Received May 12, 2020, accepted May 22, 2020, date of publication May 26, 2020, date of current version June 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997783

Improved Salp Swarm Algorithm Based on Levy Flight and Sine Cosine Operator

J. ZHANG[1] AND J. S. WANG[2], (Member, IEEE)
[1]School of Computer and Software Engineering, University of Science and Technology Liaoning, Anshan 114000, China
[2]School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan 114000, China

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

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

ABSTRACT The salp swarm algorithm (SSA) is a swarm intelligence optimization algorithm that simulates the chain movement behavior of salp populations in the sea. Aiming at the shortcomings of the SSA, such as low precision, low optimization dimension and slow convergence speed, an improved salp swarm algorithm based on Levy flight and sine cosine operator (LSC-SSA) was proposed. The Levy flight mechanism uses the route of short walks combined with long jumps to search the solution space, which can effectively improve the global exploration capability of the algorithm. Improved sine cosine operator use sine search for global exploration and cosine search for local exploitation. At the same time, an adaptively switching between the two function search methods can achieve a smooth transition between global exploration and local exploitation. In the simulation experiment, salp swarm algorithm (SSA), whale optimization algorithm (WOA), particle swarm algorithm (PSO), sine cosine algorithm (SCA), firefly algorithm (FA) and LSC-SSA were adopted for solving function optimization problems. Then, the feasibility of the improved algorithm for solving high-dimensional large-scale optimization problems and the effectiveness of the improvement strategy are evaluated. Finally, LSC-SSA was applied to train muti-layer perceptron neural network. Simulation results show that the introduction of Levy flight and improved sine cosine operator in LSC-SSA significantly improves optimization accuracy and convergence speed compared with other swarm optimization algorithms. In addition, the improved algorithm can effectively solve high-dimensional large-scale optimization problems. In the application of training muti-layer perceptron NN, the improved algorithm can avoid falling into the local optimal value and obtain the ideal classification accuracy.

INDEX TERMS Salp swarm algorithm, levy flight mechanism, sine cosine algorithm, function optimization, muti-layer perceptron.

I. INTRODUCTION

The meta-heuristic algorithm has attracted researchers' attention due to its advantages such as simplicity, few parameters, derivation-free mechanism, and avoidance of local optimization. In addition to a large number of theoretical studies, meta-heuristic optimization algorithms have also been applied to engineering fields in different disciplines. Firstly, simplicity means that meta-heuristic algorithms are inspired by simple principles in nature and mathematical models are built by simulating evolutionary concepts, biological intelligence behaviors, and physical phenomena. Researchers create new meta-heuristic algorithms through different inspirations, and can also integrate other meta-heuristic algorithms based on meta-heuristics (PSO-GA [1], GA-DE [2], KH-CS [3], ACO-DE [4]) or adding search operators to propose an improved meta-heuristic algorithm [5]–[8]. Secondly, the meta-heuristic algorithm controls fewer parameters. In general, the original intention of setting parameters is to have a beneficial impact on the algorithm. However, the searching space for practical problems is complex. Too many parameters may increase complexity and calculation scale, which will adversely affect the algorithm. Fewer parameters can avoid the above problems and make the algorithm flexibly applied to optimization problems in different fields. Thirdly, derivation-free mechanism exists in most meta-heuristic algorithms. The information contained in the actual problem may derive more information. The

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

gradient-based optimization method needs to calculate the gradient information of the searching space, but the meta-heuristic algorithm generates the next solution from a random solution. The meta-heuristic algorithm treats the optimization problem as a black box, and only needs to consider the input and output to solve the optimization problem, without calculating the derivative of the searching space. Therefore, meta-heuristic algorithms are more suitable for practical problems with complex information. Finally, the meta-heuristic algorithms have the characteristic of avoiding falling into local optimum. The searching space for practical problems has a large number of local optimal values, which makes the optimization process difficult. The traditional optimization method is easy to fall into the local optimum and ignore the global optimum. The random optimization of the meta-heuristic algorithms make the searching agents widely distributed in the searching space, which will reduce the probability of falling into the local optimum.

Generally, meta-heuristic algorithms can be divided into two categories: individual-based algorithms and population-based algorithms. Individual-based algorithms initialize a candidate solution and improves the candidate solution during the optimization process. For example, Simulated Annealing (SA) [9]–[11], Tabu Search (TS) [12], Iterative Local Search (ILS) [13] are individual-based algorithms. However, the population-based algorithms obtain a set of candidate solutions by initializing the searching agent population, and optimizes this set of candidate solutions in subsequent iterations. Compared with individual-based algorithms, population-based algorithms have global explorability that can avoid falling into a local optimum. At the same time, the information exchange mechanism between populations is conducive to optimizing candidate solutions.

There are several main branches of population-based algorithms: evolution-based algorithms, physical phenomenon-based algorithms, and swarm intelligent (SI)-based algorithms. The evolution-based algorithms use the idea of natural evolution. The initial population retains the best individuals and eliminates the poor ones through combination, crossover, and mutation so as to ensure that the newly generated population is always better than the previous generation. Evolution-based algorithms include Evolutionary Strategy (ES) [14], Differential Evolutionary Strategy (DE) [15], [16], Biography-based Optimization Algorithm (BBO) [17], Probability-Based Incremental Learning (PBIL) [18], etc. For example, Genetic Algorithm (GA) [19] takes Darwinian evolution as the inspiration and inherits better individuals to the next generation of individuals, so that the initial population is continuously optimized in iteration. Physical phenomenon-based algorithms use gravitational, inertial, gravity, electromagnetic, etc. in physical phenomena to perform information exchange and mobile search between populations in solution space. Physical phenomenon-based algorithms mainly include Water Cycle Algorithm (WCA) [20], Henry gas solubility optimization (HGSO) [21], Electrostatic Discharge Algorithm (ESDA) [22]. For example, the Black Hole Algorithm (BH) [23] takes black hole and universal gravity as inspiration. Through hundreds of years of evolution, biological populations can effectively organize hunting, sailing, defending, and foraging behaviors. Birds sailing in a V-shape can evenly distribute resistance among populations to save energy. Bees search for food and use pheromone to mark the path, guiding other individuals to find the shortest path from the hive to the food. Swarm Intelligent (SI)-based algorithms are inspired by the intelligent behavior of biological swarms, including Marine Predators Algorithm (MPA) [24], Seagull Optimization Algorithm (SOA) [25], Spotted Hyena Optimizer (SHO) [26], and Naked Mole-Rat algorithm (NMR) [27], Equilibrium Optimizer (EO) [28], Parasitism Predation Algorithm (PPA) [29], Manta ray foraging optimization (MRFO) [30], Social Ski-Driver optimization algorithm (SSD) [31], etc. For example, Ant colony algorithm (ACO) [32] takes ant colony as inspiration. Ant colonies can use information exchange mechanisms to find the shortest path to food sources in different environments. SI-based algorithms are usually equipped with fewer parameters and operators, which will reduce the complexity and computational scale of the algorithm. More importantly, SI-based algorithms usually retain information in the searching space for communication among individuals in the swarm. Therefore, SI-based algorithms are superior the evolution-based algorithms and physical phenomenon-based algorithms.

In general, SI-based algorithms are inspired by the intelligent social behavior of biological swarms. The Salp Swarm Algorithm (SSA) [33] was proposed by Seyedali Mirjalili. SSA solves optimization problem by establishing mathematical model that simulates the salp swarm. SSA is equipped with fewer parameters and operators, and the structure is simple and easy to implement. However, the shortcomings of SSA are also obvious. When the leader falls into the local optimum, it will mislead other search agents (followers) to stagnate in the local optimum. At the same time, the leader's location update model reduces the search efficiency of food. In addition, the mathematical model of SSA lacks the transition between exploitation and exploration. Finally, the improved methods of SSA are mostly for solving low-dimensional optimization problems. The performance of SSA in solving high-dimensional optimization is unknown.

This paper uses two improvement strategies to make up for the shortcomings of SSA, and the improved SSA based on Levy flight and sine cosine operator (LSC-SSA) was proposed. First, Levy flight with step size control factor is used to increase the traversal and exploration capabilities of search agents. Second, improved sine cosine operator is used to improve the search efficiency of leader. At the same time, improved sine cosine operator is used to improve the balance between exploitation and exploration. Levy flight is a function that simulates animal foraging routes, which was proposed by French mathematician Paul Pierre Lévy. Researchers found that most animals' foraging routes followed Levy flight. The long-term short-step local search in the Levy flight mechanism can improve the diversity and traversal of the algorithm. The short-term long-step global jump can make the search agents jump out of the local optimum and improve the global exploration capability. As a global search operator, Levy flight is applied to many improved algorithms [34]–[36]. Xin-She Yang and Suash Deb proposed the Cuckoo Search Algorithm (CS) in 2009 [37]. Levy flight was introduced to update the bird's nest position, which effectively improved the global exploration capability of algorithm. In addition, as a novel branch of heuristic algorithm, there are little related literature on mathematical rules based algorithms. The Basic Optimization Algorithm (BOA) [38] uses basic mathematical operators and shrinking lengths to guide search agents closer to the optimal value. The Sine Cosine Algorithm (CSA) [39] builds mathematical model by adaptively and equally using sine and cosine search methods. As the name implies, mathematical rules based algorithms are inspired by mathematical rules. This type of algorithm can well balance exploitation and exploration capabilities.

The innovation of this paper is to introduce the Levy flight mechanism with step size control factor into the salp swarm algorithm, which improves the traversal and global exploration ability of the algorithm. In addition, the improved sine cosine operator and introduced into the salp swarm algorithm. The improved sine cosine operator use sine search for global exploration and cosine search for local exploitation. The improved sine cosine operator uses a more effective convergence factor. At the same time, the logarithmic spiral search route also introduced into the sine cosine operator. At the same time, according to the no free lunch theorem (NFL) [40], the effectiveness of the algorithm in a set of optimization problems may not be extended to other optimization problems. In other words, there is no one algorithm that can solve all optimization problems. Because the improved algorithm may be superior to other algorithms on some optimization problems, the innovation and motivation of this paper are strongly supported. The paper is organized as follows. Section III introduces the SSA. Section IV introduces the Levy flight mechanism, sine cosine algorithm and the proposed LSC-SSA in details. Section V selects the Salp Swarm Algorithm (SSA), Particle Swarm Optimization (PSO) [41], Whale Optimization Algorithm (WOA) [42], Sine Cosine Algorithm (CSA), Firefly Algorithm (FA) [43] and LSC-SSA to carry out the function optimization comparison experiments. The experimental results show that LSC-SSA has the advantages of high optimization accuracy and fast convergence speed. The feasibility of the improved algorithm for solving high-dimensional function optimization and the effectiveness of the improvement strategy are verified. Section VI applies LSC-SSA to train muti-layer perceptron neural network. Section VII is the conclusion of this paper.

II. RELATED WORK OF SSA

The SSA algorithm relies on the concept of swarm intelligence and has a simple structure, which has attracted the attention of many scholars. With the deepening of research, many improved methods and practical applications of SSA have been proposed [44]–[47]. Neggaz, N et al used the sine and cosine algorithm and the disrupt operator to improve SSA and proposed ISSAFD [48]. This mechanism can improve the exploration phase and avoid local stagnation. Experimental results show that ISSAFD has good performance in terms of accuracy and number of features. However, the performance of convergence speed and optimization accuracy is not shown. At the same time, the SCA algorithm has the drawback of slow convergence. SCA should be modified to introduce SSA. Tubishat, M et al proposed an improved Salp Swarm algorithm (ISSA) [49] to solve the feature selection problem and select the best subset of features in the packaging mode. Experiments show that ISSA is superior to other algorithms in accuracy, convergence and feature reduction. However, the improved algorithm needs to be further improved in terms of optimization accuracy. Panda, N et al used spatial transformation search (STS) to improve the performance of SSA and proposed STS-SSA [50]. Experimental results show that STS-SSA can effectively solve the optimization problem. Abd Elaziz, M et al proposed a multi-objective big data optimization method based on hybrid SSA algorithm and DE algorithm [51]. The experimental results of the test problems in the 2015 big data optimization competition show that the proposed method is superior to other methods on all test problems. However, the performance of the improved algorithm in function optimization has not been verified. Faris, H *et al.* Proposed two methods for feature selection using SSA as a search strategy [52]. The experimental results of 22 UCI data sets show that the proposed method is obviously superior to other methods. El-Fergany, AA used SSA to optimize the optimal values of unknown parameters of the polymer exchange membrane fuel cell model [53]. Simulation results show that the proposed SSO-based method can effectively solve the optimal solution of the model. Yang, B *et al.* Expanded the salp population into multiple independent salp chains and proposed the modular salp swarm algorithm (MSSA) [54]. Simulation results show that MSSA is superior to the other eight algorithms.

III. SALP SWARM ALGORITHM

A. MATHEMATICAL MODEL OF SALP SWARM ALGORITHM

Establishing a mathematical model that mimics the intelligent behavior of swarm is the basic step for SI-based algorithms to solve an optimization problem. Mathematical models with fish swarm, bird swarm, and ant swarm have been widely used in optimization problems. In order to model the salps chain formed by end-to-end individuals, the individuals in the salps swarm are divided into two categories: leader and followers. The leader is the foremost individual of the salps chain to determine the movement direction and foraging route of the population, and guide the salps chain toward the food. The remaining individuals are followers. They follow the leader in turn to form a chain structure. However,

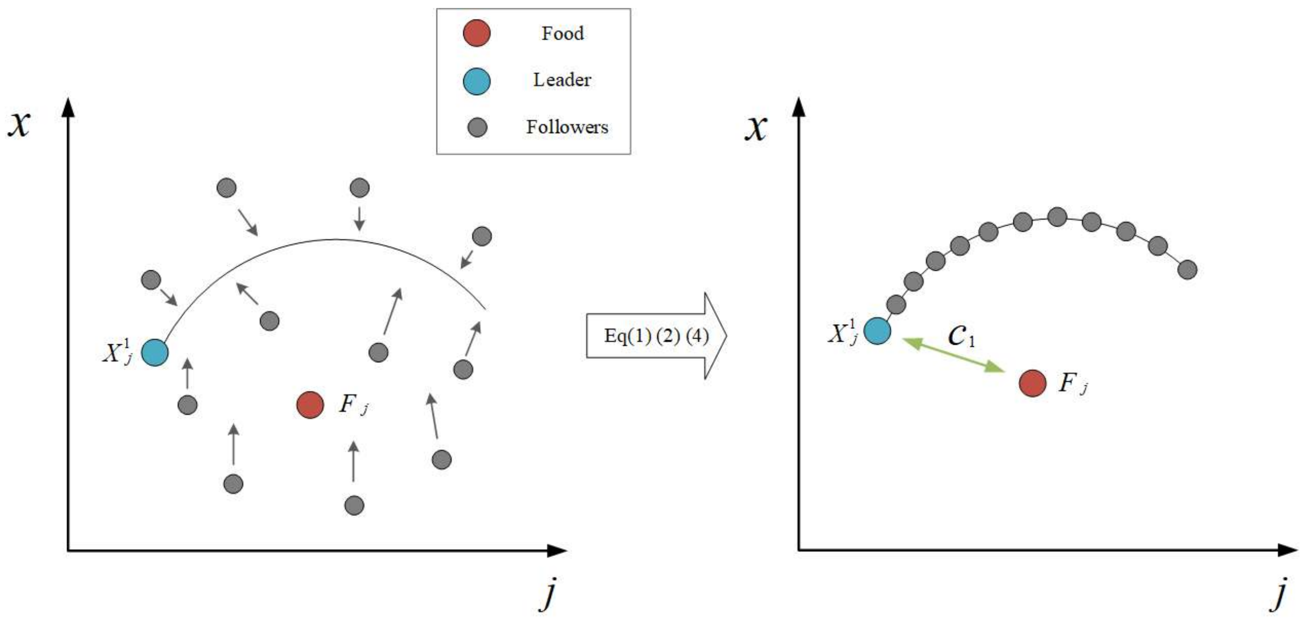


FIGURE 1. **Chain movement of salp swarm in searching space.**

the mathematical model only simulates the generation of the salps chain, and cannot directly solve the optimization problem. The mathematical model needs to be adjusted to adapt to the optimization problem. Determining the global optimal value is the goal of the optimization problem, so the global optimal value is used as the food that the salps chain needs to find. The position of the global optimal value in the optimization problem is unknown. Therefore, taking the optimal value in the current iteration as the global optimal value (food), the salps chain model can be moved closer to the target value. According to the position of the food update the leader, the entire salps chain can be brought closer to the food. This process is represented by the following equation:

$$X_j^1 = \begin{cases} F_j - c_1\left[\left(ub_j - lb_j\right)c_2 + lb_j\right] & if\ c_3 < 0.5 \\ F_j + c_1\left[\left(ub_j - lb_j\right)c_2 + lb_j\right] & if\ c_3 \geq 0.5 \end{cases} \tag{1}$$

where, X_j^1 indicates the position of the leader (the frontmost individual of the salps chain) in the j-th dimension; F_j indicates the position of the food in the j-th dimension; ub_j is the lower bound of the j-th dimension; lb_j is the upper bound of the j-th dimension; Upper and lower bounds are used to limit leader from exceeding the searching space. Parameter c_2 is a random number between [0.1], which is used to control the leader's moving step. Parameter c_3 is a random number between [0.1], which is used to equally select whether the leader's moving direction is closer or farther from the food location. Parameter c_1 shown in Eq. (2) is an adjustment factor that is used to balance global exploration and local exploitation.

$$C_1 = 2e^{-(4t/T)^2} \tag{2}$$

where, t is the current number of iterations, and T is the total number of iterations.

It can be seen from Eq (2) that the adjustment factor c_1 will adaptively decrease with the iterative process. At the beginning of the iteration, the decreasing trend of the adjustment factor c_1 is slow, which drives the leader to conduct a large-scale global exploration. In the later iterations, the decreasing trend of the adjustment factor c_1 is obvious, and the leader can carry out the detailed exploitation. In order to make followers follow the leader to form a chain structure, Newton's law of motion is used to update the position of followers, which is described as:

$$X_j^i = \frac{1}{2}a{\cdot}t^2 + v_0 \cdot t \tag{3}$$

where, X_j^i indicates the position of the i-th follower in the j-th dimension when $i \geq 2$; t represents time; v_0 represents the initial speed, and the acceleration of the follower's movement $a = v_{final}/v_0$; The speed of the follower is $v = (x - x_0)/t$. The time variable of the optimization problem is represented by the number of iterations, so the iteration interval represents the time interval, $t = 1$. The follower's initial speed $v_0 = 0$. Eq (3) can be updated as:

$$X_j^i = \frac{1}{2}\left(X_j^i + X_j^{i-1}\right) \tag{4}$$

In the process of following the leader to update the position, the followers may reach a position better than the current best solution (food). At this time, the food is replaced to the better position, and the updated leader guides the followers to move in the direction of food. The chain movement of salp swarm in searching space is shown in Fig. 1.

The advantages of SSA are as follows: 1) It can be seen from the mathematical model of SSA that it has a simple structure and is equipped with few parameters and operators, so it is easy to implement. 2) In the process of optimization, SSA only uses the optimal solution in the current iteration as food. Even the deterioration of the fitness of the entire population will not affect the quality of the food. 3) Leaders can explore and get closer based on the location of the food. The followers only need to move in a chain according to the position of the leader, which reflects the simplicity of the algorithm.

When considering the advantages, it is also necessary to discuss the shortcomings of SSA. The shortcomings of SSA are as follows: 1) Followers only need to follow the leader, which embodies simplicity. However, when the leader falls into the local optimum, it will mislead the entire population into the local optimum.. 2) The leader updates the location based on the food. However, updating the position of the leader requires calculating boundaries, which reduces the direct interaction between the leader and the food. 3) The mathematical model of SSA lacks the transition between exploitation and exploration, which leads to a low precision of the algorithm.

