Received December 10, 2019, accepted January 7, 2020, date of publication January 17, 2020, date of current version February 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2967399

Improved Grasshopper Algorithm Based on Gravity Search Operator and Pigeon Colony Landmark Operator

S. S. GUO, J. S. WANG, (Member, IEEE), W. XIE, M. W. GUO, AND L. F. ZHU
School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan 114044, China

Corresponding author: J. S. Wang (wang_jiesheng@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 21576127, in part by the Basic Scientific Research Project of Institution of Higher Learning of Liaoning Province under Grant 2017FWDF10, and in part by the Liaoning Provincial Natural Science Foundation of China under Grant 20180550700.

ABSTRACT The grasshopper optimization algorithm (GOA) is a new meta-heuristic algorithm inspired by the behavior of grasshopper groups. Aiming at the shortcomings of poor development ability and low convergence accuracy of GOA, this paper introduces the gravity search operator into the optimization process of GOA to improve the grasshopper's global exploration and avoid falling into local optimum in advance. At the same time, a pigeon search operator-landmark operator is introduced to improve and balance the algorithm's exploration and development capabilities. In order to verify the validity of the improved algorithm, this paper will adopts the gravity search operator and a deterrent landmark operator hybrid grasshoppers algorithm (HGOA) with basic grasshopper algorithm (GOA), particle swarm optimization (PSO) algorithm, sine and cosine algorithm (SCA), moth-flame optimization (MFO) algorithm, salp swarm algorithm (SSA), and bat algorithm (BA) to optimize 28 test functions. And the analysis and comparison of the obtained statistical data results finally show that the proposed improved grasshopper algorithm has better optimization ability.

INDEX TERMS Grasshopper algorithm, gravity search operator, Pigeon landmark operator, function optimization.

I. INTRODUCTION

Optimization is the process of finding the best solution for a particular problem. As the complexity of the problem increases, the need for new optimization techniques has become more apparent in the past few decades than before. Before heuristic optimization was proposed, mathematical optimization was the only tool used to optimize problems. Mathematical optimization methods are mainly deterministic and have a major problem: local optimal notch waves, which makes them extremely inefficient in solving practical problems. Natural heuristic algorithms can effectively help solve many optimization problems encountered in actual processes, which makes natural heuristic algorithms a research hotspot for optimization problems in recent years [1]. Natural heuristic algorithm includes heuristic algorithm and meta-heuristic algorithm. Heuristic algorithms are the process of finding solutions through trial and error [2]. Many heuristic algorithms, such as simulated annealing (SA) and artificial neural network (ANN), have been widely used in the field of optics and combination optimization. The meta-heuristic algorithm is a stochastic optimization algorithm with prior knowledge of random search. It is an optimization process that starts with a random solution and then randomly explores and utilizes the search space with a specific probability. These algorithms can use the useful information of the population to find the optimal solution [3]. These efficient and robust algorithms are used to solve a variety of problems, such as path planning [4], economic scheduling problem [5], inverter parameter identification [6], backpack problem [7] and location problem [8]. So far, various researchers have conducted in-depth research on these algorithms, and introduced many naturally-inspired meta-heuristic algorithms, such as particle swarm optimization (PSO) algorithm [9], bacterial foraging algorithm (BFA) [10], artificial fish swarm algorithm (AFSA) [11], artificial bee colony (ABC) algorithm [12], cuckoo search (CS)

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

algorithm [13], bat algorithm (BA) [14], ant lion optimizer (ALO) [15], moth-flame optimization (MFO) algorithm [16], and salp swarm algorithm (SSA) [17], etc.

Grasshopper algorithm is a new meta-heuristic algorithm proposed by Saremi *et al.* in 2017 [18], which is inspired by the swarm behavior of locusts. Since the algorithm was proposed, it has been widely used in many fields, such as optimal power flow [19], feature selection [20], financial stress prediction [21], and image segmentation [22]. In fact, this swarm based meta-heuristic algorithm avoids the stagnation of local optimality to some extent, but it is found in the test function that GOA has the disadvantages of poor global search ability and slow convergence speed. Therefore, in some cases, GOA cannot find a global optimal solution. The search strategy used in the base GOA is primarily based on minimums and random walks, so it cannot always successfully solve the optimization problems. Thus many researches focus on improving the performance of meta-heuristic algorithms by promoting exploration and exploitation.

Wu *et al.* [23] introduced natural selection strategy and democratic decision-making mechanism into GOA to avoid falling into local optimization. In addition, based on the dynamic feedback mechanism, the 1/5 principle is also introduced to adjust the parameter c to better balance global and local search. Liu *et al.* [24] proposed an integration strategy combining linear weighting and grasshopper optimization algorithm to solve the problem of energy management, improve the comprehensive energy efficiency and realize the regional coordination optimization. Ewees *et al.* [25] proposed an improved GOA. The proposed OBLGOA algorithm includes two stages: the first stage uses the OBL strategy to generate the initial population, and the second stage uses OBL as an additional stage to update the GOA population in each iteration. Luo *et al.* [21] combined three strategies to achieve a more appropriate balance between exploration and exploitation. The Gaussian mutation was adopted to increase population diversity and make GOA have stronger exploitation capabilities. Then, Levy flight was used to enhance the randomness of the search agent's movement and make GOA have a stronger global exploration ability. In addition, the opposition-based learning was introduced into GOA to achieve a more efficient search solution space. Arora and Anand [3] introduced chaos theory into the optimization process of GOA, which improved the global convergence speed of GOA. In the optimization process, the use of chaotic maps effectively balanced the exploration and exploitation among locusts, and reduced the repulsion and attraction between locusts. Sulaiman *et al.* [26] updated GOA with a better initialization strategy to balance GOA's search capabilities. Yan and Ye [27] proposed a hybrid locust optimization algorithm based on the idea of quantum computing in order to overcome the shortcomings of the algorithm with low convergence accuracy and used quantum revolving gate operation to improve it. Liang *et al.* [28] used Levy flight to balance the exploration and development of GOA so as to improve the convergence speed and accuracy of the algorithm. Taher *et al.* [19] applied random mutations of GOA to enhance the new search area to avoid local optimal value stagnation. Jia *et al.* [22] proposed an effective satellite image segmentation method based on hybrid GOA and minimum cross entropy (MCE), and they combined GOA with adaptive differential evolution (DE) to improve search efficiency, especially to preserve population diversity in subsequent iterations. An improved grasshopper algorithm based on gravity search operator and pigeon colony landmark operator was proposed. Firstly, this paper introduces the gravity search operator (GGOA) on the basis of the original GOA, which improves the randomness of the algorithm, increases the global search ability of the algorithm, and avoids falling into local optimization. On this basis, pigeon landmark operator is added to improve and balance the exploration and exploitation ability of the algorithm. The selection strategy is adopted in the location update strategy, which further increases the selection diversity of the algorithm and improves the convergence accuracy of the algorithm.

II. BASIC PRINCIPLES OF GRASSHOPPER ALGORITHM

The grasshopper optimization algorithm is inspired by the foraging behavior of grasshoppers [18]. Grasshoppers are a type of insect that are usually seen alone, but they are one of the largest groups of organisms. The life cycle of grasshopper is divided into two main stages: larval stage and adult stage. The larval stage is characterized by the slow movement and small movements of locusts. The adult stage is characterized by a large range of sudden movements. Similarly, naturally inspired algorithms logically divide the search process into two trends: exploration and exploitation. In the exploration phase, grasshoppers tend to move quickly, while in the exploitation phase, they are encouraged to move locally. Both of these functions, as well as finding food sources, are naturally realized by grasshoppers. Theoretically speaking, the swarm behavior of grasshoppers can be summarized as:

$$X_i = S_i + G_i + A_i \tag{1}$$

where, X_i represents the location of the $i-th$ grasshopper, S_i represents social interaction, G_i represents the gravity of the $i-th$ grasshopper, and A_i represents wind advection.

The S component in Eq. (1) can be calculated by:

$$S_i = \sum_{j=1, j\neq i}^{N} s\left(d_{ij}\right)\hat{d}_{ij} \tag{2}$$

where, d_{ij} represents the distance between the $i-th$ grasshopper and the $j-th$ grasshopper, and the expression is $d_{ij} = \left|x_j - x_i\right|$. s is the strength of the social force, and it is calculated by using Eq. (3). The unit vector between the $i-th$ grasshopper and the $j-th$ grasshopper is calculated in terms of $\hat{d}_{ij} = (x_l - x_i)\big/d_{ij}$. The social power is defined by the s function, which can be calculated by:

$$s(r) = fe^{\frac{r}{l}} - e^{-r} \tag{3}$$

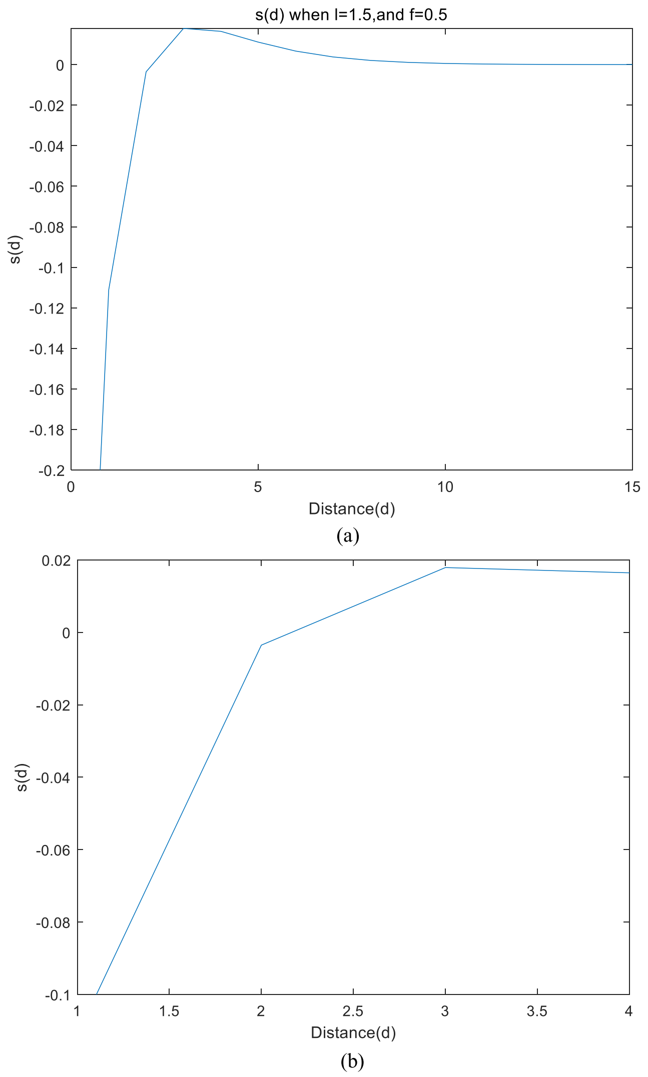


FIGURE 1. The effect of S function on grasshopper's social interaction. (a) is the effect of S function on grasshopper social interaction when l=1.5 and f=0.5; (b) is the effect of the function s on grasshopper social interaction when l=1.5, f=0.5, and when x is in [1,4].

where, f is the intensity of attraction and l is the scale of attraction length. According to literature [18], when $f = 0.5$, $l = 1.5$, the algorithm has best optimization performance.Fig. 1 shows the influence of s function on grasshopper social interaction (attraction and exclusion).

It can be seen form Fig. 1 that a distance from 0 to 15 is considered. Rejection occurs in the interval [0, 2.079]. When one grasshopper is 2.079 units away from another, it is neither attractive nor repulsive. This is called a comfort zone or comfort distance. Fig. 1 also shows that the attraction increases from 2.079 to close to 4 and then decreases gradually. The component G in Eq. (1) can be calculated by:

$$G_i = -g\hat{e}_g \tag{4}$$

where, g stands for the gravitational constant, and $\hat{e}_g$ is the unit vector pointing to the center of the earth.

The component A in Eq. (1) can be calculated by Eq. (5).

$$A_i = u\hat{e}_w \tag{5}$$

where, u represents the constant drift, and $\hat{e}_w$ is the unit vector of the wind direction.

Substitute S, G and A into Eq. (1) to obtain:

$$X_i = \sum_{j=1,j\neq i}^{N} s\left(\left|x_j - x_i\right|\right)\frac{x_j - x_i}{d_{ij}} - g\hat{e}_g + u\hat{e}_w \tag{6}$$

where, N represents the number of grasshoppers.

In the optimization algorithm, Eq. (6) is not used because it prevents the optimization algorithm from exploration and exploitation the searching space near the solution [18]. In fact, this grasshopper model is designed for colonies of grasshoppers that live in free space. In addition, the mathematical model is not directly used to solve the optimization problem because the grasshopper quickly reaches the comfort zone, while the grasshopper does not converge to the specified point. Therefore, the modified Eq. (6) is used to solve the optimization problem.

$$X_i^d = c_1\left(\sum_{j=1,j\neq i}^{N} c_2\frac{ub_d - lb_d}{2}s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}}\right) + \hat{T}_d \tag{7}$$

where, ub_d and lb_d represent the upper and lower limits respectively. $\hat{T}_d$ represents the optimal position. c_1 and c_2 represent coefficients of contraction comfort zone, exclusion zone and attractiveness.

