Memetic Computing
https://doi.org/10.1007/s12293-019-00286-1

REGULAR RESEARCH PAPER

Birds foraging search: a novel population-based algorithm for global optimization

Zhuoran Zhang[1] · Changqiang Huang[1] · Kangsheng Dong[1] · Hanqiao Huang[2]

Received: 10 October 2017 / Accepted: 29 April 2019

Abstract
Population-based algorithms have become a research hotspot for optimization problems and have been widely applied in various fields in recent decades. This paper presents the birds foraging search (BFS) algorithm, which is a novel population-based optimizer inspired by the different behaviors of birds during the foraging process for solving global optimization problems. The overall framework of the proposed algorithm involves three phases: the flying search behavior phase, the territorial behavior phase and the cognitive behavior phase. In the proposed algorithm, the first two phases balance the exploration and exploitation capabilities of the algorithm, and the third phase enhances the search efficiency. Classical benchmarks and CEC2014 benchmarks are employed to fully evaluate the performance of our BFS. The statistical results reveal that the BFS algorithm outperforms other conventional approaches and state-of-the art algorithms in terms of accuracy and convergence.

Keywords Population-based algorithm · Global optimization · Birds foraging search · Unconstrained optimization problems

1 Introduction

Optimization is the process of searching for the optimal solution for a specific problem. Mathematically, the optimal solution usually refers to the maximum or minimum of a function. Traditionally, deterministic algorithms based on numeric linear and nonlinear programming methods have been the only methods available for solving optimization problems. Because of the massive exploitation of gradients, deterministic algorithms are capable of obtaining the global optimum in certain simple and ideal systems. However, with the increasing complexity of practical problems in recent decades, the flaw in which deterministic algorithms become caught in local optima has become more prominent. Deterministic algorithms have difficulty finding global optimal solutions when applied to optimization problems with a large number of local solutions and constraints. Therefore, new optimization algorithms must be studied to overcome the drawbacks of traditional methods.

Stochastic algorithms are a class of optimization methods that search for global optimal solutions using random operators. The random strategy can effectively help the algorithms to escape the local optimum. In contrast to deterministic algorithms, stochastic algorithms do not rely on substantial gradient information during the optimization process; instead, these algorithms evaluate objective functions multiple times. Therefore, from a stochastic perspective, optimization problems are generally equivalent to black boxes, which is why these algorithms are more widely applied and are more suitable for solving an unknown search space than are traditional methods. Due to the abovementioned advantages, stochastic algorithms have replaced deterministic algorithms to become the main methods for solving modern optimization problems.

In general, stochastic algorithms can be divided into two categories: single-based and population-based algorithms. As the name implies, single-based algorithms generate only one random solution that is updated throughout the optimization process. Hill climbing [24] and simulated annealing (SA) [13] are two typical single-based algorithms. Hill climbing is a simple greedy search algorithm that selects

✉ Hanqiao Huang
cnxahhq@gmail.com

Zhuoran Zhang
zhunran1009@163.com

[1] Aeronautics and Astronautics Engineering College, Air Force Engineering University, Xi'an 710038, China

[2] Unmanned System Research Institute, Northwestern Polytechnical University, Xi'an 710072, China

Published online: 07 May 2019

an optimal solution from the neighborhood of current solutions until a local optimal solution is reached. SA simulates the annealing process of solid materials and can accept a solution that is worse than the current one with a certain probability. This mechanism helps to enhance the exploration capability of SA.

Although single-based algorithms have the advantages of lower computational cost and fewer function evaluations, they tend to converge prematurely. Population-based algorithms have a superior ability to avoid local optima at the expense of increased computational costs and function evaluation times. In general, these costs are acceptable. Thus, population-based algorithms are more suitable for solving modern complex optimization problems than are single-based algorithms.

In contrast to single-based algorithms, population-based algorithms create multiple solutions randomly and improve them over the course of optimization. The most used algorithms are the genetic algorithm (GA) [9], particle swarm optimization (PSO) [6] and ant colony optimization (ACO) [5]. GA is based on Darwin's theory of biological evolution and the survival of the fittest. In the algorithm, optimization is treated as a process of biological evolution, and the population is improved via selection, crossover and mutation, with poorer individuals phased out and superior individuals retained and allowed to reproduce. PSO is based on the flocking behavior of birds. Each particle position is updated based on its individual best position and the global optimum position found by the swarm, and current velocity information is also incorporated. The ACO is inspired by the foraging behavior of ants. Ants transfer information among individuals, which enables them to identify the shortest path between the nest and food source. According to the source of inspiration, algorithms can generally be classified into three principal categories: evolutionary, based the laws of natural selection; swarm intelligence, based on the behavior of groups of animals in nature; and physics, based on the physical rules of nature. Some of the classic and novel algorithms of each subclass are as follows:

Evolutionary: differential evolution (DE) [25], self-organizing centroid optimization (SOC-opt) [3], and monkey king evolution (MKE) [21].

Physics-based: gravitational search algorithm (GSA) [23], electro-search (ES) [27], and thermal exchange optimization (TEO) [12].

Swarm intelligence: artificial bee colony (ABC) [11], cuckoo search (CS) [32], firefly algorithm (FA) [31], grey wolf optimizer (GWO) [18], spider monkey optimization (SMO) [2], moth-flame optimization (MFO) algorithm [16], crow search algorithm (CSA) [1], whale optimization algorithm (WOA) [17], moth search (MS) [28], and satin bowerbird optimizer (SBO) [19].

Other algorithms have different sources of inspiration, such as teaching-learning-based optimization (TLBO) [22], joint operations algorithm (JOA) [26], and kidney-inspired algorithm (KA) [10].

The above algorithms share the basic idea that they gradually narrow the search range and constantly improve the accuracy based on an approximate global optimal solution, which is found by performing a wide range of searches. The main difference among the algorithms stems from the different degrees of emphasis on exploration and exploitation. Exploration embodies the global search capability of the algorithm, and exploitation represents the local search capability around the near-optimal solution. However, these two capabilities are conflicting, with one bound to weaken if the other is strengthened. For example, certain algorithms select only the individuals with better performance during the optimization process and ignore individuals with worse performance. The excessive pursuit of exploitation ability compromises the performance of algorithms through a lack of knowledge of population diversity. However, for any optimization problem, the exact tradeoff between exploration and exploitation in the algorithm is unknown. Therefore, identifying a superior method for balancing exploration and development remains the focus of optimization algorithms. Furthermore, the conflict between exploration and exploitation is especially pertinent when considering large-scale problems that adversely affect the efficiency of the algorithm. Therefore, in addition to effectively balancing exploration and exploitation, an adjustment strategy is required for a successful algorithm. This strategy can adjust the search direction of the individual through the individual's own search experience or through the gradient information of the objective function to improve the overall search efficiency.

Although new algorithms are constantly emerging and being applied in various engineering fields, a fundamental question remains: is it meaningful to propose additional algorithms? The answer to this question is affirmative. According to the no-free-lunch theorem [30], none of the algorithms can be applied to all optimization problems. From this perspective, the no-free-lunch theorem can be regarded as a conservation law: if the algorithm performs well for certain issues, it will inevitably perform poorly for other problems. Namely, the average performance of an optimization method is the same if all optimization problems are considered. Thus, a number of specific optimization problems must be addressed by new algorithms rather than current optimization techniques. Therefore, a novel population-based algorithm inspired by the behavior of birds during the foraging process, called birds foraging search (BFS), is proposed. This algorithm works in three phases: exploration, exploitation and mutation. The main contribution of this paper is to propose a new population-based algorithm to further solve the global optimization problem. Compared

with existing methods, BFS has no redundant control parameters and is thus easy to implement and to adapt to a wide variety of applications. Furthermore, BFS has advantages in terms of accuracy, convergence rate and stability.

The rest of this paper is organized as follows. Section 2 discusses the analogy between birds foraging and population-based algorithms. Section 3 describes the mathematical model of the BFS algorithm in detail. Section 4 describes the experimental setting and demonstrates the experimental results. Section 5 presents conclusions and suggestions for further work.

2 Comparison of birds foraging and population-based algorithms

Birds live and breed on all seven continents and in most terrestrial habitats. Approximately 10,000 species of birds are found worldwide. Foraging behavior accounts for the majority of their daily activity: birds need to constantly search for food to survive and adapt to changing habitats. In general, three types of behavior are observed during the foraging process: flying search behavior, territorial behavior, and cognitive behavior. First, birds rely on their flight advantages to conduct extensive reconnaissance of the ground from the air to discover new food. Raptors are typical birds with long, broad wings and sharp vision, and they use air currents for long circling flight to extensively search for prey in vast airspace [20]. Second, most birds, such as red-winged blackbirds and white-bellied antbirds, present territorial behavior [7, 33]. This process can be described as follows: certain birds in the group will occupy the area as their habitat or rut after discovering rich resources, and when other birds approach the territory, the individuals that occupy the area will drive them away by twittering to protect their territory. Finally, certain birds, such as hummingbirds and crows, exhibit cognitive behavior during foraging—a result of strong memory and learning abilities [8, 29]. For example, hummingbirds can perform targeted searches based on prior experience, which helps them judge the amount of nectar in flowers according to shape or color. If, based on the birds experience, a flower is judged to be of high quality, then the hummingbird will further exploit this flower; otherwise, the hummingbird will quickly leave and search for the next target. Thus, the efficiency of food searching is improved.

An appropriate tradeoff of exploration and exploitation is the key to obtaining powerful optimization capabilities for all population-based algorithms. Exploration is employed to enhance the ability to jump out of local optima by searching new regions using a randomization method. Exploitation is applied to improve the convergence speed of the population-based algorithms by identifying a better solution near the current optimal solution. With the development of human society, a number of extremely complex practical optimization problems continue to emerge, and "efficiency" is one of the major challenges. These problems include complicated objective functions with a large number of decision variables. Moreover, the optimal solutions for certain nonlinear constrained optimization problems with many local optima are difficult to determine, and the solving processes consume large amounts of time, which leads to low efficiency. In such situations, simply balancing exploration and exploitation is not sufficient, and an adjustment strategy for the algorithms based on previous gradient information is often required to improve the searching efficiency.

A comparison between the foraging process and the structure of population-based algorithms provides a number of interesting analogies. When explicit food information is missing, birds must expand their search area and cover as much of the search space as possible to avoid missing valuable information. Thus, flying search behavior has a role similar to that of exploration. When a bird occupies a territory, other birds will obtain directional information, which attracts them to the target area, and the search area is gradually reduced to the target area. This behavior is similar to the exploitation ability of algorithms. Cognitive behavior is similar to the adjustment strategies of algorithms. Birds will improve the search efficiency of the overall algorithm according to previous search experience, which further improves the local exploitation ability based on the balance of exploration and exploitation.

In short, these similarities provide new ideas for the novel BFS algorithm presented here. Concrete details of the BFS algorithm will be presented in the next section.

3 Birds foraging search algorithm

According to the previous analysis, the BFS can be divided into three phases: flying search behavior phase, territorial behavior phase and cognitive behavior phase.

The BFS aims to find the optimal food source using individuals. The position of the optimal food source is represented by the optimal solution, and the position of each bird should be equivalent to the position of the corresponding food source, which is a feasible solution of the optimization problem. The amount of food provided by a food source (the value of the fitness function) represents the quality of the solution. The number of birds is equivalent to the number of feasible solutions. Similar to other optimization algorithms, initialization is performed before implementing the three phases. The initial positions of the birds in space are represented as follows:

$$X_i = UB - r_1 \cdot (UB - LB) \tag{1}$$

where X_i is the *i-th* position of the bird, UB and LB represent the upper and lower bounds of the variables, respectively, and r_1 is a random value in the range [0, 1].

Before the three phases are elaborated, four ideal rules of the BFS are listed as follows:

1. A bird in the algorithm does not refer to a particular type of bird but represents an artificial bird with the characteristics of all aforementioned birds.
2. All birds live in flocks.
3. Only the current best individual possesses territory.
4. The territory is regarded as a space particle whose position is the same as that of the current best individual.

3.1 Flying search behavior phase

Raptors, such as eagles, often hover over an area with the potential to contain abundant food resources. After intensive study, researchers have determined that the flight path of a raptor is similar to the shape of a logarithmic spiral [15]. The following logarithmic spiral formula [16] can be used to describe this flight path:

$$X_i^{iter+1} = D_{i-p} \cdot e^{\theta} \cdot \cos(2\pi\theta) + PA^{iter} \tag{2}$$

where i is a random index selected from $[1, 2, 3, \ldots, N]$, with N representing the population size; θ is a random number in the range $[-1, 1]$; $D_{i-p} = \left| X_i^{iter} - PA^{iter} \right|$ indicating the distance of the *i-th* bird from the potential area, with X_i^{iter} representing the position of the *i-th* bird at the *iter-th* iteration; and PA^{iter} is the position of the potential area at the *iter-th* iteration, which indicates the current best solution, X_{best}^{iter}. The new position will replace the old position if it has a better fitness value.

As shown in Fig. 1, the value of θ can be used to define the proximity of the next position of the bird to the potential area ($\theta = -1$ is the closest position to the potential area, and $\theta = 1$ is the farthest). In detail, the bird will fly around the potential area instead of simply approaching it in a straight line (as shown in B and D). The most distinguishing characteristic of this movement is that each bird can fly along any direction in the potential area. In other words, the next position of the individual can be any direction from the best solution, which means that the next position is outside (as shown in C) or inside (as shown in E) the space between the current position and the potential area. Therefore, the flight mode has strong global search capability and simultaneously guarantees the exploitation capability of the algorithm.

3.2 Territorial behavior phase

In this phase, the current best individual is called the territory bird, and the other birds are called incursion birds. These two types of birds have different types of movement. The territory bird patrols small areas around its territory to find better food sources and protect its territory from the approach of other birds. This behavior can be formulated as follows:

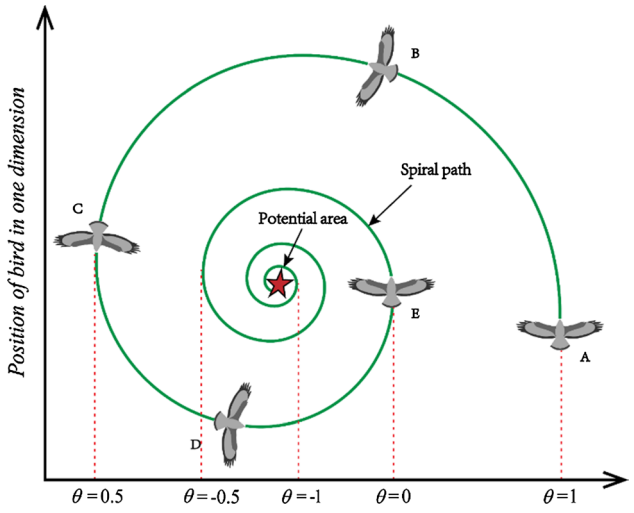


Fig. 1 Logarithmic spiral updating position in flying search behavior

$$X^{T,iter+1} = X^{T,iter} + r_d \cdot \lambda \tag{3}$$

where $X^{T,iter}$ is the position of the territory bird in the *iter -th* iteration; r_d is a random value in the range $[-1, 1]$, which represents the direction of the search; and λ is a scale factor that lets the bird move slightly around its current position. Here, λ is set to $(X^{T,iter} - X^{S,iter})$, where $X^{S,iter}$ is the position of the current suboptimal individual. $X^{T,iter+1}$ is accepted if it provides a better fitness value.

After an area is occupied by the territory bird, all incursion birds will move toward the territory bird, and the territory bird will drive other birds away by calling to defend its territory. This case includes two states.

State 1: The incursion bird j is not affected by the warning of the territory bird and moves quickly to the territory bird, and its position is updated as follows:

$$X_j^{I,iter+1} = X_j^{I,iter} + r_2 \cdot (X^{T,iter} - IF \cdot X_j^{I,iter}) \tag{4}$$

where $X_j^{I,iter}$ is the position of the *j-th* incursion birds at the *iter-th* iteration; r_2 is a random value in the range $[0, 1]$; and IF is the incursion factor that determines the positions of incursion birds to be changed. Figure 2 provides a vector representation of this state using different IF values. As shown in Fig. 2, IF=1 results in a local search ($X_j^{I,iter+1}$ lies between $X_j^{I,iter}$ and $X^{T,iter}$), whereas IF=2 leads to a global search ($X_j^{I,iter+1}$ is outside of $X_j^{I,iter}$ and $X^{T,iter}$). To balance the exploration and exploitation capabilities, the value of this factor is 1 or 2 and represents a heuristic step that is selected randomly with equal probability as $IF = round[1 + rand(0, 1)\{2 - 1\}]$.

State 2: The warning of the territory bird produces the intended effect. The incursion bird j is frightened and flees

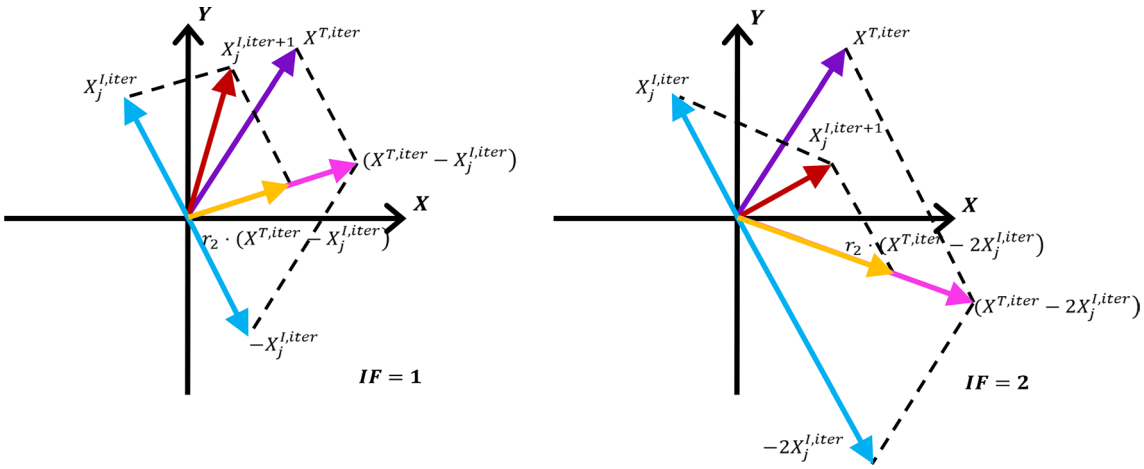


Fig. 2 Vector representation of an incursion bird's movement with different incursion factors in state 1

to the search space randomly. This process is described by the following formula:

$$X_{j,d}^{I,iter+1} = X_{j,d}^{I,iter} + r_3 \cdot (X_{k,d}^{I,iter} - X_{m,d}^{I,iter}) + r_4 \cdot (X_{l,d}^{I,iter} - X_{h,d}^{I,iter}) \tag{5}$$

where $j,k,m,l,h \in \{1,2,3,\cdots,N-1\}$, $j \neq k \neq m \neq l \neq h$, $d \in \{1,2,3,\cdots,D\}$, with D indicating the dimension of the problem; and r_3 and r_4 indicate random values in the range $[0,1]$.

