Hindawi
Complexity
Volume 2020, Article ID 6371085, 17 pages
https://doi.org/10.1155/2020/6371085

Research Article

A Modified Salp Swarm Algorithm Based on the Perturbation Weight for Global Optimization Problems

Yuqi Fan, Junpeng Shao, Guitao Sun, and Xuan Shao

Key Laboratory of Advanced Manufacturing and Intelligent Technology, Ministry of Education, School of Mechanical and Power Engineering, Harbin University of Science and Technology, Harbin 150080, China

Correspondence should be addressed to Junpeng Shao; sjp566@hrbust.edu.cn

Received 27 April 2020; Revised 30 September 2020; Accepted 26 October 2020; Published 18 November 2020

Academic Editor: Toshikazu Kuniya

Metaheuristic algorithms are often applied to global function optimization problems. To overcome the poor real-time performance and low precision of the basic salp swarm algorithm, this paper introduces a novel hybrid algorithm inspired by the perturbation weight mechanism. The proposed perturbation weight salp swarm algorithm has the advantages of a broad search scope and a strong balance between exploration and exploitation and retains a relatively low computational complexity when dealing with numerous large-scale problems. A new coefficient factor is introduced to the basic salp swarm algorithm, and new update strategies for the leader position and the followers are introduced in the search phase. The new leader position updating strategy has a specific bounded scope and strong search performance, thus accelerating the iteration process. The new follower updating strategy maintains the diversity of feasible solutions while reducing the computational load. This paper describes the application of the proposed algorithm to low-dimension and variable-dimension functions. This paper also presents iteration curves, box-plot charts, and search-path graphics to verify the accuracy of the proposed algorithm. The experimental results demonstrate that the perturbation weight salp swarm algorithm offers a better search speed and search balance than the basic salp swarm algorithm in different environments.

1. Introduction

In many engineering fields, there are numerous optimization problems that must be solved under complicated constraints, over large search domains and high complexities [1–3]. Traditional mathematical strategies, such as the steepest descent method and the variable scale method, can only calculate simple and continuous functions [4, 5]. Thus, complex features such as nonlinearity, multiple variables, multiple constraints, and multiple dimensions require new optimization strategies that have strong calculation abilities and a high degree of precision [6–8]. Intelligent metaheuristic algorithms have received considerable attention from researchers, and rapid improvements in such techniques have been made in recent years as a result of their widespread utilization, enhanced computational technologies, high practicability, and fault-tolerant abilities [9–13].

Optimization algorithms have strong prospects related to numerous practical industrial fields and theoretical mathematics applications, such as global numerical optimization [14], path planning [15], clustering analysis [16], 0-1 knapsack problems [17], image segmentation [18], PID tuning [19], obstacle avoidance of robotic manipulator [20], and feature selection [21]. All of those areas need algorithms to obtain more precise parameters. In recent years, scholars have proposed many advanced metaheuristic algorithms, such as monarch butterfly optimization (MBO) [22], beetle antennae search algorithm (BAS) [23], earthworm optimization algorithm (EWA) [24], elephant herding optimization (EHO) [25], crow search algorithm (CSA) [26], and moth search algorithm (MS) [27]. MBO, which is mainly determined by the migration operator and butterfly adjusting operator, is ideally suited for parallel searching and well capable of balancing trade-off between intensification and diversification. BAS not only has the ability of individual

recognition and environmental recognition abilities but also owns the simple code. In EWA, the addition of a Cauchy mutation can make certain earthworms escape from local optima and enhance the algorithm searching ability and can also help the whole earthworm positions proceed to a better position. EHO is divided into two operators including the clan updating operators and separating operators. The worst elephant position is replaced by randomly generated positions, which can significantly accelerate convergent speed, avoiding premature and local convergence. CSA applies a population of seekers to explore the searching space, by the use of a population, the probability of searching a feasible position and escaping from local optima increases. MS searching process can be seen as exploitation and exploration, and the act of balancing of exploitation and exploration is indeed. MS has a good performance and effectiveness.