It can be seen from Eq. (7) that the location of the grasshopper next time is determined according to its current position, target position and the location of all other grasshoppers. Eq. (7) consists of two parts: the first part considers the location of the current grasshopper relative to other grasshoppers in the region; The second part reduced the movement of grasshoppers around the target, allowing for the exploration of the entire population around the target. Specifically, the parameter c_1 is responsible for reducing the movement of locusts around the target, which means that c_1 basically balances the exploration and exploitation of the entire population around the target; The parameter c_2 reduces the attraction zone, comfort zone and exclusion zone between grasshoppers, that is to say that c_2 linearly reduces the space to guide grasshoppers to find the optimal solution in the search space. It is worth noting that the adaptive parameter c_1 helps reduce the repulsion or attraction between grasshoppers, which is proportional to the number of iterations, while c_2 reduces the search range around the target as the number of iterations increases. In order to balance exploration and exploitation, c_1 decreases proportionally as the number of iterations increases [3]. Similarly, as the number of iterations increases, the value of c_2 decreases in order to minimize the comfort zone. In [18], the parameters (c_1 and c_2) are considered as a single parameter and modified using the following methods:

$$c = c_{\max} - t\frac{c_{\max} - c_{\min}}{T} \tag{8}$$

where, $c_{\max}$ and $c_{\min}$ represent the maximum and minimum values, t represents the current number of iterations, and T is the maximum number of iterations.

The algorithm flowchart of the grasshopper algorithm is shown in Fig. 2. The pseudocode of the grasshopper algorithm is described as follows [18].

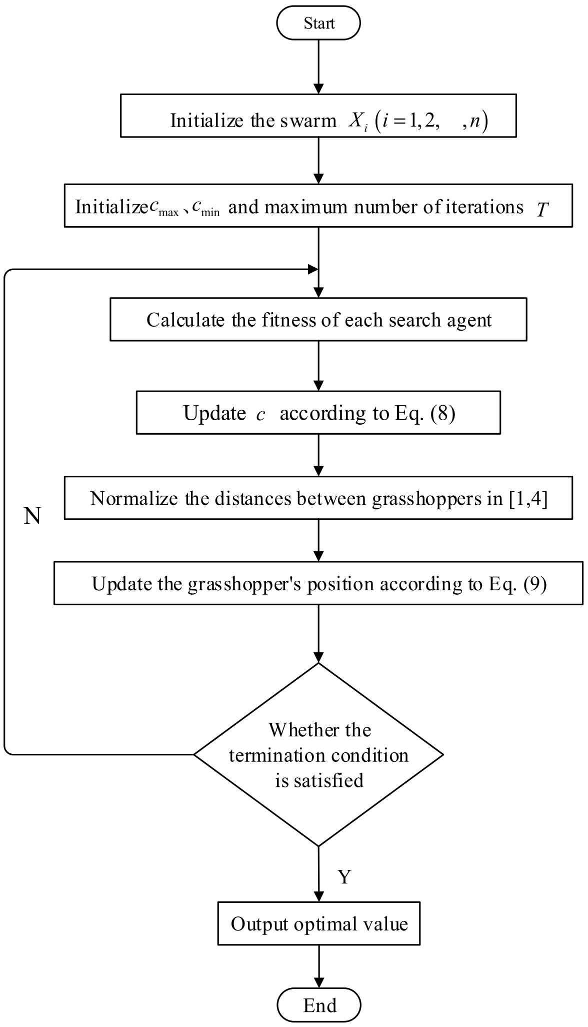

Start
Initialize the swarm X_i $(i=1,2,\quad,n)$
Initialize $c_{\max}$、$c_{\min}$ and maximum number of iterations T
Calculate the fitness of each search agent
Update c according to Eq. (8)
Normalize the distances between grasshoppers in [1,4]
Update the grasshopper's position according to Eq. (9)
Whether the termination condition is satisfied
N
Y
Output optimal value
End

FIGURE 2. Operation flowchart of grasshopper algorithm.

```
Initialize the swarm $X_i$ $(i = 1, 2, \cdots, n)$
Initialize $c_{\max}$, $c_{\min}$ and maximum number of iterations
Calculate the fitness of each search agent
Target = the best search agent
while (l<Max number of iterations)
    Update $c$ using Eq.(8)
    for each search agent
        Normalize the distances between grasshoppers in [1,4]
        Update the position of the current search agent by the Eq.(7)
        Bring the current search agent back if it outside the boundaries
    end for
    Update Target if there is a better solution
    l=l+1
end while
Return Target
```

III. HYBRID GRASSHOPPER ALGORITHM BASED ON GRAVITY SEARCH OPERATOR AND PIGEON COLONY LANDMARK OPERATOR

A. GRAVITY SEARCH OPERATOR

Inspired by the law of universal gravitation in physics, Rashedi et al. [29] proposed the Gravitational Search Algorithm (GSA) in 2009. GSA believes that each particle in the search space has a mass, and can actually perform undamped objects, and the objects are attracted to each other by gravitation. The law of motion follows Newton's second law, so that it moves toward the optimal position under the action of this law movement [30]. The standard gravitational search algorithm is described as follows. Suppose the dimension of the search space is D, the population size is N, and the initial population is $P = \{X_1, X_2, \cdots, X_N\}$, where $X_{\mathrm{i}} = \{x_i^1, x_i^2, \cdots, x_i^n\}$, $i = 1, 2, \cdots, N$ is the position of the $i-th$ particle, and x_i^k represents the position of the $i-th$ particle on the $k-th$ dimension.

(1) The inertial mass $Mass_i(t)$ of each particle i can be defined as:

$$\begin{cases} m_i(t) = \dfrac{fit_i(t) - worst(t)}{best(t) - worst(t)} \\ Mass_i(t) = \dfrac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \end{cases} \tag{9}$$

where, $fit_i(t)$ is the fitness value of particle i at time t. When solving the minimum value problem, $best(t) = \min(fit_j(t)), j = 1, 2, \cdots, N$, $worst(t) = \max(fit_j(t)), j = 1, 2, \cdots, N$; And when solving the maximum problem, $best(t) = \max(fit_j(t)), j = 1, 2, \cdots, N, worst(t) = \min(fit_j(t)), j = 1, 2, \cdots, N$.

(2) The gravity of particle j on the $k-th$ dimension of particle i can be defined as:

$$F_{ij}^k(t) = G(t)\frac{Mass_i(t) - Mass_j(t)}{R_{ij}(t) + \varepsilon}\left(x_j^k(t) - x_i^k(t)\right) \tag{10}$$

where, $Mass_i(t)$ and $Mass_j(t)$ are the inertial masses of particles i and j respectively. ε is a very small constant, $R_{ij}(t)$ is the Euclidean distance between particles i and j, and $G(t)$ is the universal gravitational constant at time t, which decreases with the age of the universe and is calculated by:

$$G(t) = G(0) \times e^{-\frac{\alpha t}{T}} \tag{11}$$

where, $G(0)$ is the universal gravitational constant at time $t = 0$, α is the attenuation coefficient, and T is the maximum number of iterations.

(3) The resultant force of particle i in the $k-th$ dimension can be calculated by:

$$F_i^k(t) = \sum_{j \in K_{best}, i \neq j} rand_j F_{ij}^k(t) \tag{12}$$

where, $rand_j$ is the random number within the interval [0,1], K_{best} is the document storing the elite particle of the population and its scale decreases linearly with the number of iterations.

According to Newton's second law, at time t, the acceleration of particle i in the $k-th$ dimension can be defined as:

$$a_i^k(t) = \frac{F_i^k(t)}{Mass_i(t)} \tag{13}$$

Therefore, the velocity and position of the particle at time $t+1$ are respectively described as:

$$v_i^k(t+1) = rand \times v_i^k(t) + a_i^k(t) \tag{14}$$

$$x_i^k(t+1) = x_i^k(t) + v_i^d(t+1) \tag{15}$$

B. PIGEON-INSPIRED ALGORITHM

Pigeon-inspired optimization (PIO) algorithm is a bionic intelligent optimization algorithm proposed by Duan et al in 2014 [31]. Each pigeon in PIO algorithm corresponds to a feasible solution, which has two attributes of position and speed, which respectively represents the current position and moving speed of the pigeon in the solution space, and evaluates the quality of the pigeon through the fitness function. The PIO algorithm uses a compass operator and a landmark operator to imitate navigation tools in different flight phases, and completes evolution and screening operations in two independent loops to achieve optimization. The landmark operator can be described as:

$$V_i(t) = V_i(t-1)\cdot e^{-Rt} + rand\cdot(X_g - X_i(t-1)) \tag{16}$$

$$X_i(t) = X_i(t-1) + V_i(t) \tag{17}$$

where, R is the map and guide factor . t represents the current number of iterations. $X_i(t)$ and $V_i(t)$ represent the position and velocity of pigeon i in generation t, X_{gbest} represents the global best position obtained by comparing the positions of all pigeons after the iteration $t-1$, and $rand$ is a random number on the scope [0,1].

The landmark operator is used to imitate the role of landmarks in individual pigeon flock evaluation and correction when the pigeons are flying close to the destination [31]. If the pigeons are familiar with landmarks, they can fly straight to their destination. Conversely, if the pigeons are unfamiliar with the landmark and stay away from the destination, they will follow the familiar pigeon to reach the destination. In the landmark operator, the number of pigeons in each generation is halved by Np. However, the pigeons are still far from their destination and they are not familiar with landmarks. Let $X_c(t)$ be the center of a pigeon's position at the $t-th$ iteration, assuming that each pigeon can fly directly to its destination. The position update rule of pigeon i at the $t-th$ iteration can be described as follows.

$$N_p(t) = \frac{N_p(t-1)}{2} \tag{18}$$

$$X_c(t) = \frac{\sum X_i(t)\cdot fitness(X_i(t))}{N_p \sum fitness(X_i(t))} \tag{19}$$

$$X_i(t) = X_i(t-1) + rand\cdot(X_c(t) - X_i(t-1)) \tag{20}$$

where, $fitness()$ represents the fitness value of pigeon.

C. IMPROVED GRASSHOPPER ALGORITHM

According to Eq.(7), after a given initial population, the search individual will converge to the target location, and continuously adjust the location according to the influence of interaction forces during the convergence process, and finally make the population more evenly distributed around the target location. This unique way of interacting with other search individuals is a significant feature that distinguishes it from other algorithms and allows the algorithm to have better exploitation capabilities. However, this remarkable feature also leads to the shortcomings of the basic locust optimization algorithm, such as rapid population diversity decline and insufficient global search ability, which leads to the algorithm falling into local optimization and insufficient convergence precision. In order to improve the performance of the algorithm, the acceleration in gravity search is introduced to enhance the global search ability, which is described as follows.

$$X_i^d = c_1\left(\sum_{j=1,j\neq i}^{N} c_2\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}} + a_i^d\right) + \hat{T}_d \tag{21}$$

where, a_i^d is the acceleration in the gravitational search. The purpose of a_i^d is to increase the randomness of grasshopper movement, improve grasshopper's global exploration and avoid falling into local optimization. Suppose $Sol_i^d = \sum_{j=1,j\neq i}^{N} c_2\frac{ub_d-lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j-x_i}{d_{ij}} + a_i^d$, Sol_i^d indicates the new position of the grasshopper formed by the effect of the force between grasshoppers. Eq. (7) can be simplified as:

$$X_i^d = c_1 Sol_i^d + \hat{T}_d \tag{22}$$

At the same time, in order to improve the convergence speed of the algorithm, this paper introduces the search operator in the pigeon algorithm after introducing the gravity search operator, which is described as follows.

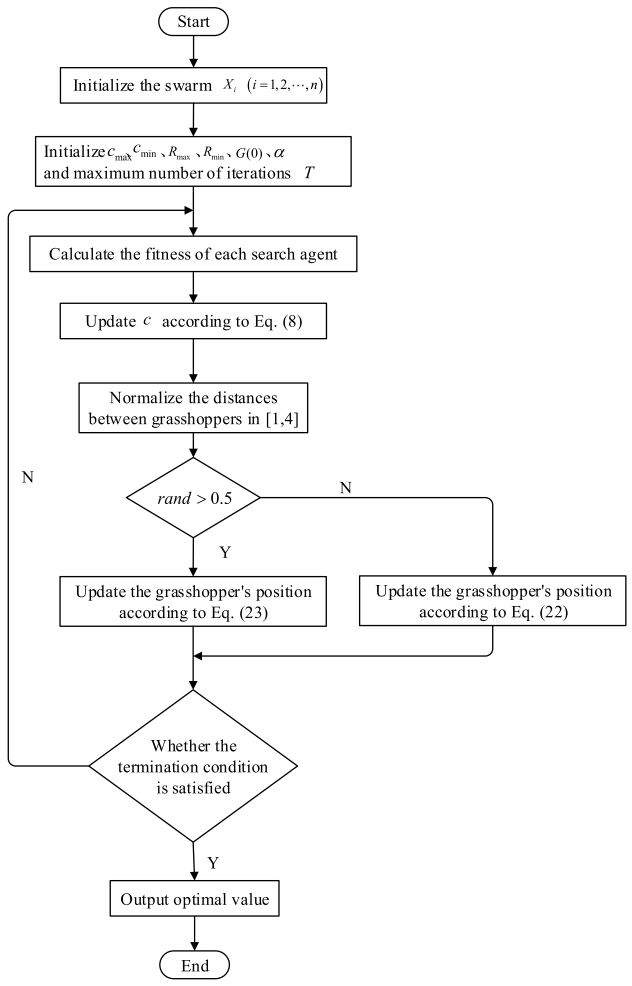
$$X_i^d = c_1 Sol_i^d + e^{-Rt}\left(X_i^d - \hat{T}_d\right) \tag{23}$$

where, R represents the map and guide factor. The research results of literature show that the smaller R is, the stronger the search ability is and the greater the development potential is [45]. In order to solve the balanced development problem, the linear variation strategy is adopted in this paper, and the calculation formula is defined as:

$$R = \left(R_{\min} + R_{\max}\frac{t}{T}\right)(1 + p_r(rand - 1)) \tag{24}$$

where, $R_{\max}$ represents the maximum map compass factor value, $R_{\min}$ represents the minimum map compass factor value, and p_r represents the probability of variation.