Overall, states 1 and 2 can be expressed as follows:

$$\begin{cases} X_j^{I,iter+1} = X_j^{I,iter} + r_2 \cdot (X^{T,iter} - IF \cdot X_j^{I,iter}) & if\ P^{iter} \leq rand \\ X_{j,d}^{I,iter+1} = X_{j,d}^{I,iter} + r_3 \cdot (X_{k,d}^{I,iter} - X_{m,d}^{I,iter}) + r_4 \cdot (X_{l,d}^{I,iter} - X_{h,d}^{I,iter}) & otherwise \end{cases} \tag{6}$$

where P^{iter} is the probability that the warning is effective, and *rand* is a random value in the range $[0,1]$. Over time, incursion birds will gradually discover the territory bird's bravado. In addition, the physical strength of the territory bird will also decline. Therefore, the warning will gradually lose efficacy. Accordingly, P^{iter} is determined as follows:

$$P^{iter} = 1 - \frac{iter}{Max - iter} \tag{7}$$

P^{iter} decreases linearly with iteration from 1 to 0. From the perspective of optimization, P^{iter} enables the algorithm to perform a global search with a larger probability in the early stages and then gradually reduces the search space and emphasizes exploitation in the later stages. Thus, P^{iter} balances the exploration and exploitation abilities of the algorithm. Finally, the incursion bird j updates its position only if the new position is better; otherwise, its position remains unchanged.

In addition, a role-switching mechanism is implemented throughout the search process. If an incursion bird finds a better area than that found by other incursion birds and the current territory bird, then in the next searching round, the incursion bird becomes the new territory bird and the original territory bird joins the team of incursion birds. This role-switching mechanism helps the algorithm avoid becoming trapped in local optima.

3.3 Cognitive behavior phase

In general, cognitive behavior is a process of self-learning that is based on accumulated historical experience; thus, birds can avoid unnecessary searching and improve their foraging efficiency based on experience gained from previous searches. To implement this process, we compare the current and last retained position information, the results of which constitute the experience for the next search. The entire cognitive behavior phase includes two states.

State 1: Birds continuously find better sources of food [i.e., the previous two positions are different ($X_i^{iter} \neq X_i^{iter-1}$)]. This state indicates that the search direction is correct and promising. As a result, birds will learn according to the original gradient information. This targeted search can accelerate the convergence rate of the algorithm. The following formula illustrates this situation:

$$X_i^{iter+1} = X_i^{iter} + r_5 \cdot (X_i^{iter} - X_i^{iter-1}) \tag{8}$$

where X_i^{iter} and X_i^{iter-1} are the positions of the bird at the *iter -th* iteration and the *iter*−1-*th* iteration, respectively; and r_5 is a random value in the range $[0,1]$.

State 2: Birds continuously search but fail to find better results [i.e., the previous two positions are the same ($X_i^{iter} = X_i^{iter-1}$)]. The search route is changed randomly because experience tells the birds that the original search direction is no longer applicable (the algorithm may fall into local optima). The process is implemented based on a Gaussian distribution:

$$X_i^{iter+1} = Gaussian(X_{best}^{iter}, \xi) \tag{9}$$

The objective of formula (9) is to enable the birds to seek feasible positions within a region determined by the best individual. $Gaussian(X_{best}^{k}, \xi)$ is a random number sampled from a Gaussian distribution with a mean value of X_{best}^{k} and standard deviation of ξ. X_{best}^{k} is the best solution in the *k-th* iteration, and ξ is calculated as follows:

$$\xi = (\log(iter)/iter) \cdot abs(X_{best}^{iter} - r_6 \cdot X_i^{iter}) \tag{10}$$

where r_6 is a random value in the range [0, 1] that is used to enhance the distribution of new individuals when X_{best}^{iter} is equal to X_i^{iter}; and $\log(iter)/iter$ is used to adjust the size of the standard deviation. The value of $\log(iter)/iter$ decreases as the number of iterations increases, and the standard deviation decreases accordingly. As a result, this process leads to a more concentrated stochastic distribution and a slight disturbance around the global best solution, which improves the local search capability. Similarly, X_i^{iter+1} is accepted only if its fitness value is better than that of X_i^{iter}.

In addition, certain individuals may search beyond the borders. To prevent this invalid search, we introduce a border control strategy as follows:

$$X_{i,d}^{iter} = UB - r_7 \cdot (UB - LB) \quad if \ \ X_{i,d}^{iter} < LB \ \ or \ X_{i,d}^{iter} > UB \tag{11}$$

where $X_{i,d}^{iter}$ is the *d-th* dimension of the *i-th* solution at the *iter-th* iteration, and r_7 is a random value in the range [0, 1]. The overall procedure of the standard BFS algorithm is shown in Algorithm 1.

Algorithm 1 Overall procedure of the BFS

Set the values of parameters UB, LB, N, and Max_iter;
Initialize the bird population with random positions in the problem space;
$iter = 1$;
while ($iter \le Max\_iter$) **do**
  // Execute flying search behavior phase
    for $i = 1$ to N **do**
      Generate X_i^{iter+1} by the spiral flight according to Eq. (2);
      Check the boundary;
      Evaluate the new positions of the birds;
      Update X_i with X_i^{iter+1};
    end for
  // Execute territorial behavior phase
      /* territory bird */
      Generate the new position of territory bird $X^{T,iter+1}$ using Eq. (3);
      Check the boundary;
      Evaluate the fitness of function for $X^{T,iter+1}$
      Update X^T with $X^{T,iter+1}$;
      /* incursion birds */
      Calculate P^{iter} via Eq. (7);
    for j=1 to N-1 **do**
      Generate the new position of incursion birds $X_j^{I,iter+1}$ via Eq. (6);
      Check the boundary;
      Evaluate the fitness of the function for $X_j^{I,iter+1}$;
      Update X^I with $X_j^{I,iter+1}$;
    end for
    if the position of an incursion bird is superior to all other birds **then**
      execute the role change mechanism
    end if
  // Execute cognitive behavior phase
    Merge population $X = \{X^T, X^I\}$
    for $i = 1$ to N **do**
      Calculate the standard deviation ξ using Eq. (10);
      if $X_i^{iter} \ne X_i^{iter-1}$
        Generate X_i^{iter+1} using Eq. (8);
      else
        Generate X_i^{iter+1} using Eq. (9);
      end if
      Check the boundary;
      Evaluate the fitness function of X_i^{iter+1}
      Update X_i with X_i^{iter+1};
    end for
    $iter = iter + 1$;
end while
Output the position of the best individual in the entire population.

4 Salient features of BFS

Our BFS algorithm is similar to previously proposed algorithms in certain respects. For example, similar to most population-based algorithms, the BFS represents the candidate solution by only its position. Moreover, during the flying search behavior phase, the spiral flight strategy is inspired by the MFO algorithm. Finally, during the territorial behavior phase, we are inspired by "DE/rand/2" to simulate the random escape of incursion birds. However, as a new optimization algorithm, the BFS presents certain salient features that differentiate it from existing algorithms. First, although certain algorithms, such as the PSO and CSA algorithms, also imitate bird foraging behavior, the BFS algorithm is different from these algorithms in terms of both the overall structure and the specific mathematical model. The BFS uses a flying search behavior phase, territorial behavior phase, and cognitive behavior phase to search for the optimal solution, which represents the first application of existing algorithms. Second, many population-based algorithms contain several parameters that need to be tuned before running. For example, the ABC should be tuned to a predetermined cycle (limit). The SMO algorithm considers the following four parameters: perturbation rate, local leader limit, global leader limit and maximum group. By contrast, specific control parameters do not need to be adjusted for the BFS. In other words, compared with previous algorithms with multiple control parameters, the BFS is easy to implement and is adaptable to a wide variety of optimization problems, which represents one of the great advantages of BFS. Third, the search steps of many algorithms, such as the CS, CSA and SBO algorithms, are fixed constants. In the BFS, when an individual moves closer to the optimal area, we use the intrusion factor [reflected in Eq. (4)] instead of the traditional fixed search step to balance exploration and exploitation. Fourth, from the population update perspective, the position update equations of many algorithms, such as DE, ABC and SMO, are based on the difference vector. The BFS is updated in a variety of ways. In addition to the difference vector, a spiral search mode and Gaussian variation are included. Multiple update methods also increase the probability that the BFS will escape the local optimum. Moreover, to improve the search efficiency, we are inspired by the cognitive behavior of the birds and use the individual's own gradient information to improve the overall exploitation ability of the algorithm, which is a novel concept. The abovementioned salient features of the BFS demonstrate that it is different from existing algorithms.

5 Experimental study

To verify the performance of the BFS algorithm, experimental studies were performed on classical and modern challenging benchmark functions in this section. All the experiments are conducted in MATLAB 2016a in a Windows 10 environment using a 2.60 GHz Intel(R) Core(TM) i7-6700HQ computer with 16 GB of RAM.

5.1 Experiment I: classical benchmarks

In total, 13 standard benchmark functions are used to test the relative performance of the BFS algorithm on classical benchmark functions. These functions are the classical functions utilized by many researchers [1, 16, 18, 19]. These functions are characterized as being unimodal, multimodal, separable and nonseparable. Nonseparable functions are more difficult to handle than separable functions because each variable in a function is related to all other variables. The difficulty of a problem also increases with the function's dimensionality. Some functions represented by f_{08} are highly deceptive because the global optimum of the function is far from the suboptimal value, and the algorithm is often induced to converge in the wrong direction. Some functions represented by f_{10} have global optimal values on the edge. If the algorithm does not have strong exploration capability, the global optimal value is difficult to find. In addition, some functions with flat surfaces, such as f_{05} and f_{06}, are difficult to handle because the flatness of the function does not provide the algorithm any useful information from which to find the global optimum. In this section, to examine the exploration and exploitation of algorithms, these benchmark functions are divided into two respective categories: unimodal (f_{01}–f_{07}) and multimodal (f_{08}–f_{13}). Unimodal signifies that the function has only one global optimal value, which can be used primarily to test the exploitation ability of the algorithm. Multimodal signifies that the function has many local optimal values, which can be used to investigate the exploration ability of the algorithm. Table 1 summarizes the 13 benchmarks, including the cost functions, bounds of variables and optimal values.

Our BFS algorithm is compared with 6 well-known algorithms that have been widely used in various fields: PSO [6], DE [25], ABC [11], CS [32], GSA [23], and FA [31]. The maximum number of function evaluations is set to 300,000 for f_{01}–f_{13}. Each algorithm is performed independently for the test functions 30 times. The control parameter settings for the compared algorithms in this test are as follows:

1. PSO: $\omega = 0.6$ as a weight factor and $c_1 = c_2 = 2$ as in [4];
2. DE: $F = 0.5$ and $CR = 0.9$ as in [25];

Table 1 Description of the classic benchmark functions (U: unimodal, M: multimodal, Dim: dimension)

Test function	Name	Type	Dim	Range	Optimum
$f_{01}(x) = \sum_{i=1}^{D} x_i^2$	Sphere	U	30	[−100,100]	0
$f_{02}(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i-1}^{D} \lvert x_i \rvert$	Schwefel 2.22	U	30	[−10,10]	0
$f_{03}(x) = \sum_{i=1}^{D} (\sum_{j-1}^{D} x_i)^2$	Schwefel 1.2	U	30	[−100,100]	0
$f_{04}(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le D\}$	Schwefel 2.21	U	30	[−100,100]	0
$f_{05}(x) = \sum_{i=1}^{D} 100(x_{i+1}^2 - x_i^2)^2 + (x_i - 1)^2$	Rosenbrock	U	30	[−30,30]	0
$f_{06}(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	Step	U	30	[−100,100]	0
$f_{07}(x) = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	Quartic	U	30	[−1.28,1.28]	0
$f_{08}(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$	Schwefel 2.26	M	30	[−500,500]	−418.9829*D
$f_{09}(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	Rastrigin	M	30	[−5.12,5.12]	0
$f_{10}(x) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i))$	Ackley	M	30	[−32,32]	8.8818E−16
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} (x_i^2) - \left(\prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}})\right) + 1$	Griewank	M	30	[−600,600]	0
$f_{12}(x) = \frac{\pi}{D}\{10\sin^2(\pi y_i) + \sum_{i-1}^{D-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)\}$ $+ \sum_{i-1}^{D} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < a \end{cases}$	Penalized	M	30	[−50,50]	0
$f_{13}(x) = 0.1\{\sin^2(3\pi x_i) + \sum_{i=1}^{D} (x_i - 1)^2[1 + \sin^2(3\pi x_i)]$ $+(x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i-1}^{D} u(x_i, 5, 100, 4)$	Penalized2	M	30	[−50,50]	0

3. ABC: $limit = (N \cdot D)/2$ and size of employed bee=onlooker bee=(colony size)/2 as in [11];
4. GSA: $G_O = 100$ and $\alpha = 20$ as in [23];
5. CS: $\beta = 1.5$ and $p_a = 0.25$ [32];
6. FA: $\alpha = 0.2$, $\beta_0 = 1$, and $\gamma = 1$ [31].

For a fair comparison, all the algorithms should have equal function evolutions for the same number of maximum iterations. In terms of the structure of the algorithms, the BFS and CS possess three phases and two phases, respectively, and the other algorithms have only one phase. Therefore, the population size of the BFS is set to 50, the population size of the CS is set to 75, and the population sizes of the remaining algorithms are set to 150. As a result, all the algorithms require the same number of function evolutions for each iteration.

Table 2 summarizes the results obtained by applying the BFS and the six abovementioned algorithms to unimodal functions (f_{01}–f_{07}). To perform a comprehensive and reliable comparison, each algorithm is evaluated using the best, worst, and mean solution and the corresponding standard deviations based on 30 independent runs. For convenience, the best and mean solutions and the standard deviations are represented by "Best", "Mean", and "SD", respectively. The algorithms are ranked from the smallest to largest mean value. Moreover, we calculate the average ranks and obtain the overall rank to reflect the performance of each algorithm more intuitively. The best results among all algorithms for each function are shown in bold text. Table 2 shows that the BFS outperforms the PSO, DE, ABC, CS, GSA and FA for f_{01}–f_{04}, f_{06}, and f_{07}, whereas its performance is not superior for f_{05}. The findings show that the BFS ranks first among the algorithms. The ABC algorithm shows the best performance for f_{05}, and the BFS presents the second-best performance for this function. The main purpose of investigating unimodal functions is to examine the exploitation capability of the algorithms. Thus, the convergence rate is more important for the algorithm than the final result. Figure 3 depicts the graphical results of all studied algorithms for 6 typical functions and shows that the BFS has a faster convergence rate than do the other algorithms for four test functions.

The experimental results obtained for PSO, DE, ABC, CS, GSA, FA, and BFS on multimodal functions (f_{08}–f_{13}) over 30 independent runs are presented in Table 3. Compared with unimodal functions, multimodal functions contain many local minima whose number increases exponentially as the problem size increases. Therefore, such functions are often used to test the global search capability of algorithms. As shown in Table 3, the BFS shows excellent performance for multimodal functions. For the 6 evaluated multimodal functions, the BFS achieves the best results for all test functions except f_{08} and f_{13}. The DE presents promising performance

Table 2 Optimization results of the unimodal functions for the comparative algorithms

No.	Algorithms	PSO	DE	ABC	CS	GSA	FA	BFS
f_{01}	Best	2.1716E−17	1.4248E−18	2.9980E−16	2.4660E−08	8.5209E−19	1.2266E−04	**0**
	Mean	1.4526E−15	6.9757E−18	4.9870E−16	4.4598E−08	1.3877E−18	2.0370E−04	**0**
	SD	2.0154E−15	7.0852E−18	5.7425E−17	1.2951E−08	2.6339E−19	5.6303E−05	**0**
	Rank	5	3	4	6	2	7	**1**
f_{02}	Best	5.8263E−13	3.2188E−09	3.2104E−15	6.7795E−03	5.0313E−09	7.3512E−03	**0**
	Mean	2.1091E−11	2.2182E−08	5.6489E−15	1.5681E−02	6.3681E−09	2.2786E+01	**0**
	SD	5.6615E−11	1.5994E−08	1.4900E−15	6.0143E−03	6.3507E−10	2.6957E+01	**0**
	Rank	3	5	2	6	4	7	**1**
f_{03}	Best	1.3213E+02	8.6812E−02	2.4495E+03	1.5999E+01	4.5699E+00	1.1675E+02	**0**
	Mean	7.3949E+02	3.6704E−01	5.1018E+03	2.9246E+01	1.9786E+01	9.1026E+02	**0**
	SD	3.6974E+02	1.9231E−01	1.4537E+03	6.3781E+00	8.7016E+00	5.0145E+02	**0**
	Rank	5	2	7	3	4	6	**1**
f_{04}	Best	8.5334E+00	6.2512E−04	1.7590E+00	2.9942E−01	4.5974E−10	1.9980E+00	**0**
	Mean	2.4814E+01	1.4940E−03	4.5136E+00	4.9345E−01	5.7234E−10	1.1571E+01	**0**
	SD	1.1988E+01	5.1795E−04	1.2316E+00	1.1839E−01	6.2144E−11	6.2446E+00	**0**
	Rank	7	3	5	4	2	6	**1**
f_{05}	Best	4.5515E+00	5.9117E+00	**1.2502E−04**	1.5752E+01	2.3917E+01	2.4718E+01	4.1218E+00
	Mean	7.7561E+01	8.0091E+00	**5.8235E−02**	2.2681E+01	2.4101E+01	2.3358E+02	8.3044E+00
	SD	1.1785E+02	1.1341E+00	**6.2685E−02**	2.2462E+00	9.5012E−02	4.9060E+02	2.0869E+00
	Rank	6	2	**1**	4	5	7	3
f_{06}	Best	7.3194E−18	3.7132E−24	3.3286E−16	1.8040E−08	1.0282E−18	1.1271E−04	**2.4868E−30**
	Mean	5.3961E−15	5.6084E−23	4.9752E−16	4.8657E−08	1.4739E−18	1.9748E−04	**4.8226E−25**
	SD	1.9034E−14	5.2095E−23	4.6401E−17	2.1338E−08	2.6821E−19	5.8394E−05	**2.3743E−24**
	Rank	5	2	4	6	3	7	**1**
f_{07}	Best	9.1002E−03	4.5807E−03	4.3468E−02	7.4703E−03	1.0275E−03	6.3340E+01	**8.2898E−05**
	Mean	2.2682E−02	8.0445E−03	1.1754E−01	1.7640E−02	2.1091E−03	7.8710E+01	**1.5482E−03**
	SD	8.7490E−03	2.3733E−03	2.4078E−02	4.9905E−03	7.3508E−04	2.9029E+00	**1.0312E−03**
	Rank	5	3	6	4	2	7	**1**
Average rank		5.1429	2.8571	4.1429	4.7143	3.1429	6.7143	**1.2857**
Overall rank		6	2	4	5	3	7	**1**

