldsymbol{X}^* = (-\pi, 12.275) = (-\pi, 2.275) = (9.42478, 2.475)$; $F(\boldsymbol{X}^*) = f_{16}(\boldsymbol{X}^*) = 0.397887$

17,18,19) f_{17}, f_{18} and f_{19} (Shekel function)

$$f_{17}(\boldsymbol{X}) = -\sum_{j=1}^{5} \left[\sum_{i=1}^{4} \left(x_i - C_{ij} \right)^2 + \beta_j \right]^{-1}$$

$$f_{18}(\boldsymbol{X}) = -\sum_{j=1}^{7} \left[\sum_{i=1}^{4} \left(x_i - C_{ij} \right)^2 + \beta_j \right]^{-1}$$

$$f_{19}(\boldsymbol{X}) = -\sum_{j=1}^{10} \left[\sum_{i=1}^{4} \left(x_i - C_{ij} \right)^2 + \beta_j \right]^{-1}$$

with

$\beta_j = 0.1 \times [1, 2, 2, 4, 4, 6, 3, 7, 5, 5]^T$

$$C_{ij} = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{bmatrix},$$

$0 < x_1, x_2, x_3, x_4 < 10$

$\boldsymbol{X}^* = (4, 4, 4, 4)$;

$F(\boldsymbol{X}^*) = f_{17}(\boldsymbol{X}^*) = -10.1532$; $F(\boldsymbol{X}^*) = f_{18}(\boldsymbol{X}^*) = -10.4029$; $F(\boldsymbol{X}^*) = f_{19}(\boldsymbol{X}^*) = -10.5364$

20) f_{20} (Goldstein & Price function)

$f_{20}(\boldsymbol{X}) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \times (30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2))$; with $-2 < x_1, x_2 < 2$;

$\boldsymbol{X}^* = (0, 1)$; $F(\boldsymbol{X}^*) = f_{20}(\boldsymbol{X}^*) = 3$

The brief descriptions of six LSGO benchmark functions are as follows, more detailed description and codes for these functions can be found in literature [40].

21) f_{21} function (Shifted Elliptic Function)

$$f_{21}(\boldsymbol{z}) = \sum_{i=1}^{d} 10^{6\frac{i-1}{n-1}} z_i^2;$$

with $= T_{osz}(\boldsymbol{X} - \boldsymbol{X}^*)$; $-100 \le x_i \le 100$; $d = 1000$; $\boldsymbol{X} = (x_1, x_2, \cdots, x_i, \cdots x_d)$; $F(\boldsymbol{X}^*) = f_{21}(\boldsymbol{X}^*) = 0$

22) f_{22} function (Shifted Rastrigin's Function)

$$f_{22}(\boldsymbol{z}) = \sum_{i=1}^{d} \left[z_i^2 - 10\cos(2\pi z_i) + 10\right]$$

with $\boldsymbol{z} = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(\boldsymbol{X} - \boldsymbol{X}^*))$; $-5 \le x_i \le 5$; $d = 1000$; $\boldsymbol{X} = (x_1, x_2, \cdots, x_i, \cdots x_d)$;

$F(\boldsymbol{X}^*) = f_{22}(\boldsymbol{X}^*) = 0$

23) f_{23} function (7-nonseparable, 1-separable Shifted and Rotated Rastrigin's Function)

$$f_{23}(\boldsymbol{z}) = \sum_{j=1}^{|S|-1} \omega_j f_{rasgrigin}(\boldsymbol{z}_j) + f_{rasgrigin}(\boldsymbol{z}_{|S|})$$

$S = \{50, 25, 25, 100, 50, 25, 25, 700\}$; $d = \sum_{j=1}^{|S|} S_j = 1000$; $\boldsymbol{y} = \boldsymbol{X} - \boldsymbol{X}^*$; $-5 \le x_i \le 5$; $\boldsymbol{X} = (x_1, x_2, \cdots, x_i, \cdots x_d)$; $y_j = y(p_{|c_{j-1}+1|} : p_{|c_j|})$, $j \in \{1, \cdots, |S|\}$; $\boldsymbol{z}_j = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(\boldsymbol{R}_j \boldsymbol{y}_j))$; $\boldsymbol{z}_{|S|} = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(\boldsymbol{y}_{|S|}))$, $j \in \{1, \cdots, |S| - 1\}$; $\boldsymbol{R}_j$: a $|S_i| \times |S_i|$ rotation matrix; $F(\boldsymbol{X}^*) = f_{23}(\boldsymbol{X}^*) = 0$

24) f_{24} function (20-nonseparable Shifted and Rotated Ackley's Function)

$$f_{24}(\boldsymbol{z}) = \sum_{j=1}^{|S|} \omega_j f_{ackley}(\boldsymbol{z}_j)$$