Therefore, the operation flow chart of the improved grasshopper algorithm is shown in Fig.3, and the pseudocode is described as follows:



Start
Initialize the swarm X_i $(i=1,2,\cdots,n)$
Initialize $c_{\max}$、$c_{\min}$、$R_{\max}$、$R_{\min}$、$G(0)$、α and maximum number of iterations T
Calculate the fitness of each search agent
Update c according to Eq. (8)
Normalize the distances between grasshoppers in [1,4]
$rand > 0.5$
N
Y
Update the grasshopper's position according to Eq. (23)
Update the grasshopper's position according to Eq. (22)
Whether the termination condition is satisfied
N
Y
Output optimal value
End

FIGURE 3. **Operation flowchart of improved grasshopper algorithm.**

```
Initialize the swarm $X_i$ $(i = 1, 2, \cdots, n)$
Initialize $c_{\max}, c_{\min}, R_{\max}, R_{\min}, G(0), \alpha$ and maximum number of iterations $T$
Calculate the fitness of each search agent
Target = the best search agent
while ($l < T$)
    Update $c$ using Eq.(8)
    for each search agent
        Normalize the distances between grasshoppers in [1,4]
        if rand>0.5
            Update the position of the current search agent by the Eq.(23)
        else if
            Update the position of the current search agent by the Eq.(22)
        end
        Bring the current search agent back if it outside the boundaries
    end for
    Update Target if there is a better solution
    l=l+1
end while
Return Target
```

D. THE TIME COMPLEXITY OF THE HWOA

The computational complexity of the algorithm is an important index to evaluate its running time, which can be defined according to the structure and implementation of the algorithm. This section uses the big oh notation, which is the worst-case time complexity. The time complexity of HWOA depends on the number of grasshoppers, the dimension of the variable, the maximum number of iterations, and the location-updating mechanism of grasshoppers in each iteration. The calculation of the specific time complexity is as follows:

① To initialize and assign the value of n grasshoppers in d- dimensional search space, dn times should be run.

② Computing the fitness values of each grasshoppers, and inertial mass to run n times, and selecting the optimal fitness value needs to run $\frac{n(n-1)}{2}$ times.

③ To calculate the distance between grasshoppers and normalize it to [1,4] requires running $\frac{n(n-1)}{2}$ times.

④ Updates of parameters c, R and G need to be run once, respectively.

⑤ According to Eq.13, the acceleration of n grasshoppers in the d dimension needs to be run $(n(n-1)d)$ times.

⑥ According to Eq.23 or Eq.22, updating the position of the grasshopper requires running n^2 times.

⑦ The optimal value and the judgment algorithm need to run once respectively.

In summary, the time complexity of HWOA is simplified as:

$$O(HWOA) = t\left(\frac{n(n-1)}{2} + 3 + n(n-1)d + n^2 + 2\right)$$
$$= \frac{(3+2d)n^2t - (1+2d)nt + 10t}{2} \tag{25}$$

IV. SIMULATION EXPERIMENTS AND RESULT ANALYSIS ON FUNCTION OPTIMIZATION PROBLEMS

A. TEST FUNCTIONS

In order to verify the performance of the proposed HGOA in this section, 28 most widely used benchmark functions are adopted for evaluation. The expressions of the 28 benchmark functions are shown in Table 1. In order to prove the superiority of the improved algorithm from multiple angles, the 28 functions are divided into three groups. $F_1 \sim F_{12}$ are unimodal functions [33]–[35], that is to say that there is only one global optimal solution in the solution space of the function, so these unimodal functions are used to test the convergence rate of the algorithm. $F_{13} \sim F_{22}$ are multimodal functions [35], [36]. There is more than one extremum in multimodal function, so there are many local optima in multimodal function, so the algorithm is easy to fall into

TABLE 1. Benchmark functions.

Function	Dim	Range	f_{min}
$F_1(x)=\sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[-10,10]	0
$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	30	[-100,100]	0
$F_4(x)=\max_i\left\{\lvert x_i\rvert,1\le i\le n\right\}$	30	[-100,100]	0
$F_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	30	[-30,30]	0
$F_6(x)=\sum_{i=1}^{n} i x_i^4+rand[0,1)$	30	[-100,100]	0
$F_7(x)=x_1^2+10^6\sum_{i=2}^{n} x_i^{\;2}$	30	[-10,10]	0
$F_8(x)=\sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	30	[-1,1]	0
$F_9(x)=\sum_{i=1}^{n}\lvert x_i\rvert$	30	[-100,100]	0
$F_{10}(x)=\sum_{i=1}^{n} x_i^{\;10}$	30	[-10,10]	0
$F_{11}(x)=\sum_{i=1}^{n-1}\left(x_i^2\right)^{\left(x^2_{i+1}+1\right)}+\left(x^2_{i+1}\right)^{\left(x_i^2+1\right)}$	30	[-1,4]	0
$F_{12}(x)=\left(x_1-1\right)^2+\sum_{i=1}^{n} i\left(2x_i^2-x_{i-1}\right)^2$	30	[-10,10]	0
$F_{13}(x)=\sum_{i=1}^{n}-x_i^2\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[-500,500]	-418.9829×Dim
$F_{14}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	30	[-32,32]	0
$F_{15}(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[-600,600]	0
$F_{16}(x)=\frac{\pi}{n}\{10\sin\left(\pi y_1\right)+\sum_{i=1}^{n-1}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]$ $+\left(y_n-1\right)^2\}+\sum_{i=1}^{n} u\left(x_{i,},10,100,4\right)$ $y_i=1+\frac{x_i+1}{4}$ $u\left(x_i,a,k,m\right)=\begin{cases}k\left(x_i-a\right)^m & x_i>a\\ 0 & -a<x_i<a\\ k\left(-x_i-a\right)^m & x_i<-a\end{cases}$	30	[-50,50]	0
$F_{17}(x)=0.1\{\sin^2\left(3\pi x_i\right)+\sum_{i=1}^{n}\left(x_i-1\right)^2\left[1+\sin^2\left(3\pi x_i+1\right)\right]$ $+\left(x_n-1\right)^2\left[1+\sin^2\left(2\pi x_n\right)\right]\}+\sum_{i=1}^{n} u\left(x_i,5,100,4\right)$	30	[-50,50]	0
$F_{18}(x)=\sum_{i=1}^{d}\lvert x_i\sin\left(x_i\right)+0.1x_i\rvert$	30	[-10,10]	0
$F_{19}(x)=\sum_{i=2}^{n}\left[\left(x_i-1\right)^2+\left(x_1-x_i^2\right)^2\right]$	30	[0,10]	0
$F_{20}(x)=\sin^2\left(\pi w_1\right)+\sum_{i=1}^{n-1}\left(w_i-1\right)^2\left[1+10\sin^2\left(\pi w_i+1\right)\right]$ $+\left(w_n-1\right)^2\left[1+\sin^2\left(2\pi w_n\right)\right]$ $w_i=\frac{x_i-1}{4}$	30	[-10,10]	0
$F_{21}(x)=\left[\frac{1}{n-1}\sum_{i=1}^{n-1}\left(\sqrt{s_i}\left(\sin\left(50.0s_i^{0.02}\right)+1\right)\right)\right]^2$ $s_i=\sqrt{x_i^2+x^2_{i+1}}$	30	[-10,10]	0
$F_{22}(x)=-\cos\left(x_1\right)\cos\left(x_2\right)e^{\left[-\left(x_1-\pi\right)^2-\left(x_2-\pi\right)^2\right]}$	30	[-100,100]	0
$F_{23}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right)^{-1}$	2	[-65,65]	1
$F_{24}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_i x_2\right)}{b_i^2+b_i x_3+x_4}\right]^2$	4	[-5,5]	0.00030
$F_{25}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	[-5,5]	-0.398
$F_{26}(x)=-\sum_{i=1}^{5}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	[0,10]	-10.1532
$F_{27}(x)=-\sum_{i=1}^{7}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	[0,10]	-10.4028
$F_{28}(x)=-\sum_{i=1}^{10}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	[0,10]	-10.5363

local optima. The test results of multimodal function can well show the function optimization ability of the algorithm. $F_{23} \sim F_{28}$ are combination functions [33], which are formed by rotation, displacement and offset of various benchmark functions. Such functions have low dimensions and a small number of local optimal values [36]. Both unimodal and multimodal functions are tested in 30 dimensions, and composite functions are tested in their own dimensions.

B. SIMULATION EXPERIMENTS AND PERFORMANCE ANALYSIS WITH OTHER SWARM INTELLIGENT ALGORITHMS

1) TEST FUNCTION RESULTS AND ANALYSIS

In order to verify the performance of the improved algorithm, this paper compares the improved hybrid grasshopper algorithm (HGOA) with the original grasshopper algorithm (GOA), particle swarm optimization (PSO) algorithm, sine cosine algorithm (SCA), moth flame optimization (MFO) algorithm, salp swarm algorithm (SSA) and bat algorithm (BA). The parameter settings of each algorithm are shown in Table 2 . The simulation results of 28 test functions are shown in Fig.4, and the performance comparison simulation results under 10 runs are shown in Table 4. The optimal value, mean value and variance of the simulation results for 10 times are given in Table 3 , in which bold data is the best result.

From the above experimental results, we can see that the function optimization ability of HGOA is better than other optimization algorithms. For the test results of the unimodal functions $F_1 \sim F_{12}$, HGOA algorithm gets the best results. In most single-mode test functions, the performance of the improved algorithm has been significantly improved, showing the high performance of the algorithm. The single-modal test function has only one global optimum, and the results show that HGOA has a high exploitation ability. It can be seen from the data table of test results, HGOA greatly improves the accuracy of optimization results. In function F_{10}, HGOA improves the accuracy by 58% compared with GOA, and in function F_8, it improves the accuracy by 37%. However, according to the optimal convergence curve, it can be seen that although HGOA effectively improves the convergence accuracy of the algorithm, for some functions, its convergence speed needs to be further improved. For the multimodal functions $F_{13} \sim F_{22}$, the optimization results of HGOA are all near the optimal value. For the function F_{13}, the optimal value is basically found. In other functions, the optimization accuracy is also improved compared with other algorithms in different degrees. Because there are a large number of local solutions in the multi-modal test function, these results quantitatively show the effectiveness of the algorithm to avoid local solutions in the optimization process. For the test function with fixed dimension, HGOA shows a good optimization effect. Except for the test function F_{25}, the optimal value of the function can be found, and the convergence speed of the algorithm can be improved to a large extent. At the same