The salp swarm algorithm (SSA), a nature-inspired metaheuristic algorithm, is proposed by Mirjalili et al. in 2017 [28], which displays some promising performance for global optimization functions. SSA imitates the salp living and predation habits, and the mathematical model of SSA can be divided into two groups including one leader and followers. The leader is the first salp in front of the salp chain, whereas other salps can be seen as followers. The leader indirectly guides the salp swarm to follow each other. SSA has exploration and local optima avoidance abilities, which originate from the reason that salps tend to interact with each other; so, salps do not gravitate towards a local feasible solution easily. The salp chain makes the SSA search the finding space and gradually move to the global optimum, which demonstrates the superior exploitation of SSA. SSA converges towards the food position proportional to the iteration number because the connections between the leaders also pull other salps towards the food position. In addition, it is observed that SSA can balance exploration and exploitation. Owing to the distinguishing characteristics including simple code and easy implementation, it is becoming one of the most studying hot areas in algorithm fields, such as node localization in wireless sensor networks [29], the Takagi–Sugeno fuzzy logic controller design [30], the extreme learning machine optimization [31], the IIR wideband digital differentiators and integrators design [32], the photovoltaic cell models parameters identification [33], PEM fuel cells parameters extracting [34], the passive sonar target classification [35], the airfoil-based savories wind turbine optimization [36], the model predictive controller devising [37], and the soil water retention curve parameter estimation [38]. There are different salp swarm algorithm variants that are used in many areas. Wan et al. [39] proposed that the MPPT controller is achieved by combining the salp swarm algorithm with the grey wolf optimizer. Gao et al. [40] combined SSA with quantum swarm intelligence and proposed the quantum salp swarm algorithm to solve Nakagami-m quantile functions. Xing and Jia [41] introduced the Lévy flight salp swarm algorithm which can eliminate the problem of getting stuck in local optima and applied the proposed algorithm for multilevel color image segmentation. The literature [42] designed an advanced Lévy flight salp swarm algorithm for hydraulic systems. Majhi et al. [43] drafted a chaotic salp swarm algorithm based on the fire neural model and quadratic integration. Neggaz et al. [44] created improved leaders of the salp swarm algorithm using the sine cosine algorithm and disrupt operator, the updating position it consists to update the leader position by the sine cosine algorithm and applying the disrupt operator. The literature [45] applied diversities of the moth-flame optimization (MFO) algorithm to weaken the limitations of basic SSA and the proposed algorithm called SSAMFO. Ibrahim et al. [46] devised the hybridization algorithm SSAPSO between SSA and PSO, in which the exploration and the exploitation steps of SSA are improved. Panda and Majhi [47] introduced an improved version of SSA, which can boost the *s* searching performance of SSA by using space transformation search. In literature [48], a new SSA binary version called BSSA was drafted based on an Arctan transformation. In literature [49], a novel hybrid algorithm based on SSA and chaos theory was proposed, and the capability of proposed algorithm in finding an optimal feature subset can enhance the classification accuracy. Wu et al. [50] proposed an improved salp swarm algorithm based on weight factor and adaptive mutation, and testing results showed the good convergence performance of escaping local optimum when compared with basic SSA.

Xiang et al. [51] proposed a modified salp swarm algorithm called polar coordinate salp swarm algorithm (PSSA), which is inspired by the spiral aggregation chain and foraging trajectory of salps. Hegazy et al. [52] added a new control parameter in basic SSA to adjust the present best solution, and the new method is called the improved salp swarm algorithm (ISSA). Qais et al. [53] introduced an enhanced salp swarm algorithm (ESSA) to improve the power point tracking and the fault ride-through capability of a grid-tied permanent magnet synchronous generator driven by a variable speed wind turbine.

The leader salp searches for the best solution to the given problems using the difference between the lower searching bound and the upper searching bounds, which causes that the local optimum cannot be sufficiently utilized for the optimization procedure in basic SSA. The expression factor with a fixed coefficient is the *e* exponential function, and the exponential function will emerge the exponential disaster in the later iteration phase, which causes premature convergence. The followers update their next positions by applying for their neighbor positions. This single updating mechanism is unfavorable in terms of algorithm diversity. To overcome the above problems and enhance the performance of SSA, this paper describes the perturbation weight salp swarm algorithm (PWSSA). A new coefficient factor, a new leadership position updating strategy, and new followers updating strategy are added to the basic salp swarm algorithm. PWSSA uses the perturbation weight mechanism to weaken the distance difference of the best solution and each solution and applies the asymptotic circular searching mechanism to find a better leader position at a faster speed. Followers' positions will be changed more and more slightly with increasing iterations in PWSSA. PWSSA can balance the leader position, and other followers' positions can weaken the exponential explosion problem in basic SSA. As

a result, the effectiveness of search orientation is significantly enhanced. For experiments and discussion, this paper used different algorithms to carry on different function experiments including low-dimension functions and variable-dimension functions, and iteration curves, box-plot charts, and searching path graphics were given to show the strong searching performance of PWSSA. All experiment results demonstrate that the proposed algorithm has a stronger searching accuracy and larger exploration balance than the basic salp swarm algorithm.