B. PROCEDURE OF SALP SWARM ALGORITHM

The procedure of standard SSA is described as follows.

Step 1: Initialize the algorithm parameters: Number of iterations T, number of ascidian populations N, test function dimension D.

Step 2: Initialize the salp population according to the upper and lower bounds, $t = 1$.

Step 3: Calculate the fitness value of each search individual, and treat the individual with the best fitness value in the current population as food F_j.

Step 4: Update c_1 according to Eq. (2) and generate random numbers c_2 and c_3.

Step 5: If $i = 1$, update the leader's position according to Eq.(1). If $i \geq 1$, update the follower's position according to Eq. (4). $t = t + 1$.

Step 6: Determine whether the algorithm has reached the maximum number of iterations or found the optimal value. If the algorithm's end condition is met, the optimal value is returned and exited; otherwise, go to Step 3.

The flowchart of salp swarm algorithm is shown in Fig. 2.

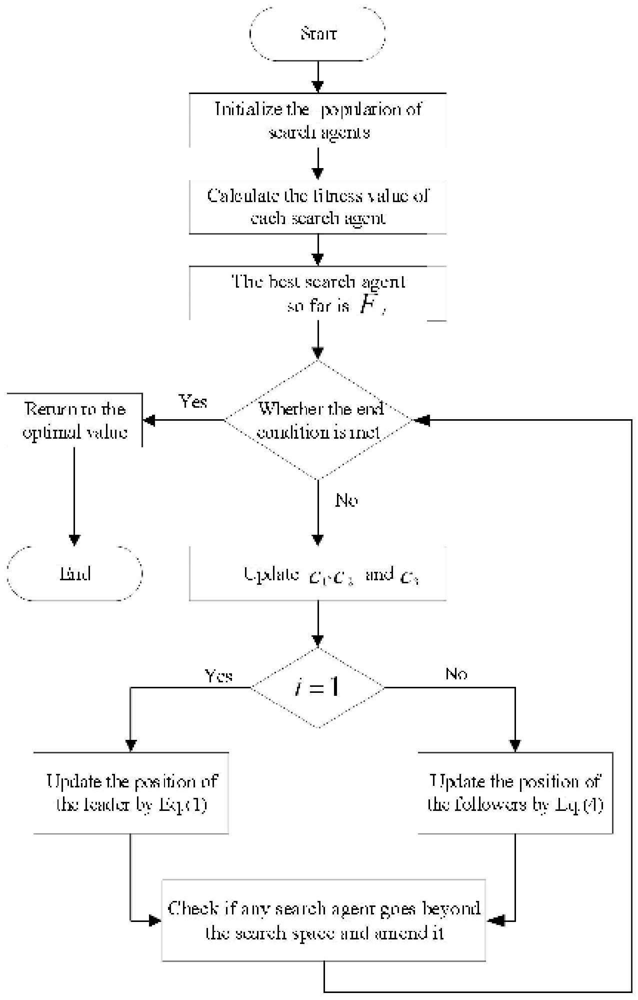

Start
Initialize the population of search agents
Calculate the fitness value of each search agent
The best search agent so far is F_j
Whether the end condition is met
Yes
Return to the optimal value
End
No
Update c_1, c_2 and c_3
$i = 1$
Yes
No
Update the position of the leader by Eq.(1)
Update the position of the followers by Eq.(4)
Check if any search agent goes beyond the search space and amend it

FIGURE 2. The flowchart of SSA.

IV. IMPROVED SALP SWARM ALGORITHM BASED ON LEVY FLIGHT AND SINE COSINE OPERATOR

Levy flight is a function that simulates animal foraging routes, which was proposed by French mathematician Paul Pierre Lévy. Researchers found that most animals' foraging routes followed Levy flight. The long-term short-step local search in the Levy flight mechanism can improve the diversity and traversal of the algorithm. The short-term long-step global jump can make the search agents jump out of the local optimum and improve the global exploration capability. In addition, as a novel branch of heuristic algorithm, there is very little related literature on mathematical rules based algorithms. The Sine Cosine Algorithm (CSA) builds mathematical model by adaptively and equally using sine and cosine search methods. SCA algorithm can well balance exploitation and exploration capabilities.

A. LEVY FLIGHT

Levy flight is a probability distribution proposed by the French mathematician Paul Pierre Lévy (1886-1971) in the 1930s [55], which is used to simulate bird foraging routes. So far, some scholars have shown that the foraging trajectories of many birds and insects in nature (such as albatross, bees and fruit flies) conform to the Levy distribution. Even more novel is that some marine animals (tuna, swordfish, some sharks, etc.) also follow the mathematical strategy of Levy distribution when foraging. These studies formed the Levy flight foraging hypothesis: Levy flight can improve the efficiency and accuracy of biological foraging, and it is more naturally adaptable.

As a global searching operator, Levy flight mechanism searches for space using short-distance walking combined with long-distance jumping routes. Among them, long-term short-distance walking can enable the search agent to

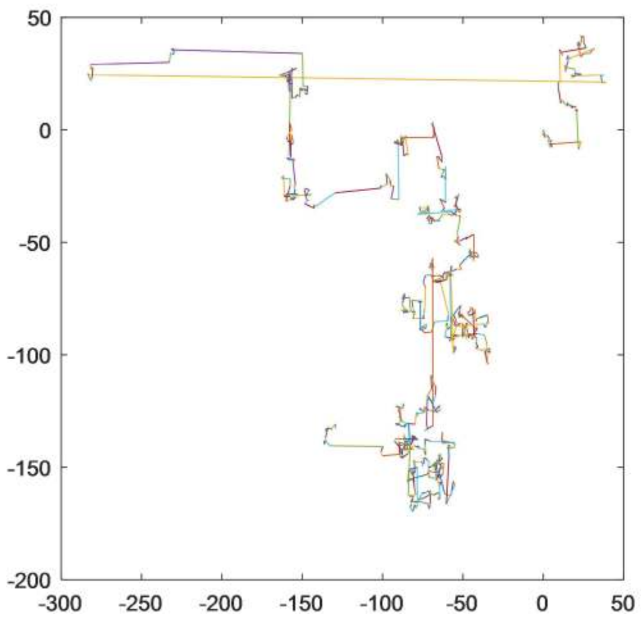


FIGURE 3. 500-step trajectory of Levy flight.

carefully search the area near it, which improves the diversity and local exploitation ability of the population. The directional variability of the occasional long-distance jump guarantees a large probability search of the entire region by the population, and the abrupt change has a great advantage for exploring problems in a large space. The combination of short-distance and long-distance methods achieves sufficient optimization of the solution domain, which greatly improves the global search ability of the algorithm. The 500-step motion trajectory of the Levy flight within the search range is shown in Fig. 3, which fully verified the characteristics of the short distance of Levy flight combined with the occasional long distance jump, and fully explored the solution domain.

The probability density function of Levy flight obeys the Levy distribution, which can be described as follows.

$$Levy\,(s) = \frac{1}{\pi}\int_0^\infty \exp\left(-\beta|k|^\lambda\right)\cos\,(ks)\,dk \tag{5}$$

where, $0 < \lambda \leq 2$ to control the peak sharpness of the Levy distribution graph; $\beta > 0$ to control the span of the distribution graph. When $\lambda = 2$, the Levy distribution is transformed into a Gaussian distribution; when $\lambda = 1$, the Levy distribution is transformed into a Cauchy distribution. There is no clear analysis of the integral formula, and it is more difficult to generate a random number that obeys the distribution. However, when $s \gg s_0 > 0$, that is to say $s \to \infty$, Eq. (5) can be updated as:

$$Levy\,(s) \approx \frac{1}{\pi}\cdot\lambda\beta\cdot\Gamma\,(\lambda)\sin\left(\frac{\pi\lambda}{2}\right) \tag{6}$$

The approximate distribution exhibits power-law behavior, and the variance exhibits an exponential relationship with time, that is to say $\sigma^2\,(t) \sim t^{3-\beta}$. So Levy flight is better than Brown sport. Since then, many scholars have proposed many implementation methods for generating random numbers obeying the Levy distribution according to this approximate formula, which includes a method proposed by Mantegna in 1994 to solve random numbers using the normal distribution, sometimes called the Mantegna method [56]. The Mantegna method for generating random step sizes obeying the Levy distribution is described as follows:

$$S = \frac{u}{|v|^{1/\beta}} \tag{7}$$

where, u and v obey the following Gaussian distribution.

$$u \sim \left(0, \sigma_u^2\right), \quad v \sim \left(0, \sigma_v^2\right) \tag{8}$$

$$\sigma_u = \left[\frac{\Gamma\,(1+\beta)\cdot\sin\,(\pi\beta/2)}{\Gamma\,[(1+\beta)/2]\,\beta\cdot 2^{\beta-1/2}}\right]^{1/\beta} \tag{9}$$

$$\sigma_v = 1 \tag{10}$$

where, $\beta = 1.5$; Γ is a Gamma function, which is calculated by:

$$\Gamma\,(z) = \int_0^\infty t^{z-1}e^{-t}dt \tag{11}$$

When $z = n$, $\Gamma\,(n) = (n-1)!$. A large number of studies have shown that Levy flight mode can maximize the efficiency of search targets under uncertain conditions [44]. When solving the function optimization problems, the equation for updating the population position by the Levy flight mechanism can be described as:

$$X_i\,(t+1) = X_i\,(t) + S \otimes X_i\,(t) \tag{12}$$

where, $X_i\,(t+1)$ indicates the position of the population after the Levy flight operation; $X_i\,(t)$ is the position of the current population; s is a random step that obeys the Levy distribution shown in Eq. (7); $\otimes$ indicates the dot product between elements.

B. SINE COSINE ALGORITHM

Sine Cosine Algorithm (SCA) is a novel mathematical rules based algorithm. As the name implies, this algorithm uses the sine function combined with the cosine function in the mathematical rules to solve the optimization problem. SCA has fast convergence speed, simple structure, and can well balance the global exploration ability and local exploitation ability. In general, a population-based algorithm initializes a set of random solutions. After evaluation of the objective function of the optimization problem, this set of random solutions will be improved. If the distribution of the agents in the searching space are too concentrated, it will fall into the local optimal value and ignore the global optimal value, which will reduce the global explorability of the algorithm. On the contrary, if the distribution of the agents are too scattered, the local optimal value will be ignored, which will reduce the local exploitation of the algorithm. Therefore, balancing global exploration and local exploitation is an important part of optimization algorithms. In order to achieve this function,

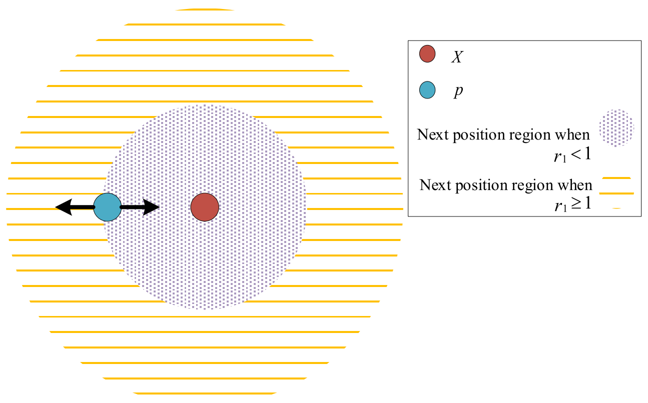

X
p
Next position region when $r_1 < 1$
Next position region when $r_1 \geq 1$

FIGURE 4. The model combining sine search and cosine search.

the sine cosine algorithm uses the sine search method for exploration and the cosine search method for exploitation. The equation of sine search and cosine search are described as follows:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \cdot \sin(r_2) \cdot \left| r_3 p_i^t - X_i^t \right| & \text{if } r_4 < 0.5 \\ X_i^t + r_1 \cdot \cos(r_2) \cdot \left| r_3 p_i^t - X_i^t \right| & \text{if } r_4 \geq 0.5 \end{cases} \tag{13}$$

where, X_i^t indicates the position of the individual in the i-th dimension in the t-th iteration and p_i^t indicates the position of the current optimal individual in the i-th dimension.

It can be seen from Eq. (13) that there are four main parameters in the sine cosine algorithm: r_1, r_2, r_3 and r_4. r_2 is a random number between $[0,2\pi]$ to control the moving distance of the search agent. r_3 is a random number to provide weight to the search agent to enhance ($r_3 > 1$) or weaken ($r_3 < 1$) the effect of the individual's moving distance. r_4 is a random number between [0,1] to control the equal use of two search methods. r_1 can adaptively guide the moving direction of the search agent (the location to be searched next time), which is calculated by:

$$r_1 = a - t\frac{a}{T} \tag{14}$$

where, t indicates the current number of iterations; T indicates the maximum number of iterations; a is a constant that limits the size of r_1, generally $a = 2$.

As an adaptive guide factor, r_1 can guide the search agent's movement direction (next search position). When $r_1 < 1$, r_1 guides the search agent to the area near the optimal value. when $r_1 \geq 1$, r_1 guides the search agent to spread beyond the optimal value. The effect of adaptive guidance factor r_1 on Eq. (13) is shown in Fig. 4. Fig. 4 illustrates that Eq. (13) defines the area of the search agent and optimal value in the searching space, and the movement direction of the search agent can be changed by the adaptive guidance factor r_1.

Normally, the range of values for the sine and cosine functions is [−1,1]. By expanding the range of the sine and cosine functions to [−2, 2], Eq. (13) can be extended to higher dimensions to accommodate the complex searching space of the optimization problem. At this time, parameter r_2 (controlling the moving distance of the search agent) can ensure that the agent switches between the two search ranges ([−1,1] and [−2,2]). This mechanism effectively ensures the coordination of exploration and exploitation in the searching space.

The pseudo code of the Sine Cosine Algorithm (SCA) is described as follows:

```
Initialize the search agents population $X_i^t$ $(i = 1, 2, 3 \ldots n)$
Calculate the fitness of each search agent, t = 1
$p_i^t$ = the best search agent so far
While(t < maxmum numer of iterations)
    Update $r_1$, $r_2$, $r_3$ and $r_4$
        IF ($r_4 < 0.5$)
            Update the position of the search agents by the sine search of Eq.(13)
        else if ($r_4 \geq 0.5$)
            Update the position of the search agents by the cosine search of Eq.(13)
        end if
        END IF
Check if any search agent goes beyond the search space and amend it
Calculate the fitness of each search agent
Update $p_i^t$ if there is a better solution
t = t + 1
End while
Return $p_i^t$
```

Seen from the pseudo-code and mathematical model of sine cosine algorithm, it is known that the algorithm is optimized based on the search method generated by the sine function and cosine function, and the structure is simple. As a population-based algorithm, SCA continuously improves the initialized random solution to avoid falling into local optimal values. SCA adaptively adjusts the search area of the agents using the sine search method and cosine search method, and saves the current best solution as the target value (global optimal value). This mechanism can balance global exploration and local exploitation without losing the target value, and develop towards the best area of the searching space.

C. IMPROVED SALP SWARM ALGORITHM BASED ON LEVY FLIGHT AND SINE COSINE OPERATOR

In the process of solving optimization problems, how to make the algorithm avoid getting stuck in the local optimal value is a challenge. Avoiding local optimization requires the agent to search in the solution space as widely as possible. As a search operator with strong global performance, the Levy flight mechanism can improve the global exploration capability of the salp swarm algorithm. It uses short-distance walking combined with long-distance jumping routes to conduct a full search of the solution space. Among them, long-term short-distance walking can enable the population to carefully search the area nearest to it, which improves the diversity and local exploitation capacity of the population. The directional variability of the occasional long-distance jump ensures a large probability search of the entire area by the population and improves the global exploration capability of the algorithm. This paper uses improved Levy operator to update the position of salp swarm. The improved Levy operator adds a step size control factor to the Levy flight, which can control (weaken or enhance) the walking step size to suit the searching space of different optimization problems. When the step size control factor is small, the agent can be searched carefully in a small range so as to enhance the exploitation ability of the algorithm without affecting the exploration ability, which is suitable for optimization problems with small searching space. When the step size control factor is large, the agent can search extensively in a wide range so as to increase the probability of the algorithm jumping out of the local optimal value, which is suitable for high-dimensional large-scale optimization problems. The equation of improved Levy operator to update the position of salp swarm is described as follows.

$$X_j^i = X_j^i + a \cdot S \otimes X_j^i \tag{15}$$

where, X_j^i indicates the position of the i-th follower in the j-th dimension when $i \geq 2$; s is the random step size following the Levy distribution generated by the Mantegna method shown in Eq. (7); $\otimes$ indicates the dot product between elements; a is the step size control factor. When the step size control factor is small, the search agent can carefully search in a small range. In this paper, $a = 0.01$.

At the same time, this paper also introduces an improved sine cosine operator to update the position of leader. In the salp swarm algorithm, the leader guides the followers so that the population can move according to the position of the food. In other words, just updating the leader's position can realize the chain movement of the entire salp swarm. The leader position update method shown in Eq. (1) is similar to the population update method shown in Eq (13) of the sine cosine algorithm. The same point is that both of them select the search method equally, and update the positions of the remaining individuals according to the position of the current optimal value. But the difference is that the selection of the former searching method is determined only by probability, while the latter can adaptively switch in the search method according to the information returned by the searching space. Compared with the salp swarm algorithm, the population update method of the sine cosine algorithm can better reflect the balance between exploration and exploitation. Therefore, this paper proposes an improved sine cosine operator to update the position of the leader. Firstly, the parameter r_1 of the sine cosine algorithm is replaced with the parameter c_1 of the salp swarm algorithm. Essentially, parameters r_1 and c_1 have the same effect. As global convergence factors, parameters r_1 and c_1 adaptively decrease with the iterative process, which makes the searching agent gradually converge from global to local. This mechanism guarantees the global convergence of the algorithm. However, the convergence effects of the two global convergence factors are different. The convergence effect of parameters r_1 and c_1 in 1000 iterations is shown in Fig. 5. It can be seen that the convergence effect of

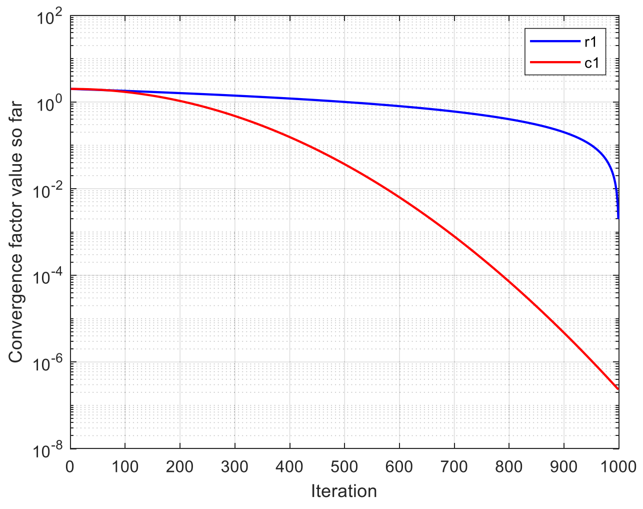


FIGURE 5. **Comparison of convergence effects.**

c_1 is significantly better than r_1, so use c_1 instead of parameter r_1. Second, the exponential function e^x is introduced so that the leader can form a logarithmic spiral path close to the target value. Finally, the random parameter r_3 has limited usefulness to the algorithm. At the same time, too many parameters will increase the randomness of the algorithm, so the parameter r_3 is removed from the sine cosine operator.