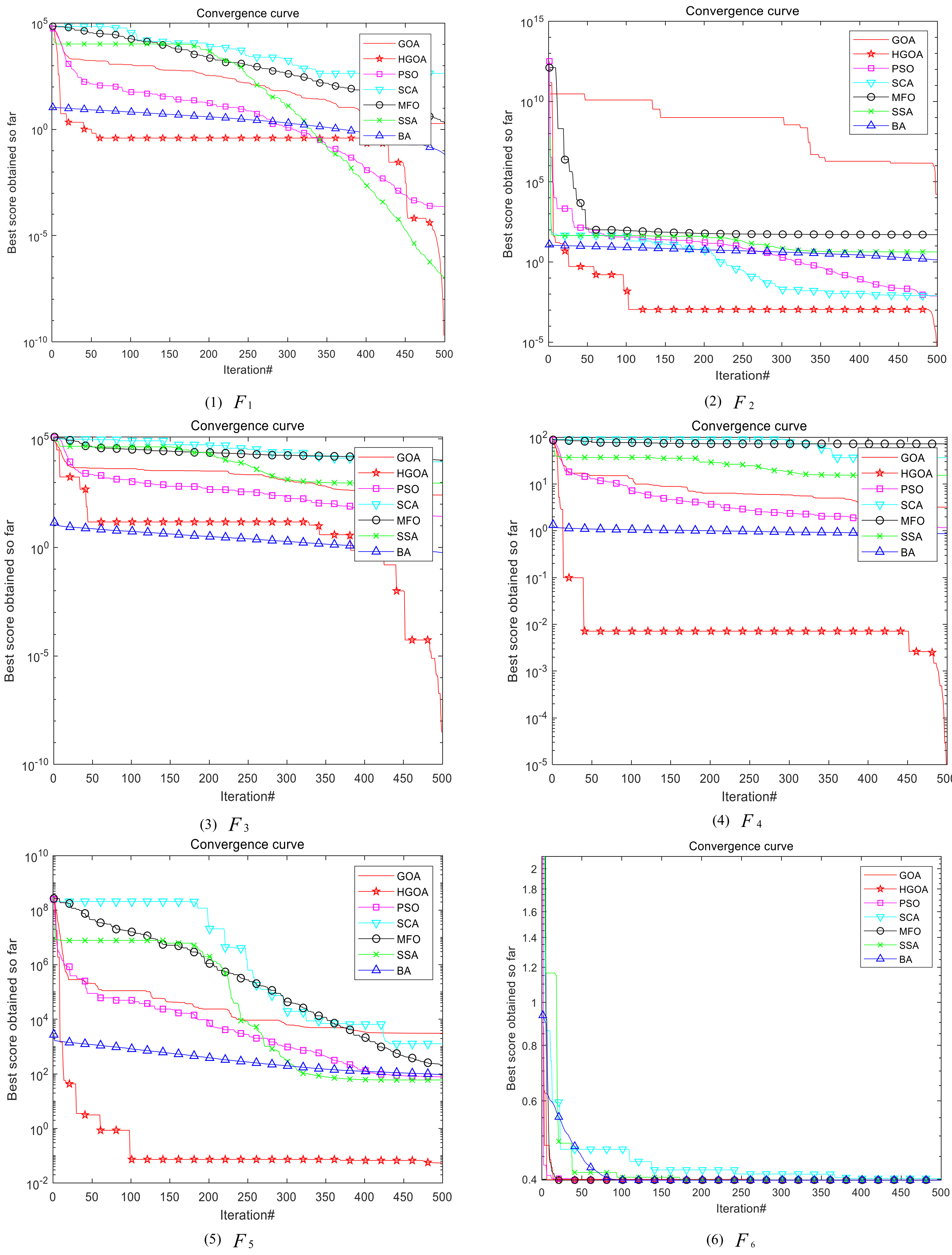
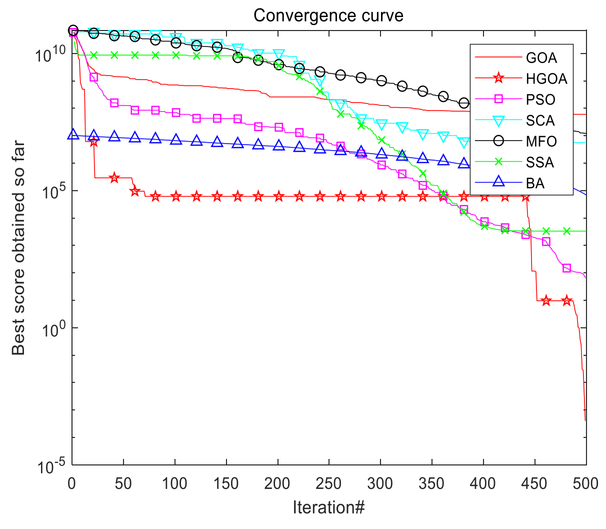
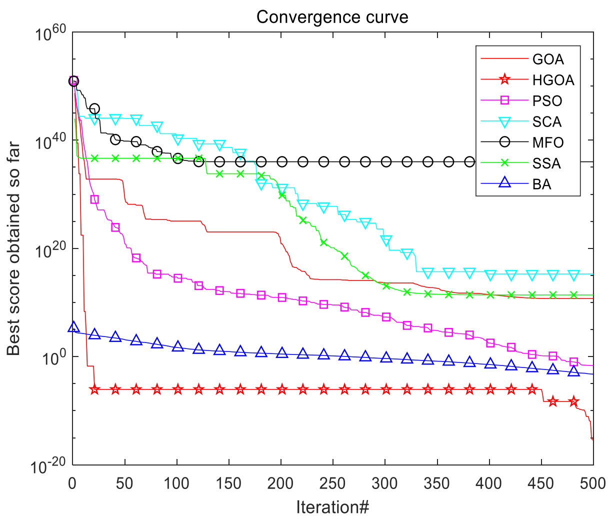


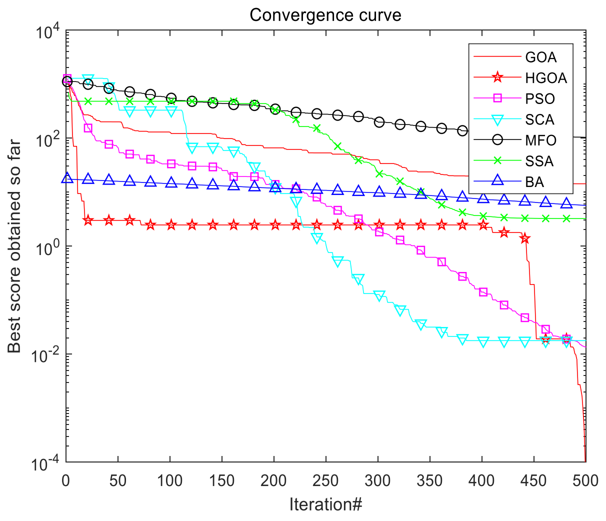
FIGURE 4. The convergence curves of test function.



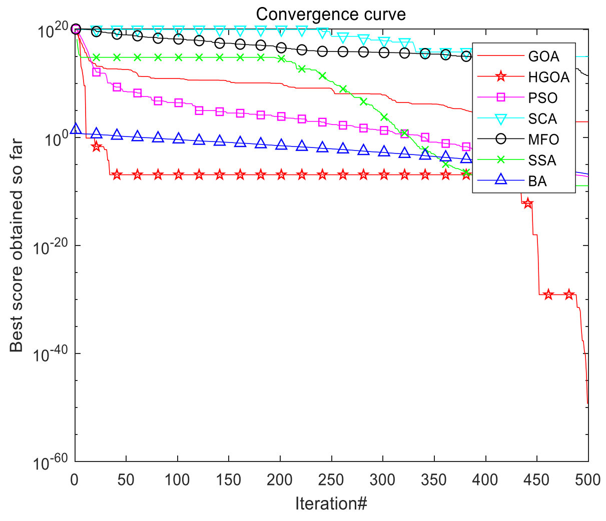
(7) F_7



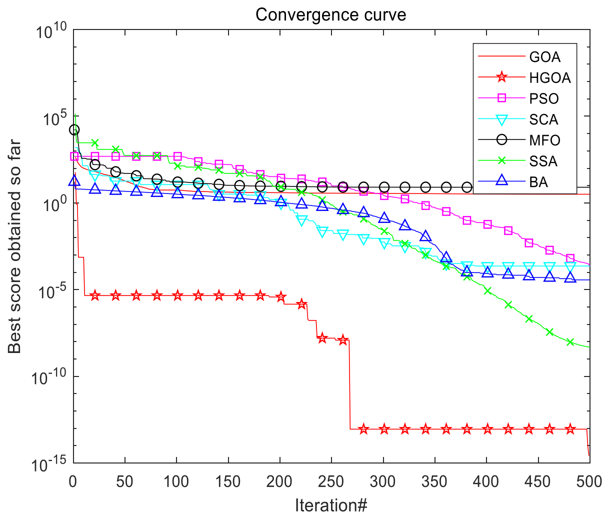
(8) F_8



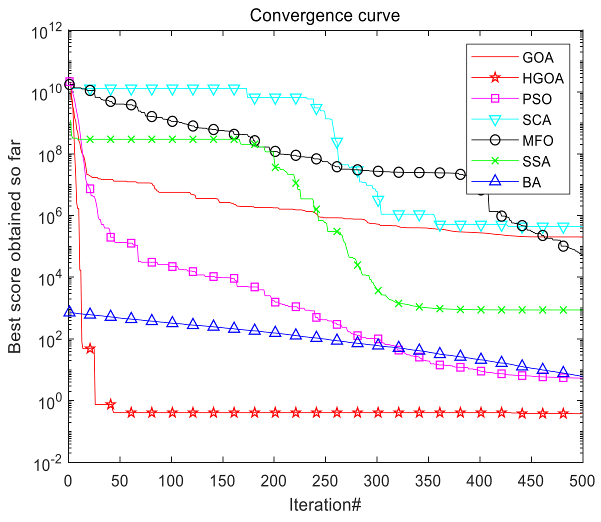
(9) F_9



(10) F_{10}

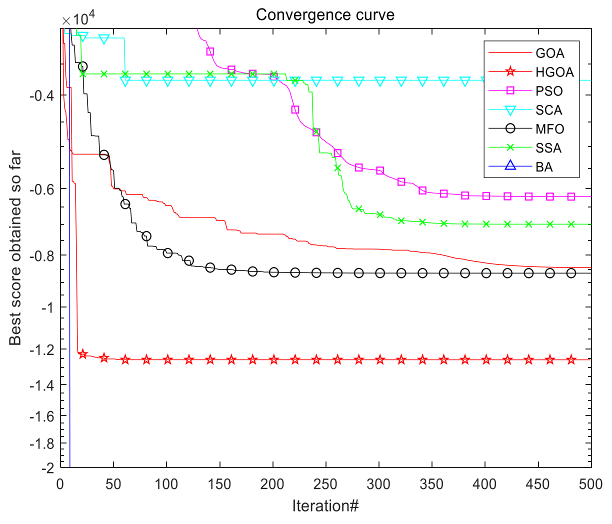


(11) F_{11}

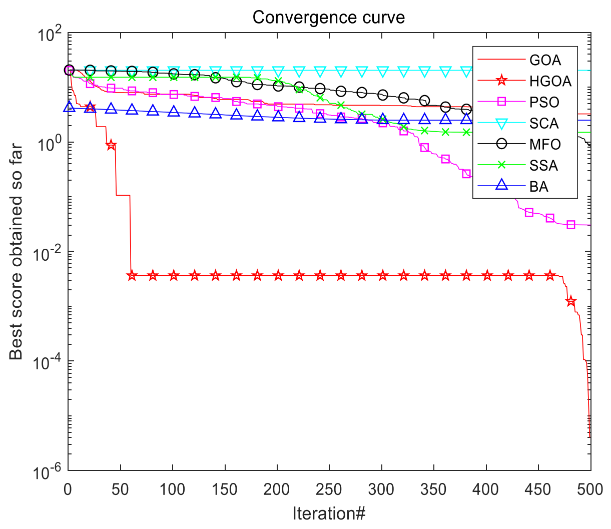


(12) F_{12}

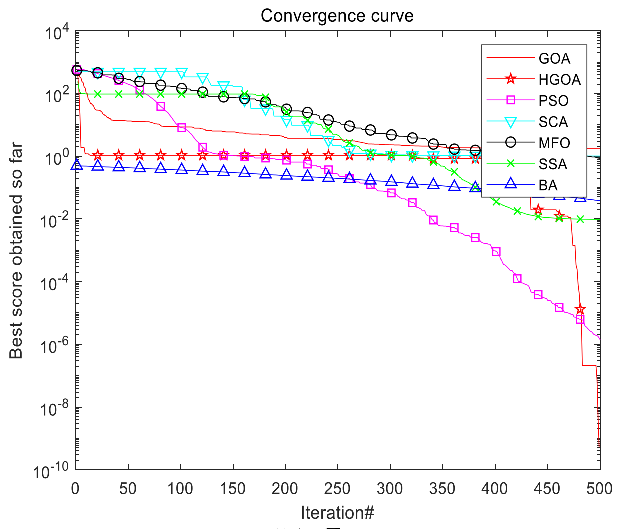
FIGURE 4. ***(Continued.)*** **The convergence curves of test function.**