The rest of this paper is organized as follows: in Section 2, the basic salp swarm algorithm is presented. In Section 3, the perturbation weight salp swarm algorithm is proposed. Experimental parameters, experimental environments, results, and discussion are given in Section 4. In Section 5, the conclusion is drawn.

2. Salp Swarm Algorithm

The salp swarm living in the sea is a transparent organism that is similar to jellyfish. Mirjalili et al. introduced the salp swarm algorithm depending on the salp predation strategy, which is a chain-like behavior relying on the chain mechanism of the group. SSA, which is based on chain behavior to find the optimal solution, is one of the evolutionary metaheuristic algorithms. In the salp swarm, all salps are divided into two parts including a leader and followers. The salp presented at the front of the salp chain is the leader, whereas other salps can be seen as the followers. In the procedure of the salp predation mechanism, the leader in the front of the chain guides followers search food, and all followers, which follow the recent salp, deliver food signals to keep the flexibility chain shape.

In this paper, each salp position is set to find food in an $N \times D$ dimension searching space, where N represents the population size, and D represents the searching dimension. Hence, i^{th} salp position $x^i_{d\,(i=1,2,\ldots,N),\,(d=1,2,\ldots,D)}$ in the in the d^{th} dimension can be represented as

$$\mathbf{x}_d^i = \begin{bmatrix} \mathbf{x}_1^1 & \mathbf{x}_2^1 & \cdot & \mathbf{x}_D^1 \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \cdot & \mathbf{x}_D^2 \\ \cdot & \cdot & \cdot & \cdot \\ \mathbf{x}_1^N & \mathbf{x}_2^N & \cdot & \mathbf{x}_D^N \end{bmatrix}. \tag{1}$$

The leader position $x^1_{d\,(d=1,2,\ldots,D)}$ is assigned in d-dimension searching space.

The food source, which can be seen as the best solution in functions, is also set to be present in the searching area and is targeted by the salp swarm chain. The leader updates its position according to the food source position. The position of the leader can be represented as

$$x_d^1 = \begin{cases} F_d + c_1\left((\mathrm{ub}_d - \mathrm{lb}_d)c_2 + \mathrm{lb}_d\right), & c_3 \geq p, \\ F_d - c_1\left((\mathrm{ub}_d - \mathrm{lb}_d)c_2 + \mathrm{lb}_d\right), & c_3 < p, \end{cases} \tag{2}$$

where x_d^1 is the leader position, F_d is the food position, ub_d is the upper bound of d^{th} dimension searching space, and lb_d is the lower bound of d^{th} dimension searching space. Parameters p, c_2, and c_3 are random numbers uniformly obtained in the interval of $[0, 1]$. The parameter c_1 indicates the expression coefficient and can be represented as

$$c_1 = 2e^{-(4t/T)}, \tag{3}$$

where t is the current iteration number, T is the maximum number of iterations, and e is the natural base.

In each searching procedure, each follower tracks the leader position by following other followers. Each follower position can be defined as follows:

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

where $i \geq 2$, x_d^i, and x_d^{i-1} mean one ith follower position and its neighbor position in d^{th} dimension.

The pseudocode of the basic SSA is given in Algorithm 1. The SSA main step can be summarized in the pseudocode as follows:

3. Perturbation Weight Salp Swarm Algorithm

The leader guides followers find food according to the difference of the lower searching bound and the upper searching bound in SSA, but if the searching problem has large-scale optimization fields, the large searching scope makes that the local searching is not a sufficient optimization process, and the neighborhood information near the local optimization optimum is insufficiently applied. The expression factor c_1 is the e exponential function with a fixed coefficient, and the exponential equation will grow explosively at the later of iteration; therefore, the leader searching strategy has drawbacks of premature convergence and low searching precision. The parameter c_2 is randomly selected in the range of $[0, 1]$, which is not suitable for high-precision searching in the later of iterations. Positions of other salps are updated by the average position of each follower and its neighbor. Updated positions of followers have a single direction and blindness, which cause that SSA falls into the local extremum and maximize the viciousness of iterations. To get a better searching strategy and avoid the blindness of the searching process, this paper added the variable perturbation weight mechanism into the basic SSA, and the proposed algorithm is called the perturbation weight salp swarm algorithm (PWSSA).