$S = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100, 25, 100, 50, 25, 25, 25, 100, 50, 25\}$; $d = \sum_{j=1}^{|S|} S_j = 1000$; $\boldsymbol{y} = \boldsymbol{X} - \boldsymbol{X}^*$; $-32 \le x_i \le 32$; $\boldsymbol{X} = (x_1, x_2, \cdots, x_i, \cdots x_d)$; $y_j = y(p_{|c_{j-1}+1|} : p_{|c_j|})$, $j \in \{1, \cdots, |S|\}$; $\boldsymbol{z}_j = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(\boldsymbol{R}_j \boldsymbol{y}_j))$, $j \in \{1, \cdots, |S| - 1\}$; $\boldsymbol{R}_j$: a $|S_i| \times |S_i|$ rotation matrix; $F(\boldsymbol{X}^*) = f_{24}(\boldsymbol{X}^*) = 0$

25) f_{25} function (Shifted Rosenbrock's Function)

$$f_{25}(\boldsymbol{z}) = \sum_{i=1}^{d-1} \left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right];$$

with $d = 1000$; $-100 \le x_i \le 100$; $\boldsymbol{X} = (x_1, x_2, \cdots, x_i, \cdots x_d)$; $F(\boldsymbol{X}^* + 1) = f_{25}(\boldsymbol{X}^* + 1) = 0$

26) f_{26} function (Shifted Schwefel's Function)

$$f_{26}(\boldsymbol{z}) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_i\right)^2$$

with $d = 1000$; $-100 \le x_i \le 100$; $\boldsymbol{X} = (x_1, x_2, \cdots, x_i, \cdots x_d)$; $\boldsymbol{z} = T_{asy}^{0.2}(T_{osz}(\boldsymbol{X} - \boldsymbol{X}^*))$

$F(\boldsymbol{X}^*) = f_{26}(\boldsymbol{X}^*) = 0$

References

[1] K.A. Dowsland, J.M. Thompson, Simulated Annealing. Handbook of Natural Computing, Vol. 162, Springer, Berlin, Heidelberg, 2012, pp. 3–1655.
[2] K. Sastry, D.E. Goldberg, G. Kendall, Genetic Algorithms. Search Methodologies, Vol. 9, Springer, US, 2014, pp. 3–117.
[3] M. Dorigo, T. Stuzle, Ant Colony Optimization, MIT Press, Cambridge, Mass, 2004.
[4] A. Khare, S. Rangnekar, A. review of particle swarm optimization and its applications in Solar Photovoltaic system, Appl. Soft Comput. 13 (5) (2013) 2997–3006.
[5] A.R. Yildiz, A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations, Appl. Soft Comput. 13 (3) (2013) 1561–1566.
[6] K.M. Passino, Passino Bacterial Foraging Optimization, Innovations and Developments of Swarm Intelligence Applications, IGI Global Press, 2012, pp. 219–234.
[7] E.E. Eldukhri, H. G.Kamil, Optimisation of swing-up control parameters for a robot gymnast using the Bees Algorithm, J. Intell. Manuf. 26 (5) (2015) 1039–1047.
[8] D.W. Stephens, J.S. Brown, R.C. Ydenberg, Foraging: Behavior Chicago Ecology, The University of Chicago Press, 2007.
[9] C.J. Yahnke, Optimal foraging theory, The Amer. Biol. Teacher 68 (October (8)) (2006) 471–475.
[10] W.-B. Zhang, G.-Y. Zhu, A new optimization algorithm based on the optimal foraging theory, ICIC Express Letters (5) (2011) 3967–3972.
[11] S. Das, A. Biswas, S. Dasgupta, A. Abraham, Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications, in Found. of Comput. Intell. Volume 3: Global Opt., in: Stud. in Comput. Intell., Springer Verlag Germany, 2009, pp. 23–55.
[12] H.S. Li, A set of intelligence algorithm based on bee foraging model, Computer and Modernization (1) (2010) 7–10 (In Chinese).
[13] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[14] H.M. Hasanien, Shuffled frog leaping algorithm for photovoltaic model identification, IEEE T. Sustain. Energ. 6 (2) (2015) 509–515.
[15] G.Y. Zhu, W.B. Zhang, An improved Shuffled Frog-leaping Algorithm to optimize component pick-and-place sequencing optimization problem, Expert. Syst. Appl. 41 (15) (2014) 6818–6829.
[16] G.H. Pyke, H.R. Pulliam, E.L. Charnov, Optimal foraging: a selective review of theory and tests, Q. Rev. Biol 52 (June (2)) (1977) 137–154.
[17] P.H. Raven, G.B. Johnson, Biology, Tsinghua University Press, Beijing, 2002.
[18] J.R. Krebs, J.T. Erichsen, Michael I. Webber, Optimal Prey Selection in the great tits (parus major), Anim. Behav. 25 (February (1)) (1977) 30–38.
[19] R.J. Cowie, Optimal foraging in great tits (Parus major), Nature 268 (July) (1977) 137–139.
[20] R.H.E.R. MacArthur Pianka, On optimal use of a patchy environment, Am. Nat. 100 (November–December (916)) (1966) 603–609.
[21] E.L. Charnov, Optimal foraging, the marginal value theorem, Theor. Pop. Biol. 9 (April (2)) (1976).
[22] G.H. Pyke, Optimal foraging theory: a critical review, Ann. Rev. Ecol. Syst 15 (November) (1984) 523–575.