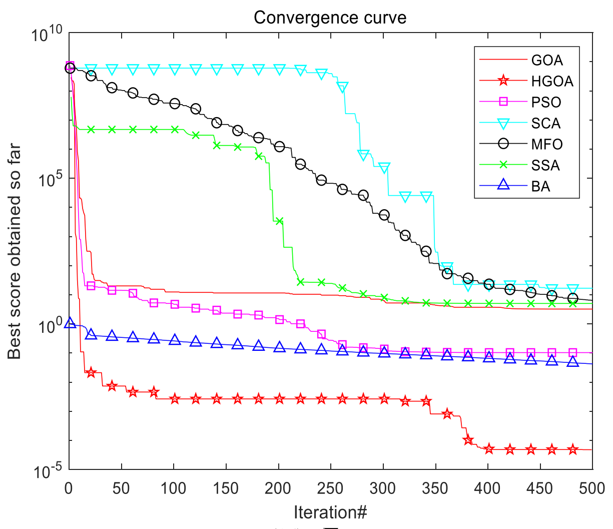
(13) F_{13}



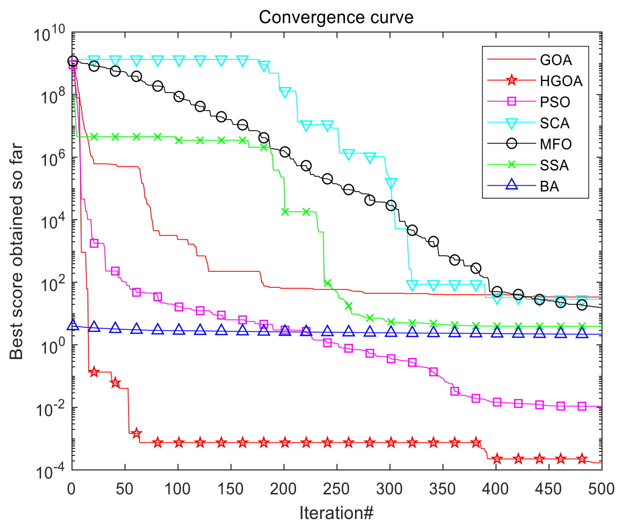
(14) F_{14}



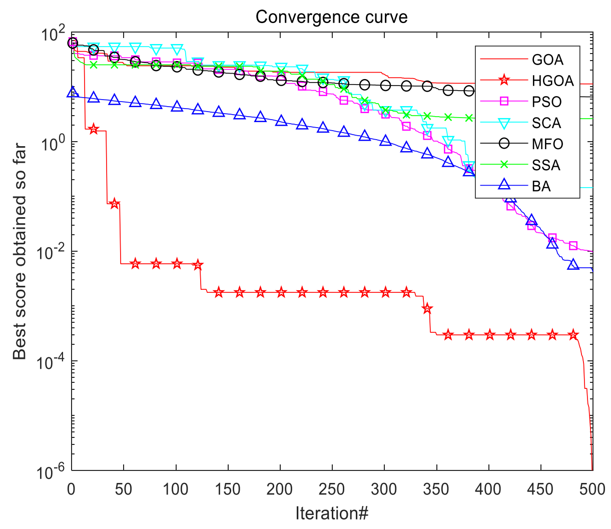
(15) F_{15}



(16) F_{16}

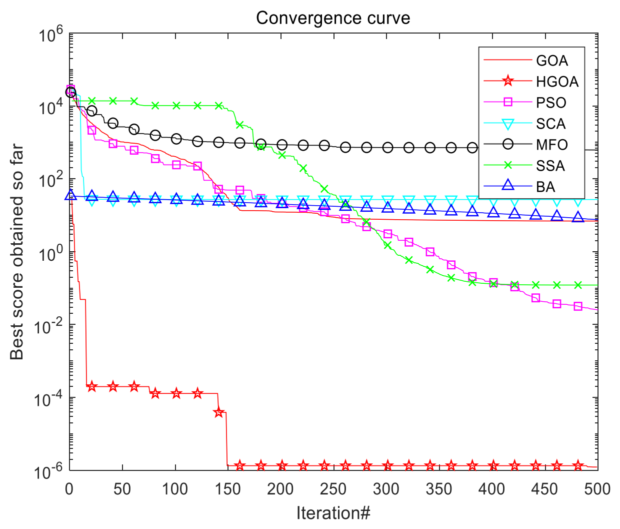


(17) F_{17}

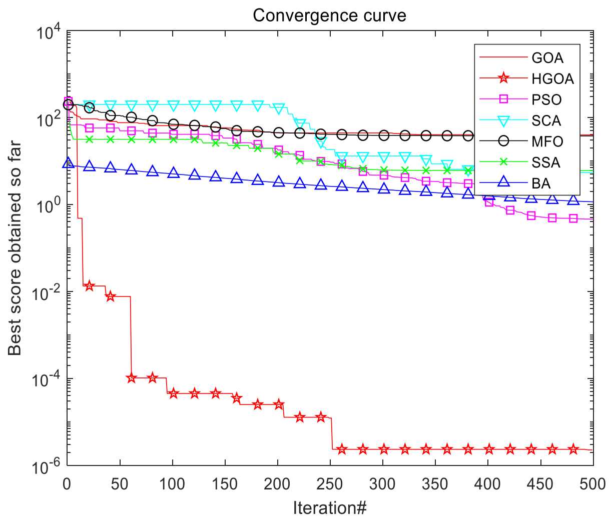


(18) F_{18}

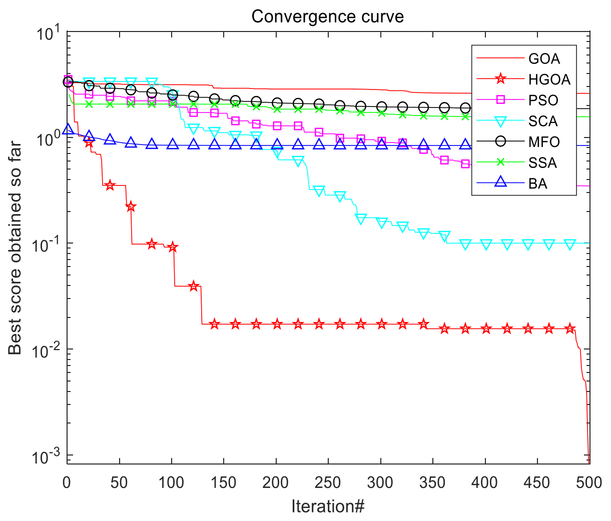
FIGURE 4. ***(Continued.)*** **The convergence curves of test function.**



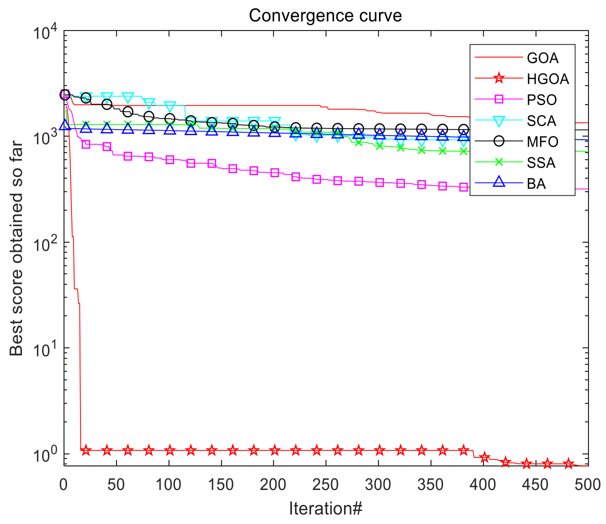
(19) F_{19}



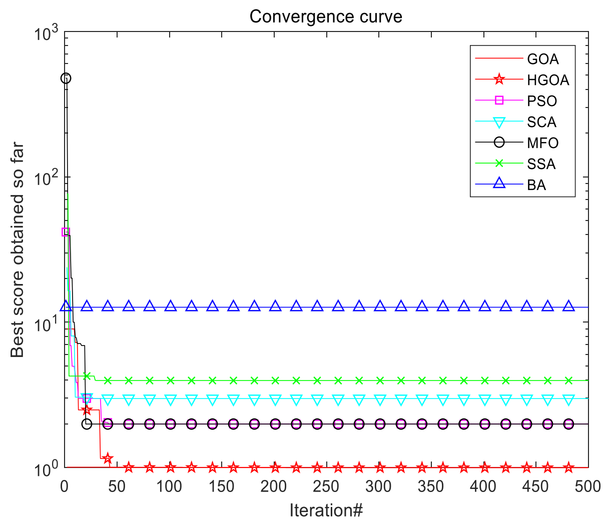
(20) F_{20}



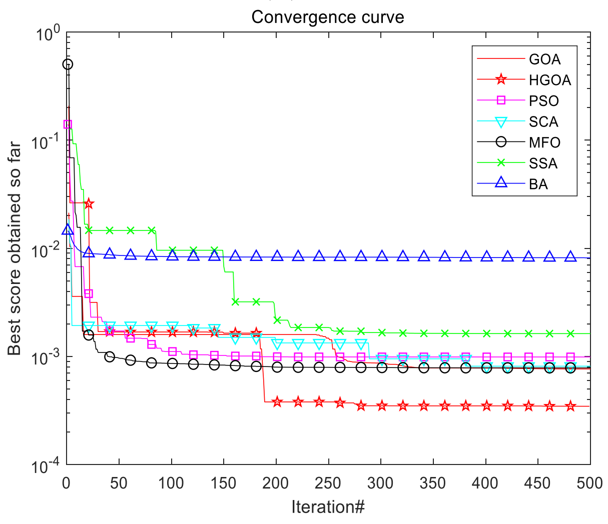
(21) F_{21}



(22) F_{22}

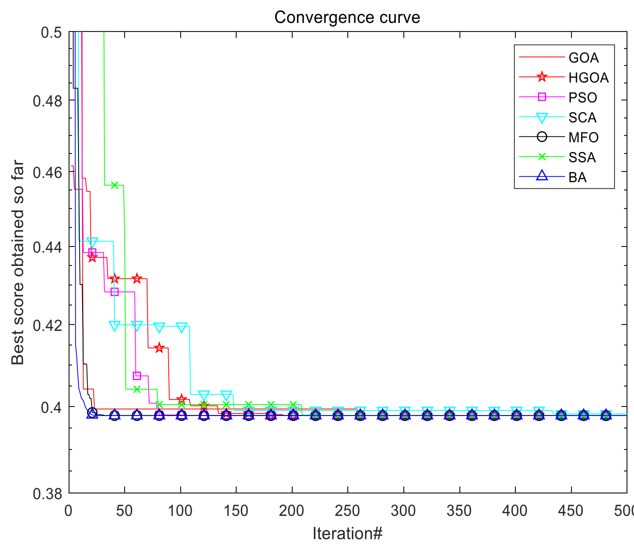


(23) F_{23}

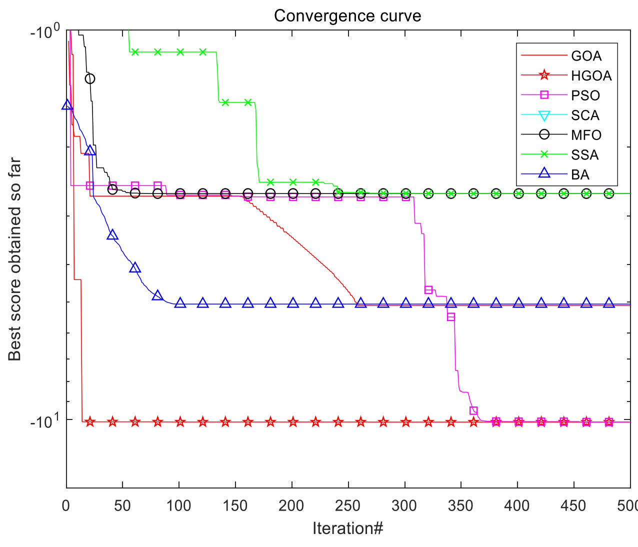


(24) F_{24}

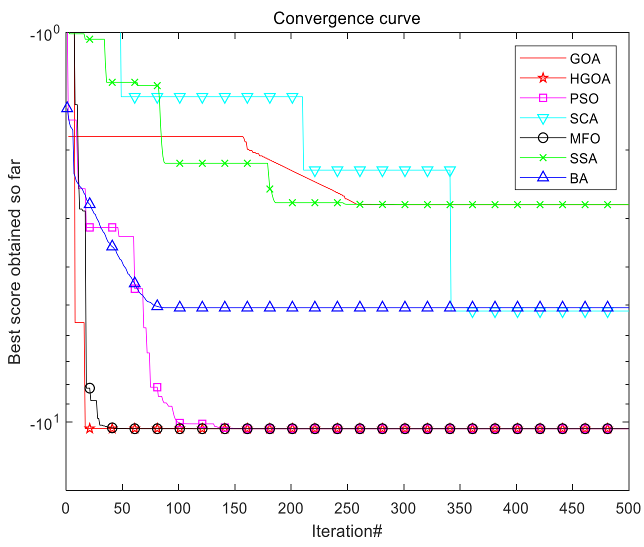
FIGURE 4. ***(Continued.)*** **The convergence curves of test function.**



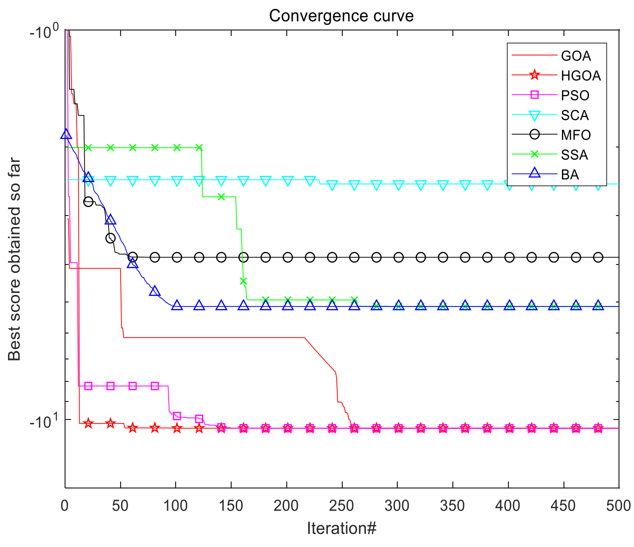
(25) F_{25}



(26) F_{26}



(27) F_{27}



(28) F_{28}

FIGURE 4. ***(Continued.)*** **The convergence curves of test function.**

TABLE 2. **Parameter Settings for each algorithm.**

Algorithm	Main parameters Settings
GOA	Particle number $n=30$; $c_{\max}=1$, $c_{\min}=0.00004$
HGOA	Particle number $n=30$; $c_{\max}=1$, $c_{\min}=0.00004$; $G0=100$, $a_0=20$, $R_{\max}=15$, $R_{\min}=0.00004$, $p_r=0.5$
PSO	Particle number $n=30$; $c_1=2$, $c_2=2$; $w_{Max}=0.9$, $w_{Min}=0.1$
SCA	Particle number $n=30$;
MFO	Particle number $n=30$
SSA	Particle number $n=30$
BA	Particle number $n=30$

time, it can be seen from the difference column of the test data that HGOA is very stable in the 10 times of optimization process. The fixed dimension function is more challenging than the multimodal test function, which requires a proper balance between expansion and development. Therefore, it can be said that HGOA can well balance exploration and

TABLE 3. Performance comparison of test functions.