The perturbation weight mechanism works by changing the distance difference of the optimum solution and the population solution. The searching range is regulated by applying the asymptotic circular searching to obtain a better leader searching strategy with a faster speed and higher balance. The followers' position will achieve better, and the position adjusting will change more and more slightly with increasing iterations. The perturbation weight mechanism will make SSA get the optimum solution better. New factors c_1 and c_2 can be updated as follows:

$$c_{1\mathrm{new}} = u_1 \cdot \left(1 - \frac{t}{T}\right), \tag{5}$$

$$c_{2\mathrm{new}} = u_2 \cdot \left(1 - \frac{t}{T}\right), \tag{6}$$

Input: fitness function $F(\cdot)$. Dimension d. Population size N. Each i^{th} salp position x_d^i. Best position F_d. Best function value F^*. $[\text{lb}_d \cdot \text{ub}_d]$. $t = 0$. Set T and p.
Output: F^*.
while ($t < T$)
  $c_1 = 2e^{-(4t/T)}$
  $c_2 \in [0, 1]$
  $c_3 \in [0, 1]$
  for 1 ($i = 1 : N$)
    if 1 ($i = 1$)
      if 2 ($c_3 \geq P$)
      $x_d^1 = F_d + c_1$ $((\text{ub}_d - \text{lb}_d)c_2 + \text{lb}_d)$
        else
      $x_d^1 = F_d - c_1$ $((\text{ub}_d - \text{lb}_d)c_2 + \text{lb}_d)$
      end if 2
    else
      $x_d^i = (x_d^i + x_d^{i-1})/2$
    end if 1
  if 3 F (x_d^i) is better than F^*
      $F_d = x_d^i$
      $F^* = F$ (x_d^i)
    end if 3
  end for 1
  $t = t + 1$
end while

Algorithm 1: SSA.

where u_1 and u_2 meet the standard normal distribution, $u_1 \sim N(0, 1)$ and $u_2 \sim N(0, 1)$. The standard normal fraction has advantages of the concentration, the symmetry, and the uniform variability.

The new leader position can be updated as follows:

$$x_d^1 = \begin{cases} F_d + c_{1\text{new}}\left(\left(F_d - x_d^1\right)c_{2\text{new}} + \text{lb}_d\right), & c_3 \geq p, \\ F_d - c_{1\text{new}}\left(\left(F_d - x_d^1\right)c_{2\text{new}} + \text{lb}_d\right), & c_3 < p. \end{cases} \tag{7}$$

To increase diversities of followers' positions, the multidirectional crossing searching strategy is added in the basic SSA:

$$x_d^i = w_1\left[\left(w_2 \cdot F_d - x_d^i\right) + \left(w_3 \cdot F_d - x_d^{i-1}\right)\right], \tag{8}$$

where w_1, w_2, and w_3 are random parameters in the range of $[-1, 1]$.

The specific steps of PWSSA are described as follows:

Step 1. Set salp population size N and the searching dimension D. Define the maximum number of iterations T. Determine probability coefficient p. Let $t = 0$. Each i^{th} salp position can be seen as x_d^i (i=1,2,. . .,N) and (d=1,2,. . .,D). Set the probability coefficient p.

Step 2. Begin the iteration. Judge whether i is equal to one. If i is equal to one, jump into Step 3. If i is not equal to one, jump into Step 4.

Step 3. Use equation (5) to compute the factor $c_{1,\text{new}}$. Use equation (6) to compute the factor $c_{2,\text{new}}$. Set c_3 in the range of [0, 1]. Compute the current optimal solution F_d. Judge whether c_3 is larger than p. If c_3 is larger than p, the leader position can be expressed by part one of equation (7). Otherwise, the leader position can be expressed by part two of equation (7).

Step 4. Set parameters w_1, w_2, and w_3 in the range of $[-1, 1]$. Update followers' positions using equation (8).

Step 5. Record the global optimal solution. If there is a better solution, replace F_d.

Step 6. Set $t = t + 1$. Judge whether the current iteration t is equal to the maximum number of iterations T; if t is equal to T, stop the iteration. If not, jump to Step 2.

The PWSSA main step can be summarized in the pseudocode shown in Algorithm 2, and the PWSSA main step flow chart is shown in Figure 1.