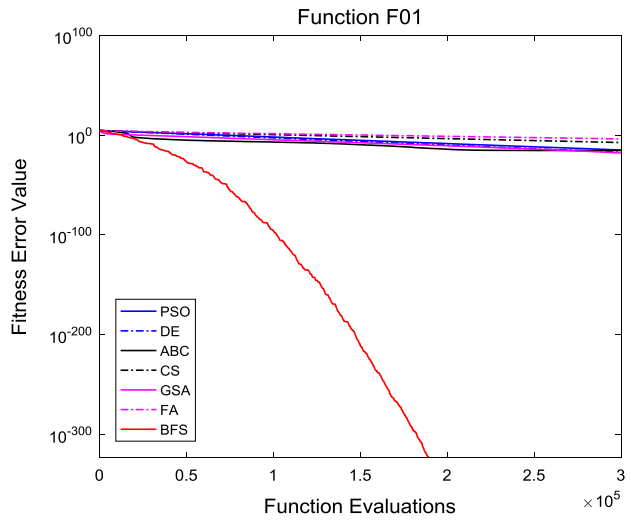
The best results among all algorithms for each function are shown in bold to highlight the best algorithm for each test function

for optimizing f_{11} and f_{13}, and the ABC performs the best for f_{08}. Moreover, for all multimodal functions, the BFS presents the overall best solutions, which implies that the BFS has a powerful ability to search for globally optimal solutions. To demonstrate the convergence characteristics of the seven overall algorithms, we also select four functions and plot their convergence curves for 30 independent runs (Fig. 4). As shown, the BFS outperforms the compared algorithms in terms of the convergence rate.

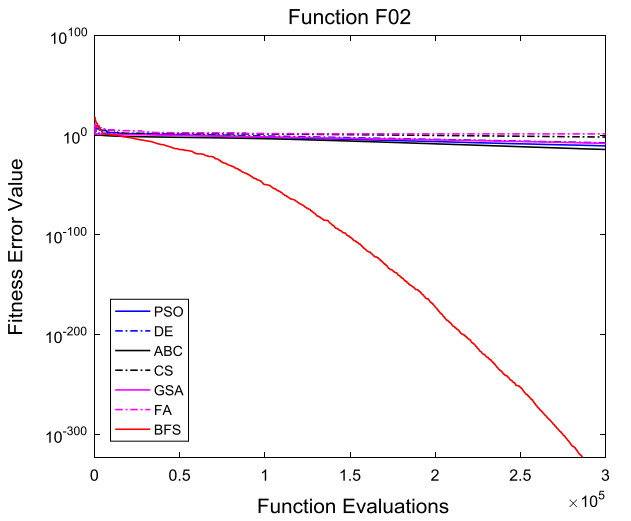
In total, 13 functions are used to test whether significant differences exist among the BFS and its competitors. The Wilcoxon signed rank test, a nonparametric test method that is not subject to the overall distribution of the samples and can be used to test for statistically significant differences between two samples, is used here at the 0.95 confidence level ($\alpha = 0.05$). Table 4 shows the statistical results for each function, and the signs "+", "≈" and "−" correspond to the following three cases: the BFS is significantly better than, significantly worse than and nonsignificantly different from the compared algorithms. "R+" denotes the sum of the ranks for which the BFS excelled compared with the corresponding competitor, and "R–" represents the sum of the ranks for the opposite case. The results in the last row of Table 4 with "+/≈/–" show that the BFS has a greater number of "+" signs than its competitors, which reflects that the BFS has excellent performance compared to that of all other algorithms in the Wilcoxon signed rank test at the 0.95 confidence level.

5.1.1 Time consumption analysis of BFS

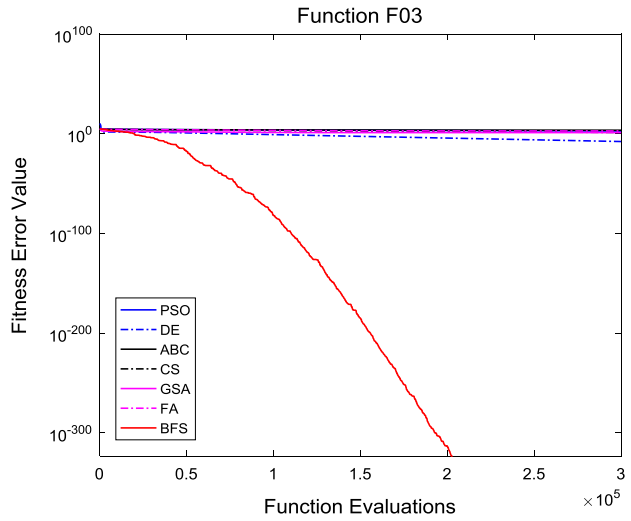
Time is an important indicator for measuring the performance of algorithms, and implementation processes that require less time are preferable. Thus, the mean time spent on the BFS and the other 6 algorithms over 30 independent runs for the 13 test functions is summarized in Table 5.



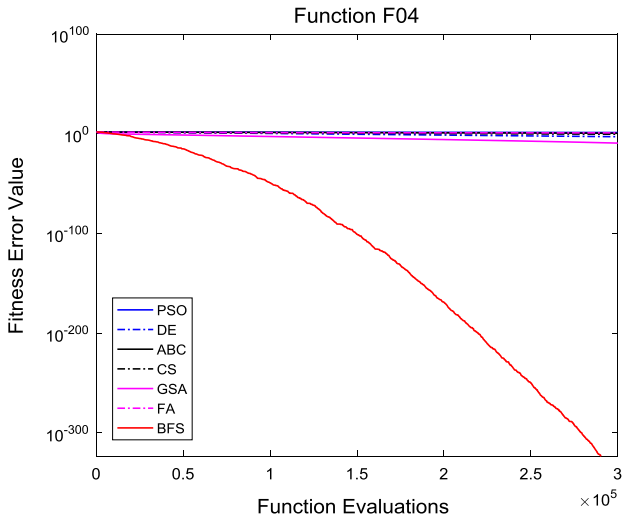
(a) f_{01}



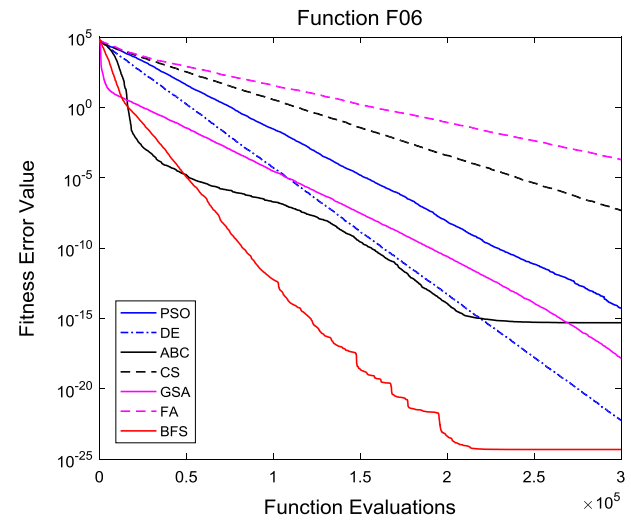
(b) f_{02}



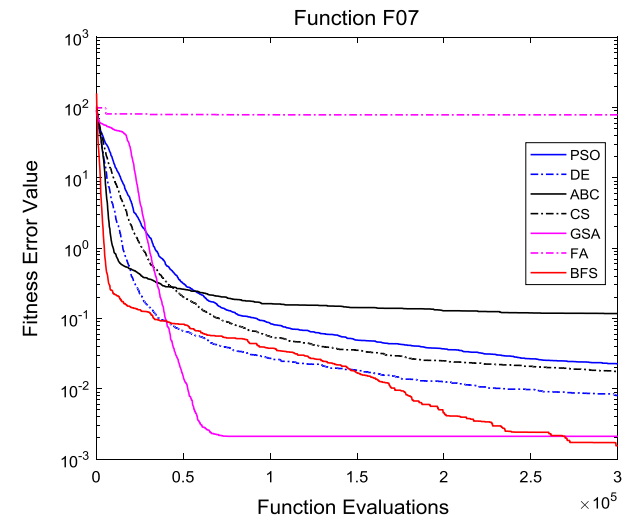
(c) f_{03}



(d) f_{04}



(e) f_{06}



(f) f_{07}

Fig. 3 Convergence curves of PSO, DE, ABC, CS, GSA, FA and BFS for 6 classic unimodal benchmarks

Table 3 Optimization results of the multimodal functions for different algorithms

No.	Algorithms	PSO	DE	ABC	CS	GSA	FA	BFS
f_{08}	Best	−1.0664E+04	−7.6859E+03	**−1.2569E+04**	−9.7379E+03	−4.2250E+03	−9.6872E+03	**−1.2569E+04**
	Mean	−9.2772E+03	−6.1831E+03	**−1.2569E+04**	−9.3617E+03	−3.0932E+03	−8.1182E+03	−1.2440E+04
	SD	9.2169E+02	5.8476E+02	**3.6723E−02**	1.9562E+02	3.8754E+02	6.8383E+02	6.4813E+02
	Rank	4	6	**1**	3	7	5	2
f_{09}	Best	1.2988E+01	1.3998E+02	0	4.7170E+01	0	2.6864E+01	**0**
	Mean	2.2049E+01	1.7249E+02	3.7895E−15	6.9658E+01	2.4874E+00	6.0460E+01	**0**
	SD	5.4619E+00	1.3607E+01	1.4422E−14	8.9198E+00	1.5840E+00	1.7750E+01	**0**
	Rank	4	7	2	6	3	5	**1**
f_{10}	Best	1.7774E−09	6.0969E−10	1.2523E−13	2.7970E−02	7.4250E−10	2.8271E−03	**8.8818E−16**
	Mean	1.5461E−08	1.0994E−09	2.1820E−13	3.9542E−01	9.4220E−10	1.6776E+00	**8.8818E−16**
	SD	2.3333E−08	3.9615E−10	4.8436E−14	4.1611E−01	9.4253E−11	8.8606E−01	**0**
	Rank	5	4	2	6	3	7	**1**
f_{11}	Best	1.1102E−16	**0**	0	1.0511E−05	0	5.2303E−04	**0**
	Mean	1.9500E−02	**0**	1.0362E−16	1.6325E−04	4.0977E−03	9.4129E−03	**0**
	SD	1.6517E−02	**0**	1.0072E−16	2.1666E−04	9.3092E−03	1.3623E−02	**0**
	Rank	7	**1**	3	4	5	6	**1**
f_{12}	Best	1.8576E−14	1.0318E−19	2.8075E−16	4.9465E−03	4.8063E−20	6.8989E−01	**2.3828E−31**
	Mean	9.0208E−08	1.6172E−18	4.4455E−16	2.3079E−01	1.3823E−02	4.1482E+00	**1.8612E−26**
	SD	4.6352E−07	1.4847E−18	9.4511E−17	2.6257E−01	3.5843E−02	2.0577E+00	**5.0463E−26**
	Rank	4	2	3	6	5	7	**1**
f_{13}	Best	2.8496E−14	1.6876E−18	3.0553E−16	3.2589E−06	6.5160E−19	9.0324E−06	**2.6426E−30**
	Mean	1.2105E−06	**8.3577E−18**	4.5801E−16	2.4314E−05	3.6625E−04	6.2792E−03	3.9970E−03
	SD	6.6300E−06	**6.4240E−18**	7.4968E−17	1.6408E−05	2.0060E−03	1.1328E−02	9.2807E−03
	Rank	3	**1**	2	4	5	7	6
Average rank		4.5000	3.5000	2.1667	4.8333	4.6667	6.1667	**2.0000**
Overall rank		4	3	2	6	5	7	**1**

The best results among all algorithms for each function are shown in bold to highlight the best algorithm for each test function

The mean time of each algorithm is ranked from short to long. The results show that the PSO algorithm consumes the least time and ranks first, followed by the CS and ABC algorithms. The BFS algorithm ranks fourth but is significantly better than the remaining three algorithms. The BFS is followed by the DE and FA algorithms; the GSA is the most time-consuming algorithm and ranks last. Based on the excellent performance for the 13 benchmark functions, the slight increase in the time consumption of the BFS is acceptable.

5.1.2 Search behavior and parameter analysis of BFS

The results of the preliminary studies showed that the BFS algorithm presents excellent performance when solving optimization problems. However, before further discussing the BFS, a question should be answered: must the three phases be simultaneously adopted for optimization problems? Therefore, in this experiment, the proposed algorithm configuration is compared with three different algorithm configurations: (1) contains only flying search behavior and territorial behavior, denoted BFS-C; (2) contains only territorial behavior and cognitive behavior, denoted BFS-F; and (3) contains only flying search behavior and cognitive behavior, denoted BFS-T. The populations of the three tested algorithms are set to 75 because each has two phases. These three configurations, along with the standard BFS, are evaluated using 8 typical test functions from the abovementioned classical benchmark functions. Table 6 lists the average results of 30 runs for BFS-C, BFS-F, BFS-T and BFS, and Fig. 5 presents the convergence graph for each algorithm for several functions.

The results in Table 6 and Fig. 5 lead to three significant conclusions. First, the standard BFS is superior to the other three configurations in terms of its accuracy and convergence speed, which indicates that the inclusion of flying search behavior, territorial behavior, and cognitive behavior is important. In addition, the performance of BFS-T is the worst, which means that territory behavior has the most important impact on the performance of the BFS. Finally, although flying search behavior has less impact on the unimodal function than do the other two behaviors and although cognitive behavior has less influence on multimodal functions than doe the other two behaviors, flying

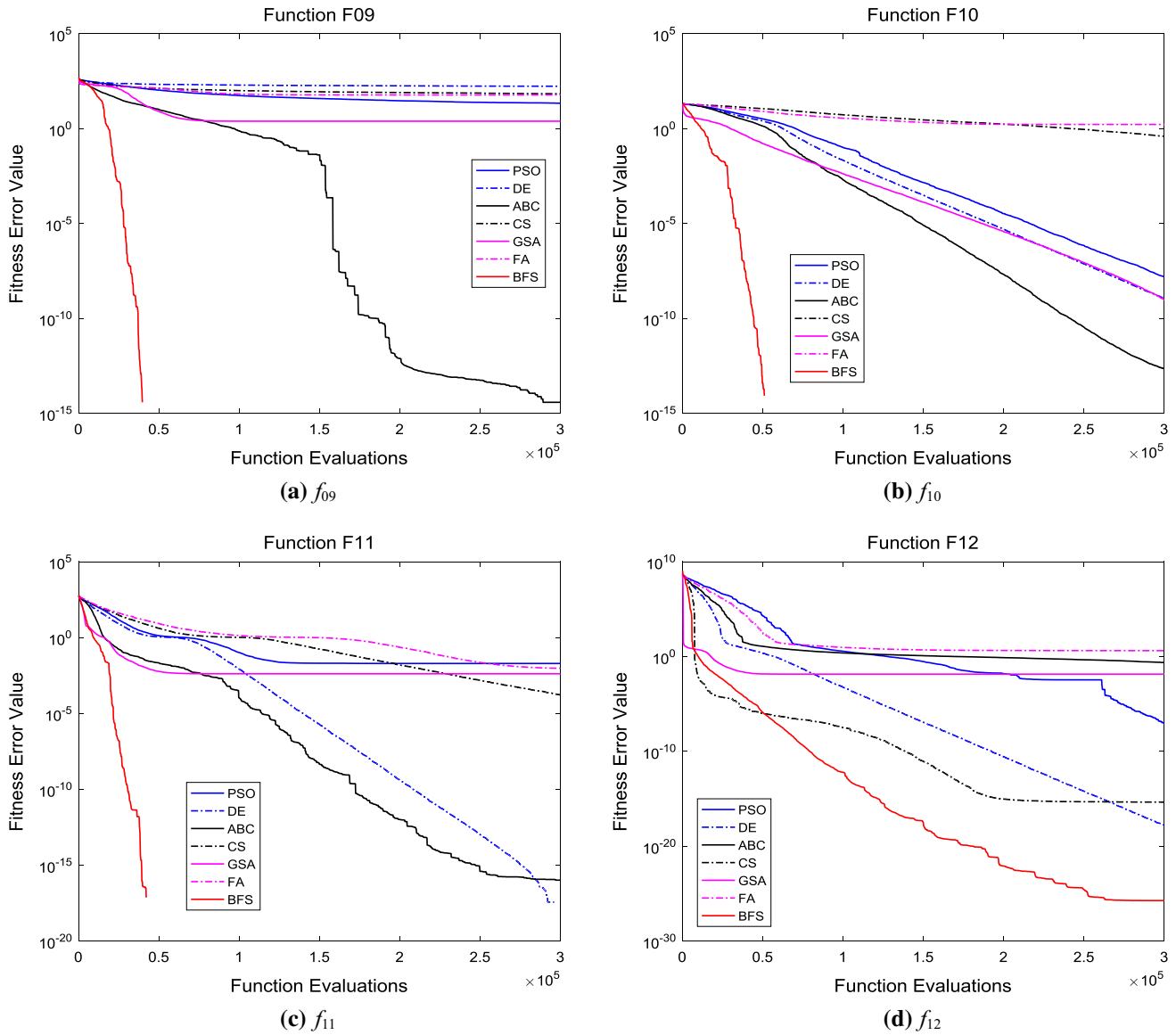


Fig. 4 Convergence curves of PSO, DE, ABC, CS, GSA, FA and BFS for 4 classic multimodal benchmarks

search behavior and cognitive behavior have important effects on the performance of the BFS. In general, the simulation results demonstrated that flying search, territorial and cognitive behavior have unique functions in the BFS and affect the quality of the final optimization results to varying degrees. Therefore, every behavior is indispensable for the BFS algorithm. Furthermore, the results demonstrate that the standard BFS is stable and reliable for solving global optimization problems.