Function	Algorithms	Best	Ave	Std
F_1	GOA	44.9775	3.6601E+02	3.9169E+02
	HGOA	4.3310E-10	**6.5062E-10**	1.1164E-10
	PSO	1.1434E-05	1.5304E-04	1.2353E-04
	SCA	0.0334	6.9835	11.0410
	MFO	0.5617	3.0124E+03	4.5752E+03
	SSA	4.1635E-08	1.3420E-07	9.2057E-08
	BA	2.2642E-05	0.2157	0.6233
F_2	GOA	4.2691	72.4397	59.8905
	HGOA	8.0858E-06	**9.3861E-06**	1.0799E-06
	PSO	8.1035E-03	4.1658E-02	3.3409E-02
	SCA	8.2068E-04	1.2217E-02	1.0720E-02
	MFO	1.3775	24.2605	17.2174
	SSA	0.3066	1.5303	1.5086
	BA	1.5856	2.4931	0.6444
F_3	GOA	2.6216E+02	9.7724E+02	6.9810E+02
	HGOA	1.9869E-10	**1.5404E-09**	9.3521E-10
	PSO	27.2366	81.8572	36.2547
	SCA	2.4586E+03	1.0679E+04	7.6250E+03
	HFO	8.7188E+03	1.8903E+04	1.0541E+04
	SSA	7.2685E+02	1.6487E+03	1.0004E+03
	BA	0.2334	3.6608	4.0681
F_4	GOA	2.3208	5.7757	2.8539
	HGOA	2.9394E-06	**7.0056E-06**	2.4155E-06
	PSO	1.0213	1.1887	0.1490
	SCA	8.7561	34.0384	12.9850
	MFO	57.6726	70.0633	5.6430
	SSA	3.6143	11.7734	4.4597
	BA	0.2827	0.5460	0.1871
F_5	GOA	98.2408	9.8179E+02	1.3317E+03
	HGOA	1.9920E-03	**2.3461E-02**	2.1591E-02
	PSO	29.6011	66.8661	23.6728
	SCA	1.2731E+03	2.6522E+04	2.3091E+04
	MFO	2.0963E+02	1.0397E+04	2.6661E+04
	SSA	29.0483	1.6995E+02	1.7059E+02
	BA	27.8161	64.5438	50.3435
F_6	GOA	0.3303	0.5115	0.1216
	HGOA	5.5559E-07	**3.0389E-05**	2.3142E-05
	PSO	0.0605	0.1657	7.4477E-02
	SCA	2.0865E-02	0.1202	0.1006
	MFO	0.1226	7.4656	8.2143
	SSA	0.0732	0.1487	7.8556E-02
	BA	9.7673	31.6906	16.2933
F_7	GOA	1.7428E+05	5.8891E+07	7.6357E+07
	HGOA	1.1117E-04	**4.2509E-04**	1.8678E-04
	PSO	9.2277	1.2100E+02	88.3537
	SCA	2.0253E+04	9.1122E+06	1.1946E+07
	MFO	1.6218E+05	2.0027E+09	3.9998E+09
	SSA	0.8427	1.2831E+03	1.6218E+03
	BA	17.4572	2.2063E+05	6.3358E+05

TABLE 3. *(Continued.)* **Performance comparison of test functions.**

F_8	GOA	9.7890E+04	3.0384E+22	9.1151E+22
	HGOA	1.1894E-16	**5.6666E-15**	4.5343E-15
	PSO	9.3227E-05	4.3238E-02	9.2559E-02
	SCA	98.5254	5.5128E+17	1.6532E+18
	MFO	1.0101E+28	1.0001E+43	3.0000E+43
	SSA	2.0361E+08	5.7533E+17	1.1661E+18
	BA	7.6487E-07	1.5342E-04	3.0675E-04
F_9	GOA	9.0140	21.4275	9.8625
	HGOA	4.3075E-05	**8.0291E-05**	2.7702E-05
	PSO	2.8335E-03	1.7310E-02	1.7716E-02
	SCA	3.7097E-03	0.4201	0.9460
	MFO	0.7157	91.5758	69.3906
	SSA	2.9331	16.5740	8.6873
	BA	1.4315	3.0777	1.2034
F_{10}	GOA	9.8450E-03	7.0908E+07	1.1986E+08
	HGOA	2.7025E-53	**7.2542E-51**	1.2936E-50
	PSO	7.3315E-09	5.8141E-06	9.0929E-06
	SCA	4.2860E+04	4.5152E+14	8.1608E+14
	MFO	2.4277E+09	1.1215E+12	1.5602E+12
	SSA	1.1401E-09	6.5487E+03	1.5744E+04
	BA	4.1754E-25	5.3261E-06	1.5924E-05
F_{11}	GOA	2.2642	17.4256	18.9014
	HGOA	2.2364E-15	**4.3966E-13**	2.7795E-13
	PSO	4.7596E-05	5.5339E-04	6.9750E-04
	SCA	1.0537E-04	4.6765E-03	7.2856E-03
	MFO	5.0002	16.7214	10.9969
	SSA	2.8185E-10	2.4727E-09	3.0402E-09
	BA	3.6462E-05	7.9141E-02	0.1250
F_{12}	GOA	8.4815E+02	5.6260E+04	6.1685E+04
	HGOA	0.3050	**0.7747**	0.2468
	PSO	0.8466	2.5304	1.5133
	SCA	2.8222E+02	1.6827E+06	2.5229E+06
	MFO	4.3945E+03	5.5868E+08	1.6760E+09
	SSA	62.5414	1.0979E+03	1.2740E+03
	BA	1.1164	6.0532	4.1903
F_{13}	GOA	-8711.8189	-7296.3449	6.8091E+02
	HGOA	-12569.48017	**-12569.4225**	0.0910
	PSO	-6527.7196	-5302.6887	8.7555E+02
	SCA	-3855.8904	-3697.6173	90.84612166
	MFO	-9637.3793	-8605.2191	9.4624E+02
	SSA	-8359.9017	-7152.0633	5.9087E+02
	BA	-9.2978E+269	-1.0378E+269	Inf
F_{14}	GOA	2.6569	15.0001	2.5681
	HGOA	2.4883E-06	**4.8242E-06**	1.3172E-06
	PSO	2.6301E-03	0.1594	5.5263E-03
	SCA	0.2542	15.6714	8.6095
	MFO	0.8323	14.7045	1.3302
	SSA	1.3404	2.4012	0.7881
	BA	1.8998	2.1697	0.1819

TABLE 3. ***(Continued.)*** **Performance comparison of test functions.**

	GOA	0.3350	3.3628	2.9376
	HGOA	4.4986E-10	**6.7444E-10**	2.4514E-10
	PSO	1.3963E-06	5.9263E-03	5.0758E-03
F_{15}	SCA	9.3831E-02	1.2176	0.9080
	MFO	0.7634	0.9266	0.1277
	SSA	3.7470E-03	1.2862E-02	7.3918E-03
	BA	6.1783E-06	1.2780E-02	1.2903E-02
	GOA	3.2364	5.4450	1.6698
	HGOA	7.7119E-09	**1.6616E-05**	3.0514E-05
	PSO	3.0180E-07	3.1106E-02	4.7506E-02
F_{16}	SCA	2.1403	2.0820E+03	6.2074E+03
	MFO	2.0498	6.9777	5.8609
	SSA	2.0917	5.9218	2.4441
	BA	0.0423	0.1291	7.5873E-02
	GOA	0.4500	3.5298	3.1923
	HGOA	7.3136E-08	**9.4120E-05**	1.4902E-04
	PSO	3.1446E-05	6.5784E-03	7.0389E-03
F_{17}	SCA	8.6177	1.7641E+05	3.0730E+05
	MFO	6.8560	99.2959	1.9769E+02
	SSA	0.6349	15.6610	11.1654
	BA	0.7710	1.8994	0.7289
	GOA	6.1118	11.4837	4.3237
	HGOA	4.0125E-07	**8.8038E-07**	2.4074E-07
	PSO	2.2831E-03	3.4222E-02	6.8672E-02
F_{18}	SCA	1.0770E-02	1.2902	2.0452
	MFO	3.2785E-02	4.2063	3.2684
	SSA	1.5101	2.6123	0.7412
	BA	4.1720E-03	0.1047	0.1962
	GOA	4.9797	92.5859	1.4483E+02
	HGOA	1.6366E-13	**7.9376E-12**	1.0701E-11
	PSO	3.4765E-02	0.1655	0.2702
F_{19}	SCA	25.4747603	26.4568	0.6365
	MFO	7.6702	12.9831	2.7605
	SSA	6.6085E-06	1.8165	4.8746
	BA	5.5167	9.7739	3.2181
	GOA	28.1076	50.0637	15.1512
	HGOA	9.9992E-10	**4.6011E-07**	6.5357E-07
	PSO	1.8596E-04	0.4823	0.5031
F_{20}	SCA	2.1412	4.1274	2.8273
	MFO	19.2212	41.9444	24.7528
	SSA	3.5941	8.0976	3.3762
	BA	1.0965	1.4220	0.2172
	GOA	2.3137	2.5767	0.1417
	HGOA	7.5472E-04	**8.0254E-04**	2.4498E-05
	PSO	0.2630	0.5582	0.2677
F_{21}	SCA	5.4818E-03	7.4004E-02	5.1362E-02
	MFO	1.5615	2.0630	0.2490
	SSA	1.1013	1.4543	0.1787
	BA	0.7246	0.8493	0.1004
	GOA	8.4187E+02	1.2047E+03	1.8721E+02
F_{22}	HGOA	0.3394	**0.6106**	0.1884
	PSO	1.8995E+02	3.6069E+02	1.0304E+02

TABLE 3. ***(Continued.)*** **Performance comparison of test functions.**

	SCA	7.8573E+02	9.0950E+02	63.5996
	MFO	5.9119E+02	9.1522E+02	1.8479E+02
	SSA	3.9865E+02	6.3873E+02	1.5661E+02
	BA	7.1141E+02	9.3240E+02	87.6770
	GOA	0.9980	**0.9980**	4.6444E-16
	HGOA	0.9980	**0.9980**	5.6352E-10
	PSO	0.9980	3.8544	2.7608
F_{23}	SCA	0.9980	2.7738	2.8209
	MFO	0.9980	1.7889	1.5160
	SSA	0.9980	1.5928	1.0104
	BA	2.9821	11.7017	2.9065
	GOA	0.0003	0.0065	8.4686E-03
	HGOA	0.0003	**0.0003**	4.1792E-06
	PSO	0.0007	0.0008	9.1242E-05
F_{24}	SCA	0.0008	0.0011	3.2184E-04
	MFO	0.0007	0.0049	7.7537E-03
	SSA	0.0003	0.0048	7.7714E-03
	BA	0.0003	0.0040	4.2918E-03
	GOA	0.3979	0.3979	1.0771E-13
	HGOA	0.3979	0.3979	1.2708E-05
	PSO	0.3979	0.3979	5.5511E-17
F_{25}	SCA	0.3980	0.3992	1.6948E-03
	MFO	0.3979	0.3979	5.5511E-17
	SSA	0.3979	0.3979	1.4388E-14
	BA	0.3979	0.3979	4.6497E-10
	GOA	-5.1008	-3.6291	1.2018
	HGOA	-10.1532	**-10.1532**	2.8161E-06
	PSO	-10.1532	-8.6537	2.9986
F_{26}	SCA	-4.8564	-2.5477	1.8364
	MFO	-10.1532	-4.8978	3.4405
	SSA	-10.1532	-5.3918	3.2517
	BA	-10.1532	-5.5650	1.5294
	GOA	-5.1288	-3.6609	1.0190
	HGOA	-10.4029	**-10.4026**	8.0210E-04
	PSO	-10.4029	-9.8714	1.5946
F_{27}	SCA	-6.7212	-3.6808	1.7641
	MFO	-10.4029	-7.0083	3.4478
	SSA	-10.4029	-8.8741	3.0576
	BA	-5.0877	-5.0877	3.7140E-07
	GOA	-10.5364	-5.3569	3.4832
	HGOA	-10.5364	**-10.5363**	1.3406E-04
	PSO	-10.5364	-9.8663	2.0103
F_{28}	SCA	-4.9918	-2.7142	1.4972
	MFO	-10.5364	-7.7596	3.4111
	SSA	-10.5364	-9.2291	2.6629
	BA	-5.1285	-5.1285	0.0000