The equations for updating the position of the leader by the improved sine cosine operator are described as follows.

$$X_j^1 = X_j^1 + c_1 \cdot \sin(r_2) \cdot \left| F_j - X_j^1 \right| \cdot e^1 \tag{16}$$

$$X_j^1 = X_j^1 + c_1 \cdot \cos(r_2) \cdot \left| F_j - X_j^1 \right| \cdot e^1 \tag{17}$$

The position updating method shown in Eq. (16) can use sine function for global exploration. Eq (17) can use cosine function for local exploitation. Parameter c_1 enables the search agent to adaptively switch between the two search modes for optimization, and to smoothly transition between exploration and exploitation. As a global convergence factor, parameter c_1 also makes the algorithm converge as the iteration increases. Parameter c_1 makes the sine and cosine search methods gradually converge in iteration, which is shown in Fig. 6.

After adding global convergence factor c_1 and parameter r_4, the two search methods can be used in combination. The equation is described as follows:

$$X_j^1 = \begin{cases} X_j^1 + c_1 \cdot \sin(r_2) \cdot \left| F_j - X_j^1 \right| \cdot e^1 & if\ r_4 < 0.5 \\ X_j^1 + c_1 \cdot \cos(r_2) \cdot \left| F_j - X_j^1 \right| \cdot e^1 & if\ r_4 \geq 0.5 \end{cases} \tag{18}$$

where, c_1 have the same effect as in Eq. (2), and r_2 and r_4 have the same effect as in Eq. (13).

When the range of the sine and cosine functions are expanded to $[-2, 2]$, the search space can be expanded. This mechanism guarantees that search agent can search

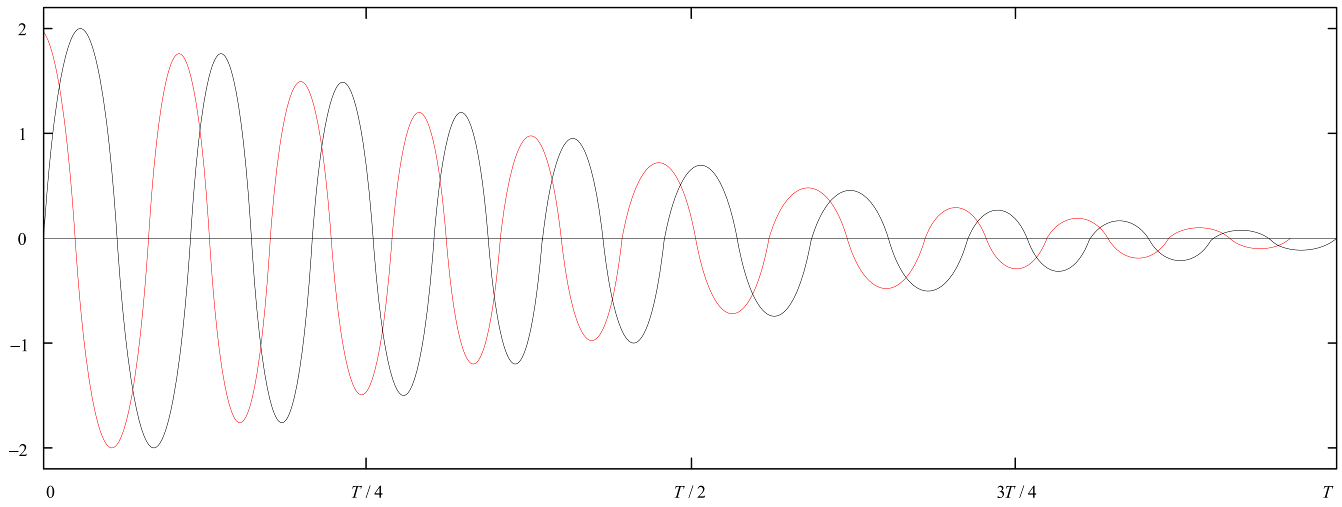


FIGURE 6. The model of sine and cosine to reduce range.

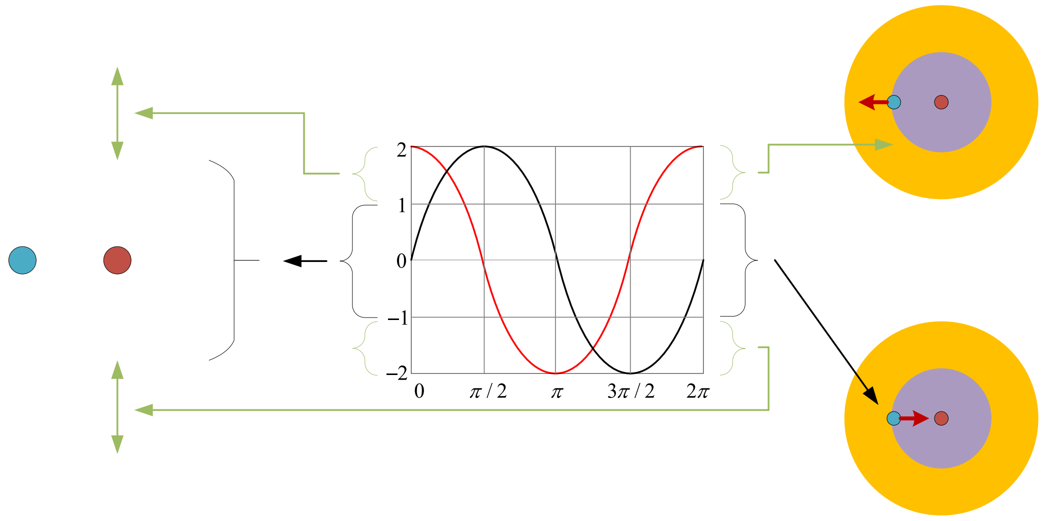


FIGURE 7. The effect of the sine and cosine functions of the range [−2, 2] on the model.

inside or outside the target value to adapt to high-dimensional optimization problems. The two search methods can also search different regions based on the returned value, which will reduce the possibility of falling into the local optimal value. When the returned value is in the range of (1, 2] and [−2, −1), the search agent will search outside the target value, which reflects the global exploration. When the return value is in the range of [−1,1], the search agent will search inside the target value space, which reflects the local exploitation. The model of the two search methods after expanding the scope is shown in Fig. 7. It shows that after expanding the range of the sine and cosine functions, the search agent can perform different search methods based on the returned value.

In addition, the improved algorithm also introduces the idea of "elite search". After the search agent performs the sine cosine operator or the Levy flight operation, the position of the search agent will change, and the updated position may be worse than the position before the update. Therefore, the updated position of the agent is compared to the position of the last iteration. If the fitness value of the agent after executing the operator is better than the fitness value of the agent without executing the operator, the agent will remain in the current position. Otherwise, the search agent will return to the location where the operator was not executed. The idea based on elitist search ensures that the search agent will develop towards a promising area in each iteration.

In order to verify the mathematical model of the improved algorithm, 30 search agents of LSC-SSA were put into the search space of sphere function for simulation experiments. The search range of the optimization function is [−100, 100], the dimension is 30, and the optimal value is 0. The distribution of 50 and 100 iterations of the search agent in the solution

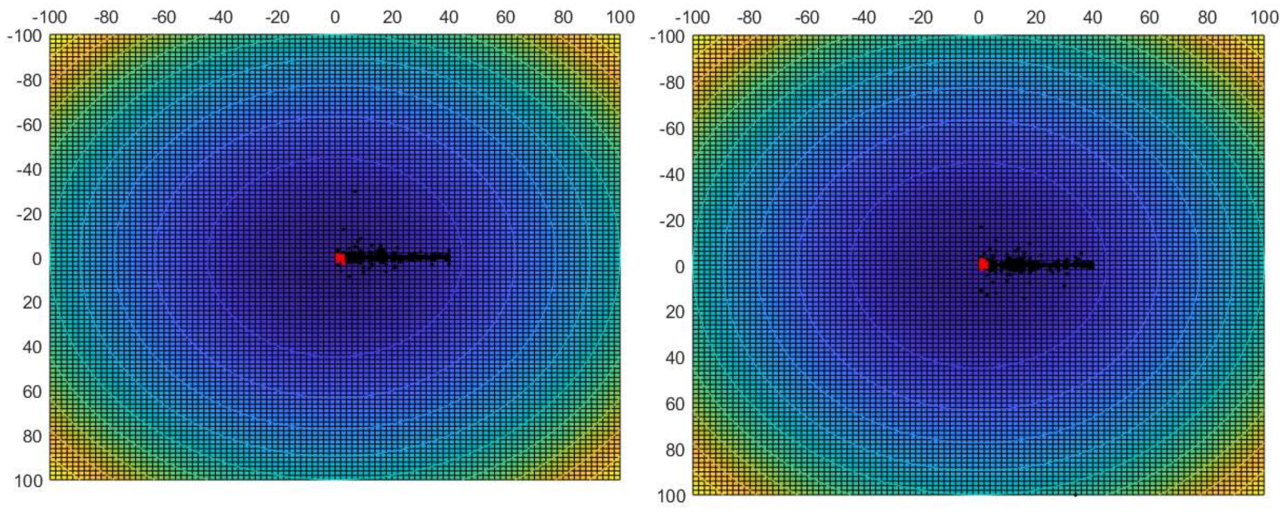


FIGURE 8. Location map of agents in searching space.

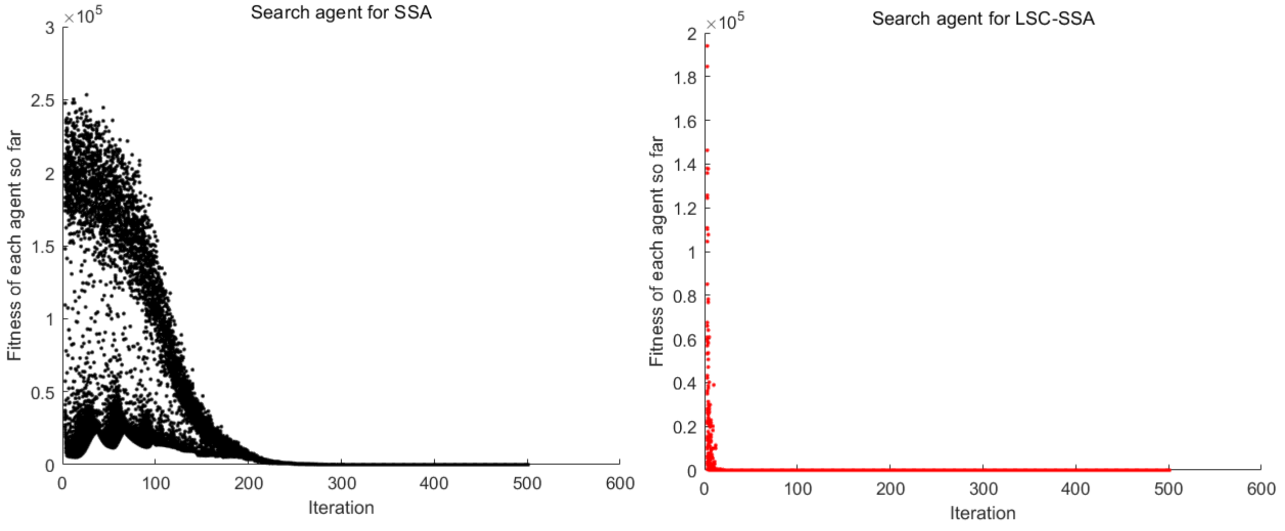


FIGURE 9. Distribution map of fitness values for agents.

space is shown in Fig. 8. The red dots indicate leader and the black dots indicate followers. It can be seen from Fig. 8 that as the iteration increases, the leader can guide the followers to move closer to the global optimal value in a chain motion. The experimental results show that the mathematical model of the improved algorithm is effective. In order to further verify the global convergence ability of the improved algorithm, SSA and LSC-SSA were selected to optimize the sphere function. The historical fitness of 40 search agents at 500 iterations is shown in Fig. 9.

The black dots indicate the fitness value of the search agent of the SSA, and the red dots indicate the fitness value of the search agent of the LSC-SSA. It can be seen from Fig. 9 that the search agent of SSA gradually converges around 200 iterations, while the LSC-SSA agent fitness value quickly completes global convergence within 50 iterations. Simulation experiments show that after the search agent executes the sine cosine operator and the Levy flight operation, the salps chain can effectively move in the searching space. At the same time, the search agent can explore and use the area near the target value, which makes the algorithm easy to jump out of the local optimal value and increase the global convergence.

The flow chart of LSC-SSA is shown in Fig. 10.

The procedure of the LSC-SSA algorithm are described as follows.

Step 1: Initialize algorithm parameters: Number of iterations T, number of ascidian populations N, test function dimensions D.

Step 2: Initialize the salp population according to the upper and lower bounds, $t = 1$.

Step 3: Calculate the fitness value of each search individual, and treat the individual with the best fitness value in the current population as food F_j.

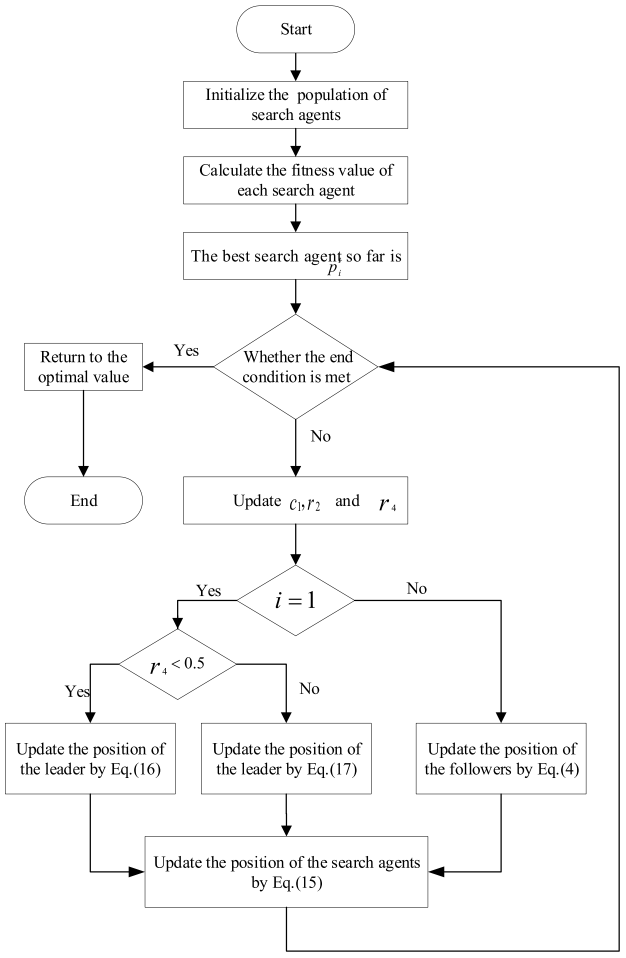

Start
Initialize the population of search agents
Calculate the fitness value of each search agent
The best search agent so far is p_i
Whether the end condition is met
Yes
No
Return to the optimal value
End
Update c_1, r_2 and r_4
$i = 1$
$r_4 < 0.5$
Update the position of the leader by Eq.(16)
Update the position of the leader by Eq.(17)
Update the position of the followers by Eq.(4)
Update the position of the search agents by Eq.(15)

FIGURE 10. **The flowchart of the LSC-SSA.**

Step 4: Update c_1 according to Eq. (2) and generate random numbers r_2 and r_4.

Step 5: If $i = 1$, update the leader's position according to Eq. (19); If $i \geq 1$, update the follower's position according to Eq. (4).

Step 6: Update the position of salp swarm according to Eq. (15). $t = t + 1$

Step 7: Determine whether the algorithm has reached the maximum number of iterations or found the optimal value. If the algorithm's end condition is met, the optimal value is returned and exited; otherwise, go to Step 3.

The pseudo code of LSC-SSA is described as follows.

```
Initialize the search agents population $X_i^t$ $(i = 1, 2, 3 \ldots n)$
Calculate the fitness of each search agent, t = 1
$p_i^t$ = the best search agent so far
while(t < maxmum numer of iterations)
    Update $c_1$, $r_2$ and $r_4$
        IF($i = 1$)
            if2($r_4 < 0.5$)
                Update the position of the leader by the Eq.(16)
            else if2($r_4 \geq 0.5$)
                Update the position of the leader by the Eq.(17)
            end if2
        ELSE IF($i \geq 2$)
                Update the position of the followers by the
                Eq.(4)
        END IF
        Update the position of the search agents by the
                Eq.(15)
    The search agent performs "elitist search" operations
    Check if any search agent goes beyond the search space
    and amend it
    Calculate the fitness of each search agent
    Update $p_i^t$ if there is a better solution
    t = t + 1
    End while
    Return $p_i^t$
```

D. TIME COMPLEXITY ANALYSIS

Time complexity is the calculation workload required to execute the algorithm, and it is an important indicator to evaluate the time consumption of the algorithm. The time complexity is usually expressed by the O symbol, excluding the low-order term and the first term coefficient of this function. In meta-heuristic algorithms, time complexity is related to the number and structure of the operating units of the algorithm. For the basic salp swarm algorithm, the time complexity mainly depends on the number of initial populations, the number of iterations, and the location update mechanism. The time complexity of the improved algorithm LSC-SSA proposed in this paper mainly depends on the number of initial populations, the number of iterations, and the location update mechanism that introduces an improved strategy. In order to evaluate the impact of the improved strategy on the time cost of the algorithm, the time complexity of the salp swarm algorithm and LSC-SSA were analyzed. The time complexity of each operation unit in the salp swarm algorithm is described as follows.

1) Initialize N populations to be distributed in the D-dimensional search space, which needs to be run $N \cdot D$ times.
2) Calculate the fitness value of each search agent and select the best agent as food, which needs to be run $[N \cdot (N - 1)]/2$ times.
3) Parameters c_1, c_2 and c_3 are updated once and need to be run 3 times.
4) The leader performs the position update operation in the D-dimensional search space, which needs to be run $1 \cdot D$ times.
5) The followers perform position update operations in the D-dimensional search space, which needs to be run $(N - 1) \cdot D$ times.
6) Select the optimal from the current population and output it, which needs to be run $N \cdot D$ times.

Each of the above operation units goes through T iterations, so the total time complexity of salp swarm is $O(SSA) = T \cdot \left[ND + \left(N^2 - N\right)/2 + 3 + D + (N - 1)D\right]$.

TABLE 1. **The specific parameter settings of the algorithm.**

Algorithm	Main parameter settings
LSC-SSA	population size n=40;T=1000 $a=0.01$
SSA	population size n=40;T=1000
WOA	population size n=40;T=1000
PSO	population size n=40;T=1000; Learning factor $c_1=2$, $c_2=2$;Inertia weight $W_{Max}=0.9$, $W_{Min}=0.1$
SCA	population size n=40;T=1000
FA	population size n=40;T=1000; $\beta_0=1$; $\gamma=1$

The time complexity of each operating unit of the improved algorithm LSC-SSA is described as follows.

1) Initialize N populations to be distributed in the D-dimensional search space, which needs to be run $N \cdot D$ times.
2) Calculate the fitness value of each search agent and select the best agent as food, which needs to be run $[N \cdot (N-1)]/2$ times.
3) Parameters c_1, r_2 and r_4 are updated once and need to be run 3 times.
4) The Leader updates position in D-dimensional search space through sine cosine operator, which needs to be run $1 \cdot D$ times.
5) The followers perform position update operations in the D-dimensional search space, which needs to be run $(N-1) \cdot D$ times.
6) The Levy flight mechanism updates the population position, which needs to be run $N \cdot D$ times.
7) Use the idea of "elite search" to test the position quality of the population, which needs to be run $N \cdot D$ times.
8) Select the optimal from the current population and output it, which needs to be run $N \cdot D$ times.