4. Results and Discussion

4.1. Experimental Parameters and Environments. Benchmark function testing is a popular and common way to indicate the performance of intelligent algorithms. This paper introduces benchmark functions to exhibit the superior performance of the proposed algorithm, and the proposed algorithm will be evaluated on classical benchmark functions in this section. To testify the ability of the proposed algorithm to solve different dimensional complex functions, eight low-dimension functions (f_1–f_8) and four variable-dimension functions (f_9–f_{12}) were chosen for algorithm testing in Table 1. In Table 1, D, scope, and aim represent the function dimension, the searching range, and the ideal value.

Low-dimension functions (f_1–f_8) are applied to measure the global searching ability of the algorithm. Variable-dimension functions (f_9–f_{12}) are very difficult to converge to the global optimal solution because of owning unevenly

Input: fitness function $F(\cdot)$. Dimension d. Population size N. Each i^{th} salp position x_d^i. Best position F_d. Best function value F^*. $[\text{lb}_d \cdot \text{ub}_d]$. $t = 0$. Set T and p.
Output: F^*.
while $(t < T)$
  $c_{1\text{new}} = u_1\ (1 - t/T)$
  $c_{2\text{new}} = u_2\ (1 - t/T)$
  $c_3 \in [0,\ 1]$
  for 1 $(i = 1 : N)$
    if 1 $(i = 1)$
      if 2 $(c_3 \geq P)$
      $x_d^1 = F_d + c_{1\text{new}}\ ((F_d - x_d^1)c_{2\text{new}} + \text{lb}_d)$
        else
      $x_d^1 = F_d - c_{1\text{new}}\ ((F_d - x_d^1)c_{2\text{new}} + \text{lb}_d)$
      end if 2
        else
      $x_d^i = w_1[(w_2 \cdot F_d - x_d^i) + (w_3 \cdot F_d - x_d^{i-1})]$
    end if 1
  if 3 $F\ (x_d^i)$ is better than F^*
      $F_d = x_d^i$
      $F^* = F\ (x_d^i)$
    end if 3
  end for 1
  $t = t + 1$
end while

Algorithm 2: PWSSA.

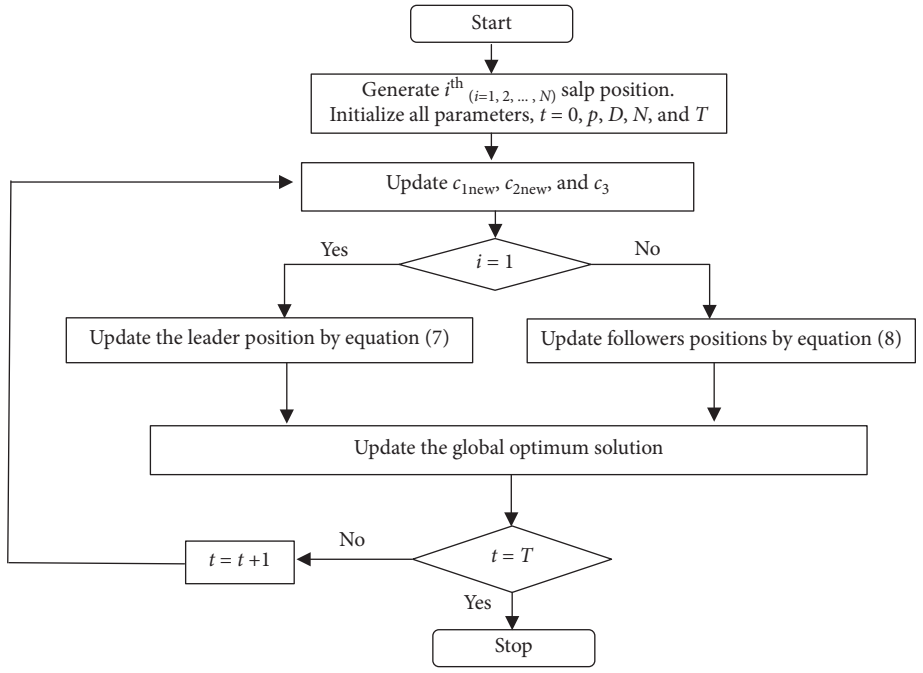

Figure 1: The flowchart for PWSSA.

distributed local optima points with strong oscillation and nonconvexity, especially in the case of being large-scale and high-dimension functions. To test the proposed algorithm in multisides, dimensions of variable-dimension functions were selected 2D and 100D in this paper. In the original SSA literature, authors compared SSA with seven popular algorithms, and performances of SSA are better than comparison algorithms. To avoid repeat and unnecessary experiments, this paper selected other algorithms to carry on comparative experiments. Comparison algorithms included the SSA, simulated annealing (SA) [54], Lévy flight trajectory-based whale optimization algorithm (LWOA) [55], and

Table 1: Basic information on benchmark functions.