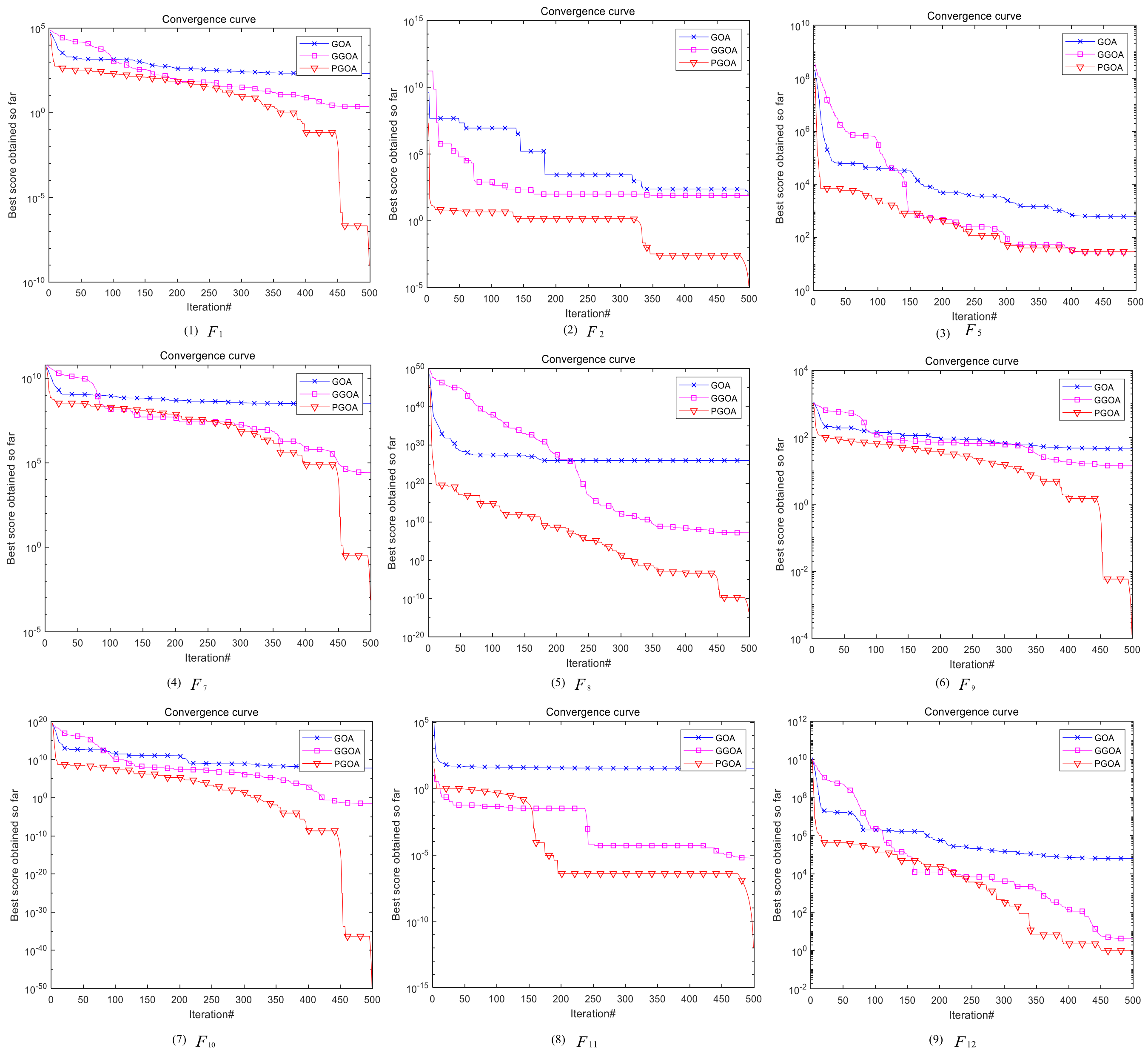


FIGURE 5. The convergence curves of test functions.

development and solve such challenging problems. According to all the test results, HGOA has been improved in most cases in terms of convergence speed, and has shown good optimization effect in terms of convergence accuracy. Especially, in the fixed dimension test functions, the algorithm's robustness is improved while ensuring the convergence speed and optimization accuracy.

2) NONPARAMETRIC TEST ANALYSIS

In order to verify the stability of the algorithm, the average value of each algorithm has been carried out the nonparametric tests. In this work, Wilcoxon rank sum test is used as nonparametric statistical test to determine the importance of the results. Table 4 depicts the 5% p value obtained from this test. It can be seen from Table 4 that the p-value of unimodal functions and multimodal functions are far less than 0.05, highlighting the significant advantages of HGOA compared with other algorithms based on p-value (less than 0.05).

C. SIMULATION EXPERIMENTS AND PERFORMANCE ANALYSIS OF IMPROVED ALGORITHM

In order to more accurately express the influence of gravity search operator and pigeon search operator on the algorithm, this section carries out simulation tests on the basic grasshopper algorithm (GOA), the grasshopper algorithm (GGOA) with gravity search operator and the grasshopper algorithm

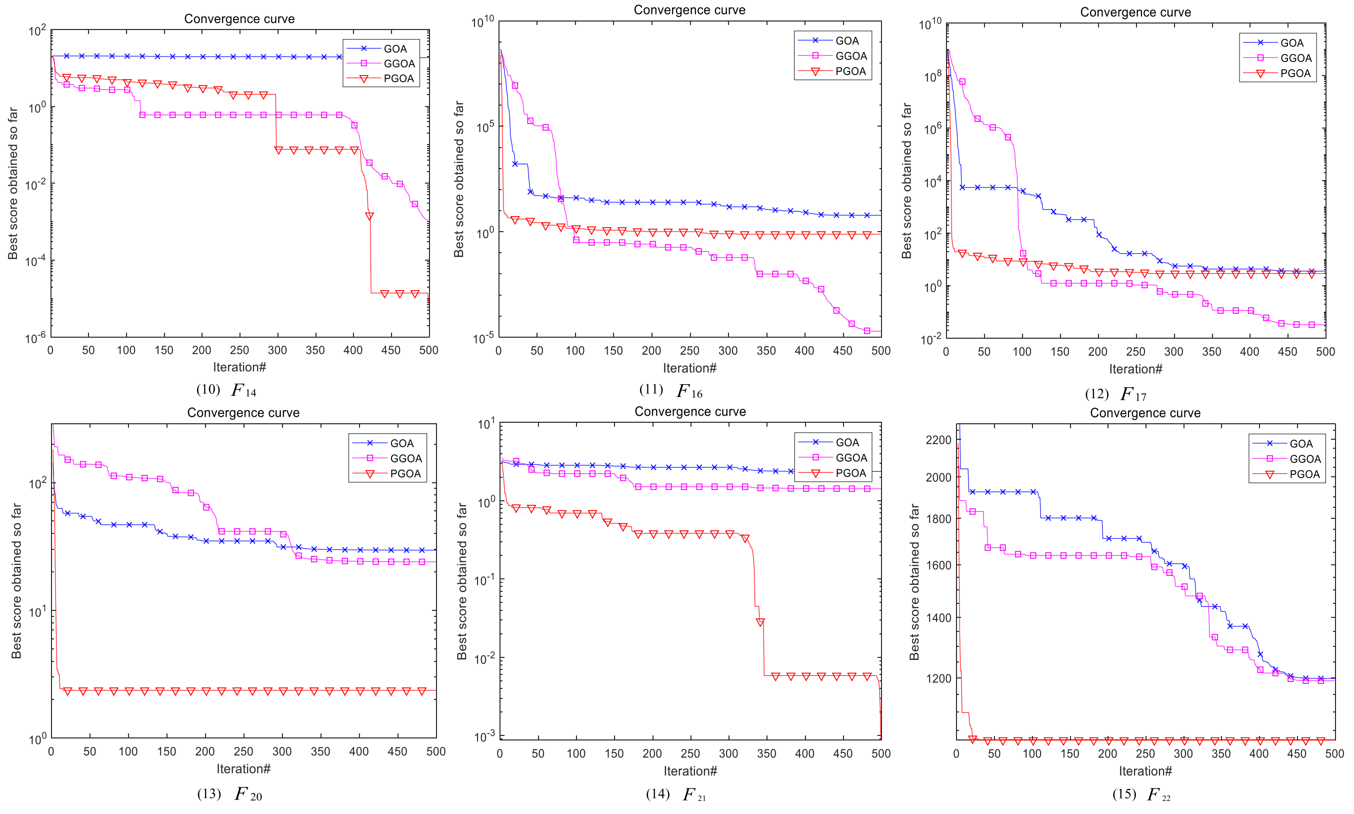


FIGURE 5. *(Continued.)* **The convergence curves of test functions.**

TABLE 4. **Nonparametric test results.**

Algorithm comparison	Unimodal functions	Multimodal functions	Combination function
HGOA vs GOA	4.6949E-05	0.0022	0.4567
HGOA vs PSO	0.0017	0.0140	0.4177
HGOA vs SCA	1.5580E-04	0.0028	0.3939
HGOA vs MFO	3.6585E-05	0.0022	0.4177
HGOA vs SSA	0.0011	0.0028	0.4177
HGOA vs BA	7.3148E-04	0.0058	0.4177

(PGOA) with pigeon search operator. In the simulation experiment, it was found that both GGOA and PGOA had different degrees of improvement in convergence speed and convergence accuracy, so the curves and data with relatively obvious effect were given. Partial convergence curves were shown in Fig.5, and the convergence results were shown in Table 5. It can be seen from the convergence curve that for unimodal functions, the influence of GGOA and PGOA on the convergence speed of the algorithm is not obvious. For multimodal functions, the convergence speeds of GGOA and PGOA have been improved to varying degrees, especially the effect of PGOA is the most obvious. It can be seen from the table 5 that PGOA has greatly improved the convergence accuracy of the algorithm, especially by 40 orders of magnitude in both functions F_8 and F_9. At the same time, combined with Fig.4 and Table 3, it can be found that HGOA has further improved convergence speed and accuracy compared with GGOA and PGOA.

V. SIMULATION AND RESULT ANALYSIS OF PRESSURE VESSEL DESIGN PROBLEM

In recent years, using stochastic optimization technology to solve structural design problems is a research hotspot in the field of structural design [37], [38]. In order to further verify

TABLE 5. Performance comparison of the test functions.

Function	Performance	GOA	GGOA	PGOA
F_1	Best	1.2100	0.2861	9.3354E-10
	Ave	76.4808	2.6472	9.3604E-10
	Std	100.5757	2.6834	1.5580E-12
F_2	Best	10.4705	71.5561	1.1480E-05
	Ave	63.7554	7.6175E+04	1.2318E-05
	Std	45.4566	2.1908E+05	5.1212E-07
F_4	Best	3.0281	1.1925	9.1611E-06
	Ave	7.6567	17.0914	9.8736E-06
	Std	3.5959	21.9562	3.5183E-07
F_5	Best	88.2889	28.5811	28.9097
	Ave	5.8744E+03	29.1271	28.9503
	Std	1.4952E+04	0.7648	0.0158
F_7	Best	2.9080E+05	4.7724E+04	7.4530E-04
	Ave	8.9874E+07	1.3090E+06	8.0112E-04
	Std	1.1451E+08	1.5174E+06	2.5914E-05
F_8	Best	7.7710E+08	3.1886E-05	8.0906E-17
	Ave	1.0000E+27	2.6642E+05	8.9695E-14
	Std	3.0000E+27	7.9926E+05	1.2417E-13
F_9	Best	18.8280	1.0351	1.2275E-04
	Ave	36.2857	2.9281	1.2686E-04
	Std	11.6065	1.3840	2.0795E-06
F_{10}	Best	0.0002	1.6861E-10	1.2272E-50
	Ave	4.2570E+07	3.1368E-06	1.9756E-50
	Std	1.2770E+08	5.8234E-06	5.1459E-51
F_{11}	Best	3.4463	4.0773E-08	9.5586E-13
	Ave	15.3420	5.2007	1.0655E-12
	Std	15.1522	15.5998	8.7809E-14
F_{12}	Best	14.3943	2.2579	0.9808
	Ave	6.5760E+04	6.4878E+02	0.9903
	Std	1.5246E+05	1.3777E+03	4.7412E-03
F_{14}	Best	3.4838	3.7573E-04	7.1484E-06
	Ave	8.3698	5.6082E-03	7.1659E-06
	Std	5.3112	7.2703E-03	5.8796E-09
F_{16}	Best	1.6443	3.9056E-05	0.5129
	Ave	4.3232	1.5959	0.6856
	Std	2.1383	3.1714	0.1169
F_{17}	Best	0.2421	0.0008	2.9665
	Ave	9.0559	0.0145	2.9909
	Std	12.1315	0.0148	0.0083
F_{20}	Best	35.5773	1.0000	2.1757
	Ave	45.7531	28.0792	2.6303
	Std	8.2278	18.7409	0.2556
F_{21}	Best	2.2172	1.8626	8.6552E-04
	Ave	2.5185	2.3507	8.7408E-04
	Std	0.1670	0.2904	6.0343E-06
F_{22}	Best	1.1912E+03	9.4810E+02	9.9758E+02
	Ave	1.4044E+03	1.1095E+03	1.0681E+03
	Std	133.2538	1.0890E+02	41.0383

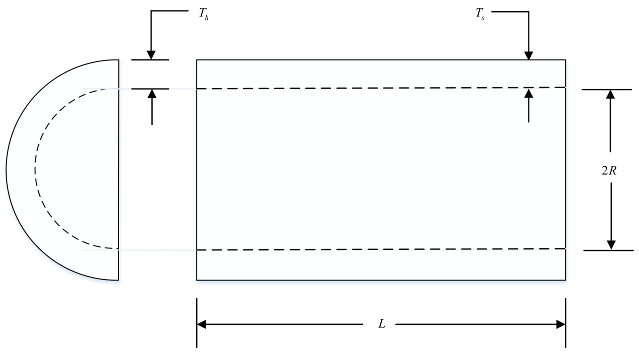


FIGURE 6. Pressure vessels and parameters.

TABLE 6. Parameters comparison results of pressure vessel design problems.

Algorithms	Ts	Th	R	L	Cost
GOA	0.908456	0.476796	47.15804	122.6225	6250.458
HGOA	0.865738	0.427684	45.02398	143.5293	**6047.587**
PSO	1.034572	0.510759	53.73043	71.25397	6478.81
SCA	1.242034	0.696617	64.18282	14.70179	7868.192
MFO	1.256757	0.618359	65.21382	10.04916	7300.218
SSA	1.332253	0.655809	69.19275	53.88863	11442.46
BA	7.287851	52.32686	59.50347	46.22813	412384.9

the effectiveness of the improved algorithm, the design of pressure vessel is optimized in this section. The objective of pressure vessel design problem is to minimize the total cost of material, forming and welding of a cylindrical vessel. Fig. 6 shows the pressure vessel and parameters involved in the design [39], [40]. In Fig. 6, T_s represents the thickness of the shell, T_h represents the thickness of the head, R represents the inner radius, and L represents the length of the cylindrical section without considering the head. The problem has four constraints, which are described as follows.