Each of the above operation units goes through T iterations, so the total time complexity of LSC-SSA is $O(LSC-SSA)=T \cdot \left[ND+\left(N^2-N\right)/2+3+D+(N-1)D\right]$. Compared with the salp swarm algorithm, LSC-SSA does not increase the time cost. The time complexity analysis shows that the introduction of the improved strategy does not destroy the simplicity of the algorithm structure, nor does it increase the computational cost of the algorithm.

V. SIMULATION EXPERIMENTS AND RESULT ANALYSIS

A. FUNCTION OPTIMIZATION

Optimization is to find the optimal value among all possible values in a given searching range and output it in a minimized or maximized form. Without loss of generality, function optimization is considered as a constrained single-objective optimization problem. Therefore, function optimization has only one target value that needs to be outputted in a minimized form. The equation for the function optimization problem can be defined as:

$$Minimize: \ F(\vec{x}) = \{f_1(\vec{x})\} \tag{19}$$

$$Subject\ to: \ g_i(\vec{x}) > 0, \quad i=1,2,\ldots,m \tag{20}$$

$$h_i(\vec{x}) = 0, \quad i=1,2,\ldots,p \tag{21}$$

$$lb_i \leq x_i \leq ub_i, \quad i=1,2,\ldots,d \tag{22}$$

where, d is the number of variables; p is the number of equality constraints; m is the number of inequality constraints; lb_i indicates the lower bound of the i-th variable, and ub_i indicates the upper bound of the i-th variable.

In order to verify the optimization performance of the improved algorithm, this paper selected different algorithms for comparative experiments. The algorithm selected in the experiment and its parameter settings are shown in Table 1.

B. BENCHMARK FUNCTION

The simulation experiments adopted 34 test functions to evaluate the optimized performance of the improved algorithm LSC-SSA. These test functions can be divided into three categories: unimodal functions, multimodal functions, and combined functions. Among them, the function F1-F22 is the test function of CEC2005. As a classic test set, they can comprehensively evaluate the performance of the algorithm. In addition, functions F22-F34 are CEC2017 test functions. As the latest test functions, they can improve the quality of experiments. Functions $F_1 - F_7$ are unimodal functions. They only have a global optimal value. These functions are used to evaluate the local exploitation ability and convergence speed of the algorithm. Functions $F_8 - F_{13}$ are multimodal functions. Multimodal functions can produce multiple local optimal values in a continuous searching space. Therefore, the algorithm is prone to fall into a local optimum when solving multimodal functions, and the optimization process is challenging. At the same time, the number of local optimal values will increase as the problem size increases, which has important reference value for evaluating the global exploration capability of the algorithm. Functions $F_{14} - F_{22}$ are

TABLE 2. Benchmark functions and related information.

Function	Dim	Range	fmin
$F_1(x)=\sum_{i=1}^{n} x_i^2$	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
$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{j} x_j\right)^2$	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
$F_6(x)=\sum_{i=1}^{n}\left(\left[x_i+0.5\right]\right)^2$	30	[-10,10]	0
$F_7(x)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	30	[-1.28,1.28]	0
$F_8(x)=\sum_{i=1}^{n}-x_i^2\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[-500,500]	-418.9829×Dim
$F_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	30	[-5.12,5.12]	0
$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	30	[-32,32]	0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[-600,600]	0
$F_{12}(x)=\frac{\pi}{n}\left\{10\sin\left(\pi y_1\right)+\sum_{i=1}^{n}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]\right\}$ $+\sum_{i=1}^{n}u\left(x_i,10,100,4\right)$ $y_i=1+\frac{x_i+1}{4},u\left(x_i,a,k,m\right)=\begin{cases}k\left(x_i-a\right)^m & x_i>a\\ 0 & -a<x_i<a\\ k\left(-x_i-a\right)^m & x_i<-a\end{cases}$	30	[-50,50]	0
$F_{13}(x)=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5x_i\right)^2+\left(\sum_{i=1}^{n}0.5x_i\right)^4$	30	[-50,50]	0
$F_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right)^{-1}$	2	[-65,65]	1
$F_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[-5,5]	0.00030
$F_{16}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	[-5,5]	0.398
$F_{17}(x)=\left[1+\left(x_1+x_2+1\right)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]$ $\times\left[30+\left(2x_1-3x_2\right)^2\times\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$	2	[-2,2]	3
$F_{18}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	3	[1,3]	-3.86
$F_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	6	[0,1]	-3.32
$F_{20}(x)=-\sum_{i=1}^{5}\left[\left(X-a\right)_i\left(X-a\right)_i^2+c_i\right]^{-1}$	4	[0,10]	-10.1532

TABLE 2. ***(Continued.)*** **Benchmark functions and related information.**

$F_{21}(x)=-\sum_{i=1}^{7}\left[(X-a)_i(X-a)_i^2+c_i\right]^{-1}$	4	[0,10]	-10.4028
$F_{22}(x)=-\sum_{i=1}^{10}\left[(X-a)_i(X-a)_i^2+c_i\right]^{-1}$	4	[0,10]	-10.5363
$F_{23}(x)=x_1^2+10^6\sum_{i=1}^{n}x_i^2$	30	[-10,10]	0
$F_{24}(x)=\sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	30	[-100,100]	0
$F_{25}(x)=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5x_i\right)^2+\left(\sum_{i=1}^{n}0.5x_i\right)^4$	30	[-5,10]	0
$F_{26}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]+(x_n-1)^2\left[1+\sin^2(3\pi x_n)\right]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	[-30,30]	0
$F_{27}(x)=\sum_{i=1}^{n-1}\left[z_i^2-10\cos(2\pi z_{i+1})\ +10\right]$ $\hat{x}=M_1\frac{5.12(x-o)}{100},$ $y_i=\begin{cases}\hat{x}_i & if\ \lvert\hat{x}_i\le 5\rvert\\ round(2\hat{x}_i)/2 & if\ \lvert\hat{x}_i>5\rvert\end{cases}$ $for\quad i=1,2,3,\ldots n$	30	[-100,100]	0
$F_{28}(x)=\frac{\pi}{n}\begin{Bmatrix}10\sin(\pi y_1)+\\ \sum_{i=1}^{n}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]\end{Bmatrix}$ $+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4},$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	30	[-50,50]	0
$F_{29}(x)=\sum_{i=1}^{n-1}\left[100(x_i^2-x_{i+1})^2+(x_i-1)^2\right]$	30	[-30,30]	0
$F_{30}(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	[-10,10]	0
$F_{31}(x)=g(x_1,x_2)+g(x_2,x_3)+\ldots+g(x_D,x_1)$ $g(x,y)=0.5+\frac{\sin^2\left(\sqrt{x^2+y^2}\right)-0.5}{1+0.001\left(x^2+y^2\right)}$	30	[-10,10]	0
$F_{32}(x)=\min\left(\sum_{i=1}^{n}(x_i-u_0)^2,dD+s\sum_{i=1}^{n}(x_i-u_1)^2+10\left(D-\sum_{i=1}^{n}\cos(2\pi z_1)\right)\right)$	30	[-10,10]	0
$F_{33}(x)=\sum_{i=1}^{n-1}\left[z_i^2-10\cos(2\pi z_{i+1})\ +10\right]$	30	[-100,100]	0
$F_{34}(x)=\sin^2(\pi w_1)+\sum_{i=1}^{n}(w_i-1)^2\left[1+10\sin^2(\pi w_i)\right]+(w_D-1)^2\left[1+\sin^2(2\pi w_D)\right]$	30	[-50,50]	0

combined functions. The combination functions are generated by the benchmark function through rotation, shift, and offset. The dimension of the combination functions are small, so the optimization is not difficult. However, the global optimal value cannot be easily found after operations such as shift and offset. The combination functions are used to verify the

TABLE 3. **Comparison of simulation performance.**

Function	Performance	LSC-SSA	SSA	PSO	WOA	SCA	FA
	Best	**0**	5.53E-09	2.63E-13	6.27E-176	8.42E-06	5.40E-14
F1	Ave	**0**	8.82E-09	5.90E-11	8.20E-169	9.94E-03	7.96E-14
	Std	**0**	2.51E-09	1.47E-10	0	1.62E-02	1.23E-14
	Best	**0**	5.71E-03	1.30E-68	3.54E-180	2.79E-08	1.63E-11
F2	Ave	**0**	4.98E-01	1.50E-64	6.83E-163	1.10E-05	6.99E-11
	Std	**0**	4.78E-01	2.69E-64	2.22E-162	1.67E-05	3.57E-11
	Best	**0**	26.2952	3.0049	6.06E+03	3.49E+02	2.28E-13
F3	Ave	**0**	78.8897	6.4375	1.84E+04	2.31E+03	6.69E-02
	Std	**0**	54.9872	3.7229	8.56E+03	2.22E+03	1.34E-01
	Best	**0**	3.449	2.09E-01	1.11E-03	1.13E+01	1.08E-07
F4	Ave	**0**	5.4442	3.28E-01	2.82E+01	2.00E+01	1.15E-07
	Std	**0**	2.3375	9.50E-02	2.66E+01	6.4751	4.82E-09
	Best	2.68E+01	2.36E+01	1.77E+01	2.60E+01	2.89E+01	2.38E+01
F5	Ave	2.74E+01	1.28E+02	4.24E+01	2.67E+01	3.94E+01	2.49E+01
	Std	6.25E-01	1.60E+02	2.67E+01	3.00E-01	1.84E+01	7.76E-01
	Best	2.46E-12	6.42E-09	5.46E-13	3.86E-03	3.5375	7.41E-14
F6	Ave	2.95E-01	1.03E-08	9.81E-11	9.46E-03	4.7038	8.30E-14
	Std	1.62E-01	2.55E-09	1.56E-10	7.40E-03	1.2479	7.15E-15
	Best	**3.43E-07**	2.94E-02	95.7908	4.23E-04	9.78E-03	2.26E-02
F7	Ave	**1.06E-05**	7.35E-02	2.39E+02	2.14E-03	3.02E-02	2.97E-02
	Std	**1.11E-05**	3.33E-02	1.27E+02	2.26E-03	2.64E-02	4.92E-03
	Best	**-12569.4781**	-8687.6661	-7334.0689	-12569.3488	-4927.4869	-10431.5066
F8	Ave	**-12569.2851**	-7429.8211	-4752.1882	-11930.1124	-4037.7271	-9944.6651
	Std	**1.81E-01**	7.79E+02	1.23E+03	1.05E+03	3.30E+02	4.39E+02
	Best	**0**	2.98E+01	18.9043	0	1.37E-04	27.8588
F9	Ave	**0**	6.07E+01	54.5318	0	1.32E+01	39.7983
	Std	**0**	1.94E+01	22.215	0	2.28E+01	6.7968
	Best	**8.88E-16**	2.19E-05	7.45E-07	8.88E-16	0.00019138	6.15E-08
F10	Ave	**8.88E-16**	1.5547	1.14E-05	3.02E-15	1.37E+01	6.47E-08
	Std	**0**	8.25E-01	9.25E-06	2.48E-15	9.5124	2.39E-09
	Best	**0**	4.19E-08	2.59E-14	0	4.40E-04	1.05E-13
F11	Ave	**0**	1.65E-02	4.19E-03	0	2.12E-01	2.71E-03
	Std	**0**	1.34E-02	5.45E-03	0	1.80E-01	4.42E-03
	Best	7.11E-13	4.02E-01	2.99E-14	3.11E-04	3.36E-01	1.61E-16
F12	Ave	1.82E-02	4.245	1.04E-02	3.66E-03	6.88E-01	2.06E-16
	Std	1.83E-02	3.2576	3.28E-02	5.54E-03	3.97E-01	2.86E-17
	Best	**0**	4.63E+05	3.21E-18	8.07E-276	8.15E-07	3.15E+07
F13	Ave	**0**	9.74E+08	2.18E-11	4.18E-237	9.50E+10	1.34E+11
	Std	**0**	1.51E+09	6.84E-11	0	3.00E+11	2.29E+11
	Best	0.998	0.998	0.998	0.998	0.998	0.998
F14	Ave	7.0403	0.998	2.1846	2.5698	1.3949	0.998
	Std	5.6087	1.28E-16	1.7852	3.027	8.37E-01	1.48E-16

TABLE 3. *(Continued.)* **Comparison of simulation performance.**

F15	Best	3.07E-04	6.21E-04	3.07E-04	3.08E-04	5.21E-04	3.07E-04
	Ave	3.65E-04	2.91E-03	6.00E-04	6.09E-04	7.88E-04	5.81E-04
	Std	8.74E-05	6.14E-03	3.31E-04	3.01E-04	2.51E-04	1.48E-04
F16	Best	0.39789	0.39789	0.39789	0.39789	0.39813	0.39789
	Ave	0.39789	0.39789	0.39789	0.39789	0.39868	0.39789
	Std	5.09E-14	3.01E-14	0	2.28E-07	3.96E-04	0
F17	Best	3	3	3	3	3	3
	Ave	3	3	3	3	3	3
	Std	9.67E-05	4.70E-14	9.00E-16	8.03E-06	1.51E-05	8.24E-16
F18	Best	-3.8626	-3.8628	-3.8628	-3.8628	-3.8548	-3.8628
	Ave	-3.8591	-3.8628	-3.8604	-3.86	-3.8544	-3.8628
	Std	3.64E-03	2.37E-14	3.81E-03	3.19E-03	4.07E-04	4.44E-16
F19	Best	-3.322	-3.322	-3.322	-3.3219	-3.1981	-3.322
	Ave	-3.1689	-3.2377	-3.2313	-3.2944	-2.9925	-3.2863
	Std	7.57E-02	5.82E-02	8.64E-02	5.76E-02	2.74E-01	5.74E-02
F20	Best	-10.1532	-10.1532	-10.1532	-10.1531	-5.8208	-10.1532
	Ave	-10.1532	-10.1532	-7.8904	-9.6427	-2.2504	-10.1532
	Std	5.72E-11	3.23E-11	3.0034	1.6119	2.0658	1.87E-15
F21	Best	-10.4029	-10.4029	-10.4029	-10.4028	-6.328	-10.4029
	Ave	-10.4029	-10.4029	-9.2077	-9.2029	-4.6722	-10.4029
	Std	8.85E-11	2.48E-11	2.5415	2.5486	8.71E-01	2.37E-15
F22	Best	-10.5364	-10.5364	-10.5364	-10.5364	-5.5242	-10.5364
	Ave	-10.5364	-9.7634	-10.0003	-9.3139	-4.472	-10.5364
	Std	7.27E-11	2.44E+00	1.6952	2.5824	1.33E+00	2.05E-15
F23	Best	**0**	6.79E-03	7.65E-07	1.18E-176	7.11E-03	6.1566
	Ave	**0**	3.5795	4.00E+01	9.35E-164	7.23E-01	4.47E+01
	Std	**0**	4.4155	5.16E+01	0	1.0726	2.56E+01
F24	Best	**0**	9.62E+04	5.92E-19	2.03E-264	8.50E-05	1.10E+09
	Ave	**0**	2.37317E+12	1.73E-13	1.07E-237	2.86E+03	2.28818E+12
	Std	**0**	6.68563E+12	5.44E-13	0	7.07E+03	4.81E+12
F25	Best	**0**	4.29E-03	3.3809	3.10E+02	3.2531	8.23E-16
	Ave	**0**	7.12E-02	1.25E+02	4.59E+02	7.8958	1.23E-15
	Std	**0**	8.93E-02	1.09E+02	1.11E+02	3.8045	2.17E-16
F26	Best	**0**	7.31E+05	3.20E-08	9.20E-172	7.25E-04	6.85E+05
	Ave	**0**	3.34E+06	1.85E-04	6.42E-163	4.92E-01	1.04E+06
	Std	**0**	1.99E+06	5.82E-04	2.22E-162	1.17E+00	2.43E+05
F27	Best	**0**	4.16E+03	1.48E-12	2.42E-177	1.90E-07	3.49E+04
	Ave	**0**	1.14E+04	7.98E-11	1.13E-162	1.79E-03	4.03E+04
	Std	**0**	6.97E+03	1.22E-10	3.14E-162	2.37E-03	2.22E+03
F28	Best	**0**	8.94E-01	1.25E-01	9.60E-68	1.44E-04	4.18E-03
	Ave	**0**	1.2246	2.81E-01	1.04E-64	5.72E-03	1.28E-02
	Std	**0**	1.65E-01	1.44E-01	2.35E-64	8.59E-03	9.05E-03
	Best	2.62E+01	2.44E+01	2.25E+01	2.60E+01	2.75E+01	2.48E+01

TABLE 3. ***(Continued.)*** **Comparison of simulation performance.**

F29	Ave	2.73E+01	3.85E+01	5.95E+01	2.66E+01	2.89E+01	2.53E+01
	Std	6.38E-01	2.27E+01	2.91E+01	3.30E-01	8.87E-01	4.02E-01
	Best	**0**	3.88E+01	2.89E+01	0	3.62E-04	2.09E+01
F30	Ave	**0**	57.21	4.78E+01	0	4.90E+00	3.32E+01
	Std	**0**	1.09E+01	1.38E+01	0	1.13E+01	6.26E+00
	Best	3.92E-12	8.80E-09	1.35E-12	4.92E-03	4.11E+00	7.46E-14
F31	Ave	0.13497	1.02E-08	2.47E-10	1.23E-02	4.35E+00	8.33E-14
	Std	0.15157	9.38E-10	4.20E-10	7.28E-03	2.88E-01	6.74E-15
	Best	2.62E+01	2.52E+01	1.88E+01	2.64E+01	2.77E+01	2.42E+01
F32	Ave	2.69E+01	45.0632	3.56E+01	2.68E+01	3.78E+01	2.50E+01
	Std	5.96E-01	3.94E+01	3.48E+01	2.35E-01	1.83E+01	4.37E-01
	Best	**0**	1.72E+02	2.40E+01	0	1.49E+00	3.00E+01
F33	Ave	**0**	275.225	3.73E+01	0	7.83E+01	4.37E+01
	Std	**0**	5.93E+01	8.57E+00	0	6.68E+01	1.10E+01
	Best	1.73E-01	1.36E+00	1.71E-13	3.90E-03	2.04E+00	3.56E-16
F34	Ave	6.24E-01	6.3488	4.54E-02	1.51E-01	2.25E+00	3.99E-16
	Std	4.96E-01	2.82E+00	1.44E-01	2.33E-01	1.84E-01	3.13E-17

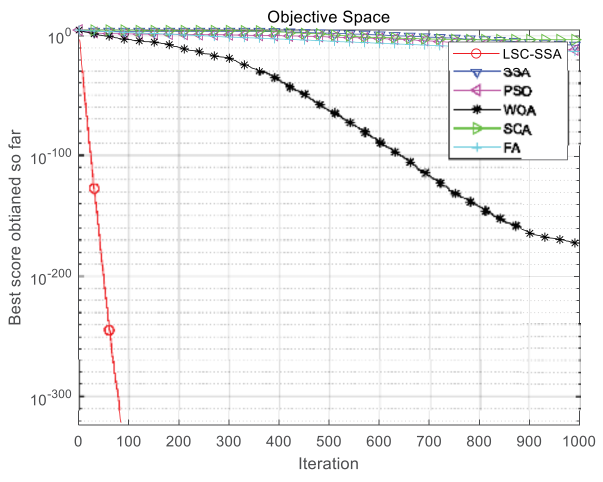
optimization accuracy of the algorithm. The specific information of the test functions are listed in Table 2.