As previously stated, the BFS does not contain any special control parameters, in contrast to other algorithms. However, as a population-based algorithm, the BFS is sensitive to the population size N. To study the influence of different population sizes on the performance of the BFS, the proposed algorithm is tested on the above eight benchmark functions with eight different population sizes for 30 independent runs. The *MaxFEs* of each function are set according to previous experiments. The statistical results of the mean and standard deviation are listed in Table 7, which shows that the BFS achieves the best performance when $N = 50$ for all test functions except f_{05} and f_{08}. Furthermore, the performance of the algorithm occasionally becomes worse as N increases for certain functions, which means that an increase in the population size does not improve, and can even degrade, the performance of the BFS. Figure 6 exhibits the influence of population size on the convergence speed for certain benchmark functions, and Fig. 7 shows the mean time consumption of the 8 test functions. Figures 6 and 7 show that the

Table 4 The results of the Wilcoxon signed ranks test based on the best solution for each function among 30 independent runs (α=0.05)

No.	PSO versus BFS				DE versus BFS				ABC versus BFS			
	p value	R+	R-	Win	*p* value	R+	R-	Win	*p* value	R+	R-	Win
f_{01}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{02}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{03}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{04}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{05}	1.9209E−06	464	1	+	3.7094E−01	189	276	≈	**1.7344E−06**	**0**	**465**	–
f_{06}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{07}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{08}	1.9209E−06	464	1	+	1.7344E−06	465	0	+	2.6016E−01	175	101	≈
f_{09}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.5730E−01	3	0	≈
f_{10}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7203E−06	465	0	+
f_{11}	1.7311E−06	465	0	+	1	0	0	≈	3.6624E−06	276	0	+
f_{12}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{13}	3.7094E−01	276	189	≈	3.7068E−01	276	189	≈	3.7068E−01	276	189	≈
+/≈/–		12/1/0				10/3/0				9/3/1		
No.	CS versus BFS				GSA versus BFS				FA versus BFS			
	p value	R+	R-	Win	*p* value	R+	R-	Win	*p* value	R+	R-	Win
f_{01}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{02}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{03}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{04}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{05}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{06}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{07}	1.7344E−06	465	0	+	2.8486E−02	339	126	+	1.7344E−06	465	0	+
f_{08}	1.9209E−06	464	1	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{09}	1.7344E−06	465	0	+	2.1938E−06	435	0	+	1.7344E−06	465	0	+
f_{10}	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
f_{11}	1.7344E−06	465	0	+	1.7148E−02	28	0	≈	1.7344E−06	465	0	+
f_{12}	1.7344E−06	465	0	+	1.7235E−06	465	0	+	1.7344E−06	465	0	+
f_{13}	3.7094E−01	276	189	≈	3.4363E−01	278.5	186.5	≈	2.5846E−03	379	86	+
+/≈/–		12/1/0				11/2/0				13/0/0		

The results that BFS is significantly worse than the compared algorithms are shown in bold

convergence rate is inversely proportional to the population size. The BFS with $N = 10$ converges faster than the BFS with other population sizes, although the computational time is the longest. According to the abovementioned analysis and considering certain factors, such as performance and time consumption, the optimal population size is 50.

5.1.3 Convergence analysis of BFS

A theoretical analysis of the convergence of the optimization algorithm will facilitate a deep understanding of the algorithm and guide its improvement. The convergence of algorithms refers to whether individuals in the entire population become global optimal solutions under the condition that the iteration time tends to infinity. In this section, we use random functional analysis theory to prove the convergence of the BFS algorithm.

Assume that $f(a)$ is the objective function. Its solution space is expressed as $\Omega = [A|A = a_1, a_2, \ldots, a_D]$, with D representing the dimensions of the solution vector; and l_i and u_i are the lower and upper bounds of a_i, respectively. Clearly, the solution space is a bounded subset of a D-dimensional Euclidean space R^n.

Definition 1 Let A be a nonempty set and d be a mapping from $A \times A$ to R. For any element a, b, c in A, if they satisfy the following three properties:

1. Nonnegativity: $d(a, b) \geq 0$; $d(a, b) = 0$ if and only if $a = b$;

Table 5 Mean time consumption in seconds for PSO, DE, ABC, CS, GSA, FA and BFS with 30 independent runs on 13 classic functions

No.	PSO	DE	ABC	CS	GSA	FA	BFS
f_{01}	2.3292E+00	1.3344E+01	6.3605E+00	3.9517E+00	9.5947E+01	8.2745E+01	9.6628E+00
f_{02}	2.6206E+00	1.3683E+01	6.8313E+00	4.4328E+00	8.7436E+01	8.4501E+01	9.4090E+00
f_{03}	1.0681E+01	2.3211E+01	1.9367E+01	1.4710E+00	9.9352E+01	9.3877E+01	2.7604E+01
f_{04}	2.3089E+00	1.3164E+01	6.7095E+00	3.9178E+00	8.7867E+01	8.4726E+01	8.1391E+00
f_{05}	2.6971E+00	8.8330E+00	6.2176E+00	4.3422E+00	8.8421E+01	8.6310E+01	8.9184E+00
f_{06}	2.3303E+00	8.0395E+00	7.1268E+00	3.9886E+00	8.8740E+01	8.5821E+01	7.9792E+00
f_{07}	4.1806E+00	1.4732E+01	6.5255E+00	5.8372E+00	9.0366E+01	8.4096E+01	8.7305E+00
f_{08}	2.9219E+00	1.4156E+01	7.2971E+00	4.9816E+00	9.2007E+01	8.4134E+01	1.0081E+01
f_{09}	2.7778E+00	1.2957E+01	6.6637E+00	4.3576E+00	9.0502E+01	7.9650E+01	1.1132E+01
f_{10}	3.0282E+00	1.3886E+01	7.5783E+00	4.7515E+00	9.1836E+01	8.3374E+01	1.1670E+01
f_{11}	3.3131E+00	1.4508E+01	7.0739E+00	5.0943E+00	9.2957E+01	8.1833E+01	1.2576E+01
f_{12}	8.2744E+00	2.0493E+01	1.4698E+01	1.0372E+01	9.6124E+01	8.6337E+01	1.9376E+01
f_{13}	8.2305E+00	2.0542E+01	1.2552E+01	1.0453E+01	9.8253E+01	8.6041E+01	1.9419E+01
Mean time	**4.2841E+00**	1.4734E+01	8.8462E+00	5.2270E+00	9.2293E+01	8.4880E+01	1.2669E+01
Overall rank	**1**	5	3	2	7	6	4

The results that consume the least time are shown in bold

Table 6 Statistical results of the investigation of different BFS behavior

No.	BFS-F		BFS-T		BFS-C		BFS	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f_{01}	**0**	**0**	1.5643E+00	1.1775E+00	2.8198E−241	0	**0**	**0**
f_{02}	**0**	**0**	5.8276E+00	5.4975E+00	8.1292E−165	5.0157E−164	**0**	**0**
f_{05}	1.3633E+01	2.2517E+00	2.0516E+02	3.3042E+02	1.8432E+01	4.5214E+00	**8.5498E+00**	**2.4488E+00**
f_{06}	1.9594E−17	5.9743E−17	4.7293E−01	1.8026E+00	2.4607E−16	9.5181E−16	**4.9290E−25**	**1.4989E−24**
f_{08}	−7.7051E+03	8.6129E+02	−5.6466E+03	9.2251E+02	−1.1291E+04	1.0992E+03	**−1.2418E+04**	**6.4702E+02**
f_{09}	**0**	**0**	1.6360E+02	2.7798E+01	**0**	**0**	**0**	**0**
f_{11}	**0**	**0**	7.3321E−01	8.4404E−01	**0**	**0**	**0**	**0**
f_{12}	3.2384E−15	6.2284E−15	1.1676E+01	6.1344E+00	5.0933E−19	1.2980E−18	**1.1510E−26**	**4.9872E−26**

The best results among all algorithms for each function are shown in bold to highlight the best algorithm for each test function

2. Symmetry: $d(a,b) = d(b,a)$;
3. Triangle inequality: $d(a,b) \le d(a,c) + d(c,b)$;

then, d is called a metric on A, and (A,d) is called a metric space.

Definition 2 If (A,d) is a metric space, and a_n is any sequence in (A,d) l for any $\varepsilon > 0$, there is a positive integer N that makes $d(a_n,a) < \varepsilon$ tenable such that for $n > N$, the sequence a_n is said to converge in (A,d).

Definition 3 The sequence a_n in the metric space (A,d) is called the Cauchy sequence, which means that for any given $\varepsilon > 0$, there is always a positive integer N that makes $d(a_m,a_n) < \varepsilon$ when $m,n > N$ or makes $d(x_m,x_n) \to 0$ when $m,n \to \infty$. If every Cauchy sequence in space (A,d) converges, then (A,d) is a complete metric space.

Definition 4 If (A,d) is a complete metric space, f is a mapping of A to A. If there is any constant $\zeta \in (0,1)$ such that for all $a,b \in A$ there is always $d(f(a),f(b)) \le \zeta d(a,b)$, then f is a contraction map on A.

Theorem 1 (Banach contraction mapping theorem) *Let (A,d) be a complete metric space and f be a compression map from A to A; then, f has one and only one fixed point $a^* \in A$ such that for any $a_0 \in A$, $a^* = \lim_{k\to\infty} f^k(a_0)$ is satisfied, where $f^0(a_0)=a_0$, $f^{k+1}(a_0)=f(f^k(a_0))$.*

Theorem 2 *When iteration $\to \infty$, the BFS algorithm gradually converges to the optimal solution of the population.*

Proof Let x_{best} be the optimal solution for the population, let X^k be the set of all individuals' positions at the $k-th$ iteration of the BFS algorithm (that is, $X^k = {x_1}^k, {x_2}^k, \ldots, {x_N}^k$), let the set of all X^k be denoted as S, and let the metric d: $S \times S \to R$ be defined as

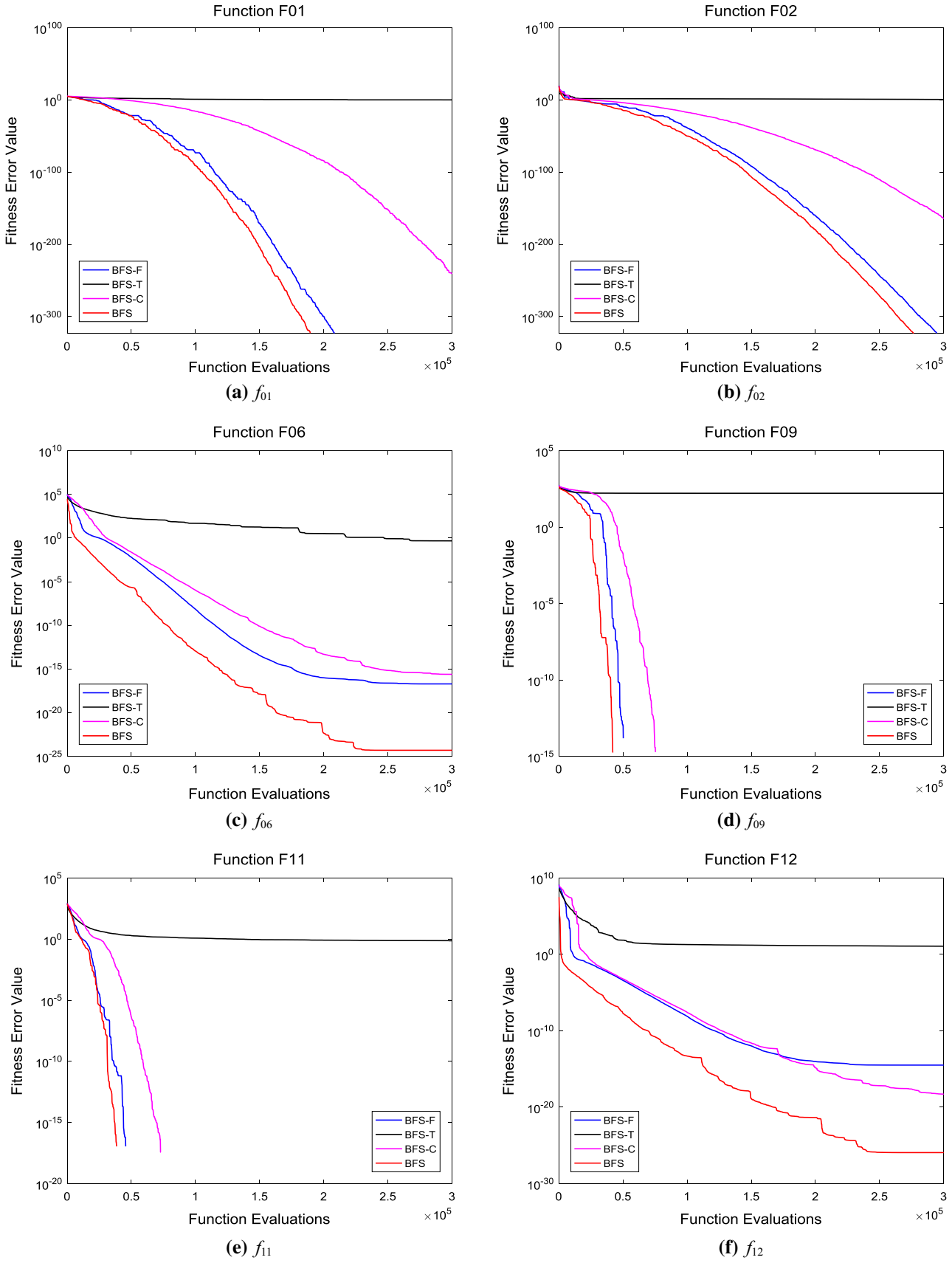


Fig. 5 Convergence rates of the BFS and three alternative configurations for 6 classic benchmark functions

Table 7 Optimum results obtained by BFS for the 8 typical classical functions with different values of N

No.		*N*=10	*N*=30	*N*=50	*N*=70	*N*=90	*N*=120	*N*=150	*N*=200
f_{01}	Mean	0	0	0	0	0	1.1903e−316	1.9567E−256	1.0013E−185
	SD	0	0	0	0	0	0	0	0
f_{02}	Mean	0	0	0	7.3512E−258	4.9942E−206	4.3652E−156	3.3164E−123	4.5455E−90
	SD	0	0	0	0	0	0	1.7948E−122	2.4861E−89
f_{05}	Mean	1.8014E+01	7.9957E+00	8.4299E+00	1.1242E+01	1.2437E+01	1.6013E+01	1.7359E+01	1.8375E+01
	SD	4.9121E+00	2.0079E+00	2.2542E+00	2.3538E+00	2.3547E+00	2.7283E+00	2.6304E+00	2.8984E+00
f_{06}	Mean	3.2218E−15	6.9616E−25	3.0683E−25	3.2560E−21	2.4767E−19	3.3845E−16	4.2897E−14	4.7576E−12
	SD	1.3887E−14	2.5570E−24	1.4051E−24	8.1516E−21	5.4767E−19	9.2569E−16	1.8905E−13	1.4522E−11
f_{08}	Mean	− 1.0761E+04	− 1.0442E+04	− 1.2292E+04	− 1.1981E+04	− 1.1784E+04	− 1.2292E+04	− 1.2218E+04	− 1.2569E+04
	SD	2.0124E+03	1.7433E+03	8.9753E+02	1.3380E+03	1.4494E+03	8.9753E+02	1.0723E+03	2.9386E−05
f_{09}	Mean	0	0	0	0	0	0	0	0
	SD	0	0	0	0	0	0	0	0
f_{11}	Mean	0	0	0	0	0	0	0	0
	SD	0	0	0	0	0	0	0	0
f_{12}	Mean	5.9185E−18	1.6196E−25	7.3624E−26	1.8980E−23	4.0693E−20	4.7660E−16	1.1479E−15	4.2359E−13
	SD	2.7006E−17	8.2722E−25	9.3347E−26	4.1943E−23	1.4644E−19	2.2623E−15	5.0903E−15	1.5920E−12

$$d(X^n, X^m) = \begin{cases} \sum_{x_i^n \in x^n} |f(x_i^n) - H| + \sum_{x_i^m \in x^m} |f(x_i^m) - H|, & X^n \neq X^m \\ 0, & X^n = X^m \end{cases} \tag{12}$$

where H is the lower bound of $f(x)$ and exists. From definition 1, it can be concluded that the three properties satisfied by the above formula are as follows:

1. $d(X^n, X^m) \geq 0$; $d(X^n, X^m)=0$ if and only if $X^n=X^m$;
2. $d(X^n, X^m)=d(X^m, X^n)$;
3. $d(X^n, X^m) = \sum_{x_i^n \in x^n} \left|f(x_i^n) - H\right| + \sum_{x_i^m \in x^m} \left|f(x_i^m) - H\right|$;
$\leq \sum_{x_i^n \in p^n} \left|f(x_i^n) - H\right| + \sum_{x_i^k \in x^k} \left|f(x_i^k) - H\right|$
$+ \sum_{x_i^k \in x^k} \left|f(x_i^k) - H\right| + \sum_{x_i^m \in x^m} \left|f(x_i^m) - H\right|$
$= d(X^n, X^k) + d(X^k, X^m)$;

Therefore, individual sequences (S, d) form a metric space.