[23] S. Braude, J. Robertson, Modeling Optimal Foraging, in An Introduction to Methods and Models in Ecology, Evolution, and Conservation Biology, Princeton Univ Press, Princeton, NJ, 2010, pp. 226–232.
[24] D.W. Bird, J.F. O'Connell, Behavioral ecology and archaeology, J. Archaeol. Res. 14 (June (2)) (2006) 143–188.
[25] R. Blasco, J. Rosell, J.F. Peris, J.L. Arsuaga, et al., Environmental availability, behavioural diversity and diet: a zooarchaeological approach from the TD10-1 sublevel of Gran Dolina (Sierra de Atapuerca, Burgos, Spain) and Bolomor Cave (Valencia, Spain), Quat. Sci. Rev. 70 (15) (2013) 124–144.
[26] J.R. Krebs, Alejandro Kacelnik, P. Taylor, Test of Optimal Sampling by foraging great tits, Nature 275 (September) (1978) 27–31.
[27] J.C. Alonso, J.A. Alonso, L.M. Bautista, R. Muńoz-Pulido, Patch use in cranes: a field test of optimal foraging predictions, Anim. Behav. 49 (May (5)) (1995) 1367–1379.
[28] M.P. Hassell, T.R.E. Southwood, Foraging strategies of insects, Ann. Rev. Ecol. Syst. 9 (1978) 75–78.
[29] R. Cogni, P.S. Oliveira, Recruitment behavior during foraging in the neotropical ant gnamptogenys moelleri (Formicidae: Ponerinae): does the type of food matter? J. Insect Behav. 17 (July (4)) (2004) 443–458.
[30] J.J. Howard, L.M. Henneman, G. Cronin, J.A. Fox, G. Hormiga, Conditioning of scouts and recruits during foraging by a leaf-cutting ant, Atta colombica, Anim. Behav. 52 (August (2)) (1996) 299–306.
[31] M. Thakur, S.S. Meghwani, H. Jalota, A modified real coded genetic algorithm for constrained optimization, Appl. Math. Comput. 235 (25) (2014) 292–317.
[32] A.M. Sánchez, M. Lozano, P. Villar, F. Herrera, Hybrid crossover operators with multiple descendents for real-Coded genetic algorithms: combining neighborhood-Based crossover operators, Int. J. Intel. Syst. 24 (5) (2009) 540–567.
[33] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Trans. Evol. Comput. 12 (February (1)) (2008) 107–125.
[34] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, Proceeding Congress on Evolutionary Computation (CEC2004) (2004) 1980–1987.
[35] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-Based differential evolution, IEEE Trans. Evol. Comput. 12 (1) (2008) 64–79.
[36] Z. Tu, Y. Lu, A robust stochastic genetic algorithm (StGA) for global numerical optimization, IEEE Trans. on, Evol. Comput. 8 (October (5)) (2004) 456–470.
[37] D.J. Cvijovic Klinowsk, Taboo search: an approach to the multiple minima problem, Science 267 (Feburary (5198)) (1995) 664–666.
[38] J.V.D. Barhen Protopopescu Reister, TRUST: a deterministic algorithm for global optimization, Science 276 (May) (1997) 1094–1097.
[39] J. Brest, S. Greiner, B. Boškovic, et al., Self-Adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.
[40] X. Li, K. Tang, M. Omidvar, Z. Yang, K. Qin, Benchmark Functions for the CEC'2013 Special Session and Competition on Large Scale Global Optimization, Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.
[41] R. Poli, W.B. Langdon, Backward-chaining evolutionary algorithmsf, Artif. Intell. 170 (2006) 953–982.
[42] F. Herrera, M. Lozano, J.L. Verdegay, Tackling real-coded genetic algorithms: operators and tools for behavioural analysis, Artif. Intell. Rev. 12 (4) (1998) 265–319.
[43] D. Kusum, T. Manoj, A new mutation operator for real coded genetic algorithms, Appl. Math. Comput. 193 (2007) 211–230.
[44] A.K. Mostafa, M. Sherif, S. Saraya, F. Areed, Parameter identification problem: real-coded GA approach, Appl. Math. Comput. 187 (2007) 1495–1501.
[45] T. Nedim, Parameter estimation in mathematical models using the real coded genetic algorithms, Expert Syst. Appl. 36 (2009) 3342–3345.
[46] K. Khalili-Damghani, A.R. Abtahi, M. Tavana, A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems, Reliab. Eng. Syst. Saf. (111) (2013) 58–75.
[47] S. Scholz, A.H. Darwish, Basic Bees Algorithm [Online], January, 2012 (Available:) http://bees-algorithm.com/modules/3/1.php.
[48] W. Korani, Bacterial Foraging [Online], January, 2012, Available: http://www.mathworks.com/matlabcentral/fileexchange/20217-bacterial-foraging.
[49] S. Mostapha Kalami Heris, Mar Shuffled Frog Leaping Algorithm (SFLA) [Online], 2016 http:// www. mathworks. com/ matlabcentral/fileexchange/52861-shuffled-frog-leaping-algorithm–sfla-.