C. COMPARISON EXPERIMENTS AND ANALYSIS WITH OTHER ALGORITHMS

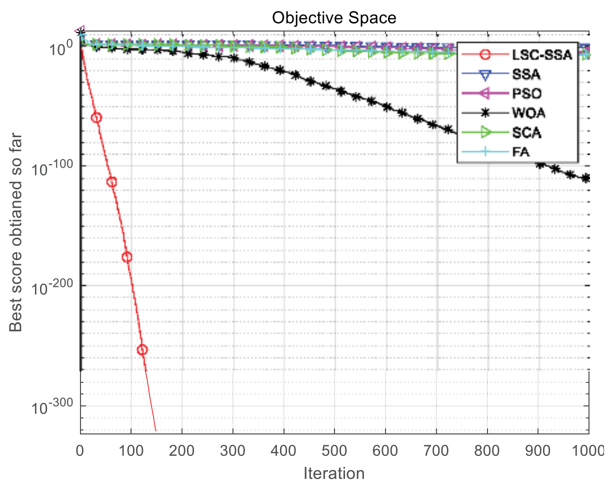
In order to verify the optimization effect of algorithm, the basic salp swarm algorithm (SSA), particle swarm optimization (PSO) algorithm, whale optimization algorithm (WOA), sine cosine algorithm (SCA), firefly algorithm (FA) and the improved algorithm proposed in this paper (LSC-SSA) were selected for carrying out the optimization comparison experiments. The algorithms set uniform parameters, and each test function runs independently 60 times. The test function convergence curves are shown in Fig. 11. The perfromance results are listed in Table 3.

The simulation results in Fig. 11 show that, except for a few functions, the optimization effect of LSC-SSA on most functions has obvious advantages. For functions F1-F5, F7-F11, F13-F15 and F23-F28, the optimization accuracy and convergence speed of LSC-SSA are significantly better than other algorithms. LSC-SSA has an advantage in 68% of functions, and has the best optimization performance in this experiment. For the unimodal functions F1-F7, LSC-SSA is only inferior to FA on function F6, and the optimal performance of other functions is the best. It shows that the improved algorithm has strong local exploitation ability and can quickly converge to the target value. For multi-modal functions F8-F13 with a large number of local optimal values, LSC-SSA has obvious advantages and can achieve higher optimization accuracy in a shorter number of iterations. LSC-SSA is inferior to PSO and FA only on function F12. This shows that LSC-SSA has a strong global exploration capability, the search agent can avoid falling into a local optimal value and develop to a promising area in the search space. For compound function F14-F22, LSC-SSA's optimization advantage on F13-F15 is the best. Among them, the problem dimensions of F16-F19 are small, so the performance difference between algorithms is not obvious. For functions F20-F22, the improved algorithm is only better than SSA and SCA, but the optimization accuracy is not inferior to other algorithms. For functions F23-28, the improved algorithm has obvious advantages. The simulation results of three types of test functions show the advantages of the proposed in global convergence and optimization accuracy.

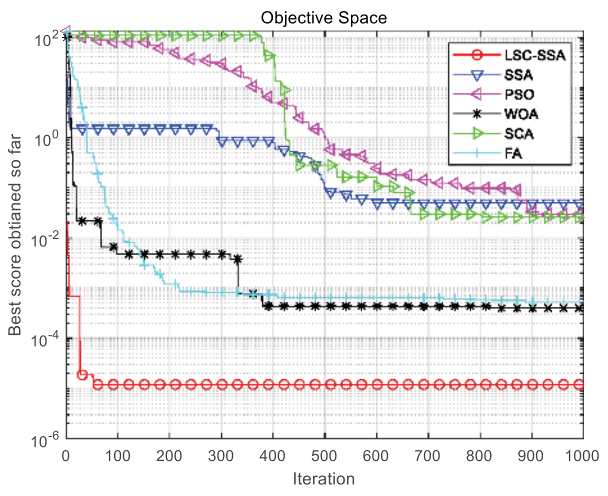
Three criteria listed in Table 3 are the optimal value, average value, and variance, which are used to evaluate the optimization accuracy, average accuracy, and stability of the algorithm. As can be seen from the optimization accuracy in the table, the optimization accuracy of the proposed is significantly better than other algorithms. The improved algorithm found the theoretical optimal value on 17 functions (F1-F4, F8, F9, F11, F13, F17, F20, F22-F28), accounting for 61%, which is the algorithm with the highest precision in this experiment. In the remaining functions, the optimization accuracy of LSC-SSA is not much different from the theoretical optimal value. In average accuracy and stability, LSC-SSA can also maintain obvious advantages. It shows that the proposed LSC-SSA is not easy to be affected by randomness, has good robustness, and can stably maintain the optimization accuracy. It is worth mentioning that the



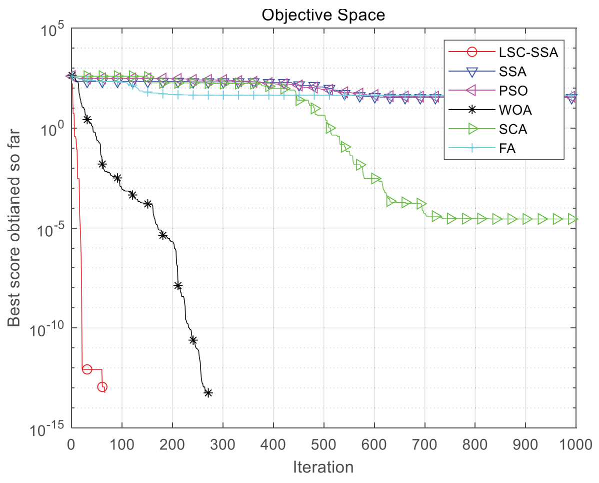
(a) Convergence curves of F_1



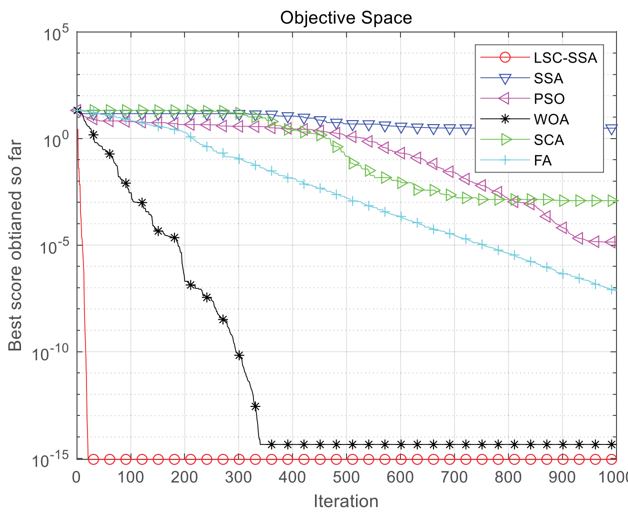
(b) Conv ergence curves of F_2



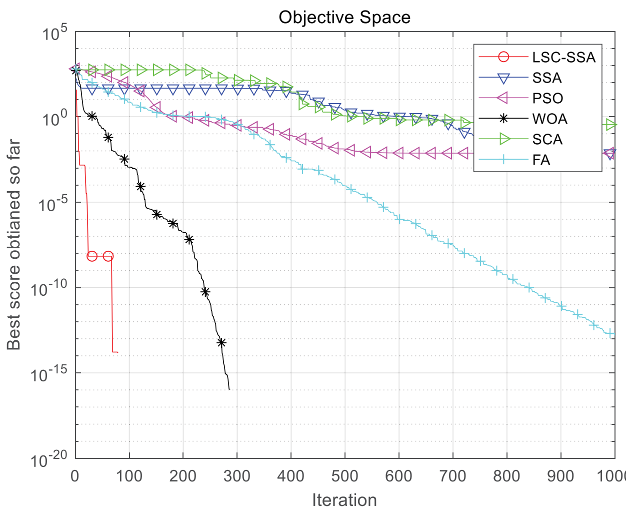
(c) Convergence curves of F_7



(d) Convergence curves of F_9

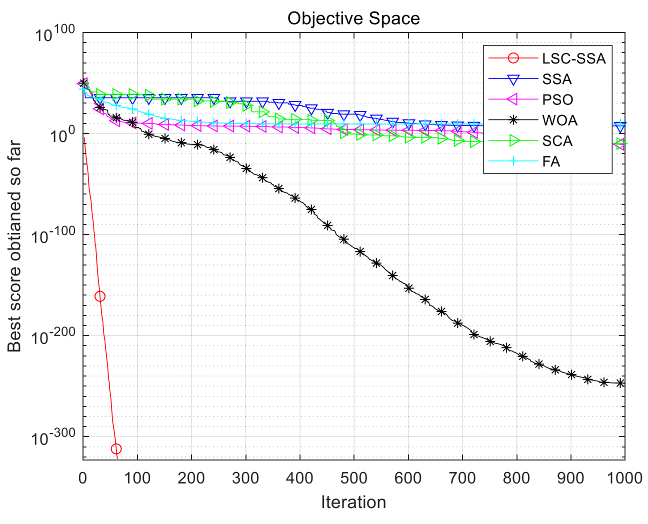


(e) Convergence curves of F_{10}

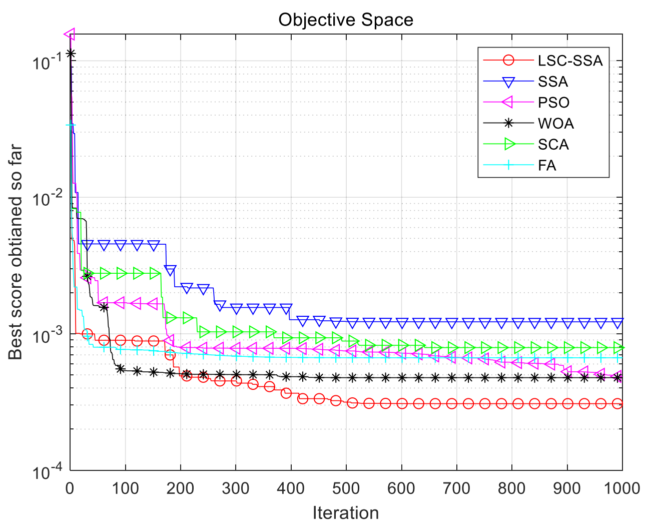


(f) Convergence curves of F_{11}

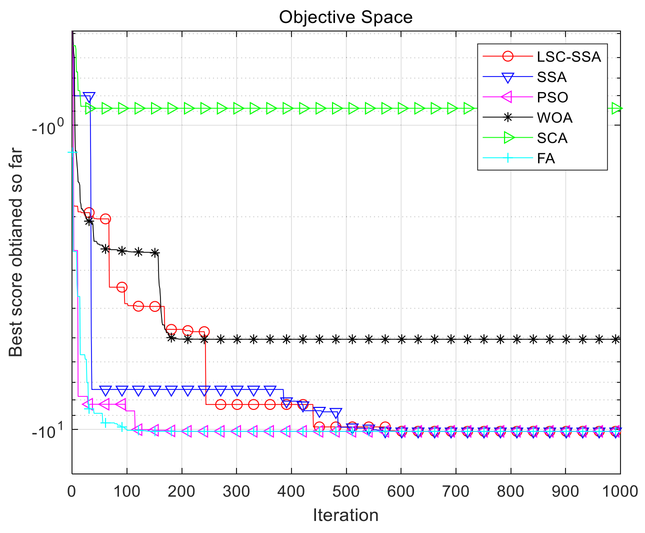
FIGURE 11. Simulation experiment results.



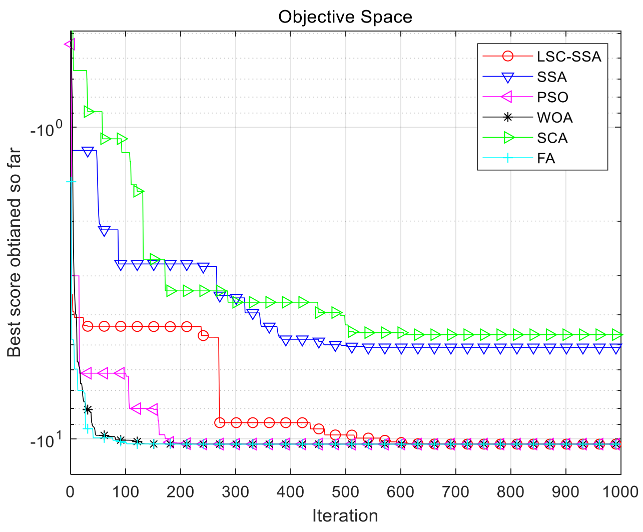
(g) Convergence curves of F_{13}



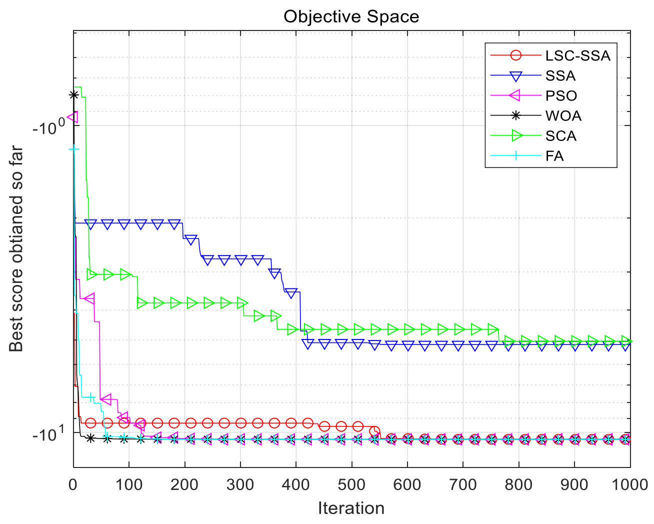
(h) Convergence curves of F_{15}



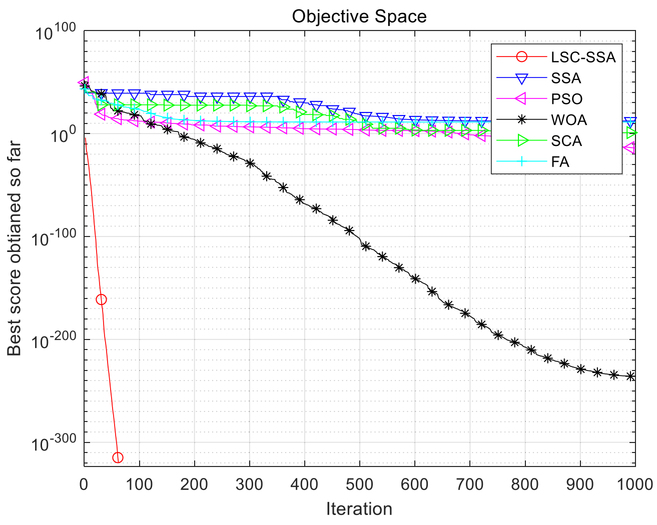
(i) Convergence curves of F_{20}



(j) Convergence curves of F_{21}

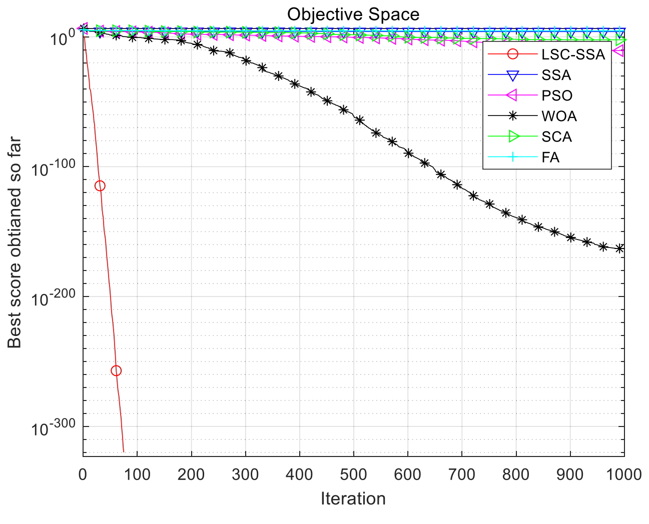


(k) Convergence curves of F_{22}

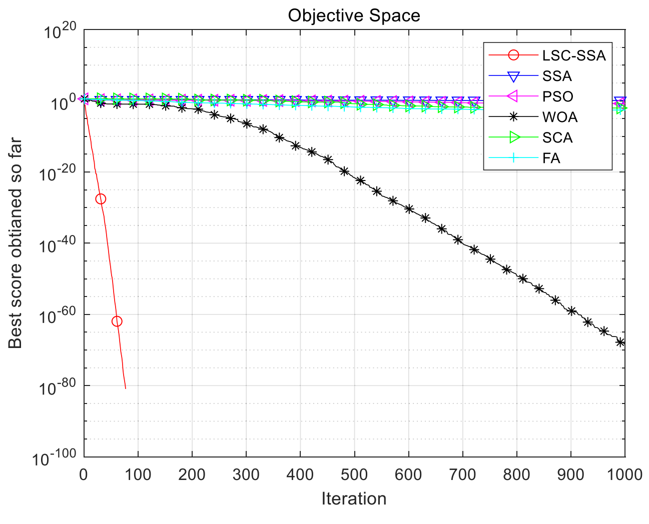


(l) Convergence curves of F_{24}

FIGURE 11. ***(Continued.)* Simulation experiment results.**



(m) Convergence curves of F_{27}



(n) Convergence curves of F_{28}

FIGURE 11. ***(Continued.)*** **Simulation experiment results.**

optimized performance of WOA is second only to LSC-SSA. Strong performance will bring more development to the algorithm. It can be expected that the whale optimization algorithm will be improved and applied to more optimization problems. It is worth mentioning that this paper uses the p value of wilcoxon rank sum test to test the performance difference between the two algorithms. When the p-value is less than 0.05, there is a significant performance difference between the two algorithms. When the p-value is greater than 0.05, there is no significant difference between the two algorithms. If the p-value result is NAN, there is no difference between the two algorithms. The test results in Table 4 show that the p value of the improved algorithm on most functions is less than 0.05, which shows that LSC-SSA has obvious advantages over other algorithms.

The optimization process of the meta-heuristic algorithm are global exploration and local exploitation. In the global exploration stage, search agents are distributed as widely as possible in the searching space and a set of random solutions are obtained. In the local exploitation phase, the algorithm will continuously improve this set of random solutions to make them develop towards global optimal values. Simulation experiments verify that LSC-SSA is more competitive than other algorithms in solving function optimization problems, indicating that the improved strategy proposed in this paper effectively improves the performance of the algorithm. First of all, the Levy flight mechanism enables the search agent to avoid falling into a local optimal value and improves the global exploration capability of the algorithm. Secondly, the introduction of improved sine cosine operator allows the algorithm to adaptively adjust the search area based on the return value of the solution space, effectively balancing exploration and exploitation. The improved strategy improves the optimization accuracy, global convergence and stability of the algorithm. The effectiveness analysis of the improvement strategy is explained in details in Section E.

D. EFFECTIVENESS ANALYSIS OF SOLVING HIGH DIMENSION FUNCTION OPTIMIZATION PROBLEM

High-dimensional, large-scale, and high-noise features are common in practical optimization problems, which make the searching space complex and difficult to optimize. In order to verify the possibility of the improved algorithm to solve practical problems and expand the theoretical research of the algorithm, this paper applies LSC-SSA to simulation experiments of high-dimensional function optimization problems. Functions $F_{14}-F_{22}$ are fixed-dimensional functions, and they are not allowed to change the number of variables in the solution space. Therefore, for functions F_1-F_{13}, the experiments increase the test function's dimensions $D = 30$ to $D = 100$, $D = 200$, and $D = 300$ to evaluate the effectiveness of the improved algorithm for solving high-dimensional large-scale optimization problems. The experimental results are listed in Table 5. In addition, Table 6 shows the p-value results of wilcoxon rank sum test.