Name	Formulation	D	Scope	A_{im}
Brent	$f_1(x) = (x_1 + 10)^2 + (x_2 + 10)^2 + e^{-x_1^2 - x_2^2}$	2	[−200, 200]	0
Cube	$f_2(x) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$	2	[−200, 200]	0
Egg crate	$f_3(x) = x_1^2 + x_2^2 + 25(\sin^2(x_1) + \sin^2(x_2))$	2	[−50, 50]	0
Himmelblau	$f_4(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	2	[−100, 100]	0
Leon	$f_5(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$	2	[−100, 100]	0
Levy13	$f_6(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2[1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + \sin^2(2\pi x_2)]$	2	[−100, 100]	0
Matyas	$f_7(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	[−100, 100]	0
Rotated ellipse	$f_8(x) = 0.7x_1^2 - 6\sqrt{3}x_1x_2 + 13x_2^2$	2	[−50, 50]	0
Ackley	$f_9(x) = -20\exp(-0.2\sqrt{(1/D)\sum_{i=1}^{D} x_i^2}) - \exp((1/D)\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + \exp(1)$	2/100	[−50, 50]	0
Levy	$f_{10}(x) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1}(w_i - 1)^2[1 + 10\sin^2(\pi w_i + 1)] + (w_D - 1)^2[1 + \sin^2(2\pi w_D)],\ w_i = 1 + ((x_i - 1)/4)$	2/100	[−50, 50]	0
Sphere	$f_{11}(x) = \sum_{i=1}^{D} x_i^2$	2/100	[−10, 10]	0
Sum of different powers	$f_{12}(x) = \sum_{i=1}^{D} \lvert x_i\rvert^{i+1}$	2/100	[−10, 10]	0

Lévy flight salp swarm algorithm (LSSA) [41]. All algorithm processes and details can be found in the original algorithm literature.

SA is inspired by analogy to the physical annealing procedure in metals, which is a local searching algorithm proposed in the early 1980s. The theory of the annealing procedure is to heat the solid-state metal to a large temperature, so that atoms of the metal are in a stochastic condition and then cool the metal down slowly according to particular procedures. Starting from some random solutions and fixed initial temperature, SA controls the process by metropolis criterion and a group of parameters called cooling schedule. SA has two initial parameters including the initial temperature T_0 and the decay factor k. For SA, parameter T_0 selects 100, and parameter k selects 0.95.

LWOA, which was proposed by Ling et al. in 2017, combines a Lévy flight trajectory and whale optimization algorithm to get a better trade-off between the exploration and exploitation for the basic whale optimization algorithm. Lévy flight is a special random searching path where walking steps are selected according to heavy power-law tails. LWOA has five initial parameters including r, b, l, p, and β. For LWOA, r and p are random numbers in [0, 1], l is automatically selected in [0, 1], $b = 2$, and $\beta = 1.5$.

LSSA was proposed by Xing and Jia in 2019. Lévy flight trajectory can not only maximize the diversity of searching domains but also enhance the global searching ability of SSA to avoid getting into local optimal values. There are two parameters in LSSA, including the power-law exponent β and the probability factor P. For LSSA, $\beta = 1.5$ and $P = 0.5$.

For PWSSA and SSA, parameter p equals to 0.5. Initial parameter values of all algorithms were chosen according to original algorithm literature, and all algorithms processes and details can be found in the original algorithm literatures. For each experiment of an algorithm on different benchmark functions, ten independent runs were performed to get a fair comparison among different algorithms. The maximum number of iterations was set to 1000, and the population size was set to 50. The best value, the worst value, the average value, and the standard deviation of each algorithm optimization were recorded. To make a fair comparison, all algorithms were programmed in MATLAB (R2014b, The MathWorks, Inc, Natick, MA, USA). All experiments were conducted on a laptop with Intel (R) Core (TM) i5-4210U CPU, 2.30 GHz, 4 GB RAM. All data and figures were completed in MATLAB (R2014b, The MathWorks, Inc, Natick, MA, USA).