By definition 3, since X^k is any sequence in (S, d), for any given $\varepsilon > 0$, there is always a positive integer K such that $d(X^n, X^m) < \varepsilon$ when $m, n > K$, or when $m, n \to \infty$, $d(X^n, X^m) \to 0$, i.e., $X^n=X^m$. Meanwhile, all the Cauchy sequences X^k in the population converge; thus, (S, d) is a complete metric space.

The workflow of the BFS algorithm shows that the process of solving BFS is a cyclical iterative process. During each iteration, the BFS algorithm completes the search through the flying search behavior phase, the territorial behavior phase and the cognitive behavior phase. Therefore, all operations of the generation of the next BFS iteration population can be regarded as a mapping $f : S \to S$ of the solution space Ω. Since the BFS algorithm uses a greedy strategy when updating individuals' positions, the population generated by the next iteration must not be worse than the population of the previous iteration. Therefore, the fitness value sequence corresponding to the individual positions formed during the entire iteration of the population is a monotonous nonincremental sequence; i.e.,$f(x_i^{k+1}) \leq f(x_i^k) \leq f(x_i^{k-1})$ or $f(f(f(x_i^k))) \leq f(f(x_i^k)) \leq f(x_i^k)$, where $f(x_i^k)$ is the fitness value of the current position x_i of the $i-th$ individual at the $k-th$ iteration. Thus, for mapping f,

$$\begin{aligned} d(f(x_i^n), f(x_i^m)) &= \sum_{x_i^n \in x^n} |f(f(x_i^n)) - H| + \sum_{x_i^m \in x^m} |f(f(x_i^m)) - H| \\ &\leq \sum_{x_i^n \in x^n} |f(x_i^n) - H| + \sum_{x_i^m \in x^m} |f(x_i^m) - H| \\ &\leq \zeta d(X_i^n, X_i^m) \end{aligned} \tag{13}$$

where $0 < \zeta < 1$. From definition 4, the map f is a contraction map. In summary, the BFS algorithm satisfies the condition of the Banach contraction mapping theorem; that is, (S, d) is a complete metric space, and the map f is a contraction map. According to Theorem 1, the algorithm is convergent, and there must be a unique fixed point $x_{best} \in S$ such that for any $x \in S$, $x_{best} = \lim_{k \to \infty} f^k(x)$ is satisfied, where $f^0(x)=x$, and $f^{k+1}(x)=f(f^k(x))$. When the values of all individuals in S are equal to x_{best}, the algorithm converges to the only fixed point x_{best}, and the theorem is confirmed.□

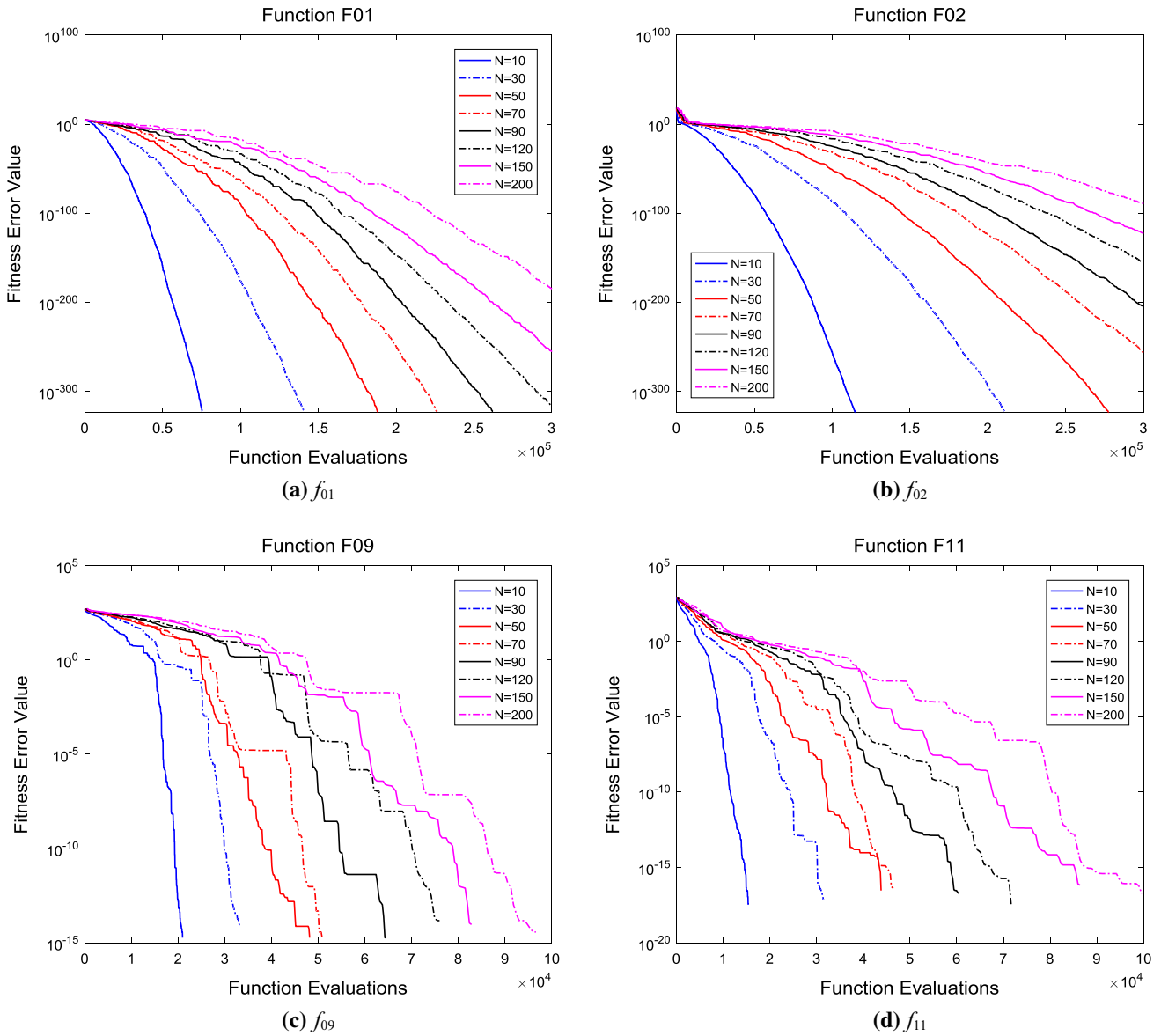


Fig. 6 Effect of N on the convergence rate for 4 classical test functions

5.2 Experiment II: CEC2014 benchmarks

In this experiment, all 30 numerical functions of the 2014 Congress on Evolutionary Computation (CEC 2014) Special Session and Competition on Single Objective Real Parameter Numerical Optimization suite listed in Table 8 are selected to test the performance of the BFS for complex real-parameter numerical optimization problems. These functions include several neoteric features, such as novel basic problems, and the test problems are constructed by a dimension-wise extraction of features from several problems, including the graded level of linkages and rotated trap problems. The traditional test function often sets the global optimal value in the central area of the search space and places the local optimal value on the coordinate axis, resulting in these functions not truly being able to reflect the actual optimization problem. The CEC2014 solves these problems by shifting the global optima and rotating the test functions. Therefore, they are more similar to the optimization problem in the real world. The CEC2014 test functions contain rotated unimodal, shifted and rotated multimodal, hybrid, and composition test functions, and the details are provided in [14]. The unimodal functions have no local optima, and there is a single global optimum. Thus, they are suitable for evaluating the exploitation of algorithms. By contrast, multimodal functions have a large number of local solutions and are useful for testing the exploration of the algorithm. In hybrid test functions, enormous numbers of

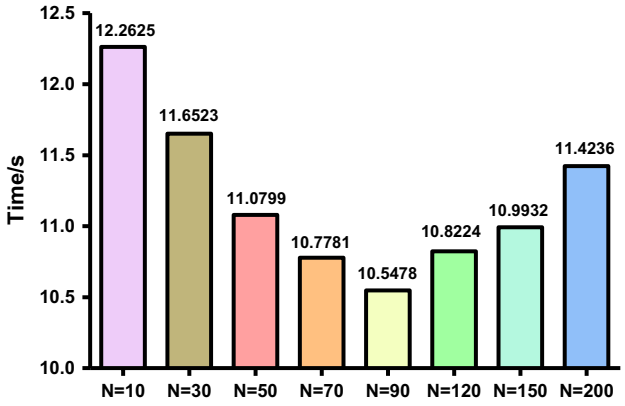


Fig. 7 Effect of N on the average time consumption for 8 classic test functions

local optimal distributions exist in the search space, and the variables are randomly divided into subcomponents, with different basic functions used for different subcomponents. The purpose of these functions is benchmarking the ability of an algorithm to escape a local optimum when faced with complex problems. The composition test functions are the rotated, shifted, biased, and combined versions of several unimodal, multimodal and hybrid functions. This set of functions is extremely complex because the functions have various shapes for different regions of the search space. These characteristics are beneficial for testing the abilities of the algorithm in terms of balanced exploration and exploitation.

The proposed BFS is compared with the following state-of-the art algorithms that have previously been demonstrated to possess superior performance compared with that

Table 8 Descriptions of the CEC2014 test functions

No.	Type	Name	Optimal
$f_{01(CEC)}$	Unimodal functions	Rotated High Conditioned Elliptic Function	100
$f_{02(CEC)}$		Rotated Bent Cigar Function	200
$f_{03(CEC)}$		Rotated Discus Function	300
$f_{04(CEC)}$	Simple multimodal functions	Shifted and Rotated Rosenbrock's Function	400
$f_{05(CEC)}$		Shifted and Rotated Ackley's Function	500
$f_{06(CEC)}$		Shifted and Rotated Weierstrass Function	600
$f_{07(CEC)}$		Shifted and Rotated Griewank's Function	700
$f_{08(CEC)}$		Shifted Rastrigin's Function	800
$f_{09(CEC)}$		Shifted and Rtated Rastrigin's function	900
$f_{10(CEC)}$		Shifted and Rotated Rastrigin's Function	1000
$f_{11(CEC)}$		Shifted and Rotated Schwefel's Function	1100
$f_{12(CEC)}$		Shifted and Rotated Katsuura Function	1200
$f_{13(CEC)}$		Shifted and Rotated HappyCat Function	1300
$f_{14(CEC)}$		Shifted and Rotated HGBat Function	1400
$f_{15(CEC)}$		Shifted and Rotated Expanded Griewank'splus Rosenbrock's Function	1500
$f_{16(CEC)}$		Shifted and Rotated Expanded Scaffer's F6 Function	1600
$f_{17(CEC)}$	Hybrid functions	Hybrid Function 1 (n = 3)	1700
$f_{18(CEC)}$		Hybrid Function 2 (n = 3)	1800
$f_{19(CEC)}$		Hybrid Function 3 (n = 4)	1900
$f_{20(CEC)}$		Hybrid function 4 (n = 4)	2000
$f_{21(CEC)}$		Hybrid function 5 (n = 5)	2100
$f_{22(CEC)}$		Hybrid function 6 (n = 5)	2200
$f_{23(CEC)}$	Composition functions	Composition function 1 (n = 5)	2300
$f_{24(CEC)}$		Composition function 2 (n = 3)	2400
$f_{25(CEC)}$		Composition function 3 (n = 3)	2500
$f_{26(CEC)}$		Composition function 4 (n = 5)	2600
$f_{27(CEC)}$		Composition function 5 (n = 5)	2700
$f_{28(CEC)}$		Composition function 6 (n = 5)	2800
$f_{29(CEC)}$		Composition function 7 (n = 3)	2900
$f_{30(CEC)}$		Composition function 8 (n = 3)	3000
	Search range: [100, − 100]	Dimension: 30	

of classic optimization algorithms: TLBO [22], GWO [18], SMO [2], MFO [16], WOA [17], CSA [1] and SBO [19]. Specific algorithm parameters are not required for TLBO, and the control parameter settings for the remaining algorithms in this test are as follows:

1. GWO: $a = 2 - 1 \times (2/MaxCycle)$ as in [18];
2. SMO: $MG = 5$, $GlobalLeader\,Limit = 50$, $LocalLeader\,Limit = 1500$ and $pr_{G+1} = pr_G + (0.4 - 0.1)/MIR$ as in [2];
3. MFO: $b = 1$ and $flame\ no = round[N - l \cdot (N - 1)/T]$ as in [16];
4. WOA: $b = 1$ and $a = 2 - 1 \times (2/MaxCycle)$ as in [17];
5. CSA: $AP = 0.1$ and $fl = 2$ as in [1];
6. SBO: $\alpha = 0.94$, $Z = 0.02$ and mutation probability of 0.05 as in [19];

The dimensions and the maximum number of function evaluations are set to 30 and 300,000, respectively. To ensure the fairness of the comparison, all algorithms should run under the same number of function evaluations and iterations. Therefore, the population sizes of the BFS and SMO are set to 50 because they contain three update phases. Correspondingly, the population size of the TLBO is set to 75 because of the inclusion of two phases, and the population sizes of the remaining 6 algorithms are set to 150 because they include one phase. Tables 9, 10, 11 and 12 present the statistical results of the BFS and 6 compared algorithms for each function from 30 independent runs. Similar to the previous test, we rank the algorithms from smallest to largest mean value. Finally, we calculate the average ranks of each algorithm and draw conclusions.

(1) Unimodal functions group ($f_{01(CEC)}$–$f_{03(CEC)}$): Table 9 provides the experimental results obtained by all algorithms from 30 independent runs. Figure 8 depicts the graphical results of the algorithms for $f_{01(CEC)}$ and $f_{03(CEC)}$. Table 9 shows that the BFS has clear advantages over the other algorithms for all functions. In addition, the BFS can find the global optimum for $f_{02(CEC)}$. The outstanding accuracy and convergence speed are highlighted in Fig. 8. The results in Table 9 show that the BFS algorithm has good exploration capability, mainly due to its full use of global optimal individual information and its own gradient information, which effectively improves the local exploitation capability of the algorithm.

(2) Simple multimodal functions group ($f_{04(CEC)}$–$f_{16(CEC)}$): The final results obtained by the BFS and 6 remaining algorithms from 30 independent runs are reported in Table 10, and the convergence rates and ANOVA tests for several typical multimodal functions are plotted in Fig. 9. Table 10 shows that the BFS obtains the best mean value for five functions ($f_{04(CEC)}$, $f_{05(CEC)}$, $f_{07(CEC)}$, $f_{14(CEC)}$, and $f_{15(CEC)}$) and almost reaches the global optimum for $f_{07(CEC)}$. As shown by the overall rank in the last line of Table 10, the BFS ranks first for the simple multimodal functions from the CEC2014. Moreover, the SMO obtains the best mean values for $f_{06(CEC)}$, $f_{08(CEC)}$, $f_{09(CEC)}$ and $f_{13(CEC)}$; the GWO performs better than the other compared algorithms for $f_{11(CEC)}$ and $f_{16(CEC)}$; and the TLBO and the SBO exhibit the best performance for $f_{10(CEC)}$ and $f_{12(CEC)}$, respectively. The graphical results shown in Fig. 9 indicate that the BFS performs better than the other algorithms in terms of the convergence rate and accuracy for the listed typical multimodal functions. Summarizing the above analysis, BFS has good exploration capability, mainly due to the extensive search of space via a logarithmic spiral pattern in the flying search behavior phase.

Table 9 Optimization results of the CEC2014 unimodal functions for 8 algorithms

No.	Algorithms	TLBO	GWO	SMO	MFO	WOA	CSA	SBO	BFS
$f_{01(CEC)}$	Best	4.5005E+04	8.7656E+06	3.3247E+05	2.8396E+06	1.3851E+07	8.1446E+05	6.1529E+05	**3.0741E+04**
	Mean	6.7457E+05	3.8280E+07	2.0851E+06	6.6872E+07	3.3578E+07	5.8487E+06	2.3383E+06	**4.3784E+05**
	SD	1.4322E+06	3.0272E+07	1.4577E+06	7.9387E+07	1.1748E+07	1.4162E+06	8.9846E+05	**4.4374E+05**
	Rank	2	7	3	8	6	5	4	**1**
$f_{02(CEC)}$	Best	2.0005E+02	1.0133E+08	2.0000E+02	3.4675E+04	6.7726E+05	3.0149E+02	2.9471E+03	**2.0000E+02**
	Mean	2.0628E+02	7.0012E+08	2.4668E+02	8.0696E+09	3.0568E+06	7.9793E+03	1.0542E+04	**2.0000E+02**
	SD	6.4622E+00	7.4408E+08	1.7744E+02	4.8025E+09	3.2021E+06	5.3857E+03	6.3512E+03	**6.6213E−03**
	Rank	2	7	3	8	6	4	5	**1**
$f_{03(CEC)}$	Best	3.4465E+02	6.9737E+03	3.4037E+02	1.4617E+04	8.9809E+03	1.1484E+03	8.5316E+02	**3.0008E+02**
	Mean	5.0532E+02	2.2562E+04	1.8891E+03	7.9283E+04	3.1998E+04	3.0025E+03	8.6355E+03	**3.3844E+02**
	SD	2.2145E+02	8.0423E+03	2.5606E+03	3.5663E+04	2.3108E+04	1.2540E+03	7.2336E+03	**1.0754E+02**
	Rank	2	6	3	8	7	4	5	**1**
Average rank		2.0000	6.6667	3.0000	8.0000	6.3333	4.3333	4.6667	**1.0000**
Overall rank		2	7	3	8	6	4	5	**1**

The best results among all algorithms for each function are shown in bold to highlight the best algorithm for each test function

Table 10 Optimization results of the CEC2014 simple multimodal functions for 8 algorithms