Consider $\vec{x} = [x_1 x_2 x_3 x_4] = [T_s T_h R L]$,

Minimize $f(\vec{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3$

$$\text{Subject to} \quad \begin{aligned} g_1(\vec{x}) &= -x_1 + 0.0193 x_3 \le 0, \\ g_2(\vec{x}) &= -x_2 + 0.00954 x_3 \le 0, \\ g_3(\vec{x}) &= -\pi x_3^2 x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \le 0, \\ g_4(\vec{x}) &= x_4 - 240 \le 0, \end{aligned}$$

$$\text{Variable range} \quad \begin{aligned} 0 &\le x_1 \le 99, \\ 0 &\le x_2 \le 99, \\ 10 &\le x_3 \le 200, \\ 10 &\le x_4 \le 200, \end{aligned}$$

TABLE 7. Performance comparison results of pressure vessel design problems.

Algorithms	Best	Worst	Ave	Std
GOA	6000.916	7699.293	6869.22	510.5817
HGOA	5904.079	6198.458	**6044.841**	75.54298
PSO	6111.89	6506.57	6312.888	134.4375
SCA	7053.085	154611.8	22612.01	44003.25
MFO	6010.09	7301.196	6721.802	495.237
SSA	5905.54	11586.24	8074.36	1875.607
BA	7956.912	428495.1	195336.1	144260

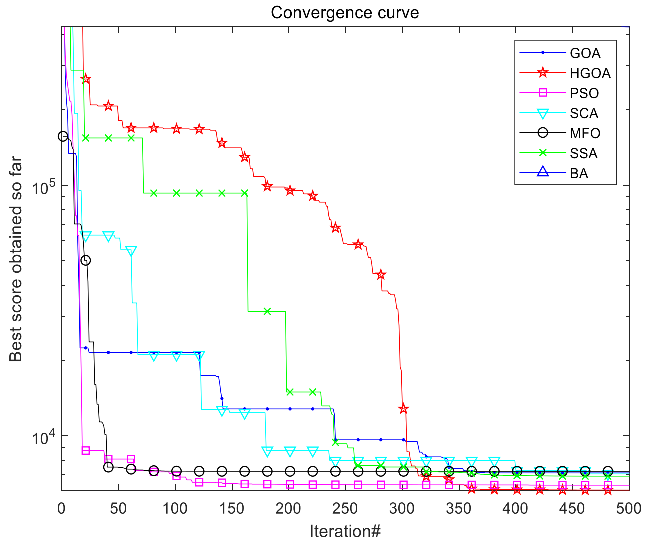


FIGURE 7. Convergence curves of pressure vessel design.

The algorithm used to test this problem are the algorithms for test function in Section 4.2. The convergence curve of each algorithm is shown in Fig. 7. The optimal value and corresponding variable value are shown in Table 6. In order to test the stability of the algorithm, run the problem 10 times, and obtain the optimal value, average value and variance as shown in Table 7. Obviously, the inspection algorithm showed that HGOA showed very competitive results compared with GOA, PSO, SCA, MFO, SSA and BA in this problem, and had better optimization ability. Moreover, according to the variance data in the table, the robustness of the algorithm is obviously better than other algorithms. Because the search space of this problem is unknown, these results provide strong evidence for the applicability of HGOA in solving practical problems. In addition, due to the limitations of case studies, it can be said that HGOA algorithm can also optimize the search space with infeasible areas. The results show that HGOA algorithm can effectively deal with the problem of limited search space.

VI. CONCLUSION

Grasshopper algorithm, as a new kind of biologically-inspired algorithm, can avoid the local optimal stagnation to some extent. However, under certain circumstances, grasshopper algorithm always fails to show the global search ability and solve the problem successfully. This paper firstly introduces the gravity search operator, which greatly improves the global search capability of GOA, and then introduces the pigeon landmark operator, which well improves and balances the exploration and development capability of the algorithm. In order to verify the effectiveness of the improved algorithm, 28 test functions were used for testing. Among them, the unimodal function indicates that HGOA has a higher mining ability; the multimodal function indicates the effectiveness of HGOA in avoiding local solutions in the optimization process; the fixed-dimensional function indicates that HGOA can balance exploration and development well. At the same time, in order to test the influence of the two operators on the improved algorithm, the grasshopper algorithm (GGOA) with only gravity search operator, the grasshopper algorithm (PGOA) with only pigeon landmark search operator and the original grasshopper algorithm (GOA) are compared. The experimental results show that PGOA plays a great role in the convergence speed and accuracy. However, combined with the data of HGOA, it can be found that HGOA has better optimization results than GGOA and PGOA. Therefore, while increasing the exploration ability of the algorithm, the balance between the exploration and development ability of the algorithm must be ensured to achieve the optimal effect of the algorithm. Of course, in the convergence curve of some functions, it can be seen that the convergence speed of HGOA is slow, so the convergence speed of the algorithm needs to be further studied in the next study. This paper also applies the improved HGOA to the design of pressure vessels. Experiment shows that HGOA has a good performance in solving the problem of unknown search space. In the future study, we should pay attention to improving the global search ability of the algorithm, while maintaining the balance between the exploration and development ability of the algorithm, and apply the idea to other algorithms to improve the optimization effect of the algorithm.

REFERENCES

[1] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci.*, vol. 237, pp. 82–117, Jul. 2013.

[2] A. R. Yildiz, "An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry," *J. Mater. Process. Technol.*, vol. 209, no. 6, pp. 2773–2780, Mar. 2009.

[3] S. Arora and P. Anand, "Chaotic grasshopper optimization algorithm for global optimization," *Neural Comput. Applic.*, vol. 31, no. 8, pp. 4385–4405, Aug. 2019.

[4] C. Q. Huang and K. X. Zhao, "Three-dimensional flight path planning of UAV based on improved antlion algorithm," *J. Electron. Inf. Technol.*, vol. 40, no. 7, pp. 13–19, 2018.

[5] T. Jayabarathi, T. Raghunathan, B. Adarsh, and P. N. Suganthan, "Economic dispatch using hybrid grey wolf optimizer," *Energy*, vol. 111, pp. 630–641, Sep. 2016.

[6] Z. Wu, D. Shen, M. Shang, and S. Qi, "Parameter identification of single-phase inverter based on improved moth flame optimization algorithm," *Electric Power Compon. Syst.*, vol. 47, nos. 4–5, pp. 456–469, Mar. 2019.

[7] Z. Y. Li, L. Ma, and H. Z. Zhang, "Application of bat algorithm in multi-target and multi-selection backpack problem," *Comput. Simul.*, vol. 30, no. 10, pp. 366–369, 2013.

[8] Y.-M. Shen and R.-M. Chen, "Optimal multi-depot location decision using particle swarm optimization," *Adv. Mech. Eng.*, vol. 9, no. 8, pp. 1–15, Aug. 2017.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, 1995, pp. 1942–1948.

[10] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52–67, Mar. 2002.

[11] X. L. Li, Z. J. Shao, and J. X. Qian, "An optimization model based on animal local government: Fish swarm algorithm," *Syst. Eng.-Theory Pract.*, vol. 22, no. 11, pp. 32–38, 2002.

[12] D. Karaboga, "Artificial bee colony algorithm," *Scholarpedia*, vol. 5, no. 3, p. 6915, 2010.

[13] R. Rajabioun, "Cuckoo optimization algorithm," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 5508–5518, Dec. 2011.

[14] X. Yang and A. H. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Eng. Comput.*, vol. 29, no. 5, pp. 464–483, Jul. 2012.

[15] S. Mirjalili, "The ant lion optimizer," *Adv. Eng. Softw.*, vol. 83, pp. 80–98, May 2015.

[16] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[17] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[18] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, Mar. 2017.

[19] M. A. Taher, S. Kamel, F. Jurado, and M. Ebeed, "Modified grasshopper optimization framework for optimal power flow solution," *Electr. Eng.*, vol. 101, no. 1, pp. 121–148, Apr. 2019.

[20] M. Mafarja, I. Aljarah, H. Faris, A. I. Hammouri, A. M. Al-Zoubi, and S. Mirjalili, "Binary grasshopper optimisation algorithm approaches for feature selection problems," *Expert Syst. Appl.*, vol. 117, pp. 267–286, Mar. 2019.

[21] J. Luo, H. Chen, Q. Zhang, Y. Xu, H. Huang, and X. Zhao, "An improved grasshopper optimization algorithm with application to financial stress prediction," *Appl. Math. Model.*, vol. 64, pp. 654–668, Dec. 2018.

[22] H. Jia, C. Lang, D. Oliva, W. Song, and X. Peng, "Hybrid grasshopper optimization algorithm and differential evolution for multilevel satellite image segmentation," *Remote Sens.*, vol. 11, no. 9, p. 1134, 2019.

[23] J. Wu, H. Wang, N. Li, P. Yao, Y. Huang, Z. Su, and Y. Yu, "Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by adaptive grasshopper optimization algorithm," *Aerosp. Sci. Technol.*, vol. 70, pp. 497–510, Nov. 2017.

[24] J. Liu, A. Wang, Y. Qu, and W. Wang, "Coordinated operation of multi-integrated energy system based on linear weighted sum and grasshopper optimization algorithm," *IEEE Access*, vol. 6, pp. 42186–42195, 2018.

[25] A. A. Ewees, M. A. Elaziz, and E. H. Houssein, "Improved grasshopper optimization algorithm using opposition-based learning," *Expert Syst. Appl.*, vol. 112, pp. 156–172, Dec. 2018.

[26] M. Sulaiman, "Implementation of improved grasshopper optimization algorithm to solve economic load dispatch problems," *Hacettepe J. Math. Statist.*, vol. 48, no. 5, pp. 1570–1589, 2019.

[27] X. Yan and C. M. Ye, "Hybrid locust optimization algorithm for job shop scheduling problem," *Comput. Eng. Appl.*, vol. 55, no. 6, pp. 263–270, 2019.

[28] H. Liang, H. Jia, Z. Xing, J. Ma, and X. Peng, "Modified grasshopper algorithm-based multilevel thresholding for color image segmentation," *IEEE Access*, vol. 7, pp. 11258–11295, 2019.

[29] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, Jun. 2009.

[30] J. Li and B. B. Xing, "Gravitational search algorithm based on dynamic gravitational constant and population decline," *J. Syst. Sci. Math.*, vol. 38, no. 1, pp. 78–85, 2018.

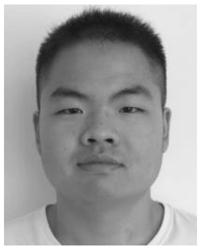
[31] H. Duan and P. Qiao, "Pigeon-inspired optimization: A new swarm intelligence optimizer for air robot path planning," *Int. Jnl. Intel. Comp. Cyber.*, vol. 7, no. 1, pp. 24–37, Mar. 2014.

[32] Z. Hongguo, Z. Changwen, H. Xiaohui, and L. Xiang, "Path planner for unmanned aerial vehicles based on modified PSO algorithm," in *Proc. IEEE Int. Conf. Inf. Automat.*, Washington, DC, USA, Jun. 2008, pp. 541–544.

[33] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proc. Swarm Intell. Symp. (SIS)*, Aug. 2005, pp. 68–75.
[34] M. Jamil, X.-S. Yang, and H.-J. Zepernick, "Test functions for global optimization: A comprehensive survey," in *Swarm Intelligence and Bio-Inspired Computation*. London, U.K.: Elsevier, 2013, pp. 193–222.
[35] F. A. Hashim, E. H. Houssein, M. S. Mabrouk, W. Al-Atabany, and S. Mirjalili, "Henry gas solubility optimization: A novel physics-based algorithm," *Future Gener. Comput. Syst.*, vol. 101, pp. 646–667, Dec. 2019.
[36] W. Xie, J. S. Wang, and Y. Tao, "Improved black hole algorithm based on golden sine operator and levy flight operator," *IEEE Access*, vol. 7, pp. 161459–161486, 2019, doi: 10.1109/ACCESS.2019.2951716.
[37] A. Kaveh and V. Mahdavi, "Colliding bodies optimization: A novel meta-heuristic method," *Comput. Struct.*, vol. 139, pp. 18–27, Jul. 2014.
[38] A. Kaveh, T. Bakhshpoori, and E. Afshari, "An efficient hybrid particle swarm and swallow swarm optimization algorithm," *Comput. Struct.*, vol. 143, pp. 40–59, Sep. 2014.
[39] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, Jan. 2013.
[40] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Appl. Math. Comput.*, vol. 188, no. 2, pp. 1567–1579, May 2007.

S. S. GUO is currently pursuing the master's degree with the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China.

J. S. WANG (Member, IEEE) received the B.Sc. and M.Sc. degrees in control science from the University of Science and Technology Liaoning, China, in 1999 and 2002, respectively, and the Ph.D. degree in control science from the Dalian University of Technology, China, in 2006. He is currently a Professor and a Master's Supervisor with the School of Electronic and Information Engineering, University of Science and Technology Liaoning. His main research interests are modeling of complex industry process, intelligent control, and computer integrated manufacturing.



W. XIE is currently pursuing the master's degree with the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China.

M. W. GUO is currently pursuing the master's degree with the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China.

L. F. ZHU is currently pursuing the master's degree with the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China.