4.2. Date Discussion. To demonstrate the optimization effect, four indicators were selected to comprehensively evaluate the competitiveness of different algorithms. Four indicators consist of the best searching value (best), the worst searching value (worst), the median (med), and the standard deviation (std). Fixed two-dimension functions testing results and two dimensions of variable-dimension functions testing results are given in Table 2. Other variable-dimension functions (100D) testing results are shown separately in Table 3. Tables 2 and 3 show that all searching values of the proposed algorithm are much closer to the ideal value in Table 1, which demonstrates that PWSSA not only can obtain the best aim but also have strong searching abilities. As the dimension of the testing function increases, the accuracy of the algorithms will decline, but the test results using PWSSA are consistently better than those using other algorithms. The convergence precision and optimization success ratio of the proposed algorithm is better than those of the other algorithms for all test functions. When a set of data changes significantly, the median can be used to illustrate the centralized trend of the data. PWSSA has the smallest median value of all the test results, indicating an outstanding performance compared with the other algorithms. PWSSA also has the smallest standard deviation of all the algorithms, demonstrating that the proposed algorithm offers good stability and produces relatively few poor results. Standard deviation, which can measure the discrete degree of a dataset, is the arithmetic square root of the variance. In other words, a large standard deviation exhibits a large difference between most values and the average value, and a small standard deviation shows that the calculated value is closer to their average value. PWSSA has the smallest standard deviation than those of other algorithms, which display that the proposed algorithm has good stability and a few poor results. In PWSSA, the good solution in the current iteration is applied by followers to find the better solution in the next iteration, and random factors can enhance diversities of solutions in nonlinear high-dimension problems, so it can be seen from testing results that in f_7, f_8, and f_{12}, PWSSA can achieve the best optimization results on best, worst, mean, and std values.

4.3. The Wilcoxon Rank Sum Test Discussion. The rank sum test is a nonparametric technique used to define whether a result is statistically significant. The nonparametric statistical test can be used in mathematics fields to check the algorithm performance [56]. The rank sum test arranges all data in order, from small to large, and has strong practicality because there is no special form of dispersed data or known distribution. However, the rank sum test ignores absolute value differences in data testing, which not only makes the test result approximate but also causes the loss of some test information. Wilcoxon improved the basic rank sum test by considering the different directions and sizes of the data. The Wilcoxon rank sum test can be applied to a distribution of data to check any differences among them and offers more effective performance than the basic rank sum test. The Wilcoxon rank sum test produces p values: if the p value is less than 0.05, there is a significant difference at a level of 0.05. To further compare PWAAS performances with those of other algorithms, the Wilcoxon signed rank test was tested in this paper. All p values are given in Table 4, and this paper applied the proposed algorithm results against those of other algorithms at the 0.05 significance level. For SSA, the p values of function 1 and function 6 are equal to 0.011 and are larger than 0.05. For SA, the p values of function 4 is equal to 0.473 and is larger than 0.05. For LWOA, the p values of function 2 and

Table 2: Comparison of testing results for functions ($D = 2$).