No.	Algorithms	TLBO	GWO	SMO	MFO	WOA	CSA	SBO	BFS
$f_{04(CEC)}$	Best	4.0006E+02	4.9468E+02	4.1877E+002	4.7090E+02	4.9217E+02	4.6823E+02	4.1103E+02	**4.0006E+02**
	Mean	4.6856E+02	5.6631E+02	4.6516E+002	1.0297E+03	5.7070E+02	5.1388E+02	4.6942E+02	**4.5725E+02**
	SD	3.2375E+01	4.8324E+01	3.0701E+001	4.5283E+02	4.7718E+01	3.6071E+01	3.7054E+01	**3.5412E+01**
	Rank	3	6	2	8	7	5	4	**1**
$f_{05(CEC)}$	Best	5.2075E+02	5.2088E+02	5.2082E+02	5.2008E+02	5.2008E+02	5.2000E+02	5.2070E+02	**5.2000E+02**
	Mean	5.2092E+02	5.2096E+02	5.2096E+02	5.2033E+02	5.2034E+02	5.2044E+02	5.2092E+02	**5.2000E+02**
	SD	5.2709E−02	4,4219E−02	5.5995E−02	1.3049E−01	1.8588E−01	3.8306E−01	7.1580E−02	**3.2060E−05**
	Rank	6	7	8	2	3	4	5	**1**
$f_{06(CEC)}$	Best	6.0727E+02	6.0628E+02	**6.0549E+02**	6.1716E+02	6.2439E+02	6.1356E+02	6.1906E+02	6.1650E+02
	Mean	6.1335E+02	6.1135E+02	**6.0862E+02**	6.2081E+02	6.3460E+02	6.2078E+02	6.2419E+02	6.2417E+02
	SD	2.9058E+00	3.1305E+00	**2.2339E+00**	2.8469E+00	4.1749E+00	3.0263E+00	2.4997E+00	4.3150E+00
	Rank	3	2	**1**	5	8	4	7	6
$f_{07(CEC)}$	Best	7.0000E+02	7.0120E+02	7.0000E+02	7.0001E+02	7.0079E+02	7.0000E+02	7.0001E+02	**7.0000E+02**
	Mean	7.0003E+02	7.0804E+02	7.0002E+02	7.6752E+02	7.0101E+02	7.0002E+02	7.0005E+02	**7.0001E+02**
	SD	3.6417E−02	6.2867E+00	1.7568E−02	3.4142E+01	6.1662E−02	2.0522E−02	1.9977E−02	**1.2349E−02**
	Rank	4	7	2	8	6	3	5	**1**
$f_{08(CEC)}$	Best	8.4676E+02	8.4100E+02	**8.0298E+002**	8.4209E+02	8.9960E+02	8.6965E+02	8.6766E+02	8.3482E+02
	Mean	8.6467E+02	8.6776E+02	**8.1011E+002**	9.1343E+02	9.8525E+02	9.1031E+02	9.0324E+02	8.9125E+02
	SD	1.2347E+01	1.9800E+01	**3.7821E+000**	3.9572E+01	4.2378E+01	1.9331E+01	1.9170E+01	2.3930E+01
	Rank	2	3	**1**	7	8	6	5	4
$f_{09(CEC)}$	Best	9.4179E+02	9.3329E+02	**9.2686E+02**	1.0194E+03	1.0495E+03	9.7761E+02	9.9452E+02	9.5970E+02
	Mean	9.6468E+02	9.8599E+02	**9.4327E+02**	1.0871E+03	1.1451E+03	1.0149E+03	1.0388E+03	9.9787E+03
	SD	1.3615E+01	3.1063E+01	**9.8253E+00**	3.9513E+01	5.6232E+01	1.7570E+01	2.1741E+01	2.5974E+01
	Rank	2	3	**1**	7	8	5	6	4
$f_{10(CEC)}$	Best	**1.8758E+03**	1.9324E+03	1.1504E+03	2.9052E+03	3.5128E+03	3.0159E+03	2.2349E+03	2.6361E+03
	Mean	**2.4826E+03**	2.9894E+03	3.0218E+03	5.5440E+03	4.9027E+03	4.0767E+03	3.3295E+03	3.7972E+03
	SD	**3.8298E+02**	6.8610E+02	1.8520E+03	7.8613E+02	6.9935E+02	5.8365E+02	4.3951E+02	5.5662E+02
	Rank	**1**	2	3	8	7	6	4	5
$f_{11(CEC)}$	Best	4.2571E+03	**2.7021E+03**	7.6061E+03	3.6982E+03	4.2042E+03	3.1232E+03	3.3211E+03	3.4211E+03
	Mean	7.4198E+03	**3.8200E+03**	8.3136E+03	5.2147E+03	5.9136E+03	4.3773E+03	4.3772E+03	4.7892E+03
	SD	7.2519E+02	**9.6597E+02**	3.0538E+02	8.4431E+02	8.4016E+02	5.5279E+02	5.7526E+02	5.3488E+02
	Rank	7	**1**	8	5	6	3	2	4
$f_{12(CEC)}$	Best	1.2019E+03	1.2001E+03	1.2022E+03	1.2001E+03	1.2008E+03	1.2001E+03	**1.2001E+03**	1.2005E+03
	Mean	1.2024E+03	1.2019E+03	1.2027E+03	1.2004E+03	1.2016E+03	1.2004E+03	**1.2001E+03**	1.2011E+03
	SD	3.3159E−01	1.0333E+00	2.1210E−01	1.9048E−01	5.0698E−01	1.7437E−01	**6.8345E−02**	3.3667E−01
	Rank	7	6	8	3	5	2	**1**	4
$f_{13(CEC)}$	Best	1.3002E+03	1.3002E+03	**1.3002E+03**	1.3005E+03	1.3003E+03	1.3003E+03	1.3002E+03	1.3002E+03
	Mean	1.3004E+03	1.3004E+03	**1.3003E+03**	1.3011E+03	1.3005E+03	1.3004E+03	1.3004E+03	1.3003E+03
	SD	8.2258E−02	8.2383E−02	**7.1021E−02**	8.1796E−01	1.3087E−01	7.4599E−02	7.1940E−02	8.4583E−02
	Rank	5	6	**1**	8	7	4	3	2
$f_{14(CEC)}$	Best	1.4002E+03	1.4002E+03	1.4002E+03	1.4010E+03	1.4002E+03	1.4002E+03	1.4002E+03	**1.4001E+03**
	Mean	1.4003E+03	1.4008E+03	1.4003E+03	1.4128E+03	1.4003E+03	1.4003E+03	1.4002E+03	**1.4002E+03**
	SD	9.7525E−02	1.5613E+00	1.0577E−01	1.1210E+01	1.7405E−01	8.5067E−02	5.1105E−02	**4.6978E−02**
	Rank	4	7	5	8	6	3	2	**1**
$f_{15(CEC)}$	Best	1.5060E+03	1.5056E+03	1.5021E+03	1.5104E+03	1.5385E+03	1.5059E+03	1.5131E+03	**1.5046E+03**
	Mean	1.5125E+03	1.5193E+03	1.5102E+03	2.7856E+04	1.5733E+03	1.5122E+03	1.5248E+03	**1.5098E+03**
	SD	5.2020E+00	1.8481E+01	5.2707E+00	4.6136E+04	2.4664E+01	5.0789E+00	7.4041E+00	**3.2165E+00**
	Rank	4	5	2	8	7	3	6	**1**
$f_{16(CEC)}$	Best	1.6108E+03	**1.6092E+03**	1.6121E+03	1.6105E+03	1.6115E+03	1.6110E+03	1.6100E+03	1.6097E+03
	Mean	1.6117E+03	**1.6108E+03**	1.6128E+03	1.6121E+03	1.6126E+03	1.6117E+03	1.6116E+03	1.6117E+03
	SD	4.0213E−01	**6.4607E−01**	3.0657E−01	6.3366E−01	4.5796E−01	3.8429E−01	6.2672E−01	7.0438E−01
	Rank	4	**1**	8	6	7	3	2	5
Average rank		4.0000	4.3077	3.8462	6.3846	6.5385	3.9231	4.0000	**3.0000**
Overall rank		4	6	2	7	8	3	4	**1**

The best results among all algorithms for each function are shown in bold to highlight the best algorithm for each test function

Table 11 Optimization results of the CEC2014 hybrid functions for 8 algorithms

No.	Algorithms	TLBO	GWO	SMO	MFO	WOA	CSA	SBO	BFS
$f_{17(CEC)}$	Best	1.5207E+04	1.1155E+05	5.9113E+04	1.1081E+05	4.5283E+05	5.5084E+03	2.3187E+05	**4.8606E+03**
	Mean	1.5599E+05	1.1034E+06	4.5547E+05	2.4535E+06	4.1068E+06	**3.6746E+04**	7.4676E+05	6.9611E+04
	SD	1.6174E+05	1.5127E+06	4.0737E+05	3.9862E+06	2.4113E+06	**2.4974E+04**	4.9216E+05	6.5949E+04
	Rank	3	6	4	7	8	**1**	5	2
$f_{18(CEC)}$	Best	1.9231E+03	2.4666E+03	1.8111E+03	2.6391E+03	2.3561E+03	1.9823E+03	1.8718E+03	**1.8409E+03**
	Mean	3.8473E+03	6.6611E+05	3.2908E+03	3.2505E+05	1.2245E+04	2.1343E+03	3.4326E+03	**2.1208E+03**
	SD	3.2354E+03	1.9910E+06	2.4023E+03	5.2344E+05	3.0541E+04	3.3642E+02	1.9633E+03	**9.2667E+02**
	Rank	5	8	3	7	6	2	4	**1**
$f_{19(CEC)}$	Best	1.9049E+03	1.9104E+03	1.9052E+03	1.9101E+03	1.9160E+03	1.9081E+03	1.9102E+03	**1.9061E+03**
	Mean	1.9171E+03	1.9193E+03	1.9112E+03	1.9306E+03	1.9609E+03	1.9209E+03	1.9168E+03	**1.9093E+03**
	SD	2.1232E+01	9.1782E+00	2.1903E+00	2.7505E+01	4.4479E+01	1.7889E+01	3.7792E+00	**1.6363E+00**
	Rank	4	5	2	7	8	6	3	**1**
$f_{20(CEC)}$	Best	2.2674E+03	2.4421E+03	6.1945E+03	1.2324E+04	7.9970E+03	2.2464E+03	3.9778E+03	**2.1432E+03**
	Mean	2.6516E+03	1.1324E+04	1.6296E+04	5.5620E+04	2.8282E+04	2.4989E+03	1.8242E+04	**2.3261E+03**
	SD	3.1088E+02	5.3227E+03	7.5653E+03	3.3215E+04	2.2536E+04	2.4728E+02	8.0243E+03	**1.0354E+02**
	Rank	3	4	5	8	7	2	6	**1**
$f_{21(CEC)}$	Best	1.2798E+04	1.6157E+04	1.5665E+04	2.0417E+04	1.0829E+05	6.0114E+03	6.3091E+04	**4.8917E+03**
	Mean	7.0859E+04	4.3372E+05	1.4959E+05	6.2307E+05	1.1820E+06	**1.3689E+04**	4.6374E+05	2.2722E+04
	SD	5.0755E+04	4.3642E+05	1.6039E+05	7.7172E+05	1.0463E+06	**7.4691E+03**	2.5146E+05	1.4734E+04
	Rank	3	5	4	7	8	**1**	6	2
$f_{22(CEC)}$	Best	2.2260E+03	2.3555E+03	**2.2277E+03**	2.5493E+03	2.4637E+03	2.2695E+03	2.5974E+03	2.2455E+03
	Mean	2.4344E+03	2.5128E+03	**2.3814E+03**	2.8185E+03	3.0345E+03	2.6284E+03	3.0060E+03	2.5685E+03
	SD	9.8211E+01	1.6185E+02	**9.2606E+01**	1.9427E+02	2.5582E+02	1.7603E+02	2.5142E+02	1.4576E+02
	Rank	2	3	**1**	6	8	5	7	4
Average rank		3.3333	5.1667	3.1667	7.0000	7.5000	2.8333	5.1667	**1.8333**
Overall rank		4	5	3	7	8	2	5	**1**

The best results among all algorithms for each function are shown in bold to highlight the best algorithm for each test function

(3) Hybrid functions group ($f_{17(CEC)}$–$f_{22(CEC)}$): Compared with the previous two sets of test functions, this set of functions has a complex form. Thus, finding the global optimal value is more difficult. Table 11 summarizes the statistical results obtained via the BFS and its competitors from 30 independent runs. The BFS is superior for $f_{18(CEC)}$, $f_{19(CEC)}$ and $f_{20\ (CEC)}$; the CSA outperforms the other algorithms for $f_{17(CEC)}$ and $f_{18(CEC)}$, although the BFS can find the minimum best solutions for these two functions; and the SMO shows the best performance for $f_{22\ (CEC)}$. The overall rank in the last line of Table 11 shows that the overall performance of the BFS for hybrid functions is the best. The graphical results are depicted for 4 typical functions in Fig. 10, which shows that the BFS performs significantly better than do the compared algorithms in terms of convergence rate and solution quality. The above results prove that the BFS algorithm has a strong ability to jump out of a local optima when applied to complex problems because the BFS algorithm incorporates various random search strategies, such as spiral search [Eq. (2)], random escape [Eq. (5)], and Gaussian variation [Eq. (9)], throughout the optimization process. Different search methods guide individuals to search by collaborating with each other, thereby improving the global search ability of the algorithm and increasing the probability of the algorithm escaping local optima.

(4) Composition functions group ($f_{23(CEC)}$–$f_{30(CEC)}$): This set of functions is extremely complex, and finding optimal solutions to these functions presents the greatest difficulty. Table 12 lists the optimization results obtained by the BFS and the other 6 compared algorithms. The BFS algorithm ranks first for five composition functions ($f_{23(CEC)}$, $f_{24(CEC)}$, $f_{25(CEC)}$, $f_{27(CEC)}$, and $f_{28(CEC)}$), and the SMO algorithm shows the best performance for $f_{26(CEC)}$, $f_{29(CEC)}$ and $f_{30(CEC)}$. The performance of the BFS for $f_{26(CEC)}$ ranks second, and for $f_{29\ (CEC)}$, the best solution of the BFS ranks first. When examining the overall rank in the last line of Table 12, the BFS undoubtedly ranks first, whereas the SMO and CSA rank second and third, respectively, and the WOA exhibits the worst performance. For a graphical perspective, the convergence rate and ANOVA test graphs of the 6 selected functions obtained using the BFS and its competitors are presented in Fig. 11. The graphical results show that the BFS

Table 12 Optimization results of the CEC2014 composition functions for 8 algorithms

No.	Algorithms	TLBO	GWO	SMO	MFO	WOA	CSA	SBO	BFS
$f_{23(CEC)}$	Best	2.6152E+03	2.6214E+03	2.6140E+03	2.6253E+03	2.6205E+03	2.6159E+03	2.6140E+03	**2500**
	Mean	2.6152E+03	2.6270E+03	2.6140E+03	2.6465E+03	2.6311E+03	2.6170E+03	2.6140E+03	**2500**
	SD	8.3036E−12	3.6089E+00	5.2054E−08	1.8803E+01	7.4246E+00	8.8061E−01	5.1543E−03	**0**
	Rank	4	6	2	8	7	5	3	**1**
$f_{24(CEC)}$	Best	2.6000E+03	2.6000E+03	2.6230E+03	2.6302E+03	2.6012E+03	2.6010E+03	2.6252E+03	**2600**
	Mean	2.6000E+03	2.6000E+03	2.6320E+03	2.6622E+03	2.6059E+03	2.6099E+03	2.6335E+03	**2600**
	SD	9.7800E−04	1.0228E−03	7.9757E+00	2.5092E+01	3.5559E+00	9.5732E+00	9.1784E+00	**0**
	Rank	2	3	6	8	4	5	7	**1**
$f_{25(CEC)}$	Best	2.7000E+03	2.7000E+03	2.7004E+03	2.7036E+03	2.7000E+03	2.7000E+03	2.7136E+03	**2700**
	Mean	2.7004E+03	2.7099E+03	2.7009E+03	2.7117E+03	2.7221E+03	2.7002E+03	2.7204E+03	**2700**
	SD	1.4157E+00	2.9834E+00	3.2022E−01	6.5284E+00	2.0223E+01	8.9289E−01	3.7561E+00	**0**
	Rank	3	5	4	6	8	2	7	**1**
$f_{26(CEC)}$	Best	2.7003E+03	2.7002E+03	**2.7002E+03**	2.7004E+03	2.7002E+03	2.7003E+03	2.7003E+03	2.7002E+03
	Mean	2.8000E+03	2.7070E+03	**2.7003E+03**	2.7010E+03	2.7004E+03	2.7004E+03	2.7337E+03	2.7004E+03
	SD	1.8184E+01	2.5282E+01	**5.8231E−02**	5.9577E−01	1.1108E−01	9.9340E−02	4.7809E+01	9.1407E−02
	Rank	8	6	**1**	5	4	3	7	2
$f_{27(CEC)}$	Best	3.1010E+03	3.1074E+03	3.0752E+03	3.1166E+03	3.1200E+03	3.1021E+03	3.1013E+03	**2900**
	Mean	3.2167E+03	3.2528E+03	3.1915E+03	3.4693E+03	3.7606E+03	3.1684E+03	3.5618E+03	**2900**
	SD	1.5296E+02	9.2503E+01	9.1780E+01	2.1949E+02	3.8415E+02	1.9529E+02	3.1184E+02	**0**
	Rank	4	5	3	6	8	2	7	**1**
$f_{28(CEC)}$	Best	3.6554E+03	3.6314E+03	3.1924E+03	3.6479E+03	3.9678E+03	4.5714E+03	5.4299E+03	**3000**
	Mean	3.9290E+03	3.7890E+03	3.2212E+03	3.7509E+03	4.8531E+03	5.7840E+03	6.7108E+03	**3000**
	SD	1.7173E+02	1.9105E+02	2.2784E+01	4.7959E+01	4.4673E+02	7.1671E+02	6.8254E+02	**0**
	Rank	5	4	2	3	6	7	8	**1**
$f_{29(CEC)}$	Best	3.7189E+03	9.3863E+03	3.1049E+03	6.2566E+03	1.5410E+04	4.6432E+03	3.1142E+03	**3.1000E+03**
	Mean	2.3989E+06	3.8031E+04	**3.1112E+03**	3.3235E+05	4.1080E+06	7.2702E+03	3.5765E+03	4.1290E+03
	SD	4.4499E+06	6.7899E+04	**1.3403E+01**	1.5765E+06	4.7446E+06	2.8521E+03	5.3867E+02	3.7328E+02
	Rank	7	5	**1**	6	8	4	2	3
$f_{30(CEC)}$	Best	3.8307E+03	1.4755E+04	**3.2477E+03**	6.2919E+03	1.8938E+04	7.7818E+03	3.2864E+03	3.3181E+03
	Mean	5.8220E+03	3.1736E+04	**3.4082E+03**	2.0015E+04	8.4912E+04	1.1983E+04	3.8764E+03	5.3549E+03
	SD	1.0936E+03	1.5259E+04	**1.3111E+02**	1.1927E+04	5.6976E+04	2.9186E+03	3.1835E+02	9.6905E+02
	Rank	4	7	**1**	6	8	5	2	3
Average rank		4.6250	5.1250	2.5000	6.000	6.6250	4.1250	5.3750	**1.6250**
Overall rank		4	5	2	7	8	3	6	**1**

The best results among all algorithms for each function are shown in bold to highlight the best algorithm for each test function

clearly outperforms the compared algorithms. In summary, BFS has a good ability to balance exploration and exploitation, as shown in Table 12 and Fig. 11, for two main reasons: 1) in each iteration, three phases are executed sequentially to jointly execute a complete search; and 2) although different phases focus on different capabilities, each phase attempts to balance exploration and exploitation. For example, in the flying search behavior phase, a spiral path also guarantees some degree of exploitation. In the territorial behavior phase, the probability P^{iter} can play a role in balancing exploration and exploitation. In the cognitive behavior phase, individuals use gradient information to increase the search efficiency while implementing Gaussian variation to improve the ability of the BFS to escape local optima.