In general, the calculation size and complexity of the test functions will increase as the dimensions increase. The expansion of the searching space and the increase of the local optimal value will reduce the probability of the algorithm finding the global optimal value. Therefore, the high-dimensional optimization process is full of challenges. Especially for multimodal functions $F_8 - F_{13}$, with multiple local optimal values increasing as the dimensions increase, search agent tend to fall into local optimal. Too many local optimal values hinder the algorithm's global explorability, which causes a dimensional disaster for large-scale problems. The experimental results show that LSC-SSA can maintain its advantages in optimization accuracy, mean

TABLE 4. **P-value results of wilcoxon rank sum test.**

Function	LSC-SSA vs SSA	LSC-SSA vs PSO	LSC-SSA vs WOA	LSC-SSA vs SCA	LSC-SSA vs FA
F1	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F2	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F3	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F4	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F5	4.27E-01	4.73E-01	2.11E-02	3.30E-04	3.30E-04
F6	4.73E-01	1.21E-01	5.71E-01	1.83E-04	1.83E-04
F7	1.83E-04	1.83E-04	1.83E-04	1.83E-04	1.83E-04
F8	1.83E-04	1.83E-04	3.08E-01	1.83E-04	2.20E-03
F9	6.39E-05	6.39E-05	NaN	6.39E-05	6.39E-05
F10	6.39E-05	6.39E-05	2.10E-03	6.39E-05	6.39E-05
F11	6.39E-05	6.39E-05	NaN	6.39E-05	6.39E-05
F12	1.83E-04	1.83E-04	1.62E-01	1.83E-04	1.83E-04
F13	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F14	6.08E-04	6.40E-03	2.90E-02	2.90E-02	6.08E-04
F15	1.83E-04	5.80E-04	5.80E-04	2.46E-04	2.57E-02
F16	9.95E-04	6.34E-05	1.82E-04	1.82E-04	6.34E-05
F17	1.79E-04	6.34E-05	9.70E-01	1.21E-01	6.34E-05
F18	1.54E-04	6.34E-05	2.41E-01	2.20E-03	6.34E-05
F19	4.73E-01	6.94E-02	1.41E-01	5.83E-04	0.0499
F20	7.91E-01	4.66E-01	1.83E-04	1.83E-04	6.39E-05
F21	3.12E-02	2.12E-02	1.83E-04	1.83E-04	6.39E-05
F22	1.41E-01	6.39E-05	1.83E-04	1.83E-04	6.39E-05
F23	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F24	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F25	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F26	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F27	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F28	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F29	1.41E-01	9.70E-01	2.20E-03	1.41E-01	1.83E-02
F30	6.39E-05	7.51E-04	6.39E-05	6.39E-05	6.39E-05
F31	1.83E-04	2.41E-01	1.83E-04	1.83E-04	1.83E-04
F32	9.70E-01	2.20E-02	1.41E-01	2.67E-03	4.40E-03
F33	6.39E-05	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F34	1.83E-04	9.10E-03	1.83E-04	1.83E-04	1.83E-04

value, and variance, and is not much different from the performance of low-dimensional (30-dimensional) test functions. On $D = 100$, $D = 200$ and $D = 300$, LSC-SSA found the theoretical optimal value (0) of 8 test functions (F1-F4, F8, F9, F11, F13), accounting for 61%. The optimal value of the function F8 will shift with the dimension. The improved algorithm can still find the theoretical optimal value of F8, which shows that the algorithm can effectively avoid the local optimal. For functions F7, F10, and F12, the improved algorithm does not fall into a dimensional disaster, and can maintain the advantages of low-dimensional functions. For functions F5 and F6, the optimization capability of LSC-

TABLE 5. Simulation comparison of high-dimensional optimization.

Function	Performance	LSC-SSA		
		D=100	**D=200**	**D=300**
F1	Best	0	0	0
	Ave	0	0	0
	Std	0	0	0
F2	Best	0	0	0
	Ave	0	0	0
	Std	0	0	0
F3	Best	0	0	0
	Ave	0	0	0
	Std	0	0	0
F4	Best	0	0	0
	Ave	0	0	0
	Std	0	0	0
F5	Best	9.85E+01	1.98E+02	2.98E+02
	Ave	1.02E+02	2.05E+02	3.09E+02
	Std	3.0907	1.63E+01	1.60E+01
F6	Best	1.7884	4.6513	8.4206
	Ave	3.5655	1.18E+01	2.28E+01
	Std	1.4016	3.8371	6.359
F7	Best	5.75E-03	9.93E-02	5.73E-03
	Ave	3.57E-01	5.30E-01	3.32E-01
	Std	3.34E-01	3.22E-01	2.75E-01
F8	Best	-32704.2898	-36064.7093	-63074.4646
	Ave	2.64E+03	2.84E+03	-2.56E+04
	Std	1.64E+04	3.13E+04	4.29E+04
F9	Best	0	0	0
	Ave	0	0	0
	Std	0	0	0
F10	Best	8.88E-16	8.88E-16	8.88E-16
	Ave	1.95E-15	1.95E-15	1.60E-15
	Std	1.72E-15	1.72E-15	1.50E-15
F11	Best	0	0	0
	Ave	0	0	0
	Std	0	0	0
F12	Best	2.00E-02	2.28E-02	4.70E-02
	Ave	5.50E-02	9.61E-02	1.38E-01
	Std	4.28E-02	3.75E-02	7.41E-02
F13	Best	0	0	0
	Ave	0	0	0
	Std	0	0	0

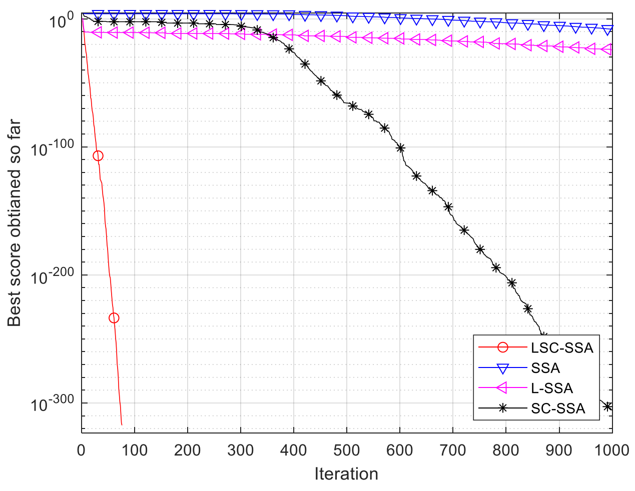
TABLE 6. P-value results of wilcoxon rank sum test.

Function	D=100 vs D=30	D=200 vs D=30	D=300 vs D=30
F1	NaN	NaN	NaN
F2	NaN	NaN	NaN
F3	NaN	NaN	NaN
F4	NaN	NaN	NaN
F5	1.86E-01	1.52E-01	3.46E-01
F6	8.90E-02	3.12E-02	3.50E-04
F7	1.86E-01	1.75E-01	1.31E-01
F8	1.83E-04	1.83E-04	1.83E-04
F9	NaN	NaN	NaN
F10	NaN	NaN	NaN
F11	NaN	NaN	NaN
F12	1.62E-01	4.52E-02	5.71E-01
F13	NaN	NaN	NaN

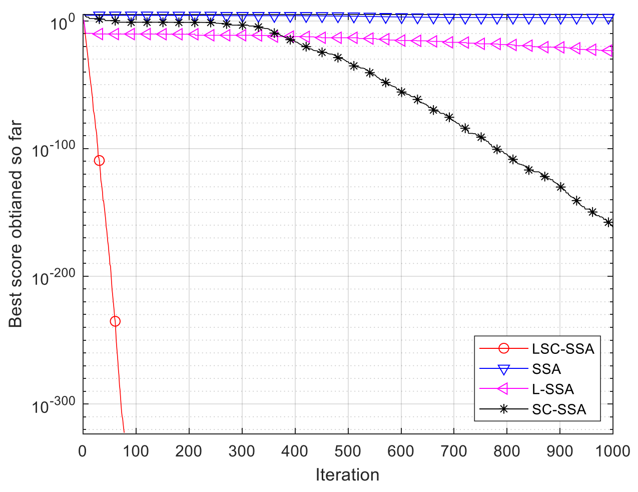
SSA will decrease as the dimension increases. The test results in Table 6 show that the performance of the algorithm in high-dimensional optimization is not significantly different from 30-dimensional, which shows that LSC-SSA will not fall into dimensional disaster. The experimental results verify that LSC-SSA can effectively avoid falling into dimensional disaster, and the optimization performance of solving high-dimensional functions is strong. The improved algorithm can still maintain strong optimization accuracy and robustness when solving large-scale problems, which lays a theoretical foundation for the application of LSC-SSA to practical problems.

E. PROBLEM EFFECTIVENESS ANALYSIS OF IMPROVED STRATEGIES

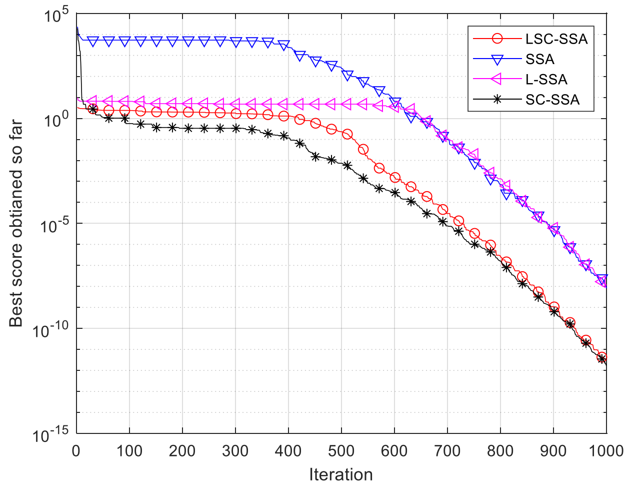
This paper makes two improvements to the basic strategy of the salp swarm algorithm. First, the introduction of the Levy flight mechanism with a step size control factor increased the population ergodicity and global explorability. Second, the position of leader is updated by improved sine cosine operator, so that the algorithm can adaptively transition between global exploration and local exploitation. The function optimization experiments in Section C have verified that the improved strategy can improve the performance of the algorithm. In order to further evaluate the effectiveness of the improved strategy, the improved algorithm (LSC-SSA), the salp swarm algorithm that only introduces Levy flight (L-SSA), the salp swarm algorithm that only introduces sine cosine operator (SC-SSA) and salp swarm algorithm (SSA) are selected for comparison experiments. The parameter settings of the algorithm are the same as in Table 1. Select functions F1-F22 from Table 2. Each function runs independently 60 times. The experimental results are listed in Table 7, and the convergence curves of some functions are shown in Fig. 12.



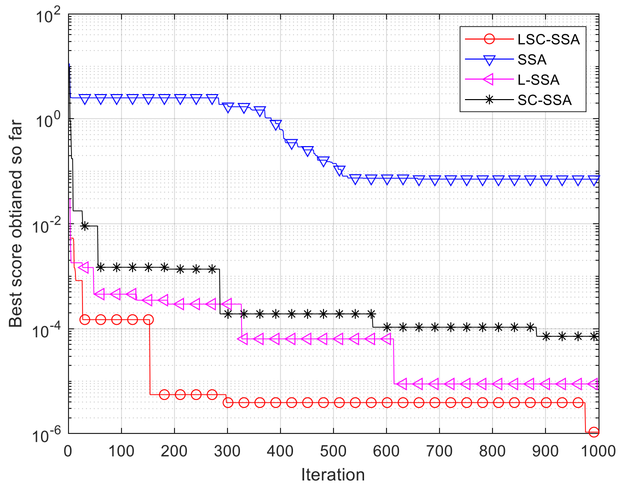
(a) Convergence curves of F_1



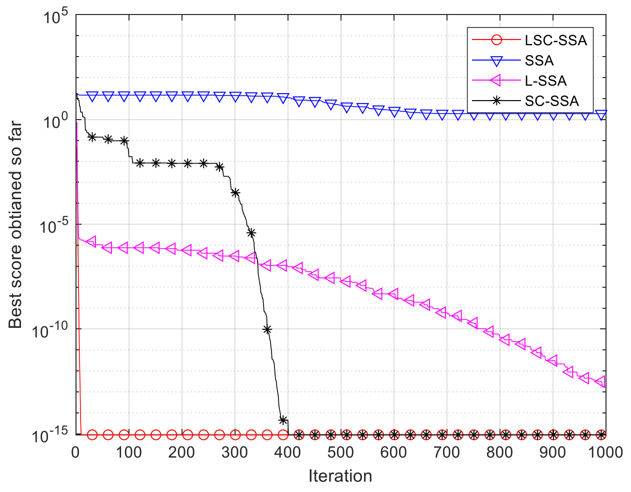
(b) Convergence curves of F_3



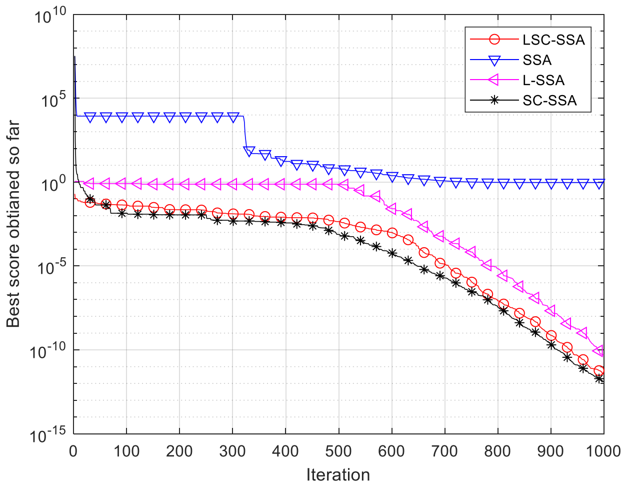
(c) Convergence curves of F_6



(d) Convergence curves of F_7



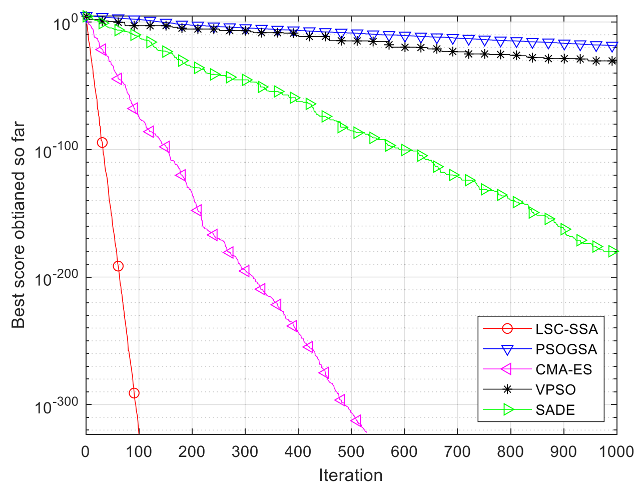
(e) Convergence curves of F_{10}



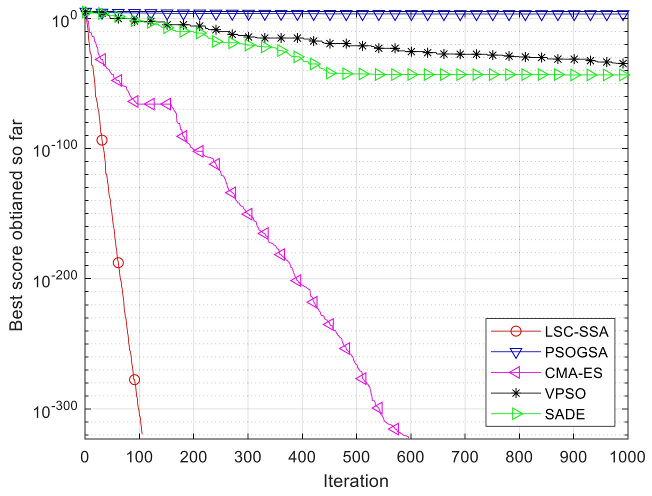
(f) Convergence curves of F_{12}

FIGURE 12. Simulation experiment results.

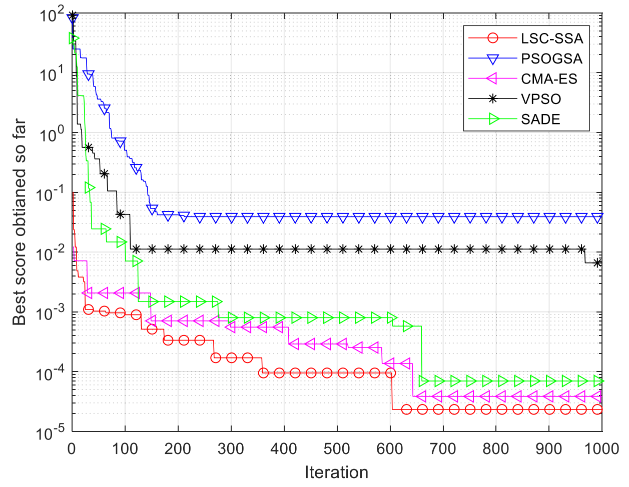
The convergence curves show that LSC-SSA is inferior to L-SSA only on F12. For the function F5, there is no difference in the optimization precision and convergence speed between LSC-SSA and L-SSA. For function F22, the convergence speed of LSC-SSA is inferior to SSA. On the other functions, LSC-SSA has the best optimization effect. The optimization



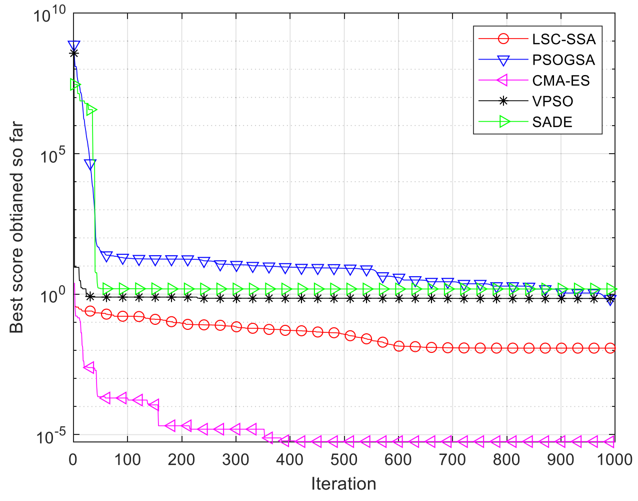
(a) Convergence curves of F_1



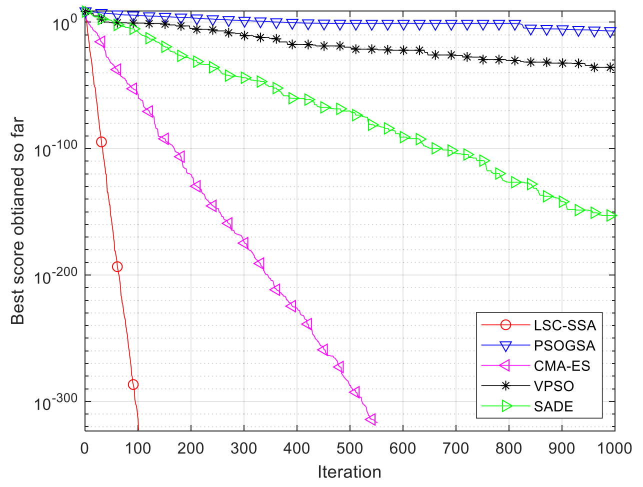
(b) Convergence curves of F_3



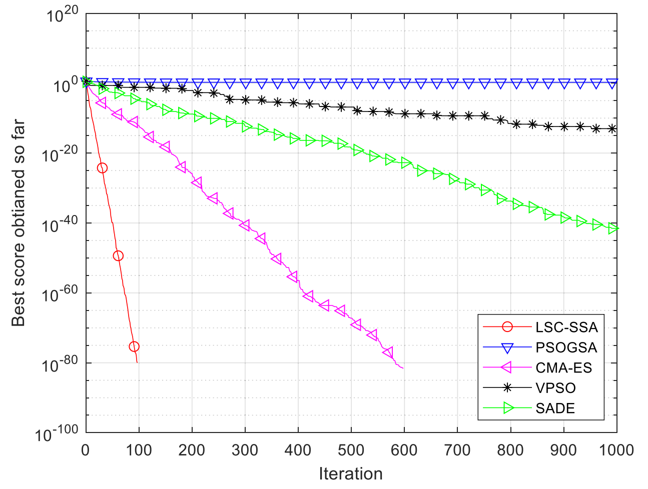
(c) Convergence curves of F_7



(d) Convergence curves of F_{12}



(e) Convergence curves of F_{23}



(f) Convergence curves of F_{28}

FIGURE 13. Simulation experiment results.

accuracy in Table 7 shows that the proportion of LSC-SSA finding the theoretical optimal value is 50%, the proportion of L-SSA finding the theoretical optimal value is 14%, and the proportion of SSA finding the theoretical optimal value is only 9%. LSC-SSA has the best optimization accuracy. Combining the performance of average accuracy and stability,

TABLE 7. Simulation comparison of improved strategies.