To verify the performance of the BFS and the other competitors via an overall comparison of the CEC2014 benchmark functions, we also employ the Wilcoxon signed rank test conducted at the 0.95 confidence level ($\alpha = 0.05$) for a paired comparison. The comparative results of each function are listed in Table 13. The last row of Table 13 shows that except for the 19 "+" in the TLBO comparison, BFS has 20 or more "+" in the comparisons with the remaining algorithms. In other words, the BFS presents more "+" values than "−" or "≈" values compared with every competitor; thus, our BFS presents statistically better performance than

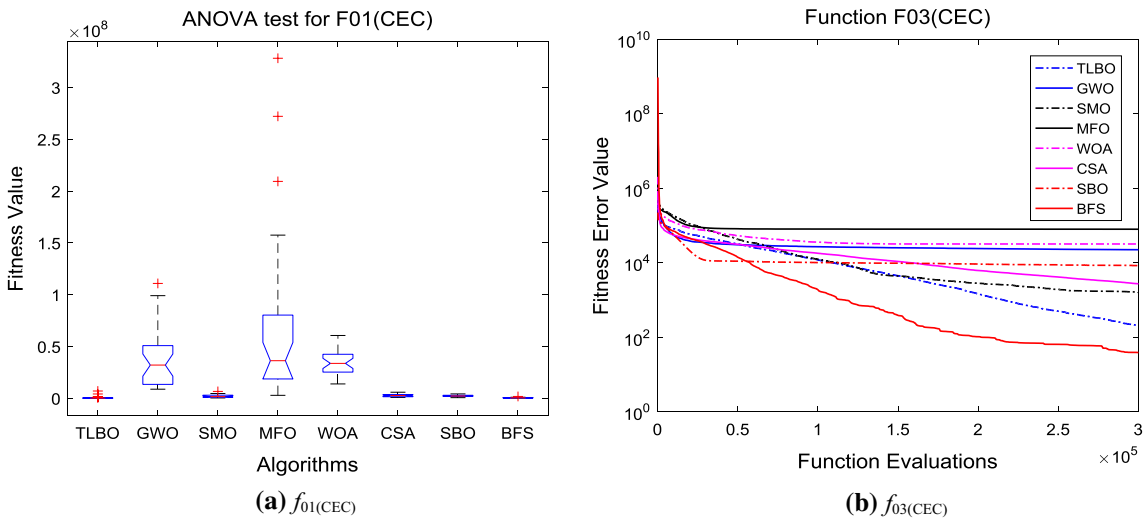


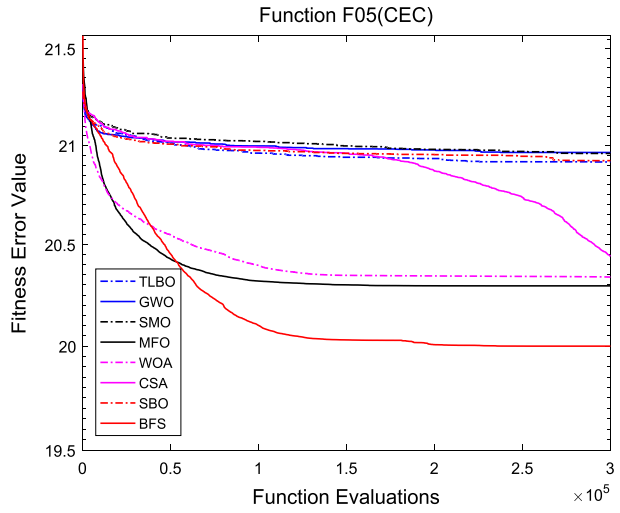
Fig. 8 ANOVA test and convergence graphs of 2 unimodal functions for all algorithms

the competitors based on the Wilcoxon signed rank test at the 0.95 confidence level. Furthermore, SMO performs better than BFS for 8 functions, which is the largest number of all competitors. WOA does not have a function that performs better or similarly to BFS. For TLBO, 5 functions are not significantly different from BFS, ranking first among all competitors.

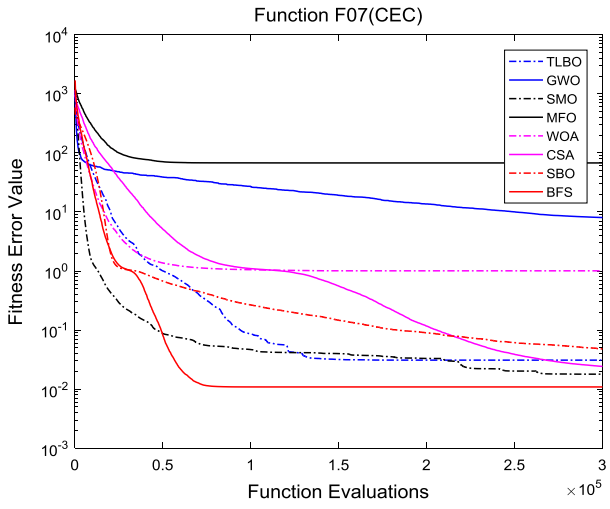
6 Conclusions

Because an increasing number of practical applications require optimization techniques, optimization algorithms will continue to be used for several decades. By simulating three different behaviors of birds during foraging, a new algorithm is introduced, and this algorithm can be divided into three phases. In the first phase, the algorithm uses the spiral path method to conduct extensive searching, thus highlighting the exploration ability of the algorithm. In the second phase, the algorithm mimics the territorial behavior of birds, thus emphasizing the exploitation ability of the algorithm. In the last phase, the algorithm simulates the cognitive behavior of birds, thus highlighting the overall search efficiency of the algorithm. In total, 13 classical benchmark functions and 30 modern complex benchmark functions of the CEC2014 are used to test the performance of the BFS. The results demonstrate that the BFS performs significantly better than other classical and state-of-the art algorithms. Therefore, the BFS has the potential for handling global optimization problems.

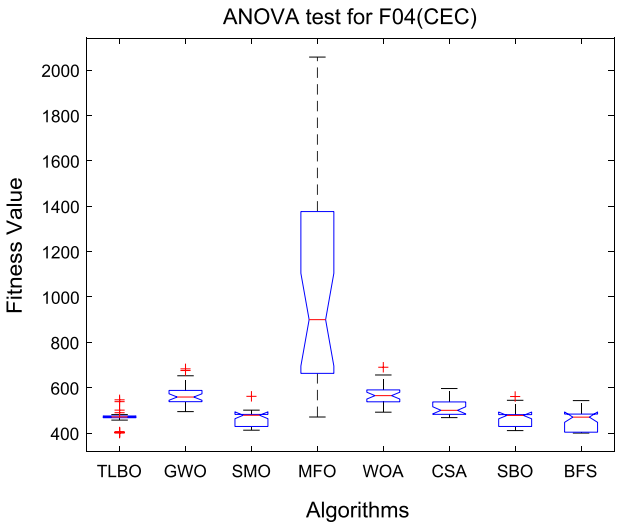
Future work can be performed in several areas. First, binary and multiobjective versions of the BFS are worth researching. Second, different flight search behaviors, such as levy flight, can be integrated into the BFS to improve its performance. Finally, studying the application of the BFS in engineering design problems in the future is of considerable practical significance.



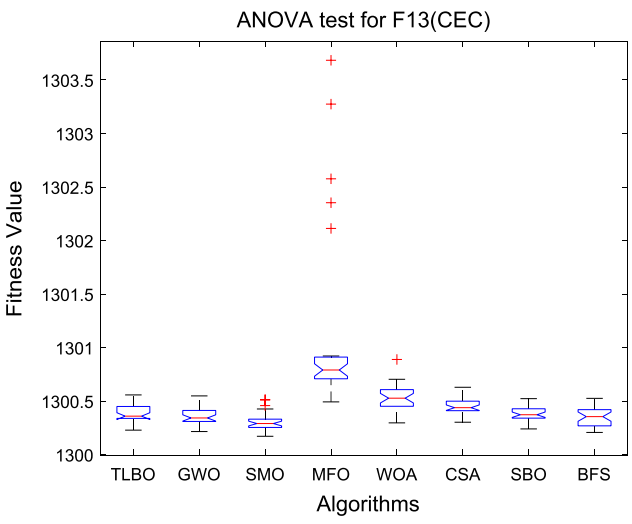
(a) $f_{05(\mathrm{CEC})}$



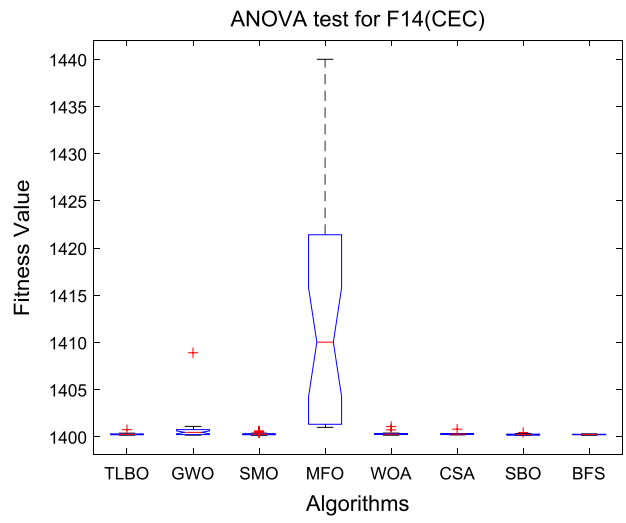
(b) $f_{07(\mathrm{CEC})}$



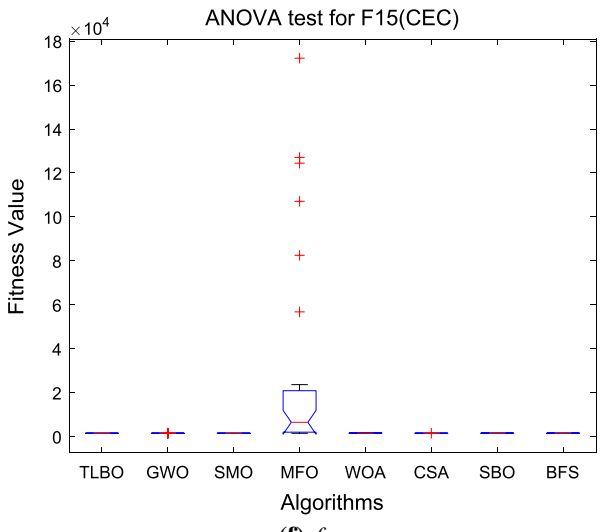
(c) $f_{04(\mathrm{CEC})}$



(d) $f_{13(\mathrm{CEC})}$



(e) $f_{14(\mathrm{CEC})}$



(f) $f_{15(\mathrm{CEC})}$

Fig. 9 Graphical results and ANOVA test of 6 CEC2014 simple multimodal functions for 8 algorithms

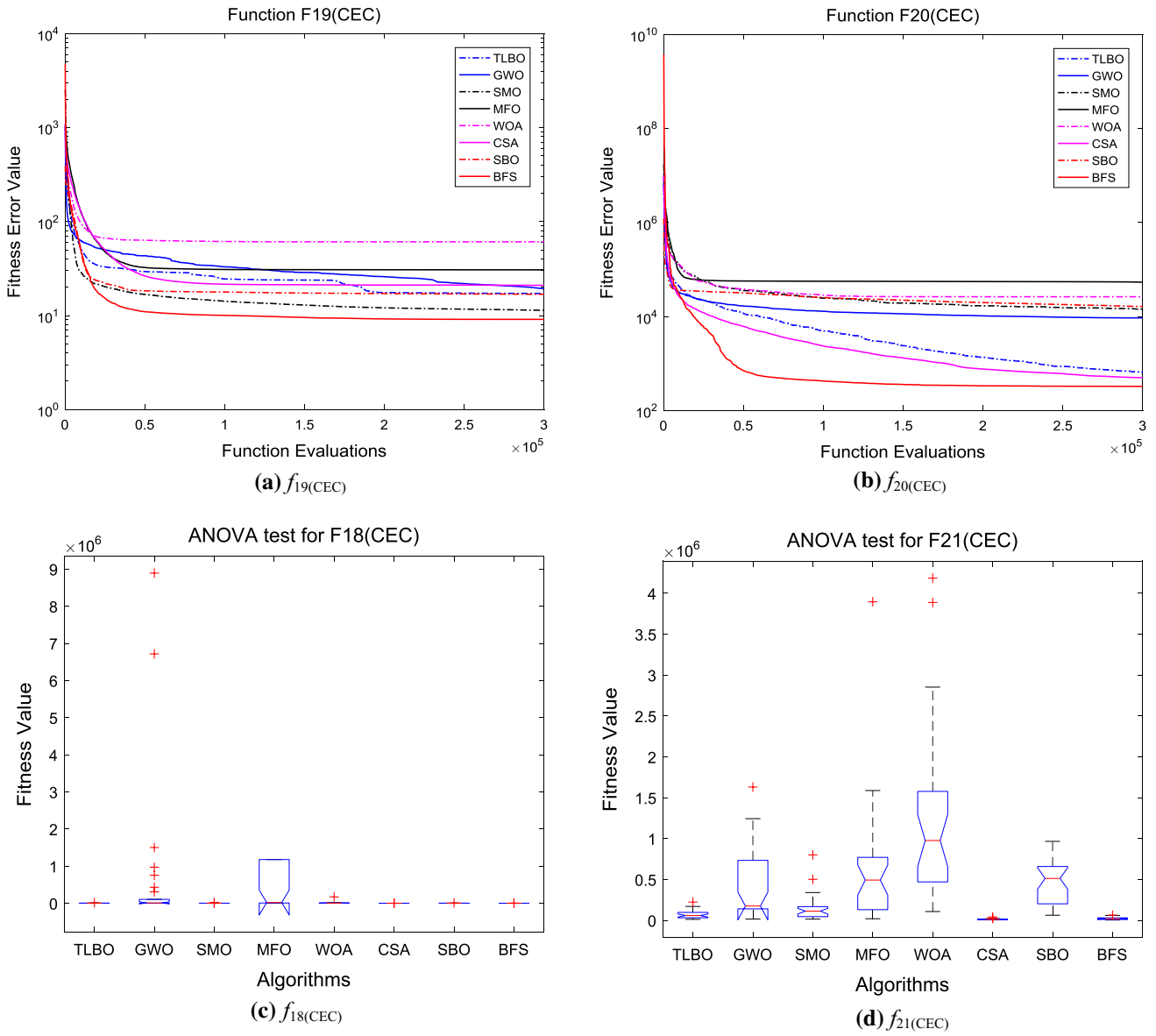
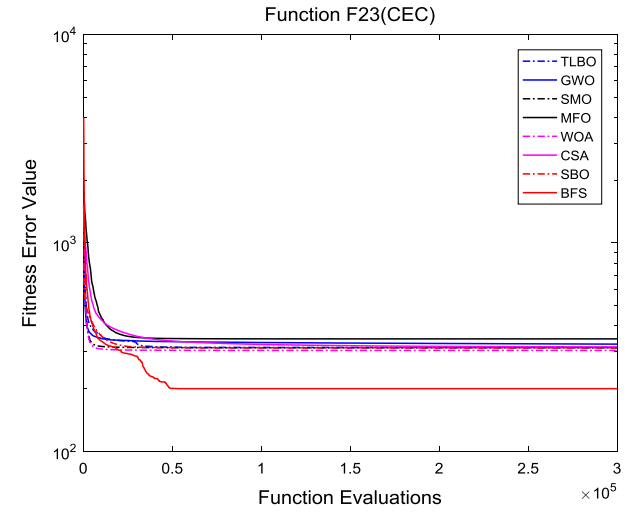
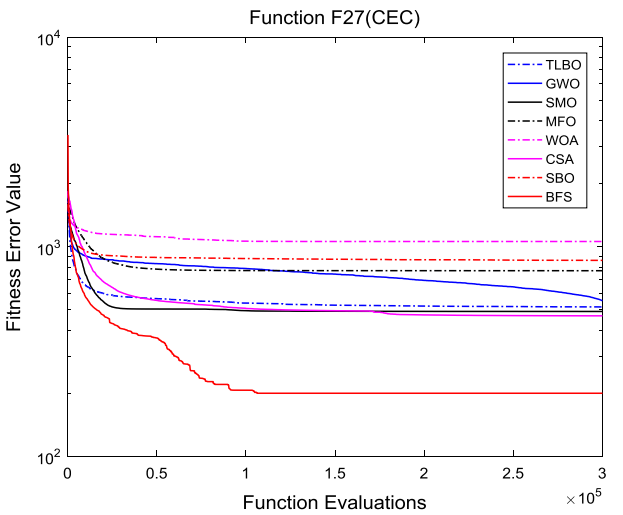


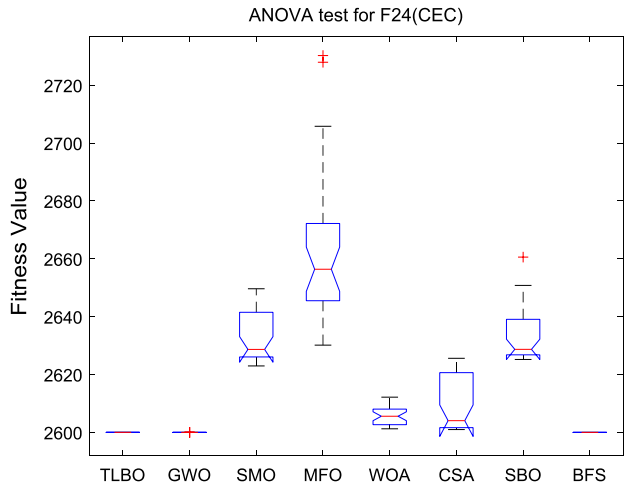
Fig. 10 Graphical results and ANOVA test of 4 CEC2014 hybrid functions for different algorithms



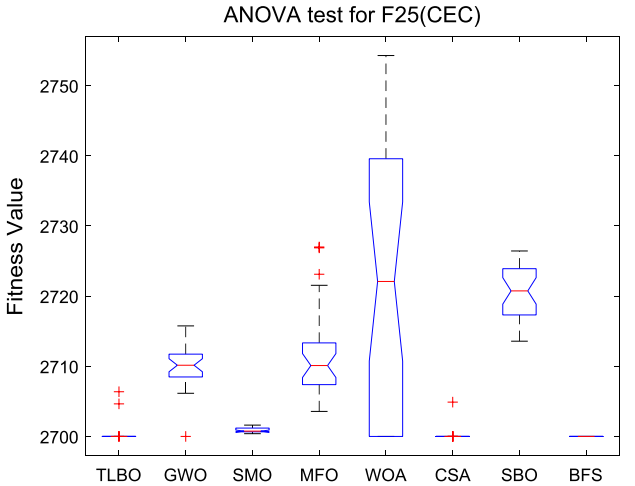
(a) $f_{23(\mathrm{CEC})}$



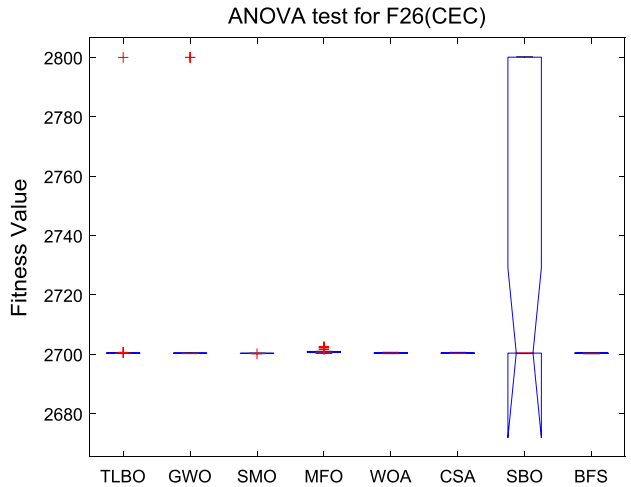
(b) $f_{27(\mathrm{CEC})}$



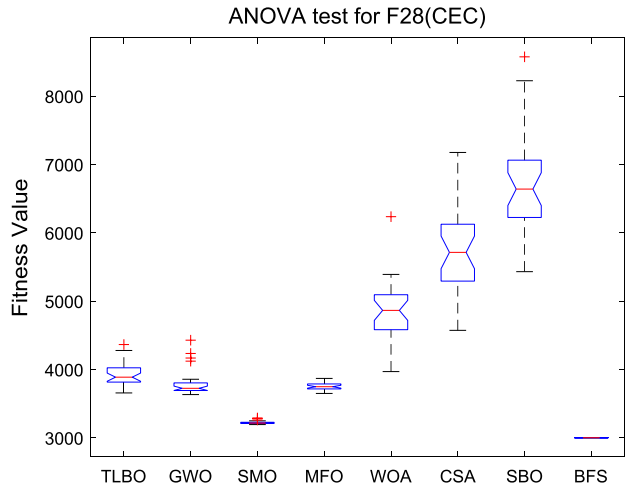
(c) $f_{24(\mathrm{CEC})}$



(d) $f_{25(\mathrm{CEC})}$



(e) $f_{26(\mathrm{CEC})}$



(f) $f_{28(\mathrm{CEC})}$

Fig. 11 Graphical results and ANOVA test of 6 CEC2014 composition functions for various methods

Table 13 The results of the Wilcoxon signed ranks test based on the best solution for each function among 30 independent runs (α=0.05)

No.	TLBO versus BFS				GWO versus BFS				SMO versus BFS			
	p value	R+	R-	Win	*p* value	R+	R-	Win	*p* value	R+	R-	Win
$f_{01(CEC)}$	7.0356E−01	214	251	≈	1.7344E−06	465	0	+	1.0246E−05	447	18	+
$f_{02(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{03(CEC)}$	1.9209E−06	464	1	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{04(CEC)}$	8.5896E−02	316	149	+	1.7344E−06	465	0	+	9.5899E−01	235	230	≈
$f_{05(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{06(CEC)}$	**1.7344E−06**	**0**	**465**	–	1.7344E−06	465	0	+	**1.7344E−06**	**0**	**465**	–
$f_{07(CEC)}$	3.6826E−02	334	131	+	1.7344E−06	465	0	+	5.1931E−02	327	138	+
$f_{08(CEC)}$	**4.4493E−05**	**34**	**431**	–	**6.1564E−04**	**66**	**399**	–	**1.7344E−06**	**0**	**465**	–
$f_{09(CEC)}$	**4.1955E−04**	**61**	**404**	–	4.1653E−01	193	272	≈	**1.7344E−06**	**0**	**465**	–
$f_{10(CEC)}$	**2.1266E−06**	**2**	**463**	–	**4.4493E−05**	**34**	**431**	–	1.8519E−02	347	118	+
$f_{11(CEC)}$	1.9209E−06	464	1	+	**6.8923E−05**	**39**	**426**	–	1.7344E−06	465	0	+
$f_{12(CEC)}$	1.7344E−06	465	0	+	5.7064E−04	400	65	+	1.7344E−06	465	0	+
$f_{13(CEC)}$	1.5886E−01	301	164	+	2.6230E−01	287	178	≈	**3.8723E−02**	**132**	**333**	–
$f_{14(CEC)}$	4.0483E−01	273	192	≈	1.1973E−03	390	75	+	3.8723E−02	333	132	+
$f_{15(CEC)}$	3.3269E−02	336	129	+	9.8421E−03	358	107	+	6.8836E−01	252	213	≈
$f_{16(CEC)}$	7.6552E−01	218	247	≈	**1.3595E−04**	**47**	**418**	–	1.7344E−06	465	0	+
$f_{17(CEC)}$	5.3197E−03	368	97	+	3.5152E−06	458	7	+	3.1817E−06	459	6	+
$f_{18(CEC)}$	2.4118E−04	411	54	+	5.2165E−06	454	11	+	2.4118E−04	411	54	+
$f_{19(CEC)}$	5.7165E−01	260	205	≈	1.7344E−06	465	0	+	2.1053E−03	382	83	+
$f_{20(CEC)}$	5.2165E−06	454	11	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{21(CEC)}$	6.3391E−06	452	13	+	2.8786E−06	465	0	+	2.8308E−04	409	56	+
$f_{22(CEC)}$	**1.1138E−03**	**74**	**391**	–	**1.6503E−01**	**165**	**300**	–	**1.2381E−05**	**20**	**445**	–
$f_{23(CEC)}$	4.3205E−08	462	0	+	1.7344E−06	465	0	+	1.7289E−06	465	0	+
$f_{24(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{25(CEC)}$	6.7889E−02	10	0	≈	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{26(CEC)}$	5.3044E−001	263	202	≈	**1.8462E−01**	**168**	**297**	–	**9.6266E−04**	**72**	**393**	–
$f_{27(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{28(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{29(CEC)}$	1.4773E−04	417	48	+	1.7344E−06	465	0	+	**1.7344E−06**	**0**	**465**	–
$f_{30(CEC)}$	1.0639E−01	311	154	+	1.7344E−06	465	0	+	**1.7344E−06**	**0**	**465**	–
+/≈/–		19/5/6				22/2/6				20/2/8		

No.	MFO versus BFS				WOA versus BFS			
	p value	R+	R-	Win	*p* value	R+	R-	Win
$f_{01(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{02(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{03(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{04(CEC)}$	2.1266E−06	463	2	+	1.7344E−06	465	0	+
$f_{05(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{06(CEC)}$	**3.3789E−03**	**90**	**375**	–	2.1266E−06	463	2	+
$f_{07(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{08(CEC)}$	6.4242E−03	365	100	+	1.9209E−06	464	1	+
$f_{09(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{10(CEC)}$	5.1931E−02	327	138	+	1.7988E−05	441	24	+
$f_{11(CEC)}$	3.5009E−02	335	130	+	5.7517E−06	453	12	+
$f_{12(CEC)}$	**1.9209E−06**	**1**	**464**	–	6.3198E−05	427	38	+
$f_{13(CEC)}$	1.7344E−06	465	0	+	2.3704E−05	438	27	+
$f_{14(CEC)}$	1.7344E−06	465	0	+	6.4242E−03	365	100	+

Table 13 (continued)

No.	MFO versus BFS				WOA versus BFS			
	p value	R+	R-	Win	*p* value	R+	R-	Win
$f_{15(CEC)}$	1.9209E−06	464	0	+	1.7344E−06	465	0	+
$f_{16(CEC)}$	2.7029E−02	340	125	+	1.7988E−05	441	24	+
$f_{17(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{18(CEC)}$	2.1266E−06	463	2	+	9.3157E−06	448	17	+
$f_{19(CEC)}$	2.3534E−06	462	3	+	1.7344E−06	465	0	+
$f_{20(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{21(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{22(CEC)}$	8.9187E−05	423	42	+	1.9209E−06	464	1	+
$f_{23(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{24(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{25(CEC)}$	1.7344E−06	465	0	+	1.1905E−05	325	0	+
$f_{26(CEC)}$	2.1266E−06	463	2	+	1.4704E−01	303	162	+
$f_{27(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{28(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{29(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{30(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
+/≈/–		28/0/2				30/0/0		
No.	CSA versus BFS				SBO versus BFS			
	p value	R+	R-	Win	*p* value	R+	R-	Win
$f_{01(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{02(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{03(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{04(CEC)}$	4.0715E−05	432	33	+	1.9152E−01	296	169	+
$f_{05(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{06(CEC)}$	**1.7088E−03**	**80**	**385**	–	8.2901E−01	222	243	≈
$f_{07(CEC)}$	1.7088E−03	385	80	+	1.7344E−06	465	0	+
$f_{08(CEC)}$	4.6818E−03	370	95	+	1.8519E−02	347	118	+
$f_{09(CEC)}$	1.3820E−03	388	77	+	1.3601E−05	444	21	+
$f_{10(CEC)}$	1.3591E−01	305	160	+	**1.7088E−03**	**80**	**385**	–
$f_{11(CEC)}$	**1.1748E−02**	**110**	**355**	–	**1.2453E−02**	**111**	**354**	–
$f_{12(CEC)}$	**2.1266E−06**	**2**	**463**	–	**1.7344E−06**	**0**	**465**	–
$f_{13(CEC)}$	4.8603E−05	430	35	+	5.7096E−02	325	140	+
$f_{14(CEC)}$	2.9575E−03	377	88	+	7.4987E−01	217	248	≈
$f_{15(CEC)}$	1.6503E−01	300	165	+	1.9209E−06	464	1	+
$f_{16(CEC)}$	3.8203E−01	190	275	≈	**2.0589E−01**	**171**	**294**	–
$f_{17(CEC)}$	**1.6566E−02**	**116**	**349**	–	1.7344E−06	465	0	+
$f_{18(CEC)}$	1.3820E−03	388	77	+	1.8910E−04	414	51	+
$f_{19(CEC)}$	3.1817E−06	459	6	+	3.1817E−06	459	6	+
$f_{20(CEC)}$	7.7122E−04	396	69	+	1.7344E−06	465	0	+
$f_{21(CEC)}$	**1.2453E−02**	**111**	**354**	–	1.7344E−06	465	0	+
$f_{22(CEC)}$	1.7791E−01	298	167	+	2.3534E−06	462	3	+
$f_{23(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{24(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{25(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{26(CEC)}$	4.9080E−01	266	199	≈	2.5846E−03	379	86	+
$f_{27(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+
$f_{28(CEC)}$	1.7344E−06	465	0	+	1.7344E−06	465	0	+

Table 13 (continued)

No.	CSA versus BFS				SBO versus BFS			
	p value	R+	R-	Win	*p* value	R+	R-	Win
$f_{29(CEC)}$	1.7344E−06	465	0	+	**4.5336E−04**	**62**	**403**	–
$f_{30(CEC)}$	1.7344E−06	465	0	+	**3.5152E−06**	**7**	**458**	–
+/≈/–		23/2/5				22/2/6		

The results that BFS is significantly worse than the compared algorithms are shown in bold

Acknowledgements The authors wish to thank the editors and anonymous reviewers whose kind assistance and constructive comments helped to significantly improve this paper. This work is supported by the National Natural Science Foundation of China under Grant No. 61601505.

References

1. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput Struct 169:1–12. https://doi.org/10.1016/j.compstruc.2016.03.001
2. Bansal JC, Sharma H, Jadon SS, Clerc M (2014) Spider monkey optimization algorithm for numerical optimization. Memetic Comp 6:31–47. https://doi.org/10.1007/s12293-013-0128-0
3. Barmada S, Raugi M, Tucci M (2016) An evolutionary algorithm for global optimization based on self-organizing maps. Eng Optimiz 10:1–19. https://doi.org/10.1080/0305215X.2015.1128424
4. Clerc M, Kennedy J (2002) The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6:58–73. https://doi.org/10.1109/4235.985692
5. Colorni A, Dorigo M, Maniezzo V (1991) Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life; Paris, France, 1991, pp 134–142
6. Eberhart RC, Kennedy J (1995) A new optimizer using particles swarm theory. In: Proceedings of the 6th international symposium on micro machine and human science, Nagoya, Japan, 1995, pp 39–43. http://dx.doi.org/10.1109/mhs.1995.494215
7. Fedy BC, Stutchbury BJM (2004) Territory switching and floating in white-bellied antbird (myrmeciza longipes), a resident tropical passerine in panama. Auk 121:486–496. https://doi.org/10.1642/0004-8038(2004)121%5B0486:TSAFIW%5D2.0.CO;2
8. Healy SD, Hurly TA (2003) Cognitive ecology: foraging in hummingbirds as a model system. Adv Stud Behav 32:325–359. https://doi.org/10.1016/S0065-3454(03)01007-6
9. Holland J (1992) Genetic algorithms. Sci Am 267:66–72. https://doi.org/10.1038/scientificamerican0792-66
10. Jaddi NS, Alvankarian J, Abdullah S (2017) Kidney-inspired algorithm for optimization problems. Commun Nonlinear Sci 42:358–369. https://doi.org/10.1016/j.cnsns.2016.06.006
11. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39:459–471. https://doi.org/10.1007/s10898-007-9149-x
12. Kaveh A, Dadras A (2017) A novel meta-heuristic optimization algorithm: thermal exchange optimization. Adv Eng Softw 110:69–84. https://doi.org/10.1016/j.advengsoft.2017.03.014
13. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 42:671–680
14. Liang JJ, Qu BY, Suganthan PN (2014) Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real parameter numerical optimization, Tech. Rep. 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China
15. Lorimer JW (2006) Curved paths in raptor flight: deterministic models. J Exp Biol 242:880. https://doi.org/10.1016/j.jtbi.2006.03.020
16. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249. https://doi.org/10.1016/j.knosys.2015.07.006
17. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
18. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
19. Moosavi SHS, Bardsiri VK (2017) Satin bowerbird optimizer: a new optimization algorithm to optimize ANFIS for software development effort estimation. Eng Appl Artif Intel 60:1–15. https://doi.org/10.1016/j.engappai.2017.01.006
20. Newton I (1979) Population ecology of raptors. J Anim Ecol 50:1–399. https://doi.org/10.2307/4081
21. Pan JS, Meng Z, Chu SC, Xu HR (2017) Monkey King Evolution: an enhanced ebb-tide-fish algorithm for global optimization and its application in vehicle navigation under wireless sensor network environment. Telecommun Syst 65:351–364. https://doi.org/10.1007/s11235-016-0237-4
22. Rao RV, Savsani VJ, Vakharia DP (2012) Teaching–learning-Based optimization: an optimization method for continuous non-linear large scale problems. Inf Sci 183:1–15. https://doi.org/10.1016/j.ins.2011.08.006
23. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inform Sciences 179:2232–2248. https://doi.org/10.1016/j.ins.2009.03.004
24. Skalak DB (1994) Prototype and feature selection by sampling and random mutation hill climbing algorithms. Mach Learn Proc 293–301. http://doi.org/10.1016/B978-1-55860-335-6.50043-X
25. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359. https://doi.org/10.1023/A:1008202821328
26. Sun G, Zhao R, Lan Y (2016) Joint operations algorithm for large-scale global optimization. Appl Soft Comput 38:1025–1039. https://doi.org/10.1016/j.asoc.2015.10.047
27. Tabari A, Ahmad A (2017) A new optimization method: electro-search algorithm. Comput Chem Eng 103:1–11. https://doi.org/10.1016/j.compchemeng.2017.01.046
28. Wang GG (2016) Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comp 1–14. https://doi.org/10.1007/s12293-016-0212-3

29. Wimpenny JH, Weir AA, Clayton L, Rutz C, Kacelnik A (2009) Cognitive processes associated with sequential tool use in New Caledonian crows. PLoS ONE 4:1–16. https://doi.org/10.1371/journal.pone.0006471
30. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82. https://doi.org/10.1109/4235.585893
31. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-inspir Com 2:78–84. https://doi.org/10.1504/IJBIC.2010.032124
32. Yang XS, Deb S (2009) Cuckoo search via lévy flights. In: Proceedings of the world congress on nature and biologically inspired computing (NaBIC 2009), Coimbatore, India, 9–11 December 2009; pp 210–214. http://doi.org/10.1109/NABIC.2009.5393690
33. Yasukawa K (1979) Territory establishment in red-winged blackbirds: importance of aggressive behavior and experience. Condor 81:258–264. https://doi.org/10.2307/1367628