Function		LSC SSA	LSSA	SSA	SC-SSA
	Best	0	2.29E-25	5.53E-09	4.78E-279
F1	Ave	0	4.32E-25	8.82E-09	3.50E-260
	Std	0	1.80E-25	2.51E-09	0
	Best	0	1.63E-13	5.71E-03	1.51E-144
F2	Ave	0	2.35E-13	4.98E-01	9.40E-136
	Std	0	3.86E-14	4.78E-01	2.86E-135
	Best	0	8.86E-25	26.2952	1.22E-189
F3	Ave	0	2.29E-24	78.8897	1.94E-151
	Std	0	9.50E-25	54.9872	6.14E-151
	Best	0	2.47E-13	3.449	7.50E-143
F4	Ave	0	3.55E-13	5.4442	1.17E-126
	Std	0	6.99E-14	2.3375	2.32E-126
	Best	2.68E+01	2.78E+01	2.36E+01	2.58E+01
F5	Ave	2.74E+01	2.80E+01	1.28E+02	2.70E+01
	Std	6.25E-01	1.21E-01	1.60E+02	7.31E-01
	Best	2.46E-12	1.48E-21	3.08E-01	6.64E-13
F6	Ave	2.95E-01	4.65E-21	1.75E+01	1.94E-12
	Std	1.62E-01	2.60E-21	1.89E+01	8.04E-13
	Best	3.43E-07	3.20E-06	2.94E-02	1.28E-05
F7	Ave	1.06E-05	1.44E-05	7.35E-02	9.48E-05
	Std	1.11E-05	1.38E-05	3.33E-02	7.27E-05
	Best	-1.26E+04	-8.48E+03	-8.69E+03	-1.26E+04
F8	Ave	-1.26E+04	-7.93E+03	-7.43E+03	-1.26E+04
	Std	1.81E-01	4.95E+02	7.79E+02	1.01E-08
	Best	0	0	2.98E+01	0
F9	Ave	0	0	6.07E+01	0
	Std	0	0	1.94E+01	0
	Best	8.88E-16	1.39E-13	2.19E-05	8.88E-16
F10	Ave	8.88E-16	1.83E-13	1.5547	8.88E-16
	Std	0	2.69E-14	8.25E-01	0
	Best	0	0	4.19E-08	0
F11	Ave	0	0	1.65E-02	0
	Std	0	0	1.34E-02	0
	Best	7.11E-13	4.93E-11	4.02E-01	5.82E-13
F12	Ave	1.82E-02	9.64E-11	4.245	1.31E-12
	Std	1.83E-02	3.14E-11	3.2576	7.20E-13
	Best	0	3.31E-38	5.25E+07	6.74E-10
F13	Ave	0	1.05E-30	1.65857E+11	1.34E-08
	Std	0	1.32E-30	3.69938E+11	6.07E-13
	Best	0.998	0.998	0.998	0.998
F14	Ave	7.0403	0.998	0.998	0.998
	Std	5.6087	1.96E-16	1.28E-16	1.48E-16
	Best	3.07E-04	3.07E-04	6.21E-04	3.07E-04
F15	Ave	3.65E-04	3.40E-04	2.91E-03	3.08E-04
	Std	8.74E-05	1.02E-04	6.14E-03	8.52E-07
	Best	0.39789	0.39789	0.39789	0.39789
F16	Ave	0.39789	0.39789	0.39789	0.39789
	Std	5.09E-14	8.43E-14	3.01E-14	5.48E-14
	Best	3	3	3	3
F17	Ave	3	3	3	3.0001
	Std	9.67E-05	2.85E-13	4.70E-14	4.82E-05
	Best	-3.8626	-3.8628	-3.8628	-3.8549
F18	Ave	-3.8591	-3.8628	-3.8628	-3.8549
	Std	3.64E-03	1.64E-14	2.37E-14	8.24E-16
	Best	-3.322	-3.322	-3.322	-3.322
F19	Ave	-3.1689	-3.2145	-3.2377	-3.1938
	Std	7.57E-02	3.78E-02	5.82E-02	1.06E-01
	Best	-10.1532	-5.0552	-10.1532	-10.1532
F20	Ave	-10.1532	-5.0552	-10.1532	-10.1532
	Std	5.72E-11	8.21E-12	3.23E-11	2.01E-12
	Best	-10.4029	-5.0877	-10.4029	-10.4029
F21	Ave	-10.4029	-5.0877	-10.4029	-10.4029
	Std	8.85E-11	6.92E-12	2.48E-11	7.20E-13
	Best	-10.5363	-5.1285	-10.5364	-10.5364
F22	Ave	-10.5364	-5.1285	-8.4608	-10.5364
	Std	8.42E-11	8.35E-12	3.402	1.23E-12

TABLE 7. *(Continued.)*Simulation comparison of improved strategies.

it can be seen that LSC-SSA has better global convergence and robustness than L-SSA and SSA.

In addition, Table 8 shows the p-value results of wilcoxon rank sum test. The statistical results show that the introduction of two improved strategies has more obvious advantages than the other cases, which shows that Levy flight and improved sine cosine operator have a synergistic effect on the improvement of algorithm performance. Experiments show that the improved sine cosine operator introduced on the basis of Levy flight can significantly improve the optimization performance of the salp swarm algorithm. The effectiveness of the diversified improvement strategies proposed in this paper is verified.

F. COMPARISON EXPERIMENTS AND ANALYSIS WITH OTHER IMPROVED ALGORITHMS

In order to verify the effectiveness of the improved algorithm, this paper selected other improved algorithms for comparative experiments, such as CMA-ES [57], PSOGSA

TABLE 8. P-value results of wilcoxon rank sum test.

Function	LSC-SSA vs SSA	LSC-SSA vs L-SSA	LSC-SSA vs SC-SSA
F1	6.39E-05	6.39E-05	6.39E-05
F2	6.39E-05	6.39E-05	6.39E-05
F3	6.39E-05	6.39E-05	6.39E-05
F4	6.39E-05	6.39E-05	6.39E-05
F5	4.27E-01	2.41E-01	1.83E-04
F6	4.73E-01	4.73E-01	1.83E-04
F7	1.83E-04	6.40E-02	1.83E-04
F8	1.83E-04	1.83E-04	1.83E-04
F9	6.39E-05	NaN	NaN
F10	6.39E-05	6.29E-05	5.00E-03
F11	6.39E-05	NaN	NaN
F12	1.83E-04	1.83E-04	1.83E-04
F13	6.39E-05	6.39E-05	1.83E-04
F14	6.08E-04	1.20E-03	3.08E-01
F15	1.83E-04	4.27E-01	1.83E-04
F16	9.95E-04	4.05E-01	1.83E-04
F17	1.79E-04	1.81E-04	1.83E-04
F18	1.54E-04	1.62E-04	1.83E-04
F19	4.73E-01	3.08E-01	1.70E-03
F20	7.91E-01	1.83E-04	7.69E-04
F21	3.12E-02	1.83E-04	2.11E-02
F22	1.41E-01	1.83E-04	1.21E-01

[58], SADE and VPSO [59]. Among them, CMA-ES is the winner algorithm of CEC2013. SADE, PSOGSA and VPSO are improved algorithms that have been verified by various optimization problems (function optimization, engineering optimization, etc.). Comparison with the four improved algorithms can more fully demonstrate the optimization capabilities of LSC-SSA. The functions F1-F28 of Table 2 were selected for experiments. The convergence curves of some functions are shown in Fig. 13. At the same time, Table 9 shows the mean and variance generated by the algorithm running 60 times independently. The mean and variance can intuitively show the average accuracy and robustness of the algorithm. In addition, p-value results of wilcoxon rank sum test are shown in Table 10.

The convergence curve shows that the improved algorithm has the best convergence speed and optimization accuracy. It can be seen from Table 9 that the improved algorithm has the best performance in terms of average accuracy and robustness. In addition, the statistical results in Table 10 show that the improved algorithm LSC-SSA has significant advantages. In summary, the improved algorithm LSC-SSA is the best algorithm in this experiment. CMA-ES is the second best algorithm. Compared with other improved algorithms, it shows the competitiveness of LSC-SSA more comprehensively.

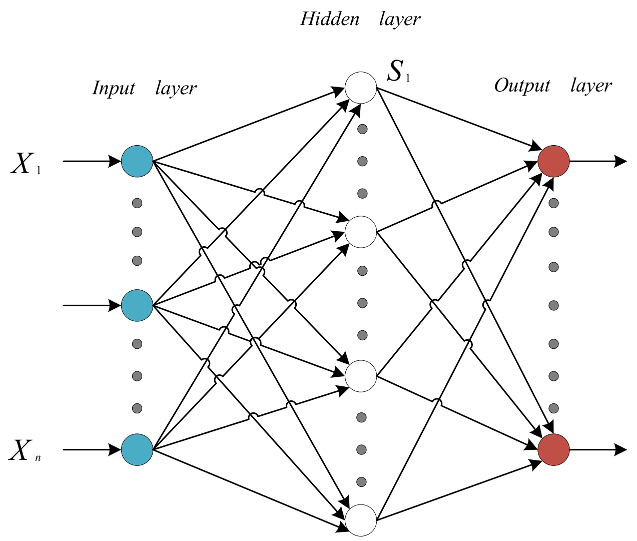


FIGURE 14. Structure of MLP.

VI. TRAINING MUTI-LAYER PERCEPTRON BY LSC-SSA

Neural Networks (NN) is an application tool in the field of intelligent computing that solves classification problems by mimicking biological neurons in the brain. Many types of neural networks have been proposed, such as Kohonen self-organizing neural networks [60], Recurrent neural networks [61], and so on. Feed-forward neural networks [62] are also one of them. In a feed-forward neural network, neurons are arranged in different parallel layers with only one-way connections between them. The first layer is used as the input layer, the last layer is used as the output layer, and the level between the input layer and the output layer is hidden layer. The input information can share the information of two neurons along one direction in the neural network. The feed-forward neural network with only one hidden layer is called Muti-Layer Perceptron (MLP). The structure of the MLP is shown in Fig. 14. The neural network can use the trainer to learn from existing experience and obtain the best connection weight and error value to ensure that the deviation of the output layer is minimized. Muti-Layer Perceptron are no exception. Neural network learning trainers are divided into deterministic learning and random learning.

Back propagation (BP) algorithm and gradient descent algorithm belong to the trainer of deterministic learning. Deterministic learning has the advantages of fast convergence, simple and efficient. But the quality of the global optimal solution depends on the initial solution, and it is easy to fall into the local optimal solution. Random learning can continuously improve the initial random solution during the learning process, which can prevent the neural network from falling into a local optimum, but the conver-

TABLE 9. Comparison of simulation performance.

Function	LSC-SSA		PSOGSA		CMA-ES		VPSO		SADE	
	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std
F1	**0**	**0**	3.00E+03	4.83E+03	0	0	5.73E-41	1.68E-40	2.46E-41	7.74E-41
F2	**0**	**0**	1.9846	6.2759	4.53E-258	0	1.39E-19	3.79E-19	8.65E-25	2.73E-24
F3	**0**	**0**	8.25E+03	8.23E+03	0	0	1.05E-37	3.00E-37	3.12E-04	9.87E-04
F4	**0**	**0**	5.88E+01	2.54E+01	3.37E-233	0	3.89E-16	1.23E-15	5.02E-19	3.50E-19
F5	2.74E+01	6.25E-01	9.03E+03	2.85E+04	2.43E-04	3.06E-04	2.90E+01	1.02E-02	2.90E+01	8.60E-03
F6	2.95E-01	1.62E-01	2.42E-19	5.08E-20	2.21E-05	2.35E-05	7.2107	4.31E-01	1.79E+01	1.17E+01
F7	1.06E-05	1.11E-05	5.23E-02	1.83E-02	5.05E-05	3.80E-05	4.48E-03	3.25E-03	1.45E+01	1.53E+01
F8	-12569.2851	1.81E-01	-7313.4425	7.18E+02	-12569.4696	2.56E-02	-24937.2467	1.78E+04	-1370.7778	1.49E+03
F9	**0**	**0**	1.28E+02	4.01E+01	0	0	0	0	1.36E+01	1.53E+01
F10	8.88E-16	0	1.61E+01	3.9301	8.88E-16	0	8.88E-16	0	1.45E+01	1.53E+01
F11	**0**	**0**	3.61E+01	4.66E+01	0	0	0	0	0	0
F12	1.82E-02	1.83E-02	7.5287	4.7674	2.05E-06	4.17E-06	1.4456	0.25629	1.2753	2.36E-01
F13	**0**	**0**	2.0011	4.2158	1.27E-254	0	6.44E-19	1.63E-18	2.9097	6.26E-02
F14	7.0403	5.6087	4.1126	5.3261	9.98E-01	4.10E-10	1.6647	0.90791	2.9467	5.55E-02
F15	3.65E-04	8.74E-05	1.01E-02	1.83E-02	3.78E-04	9.55E-05	1.13E-03	5.85E-04	1.45E-03	6.71E-04
F16	0.39789	5.09E-14	3.98E-01	0	3.98E-01	4.00E-04	0.39789	2.29E-07	0.41362	1.63E-02
F17	3	9.67E-05	3	1.50E-15	5.72E+00	8.5353	3	4.02E-05	3.0097	8.72E-03
F18	-3.8591	3.64E-03	-3.8628	8.24E-16	-3.8445	4.35E-02	-3.8627	8.42E-05	-3.8606	9.90E-04
F19	-3.1689	7.57E-02	-3.2625	6.27E-02	-3.0307	1.79E-01	-3.3209	1.64E-03	-3.1273	3.65E-02
F20	-10.1532	5.72E-11	-5.3971	3.4229	-10.153	3.13E-04	-6.5034	3.2465	-3.5819	7.46E-01
F21	-10.4029	8.85E-11	-5.5271	3.5196	-10.4024	1.08E-03	-8.6374	2.7782	-3.7699	1.2861
F22	-10.5364	7.27E-11	-4.277	3.3279	-10.5362	1.77E-04	-6.318	3.6409	-3.3155	6.00E-01
F23	**0**	**0**	1.16E-04	1.93E-04	0	0	9.40E-29	2.96E-28	2.25E-139	7.11E-139
F24	**0**	**0**	1.00E+43	3.16E+43	0	0	3.92E-40	8.76E-40	1.24E-243	0
F25	**0**	**0**	5.19E+00	1.09E+01	0	0	8.08E-28	2.55E-27	4.99E-33	1.58E-32
F26	**0**	**0**	4.41E+07	9.21E+07	0	0	8.41E-34	2.66E-33	1.27E-144	4.01E-144
F27	**0**	**0**	2.33E+04	2.15E+04	0	0	1.00E-29	3.18E-29	3.69E-154	1.17E-153
F28	**0**	**0**	1.96E+00	3.50E-01	0	0	5.83E-10	1.83E-09	1.82E-41	5.53E-41

gence is worse than deterministic learning. As a random learning trainer, the meta-heuristic algorithm can effectively solve the problem of local optimal stagnation. The function optimization in Section IV is a continuous problem. There are limited variables and target values in a limited searching space. However, training muti-layer perceptron is a discrete problem, and the values in the searching space are not continuous. At the same time, there are a large number of local optimal values for discrete problems, and the trainer may mistake the local optimal as the global optimal and reduce the

TABLE 10. P-value results of wilcoxon rank sum test.

Function	LSC-SSA vs PSOGSA	LSC-SSA vs CMA-ES	LSC-SSA vs VPSO	LSC-SSA vs SADE
F1	6.39E-05	NaN	6.39E-05	6.39E-05
F2	6.39E-05	6.39E-05	6.39E-05	6.39E-05
F3	6.39E-05	NaN	6.39E-05	6.39E-05
F4	6.39E-05	2.31E-04	6.39E-05	6.39E-05
F5	2.57E-02	1.83E-04	1.82E-04	1.83E-04
F6	1.83E-04	4.73E-01	1.82E-04	1.83E-04
F7	1.83E-04	3.30E-04	1.83E-04	1.83E-04
F8	1.83E-04	4.73E-01	5.83E-04	1.83E-04
F9	6.39E-05	NaN	NaN	1.83E-04
F10	6.39E-05	NaN	NaN	1.83E-04
F11	6.39E-05	NaN	NaN	NaN
F12	1.83E-04	1.83E-04	1.79E-04	1.83E-04
F13	6.39E-05	NaN	6.39E-05	1.83E-04
F14	5.49E-02	2.38E-02	3.51E-02	4.73E-01
F15	3.60E-03	1.70E-03	1.83E-04	1.83E-04
F16	6.39E-05	1.82E-04	1.82E-04	1.41E-01
F17	1.10E-04	1.83E-04	8.90E-02	6.40E-02
F18	6.39E-05	2.11E-02	1.30E-03	1.00E-03
F19	1.35E-01	2.80E-03	4.73E-01	3.76E-02
F20	1.39E-01	1.83E-04	1.83E-04	1.83E-04
F21	1.40E-01	1.83E-04	1.83E-04	1.83E-04
F22	2.54E-02	1.83E-04	1.83E-04	1.83E-04
F23	6.39E-05	NaN	6.39E-05	6.39E-05
F24	6.39E-05	NaN	6.39E-05	1.83E-04
F25	6.39E-05	NaN	6.39E-05	6.39E-05
F26	6.39E-05	NaN	6.39E-05	6.39E-05
F27	6.39E-05	NaN	6.39E-05	6.39E-05
F28	6.39E-05	NaN	6.39E-05	6.39E-05

optimization accuracy. Therefore, meta-heuristic algorithms are challenging to train muti-layer perceptron. In order to further verify the optimization performance of the improved algorithm and the effectiveness of solving discrete problems, LSC-SSA was applied to train muti-layer perceptron neural network.

A. LSCSSA-MLP

The goal of the meta-heuristic algorithm for training muti-layer perceptron is to find a set of connection weights and error values to optimize the classification accuracy. Generally, the data set samples have the characteristics of high dimensions, muti-modality, noise pollution, and missing data. The expression of the meta-heuristic algorithm should be changed to apply to muti-layer perceptron, so it is necessary to choose a suitable encoding mechanism. According to [63], the coding mechanism is divided into matrix coding, vector coding and binary coding. Matrix coding can simplify the decoding process of the algorithm and reduce the operation cost. Therefore, matrix coding is selected as the encoding mechanism of LCSSSA-MLP. The target variable of MLP is the connection weights and error value, and the target variable of the LSC-SSA algorithm is the global optimal value. Matrix coding is used to represent the connection weights and error values as global optimal values. The equation is described as follows.

$$\begin{aligned} V &= [W, \theta] \\ &= \left[W_{1,1}, W_{1,2}, \ldots W_{n,n}, \theta_1, \theta_2, \ldots \theta_h\right] \end{aligned} \tag{23}$$

where, n is the number of input nodes; $W_{i,j}$ indicates the connection weight from the i-th node to the j-th node; θ indicates the error value. After defining the variables of LSCSA-MLP, the objective function needs to be defined so that LSCSA-MLP can achieve the best classification accuracy in the training and test samples. The mean square error (MSE) based on all training samples is a common indicator for verifying MLP. The equation is described as follows.

$$MSE = \sum_{k=1}^{s} \sum_{i=1}^{m} \left(o_i^k - d_i^k\right)^2 / s \tag{24}$$

where, m is the number of input nodes; s is the number of training samples; d_i^k indicates the expected output value of the k-th input node when using the i-th training sample. Therefore, the objective function of the LSC-SSA-based muti-layer perceptron trainer can be defined as:

$$Minimize\ F(V) = MSE \tag{25}$$

Different data sets have different attribute ranges, so normalizing the data is an important step for LSSCA-MLP. This paper adopts the minimum-maximum normalization method to map the sample x to the intervals [a, b] and [c, d]. The equation is as follows:

$$X' = \frac{(x-a)\cdot(d-c)}{b-a} + c \tag{26}$$

After encoding, the MSE and classification accuracy obtained by the training samples after MLP learning are passed to the trainer. The LSC-SSA based trainer optimizes the connection weights and error value transmission, which further improves the MSE and classification accuracy until it finds the best classification accuracy. The muti-layer perceptron trainer based on LSC-SSA is shown in Fig. 15.

B. ADOPTED DATA SETS

The three data sets used by the LSCSSA based muti-layer perceptron trainer are from the University of California Irvine Machine Learning Database. Three categorical datasets (XOR dataset, Balloon dataset, and Breast Cancer

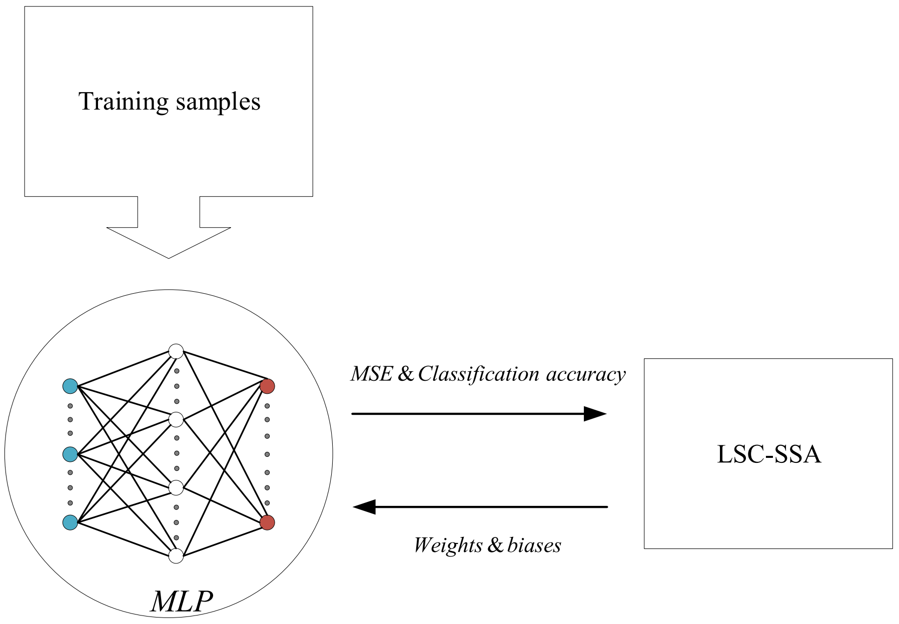


FIGURE 15. Muti-Layer Perceptron trained by LSC-SSA.

TABLE 11. Data sets and MLP structure.

Data sets	Number of attributes	Number of training samples	Number of test samples	categ ories	MLP structur e
XOR	3	8	8	2	3-7-1
Balloon	4	16	16	2	4-9-1
Breast Cancer	9	599	100	2	9-19-1

dataset) are selected. In order to effectively verify the performance of the algorithm, this paper sets different difficulties on the data set. The number of training / test samples is different, and the number of attributes is also different. Therefore, the muti-layer perceptron has different structure. The specific information of the data set and its MLP structure are listed in Table 6. It can be seen from Table 6 that the XOR dataset has 8 training / test samples, 3 attributes and 2 categories. In addition, the Balloon dataset and Breast Cancer dataset have more training / test samples than XOR dataset, so they are much more difficult than XOR dataset. The former has 16 training / test samples, 4 attributes and 2 categories, and the optimization dimension is 55. The latter has 599 training samples, 100 test samples, 9 attributes and 2 categories, and the search agent needs to optimize 209 variables. It can be seen that the classification difficulty of the three data sets is increasing.

C. EXPERIMENTAL RESULTS AND ANALYSIS

In order to ensure the objectivity of the experimental results, the mean square error (MSE) and classification accuracy of LCSSSA-MLP are compared with PSO-MLP, ACO-MLP, ES-MLP, and PBIL-MLP in the literature [64]. The experimental results are listed in Table 11. Among them, the algorithm uniformly sets the number of population to 200 and the maximum number of iterations to 250. The experimental

TABLE 12. Comparison of experimental results.

Data sets	Algorithm	MSE	Classification accuracy
XOR	LSCSSA-MLP	0.0011+2.4942e-04	87.50%
	PSO-MLP	0.08405+0.0359	37.50%
	ACO-MLP	0.18032+0.0252	62.50%
	ES-MLP	0.11873+0.0115	62.50%
	PBIL-MLP	0.03022+0.0396	62.50%
Balloon	LSCSSA-MLP	1.4908e-04+5.1924e-05	100%
	PSO-MLP	0.00058+0.0007	100%
	ACO-MLP	0.00485+0.0077	100%
	ES-MLP	0.01905+0.1702	100%
	PBIL-MLP	2.49e-05+5.27e-05	100%
Breast caner	LSCSSA-MLP	0.0016+1.9009e-05	98%
	PSO-MLP	0.03488+0.0024	11.00%
	ACO-MLP	0.01351+0.0021	40.00%
	ES-MLP	0.04032+0.0024	6.00%
	PBIL-MLP	0.03200+0.0030	7.00%

results show that, on the XOR dataset, the mean square error and classification accuracy of LCSSSA-MLP are better than other algorithms. ACO-MLP, ES-MLP and PBIL-MLP have no difference in classification accuracy, and PSO-MLP has the lowest classification accuracy. This shows that PSO algorithm falls into a local optimum in XOR dataset, and LSC-SSA can avoid falling into a local optimum. On the Balloon dataset, the mean square error of LCSSSA-MLP is second only to PBIL-MLP, and the classification accuracy is no different from other algorithms. All algorithms have reached the theoretically best accuracy. The Breast Cancer dataset has the characteristics of large scale and high dimensions, and the classification difficulty is obviously higher than the first two datasets. It is worth mentioning that the mean square error and classification accuracy of LCSSSA-MLP are significantly better than other algorithms. ACO-MLP has the second best optimization effect. The classification accuracy of the other three algorithms failed to exceed 20%. This shows that LCSSSA will not fall into a dimensional disaster and can maintain a stable optimization capability. The experimental results show that the improved algorithm can be effectively used as a trainer for muti-layer perceptron, and can match the optimal connection weights and error value. It further illustrates that LSC-SSA has good development ability and optimization performance, and can be applied to different optimization problems (continuous / discrete problems).

D. DISCUSSION OF LSC-SSA ALGORITHM

In view of the shortcomings of SSA algorithm, this paper proposes two improvement strategies. First, Levy flight with a step control factor is used to increase the global exploration capability of search agents. This mechanism makes up for the defect that the leader may mislead the population into local optimum. Second, the improved sine cosine operator introduces the convergence factor and the logarithmic spiral search route, which is used to increase the search efficiency of the leader. The combination of two improved strategies improves the balance between exploitation and exploration. In the function optimization problem, LSC-SSA has excellent convergence speed and optimization accuracy, and it shows stronger competitiveness than other algorithms. As the neural network trainer, the improved algorithm can effectively avoid local optimization. The classification accuracy of LSC-SSA is higher than other algorithms. The experiment proved the effectiveness of the improved algorithm from different angles, and laid the foundation for the application of the improved algorithm.

VII. CONCLUSION

As a swarm intelligence optimization algorithm, the salp swram algorithm has a simple structure. The algorithm is easy to implement because it has fewer parameters and operators. However, the algorithm has the disadvantages of low optimization precision and slow convergence speed. The LSC-SSA proposed in this paper first introduced the Levy flight mechanism with a step size control factor. This mechanism uses short-distance walking and long-distance jumping routes to search the space, which effectively improves the traversal and global exploration capabilities of the algorithm. In addition, LSC-SSA uses an improved sine cosine operator to update the position of leader, and uses sine search for global exploration and cosine search for local exploitation. This mechanism ensures that the algorithm adaptively switches and optimizes between two search methods to achieve a smooth transition between exploration and exploitation. In the simulation experiments, firstly, 28 benchmark test functions were used for carry out comparison experiments. LSC-SSA showed more obvious advantages. The improved algorithm has higher global convergence and optimization accuracy than other algorithms. Secondly, the high-dimensional function optimization experiments verify that the proposed LSC-SSA will not be affected by the dimensional disaster, and can still maintain the optimization accuracy and stability. LSC-SSA can effectively solve high-dimensional and large-scale optimization problems. At the same time, the effectiveness of the Levy flight mechanism and improved sine cosine operator have been verified. Finally, the muti-layer perceptron trainer based on LSC-SSA found an ideal classification accuracy rate, indicating that the improved algorithm can avoid falling into local optimal values. LSC-SSA can not only solve continuous problems (function optimization), but also effectively solve discrete problems (training muti-layer perceptron). The simulation results show that the improved algorithm has powerful optimization performance, which is of great significance for further theoretical exploration and practical application of the salp swram algorithm.

REFERENCES

[1] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and L. M. Wang, "An improved GA and a novel PSO-GA-based hybrid algorithm," *Inf. Process. Lett.*, vol. 93, no. 5, pp. 255–261, Mar. 2005.

[2] W.-Y. Lin, "A GA–DE hybrid evolutionary algorithm for path synthesis of four-bar linkage," *Mechanism Mach. Theory*, vol. 45, no. 8, pp. 1096–1107, Aug. 2010.

[3] G.-G. Wang, A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "A new hybrid method based on krill herd and cuckoo search for global optimisation tasks," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 5, pp. 286–299, 2016.

[4] H. Duan, Y. Yu, X. Zhang, and S. Shao, "Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm," *Simul. Model. Pract. Theory*, vol. 18, no. 8, pp. 1104–1115, Sep. 2010.

[5] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, Sep. 2005, pp. 522–528.

[6] K. Premalatha and A. M. Natarajan, "A new approach for data clustering based on PSO with local search," *Comput. Inf. Sci.*, vol. 1, no. 4, pp. 139–145, Oct. 2008.

[7] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 107–125, Feb. 2008.

[8] J. Levine and F. Ducatelle, "Ant colony optimization and local search for bin packing and cutting stock problems," *J. Oper. Res. Soc.*, vol. 55, no. 7, pp. 705–716, Jul. 2004.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[10] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *J. Stat. Phys.*, vol. 34, nos. 5–6, pp. 975–986, Mar. 1984.

[11] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning," *Oper. Res.*, vol. 37, no. 6, pp. 865–892, Dec. 1989.

[12] F. Glover, "Tabu search—Part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.

[13] T. Stützle, "Iterated local search for the quadratic assignment problem," *Eur. J. Oper. Res.*, vol. 174, no. 3, pp. 1519–1539, Nov. 2006.

[14] I. Rechenberg, "Evolutionsstrategien," in *Simulationsmethoden in der Medizin und Biologie*. Berlin, Germany: Springer, 1978, pp. 83–114.

[15] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[16] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2006, pp. 1–36, doi: 10.1007/3-540-31306-0.

[17] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.

[18] D. Dasgupta and Z. Michalewicz, Eds. *Evolutionary Algorithms in Engineering Applications*. Berlin, Germany: Springer, 2013.

[19] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992.

[20] H. Eskandar, A. Sadollah, A. Bahreininejad, and M. Hamdi, "Water cycle algorithm—A novel Metaheuristic optimization method for solving constrained engineering optimization problems," *Comput. Struct.*, vols. 110–111, pp. 151–166, Nov. 2012.

[21] 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.

[22] H. R. E. H. Bouchekara, "Electrostatic discharge algorithm: A novel nature-inspired optimisation algorithm and its application to worst-case tolerance analysis of an EMC filter," *IET Sci., Meas. Technol.*, vol. 13, no. 4, pp. 491–499, Jun. 2019.

[23] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Inf. Sci.*, vol. 222, pp. 175–184, Feb. 2013.

[24] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine predators algorithm: A nature-inspired Metaheuristic," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113377.

[25] G. Dhiman and V. Kumar, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems," *Knowl.-Based Syst.*, vol. 165, pp. 169–196, Feb. 2019.

[26] G. Dhiman and V. Kumar, "Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications," *Adv. Eng. Softw.*, vol. 114, pp. 48–70, Dec. 2017.

[27] R. Salgotra and U. Singh, "The naked mole-rat algorithm," *Neural Comput. Appl.*, vol. 31, no. 12, pp. 8837–8857, Dec. 2019.

[28] A. Faramarzi, M. Heidarinejad, B. Stephens, and S. Mirjalili, "Equilibrium optimizer: A novel optimization algorithm," *Knowl.-Based Syst.*, vol. 191, Mar. 2020, Art. no. 105190.

[29] A.-A.-A. Mohamed, S. A. Hassan, A. M. Hemeida, S. Alkhalaf, M. M. M. Mahmoud, and A. M. B. Eldin, "Parasitism—Predation algorithm (PPA): A novel approach for feature selection," *Ain Shams Eng. J.*, early access, Nov. 11, 2019, doi: 10.1016/j.asej.2019.10.004.

[30] W. Zhao, Z. Zhang, and L. Wang, "Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications," *Eng. Appl. Artif. Intell.*, vol. 87, Jan. 2020, Art. no. 103300.

[31] A. Tharwat and T. Gabel, "Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm," *Neural Comput. Appl.*, pp. 1–14, 2019, doi: 10.1007/s00521-019-04159-z.

[32] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artif. Life*, vol. 5, no. 2, pp. 137–172, Apr. 1999.

[33] 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.

[34] Y. Ling, Y. Zhou, and Q. Luo, "Lévy flight trajectory-based whale optimization algorithm for global optimization," *IEEE Access*, vol. 5, pp. 6168–6186, 2017.

[35] H. Haklı and H. U uz, "A novel particle swarm optimization algorithm with Lévy flight," *Appl. Soft Comput.*, vol. 23, pp. 333–345, Oct. 2014.

[36] W. Xie, J. S. Wang, and Y. Tao, "Improved black hole algorithm based on golden sine operator and Lévy flight operator," *IEEE Access*, vol. 7, pp. 161459–161486, 2019.

[37] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, 2009, pp. 210–214.

[38] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. 1st Annu. Conf. Genet. Evol. Comput.*, vol. 1. San Mateo, CA, USA: Morgan Kaufmann, 1999, pp. 525–532.

[39] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.

[40] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[41] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[42] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.

[43] X-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 2, pp. 78–84, 2010.

[44] G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel chaotic salp swarm algorithm for global optimization and feature selection," *Int. J. Speech Technol.*, vol. 48, no. 10, pp. 3462–3481, Oct. 2018.

[45] R. Abbassi, A. Abbassi, A. A. Heidari, and S. Mirjalili, "An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models," *Energy Convers. Manage.*, vol. 179, pp. 362–372, Jan. 2019.

[46] R. A. Ibrahim, A. A. Ewees, D. Oliva, M. A. Elaziz, and S. Lu, "Improved salp swarm algorithm based on particle swarm optimization for feature selection," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 8, pp. 3155–3169, Aug. 2019.

[47] S. Ahmed, M. Mafarja, H. Faris, and I. Aljarah, "Feature selection using salp swarm algorithm with chaos," in *Proc. 2nd Int. Conf. Intell. Syst., Metaheuristics Swarm Intell. (ISMSI)*, 2018, pp. 65–69.

[48] N. Neggaz, A. A. Ewees, M. A. Elaziz, and M. Mafarja, "Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection," *Expert Syst. Appl.*, vol. 145, May 2020, Art. no. 113103.

[49] M. Tubishat, N. Idris, L. Shuib, M. A. M. Abushariah, and S. Mirjalili, "Improved salp swarm algorithm based on opposition based learning and novel local search algorithm for feature selection," *Expert Syst. Appl.*, vol. 145, May 2020, Art. no. 113122.

[50] N. Panda and S. K. Majhi, "Improved salp swarm algorithm with space transformation search for training neural network," *Arabian J. for Sci. Eng.*, vol. 45, no. 4, pp. 2743–2761, Apr. 2020.

[51] M. A. Elaziz, L. Li, K. P. N. Jayasena, and S. Xiong, "Multiobjective big data optimization based on a hybrid salp swarm algorithm and differential evolution," *Appl. Math. Model.*, vol. 80, pp. 929–943, Apr. 2020.

[52] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, and H. Fujita, "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," *Knowl.-Based Syst.*, vol. 154, pp. 43–67, Aug. 2018.

[53] A. A. El-Fergany, "Extracting optimal parameters of PEM fuel cells using salp swarm optimizer," *Renew. Energy*, vol. 119, pp. 641–648, Apr. 2018.

[54] B. Yang, L. Zhong, X. Zhang, H. Shu, T. Yu, H. Li, L. Jiang, and L. Sun, "Novel bio-inspired memetic salp swarm algorithm and application to MPPT for PV systems considering partial shading condition," *J. Cleaner Prod.*, vol. 215, pp. 1203–1222, Apr. 2019.

[55] P. Lévy, *Théorie de L'Addition des Variables Aléatoires*. Paris, France: Gauthier-Villars, 1937.

[56] R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 49, no. 5, p. 4677, 1994.

[57] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, Mar. 2003.

[58] S. Mirjalili, S. Z. M. Hashim, and H. M. Sardroudi, "Training feed-forward neural networks using hybrid particle swarm optimization and gravitational search algorithm," *Appl. Math. Comput.*, vol. 218, no. 22, pp. 11125–11137, Jul. 2012.

[59] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm Evol. Comput.*, vol. 9, pp. 1–14, Apr. 2013.

[60] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[61] G. Dorffner, "Neural networks for time series processing," *Neural Netw. World*, vol. 6, no. 4, pp. 447–468, 1996, doi: 10.1016/S0375-9601(97)00753-6.

[62] W. F. Schmidt, M. A. Kraaijveld, and R. P. W. Duin, "Feed forward neural networks with random weights," in *Proc. Int. Conf. Pattern Recognit.*, 1992, p. 1.

[63] M. H. Ali, B. A. D. Al Mohammed, A. Ismail, and M. F. Zolkipli, "A new intrusion detection system based on fast learning network and particle swarm optimization," *IEEE Access*, vol. 6, pp. 20255–20261, 2018.

[64] S. Mirjalili, "How effective is the grey wolf optimizer in training multi-layer perceptrons," *Int. J. Speech Technol.*, vol. 43, no. 1, pp. 150–161, Jul. 2015.

J. ZHANG is currently pursuing the master's degree with the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China. His main research interest includes intelligent optimization algorithms.

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 include modeling of complex industry process, intelligent control, and computer integrated manufacturing.