Function	Index	Algorithm				
		PWSSA	SSA	SA	LWOA	LSSA
f_1	Best	2.59854$E-07$	1.03505$E-06$	3.80418$E-05$	1.76767$E-05$	1.62809$E-05$
	Worst	3.01968$E-05$	0.01103	7.43357$E-04$	0.00122	0.02305
	Med	1.97725$E-05$	6.51329$E-04$	2.39672$E-04$	3.45484$E-04$	0.00367
	Std	9.78409$E-05$	0.00381	2.22456$E-04$	3.90852$E-04$	0.00868
f_2	Best	9.06274$E-09$	3.98052$E-06$	0.00177	2.57169$E-05$	5.59844$E-05$
	Worst	0.00125	18.45495	1.12698	2.38152	9.20663
	Med	7.22103$E-05$	0.16124	0.01565	5.99209$E-04$	0.00964
	Std	4.52172$E-04$	6.31572	0.42126	0.87229	2.90774
f_3	Best	0	2.46394$E-05$	0.00136	5.0005$E-06$	2.77604$E-04$
	Worst	1.58274$E-28$	9.78594	37.97164	3.49800$E-04$	0.027876542
	Med	2.27881$E-30$	0.00317	9.56406	4.98979$E-05$	6.94903$E-04$
	Std	5.73257$E-29$	4.06975	14.74794	1.14505$E-04$	0.00858
f_4	Best	9.77441$E-05$	0.00285	4.40269$E-05$	1.05692$E-04$	1.26063$E-04$
	Worst	0.01444	5.97614	0.00563	0.01279	0.82895
	Med	5.39004$E-04$	0.19535	9.18479$E-04$	0.00275	0.10037
	Std	0.00492	2.24549	0.00165	0.00363	0.31098
f_5	Best	1.55739$E-09$	1.77563$E-05$	1.62255$E-04$	2.26596$E-06$	2.38314$E-05$
	Worst	4.97218$E-04$	2.69243	0.00164	8.58795$E-04$	0.20599
	Med	2.23530$E-05$	0.01289	7.99437$E-04$	3.13375$E-05$	0.02672
	Std	1.52718$E-04$	0.85991	4.86968$E-04$	2.98953$E-04$	0.07261
f_6	Best	5.49332$E-08$	1.6071$E-07$	0.00115	1.02565$E-05$	2.62883$E-07$
	Worst	2.09433$E-04$	0.11007	0.03757	1.37803$E-04$	0.00534
	Med	1.44421$E-05$	9.5265$E-04$	0.00532	8.72917$E-05$	6.50038$E-04$
	Std	6.36087$E-05$	0.04318	0.01402	4.44597$E-05$	0.00205
f_7	Best	0	1.34499$E-06$	5.73498$E-07$	1.19744$E-08$	1.40161$E-09$
	Worst	5.99251$E-30$	1.30251$E-04$	5.94346$E-06$	6.66386$E-07$	7.96877$E-06$
	Med	1.81891$E-32$	4.21287$E-06$	2.21515$E-06$	1.01826$E-07$	9.11214$E-07$
	Std	1.87875$E-30$	4.77821$E-05$	1.6056$E-06$	1.9426$E-07$	2.36125$E-06$
f_8	Best	0	2.03368$E-06$	3.47271$E-05$	7.18475$E-06$	7.18865$E-06$
	Worst	2.27824$E-29$	0.00228	8.13039$E-04$	1.67183$E-04$	3.95153$E-04$
	Med	1.59887$E-31$	0.00133	1.25073$E-04$	4.12726$E-05$	1.22018$E-04$
	Std	7.51732$E-30$	6.65682$E-04$	3.11043$E-04$	5.63652$E-05$	1.40560$E-04$
f_9	Best	8.88178$E-16$	0.00952	0.00281	7.94444$E-05$	742906$E-04$
	Worst	7.99361$E-15$	0.34629	0.13792	0.00999	0.04047
	Med	8.88178$E-16$	0.04335	0.02221	0.00126	0.00975
	Std	2.99591$E-15$	0.11375	0.05041	0.00289	0.01123
f_{10}	Best	2.39978$E-08$	1.21556$E-06$	2.50954$E-05$	2.22576$E-07$	5.94347$E-08$
	Worst	4.64332$E-06$	1.16567$E-04$	18.31331	4.86683$E-06$	4.50717$E-05$
	Med	6.04547$E-07$	5.86534$E-05$	3.18643	8.57693$E-07$	8.81307$E-06$
	Std	1.38693$E-06$	3.70681$E-05$	6.67089	1.52368$E-06$	1.33158$E-05$
f_{11}	Best	2.67505$E-30$	8.45269$E-08$	6.07517$E-06$	1.27075$E-07$	1.4559$E-09$
	Worst	4.22997$E-27$	6.40967$E-05$	6.85882$E-05$	5.10776$E-06$	1.59526$E-06$
	Med	8.13894$E-28$	4.51534$E-06$	2.80607$E-05$	9.86996$E-07$	1.22476$E-07$
	Std	1.40801$E-27$	2.2201$E-05$	2.19078$E-05$	1.84301$E-06$	6.34252$E-07$
f_{12}	Best	0	1.16251$E-08$	1.89741$E-09$	2.61773$E-10$	5.38026$E-13$
	Worst	3.14782$E-31$	9.51884$E-06$	1.087$E-05$	1.33367$E-07$	4.76809$E-07$
	Med	1.05963$E-35$	2.58401$E-07$	1.45518$E-06$	1.36857$E-08$	1.36892$E-07$
	Std	1.03358$E-31$	2.92926$E-06$	3.23044$E-06$	4.08975$E-08$	1.60624$E-07$

functions 4–6 are larger than 0.05. For LSSA, the p values in function 6 is equal to 0.011 and is larger than 0.05. Other results are all less than 0.05. The Wilcoxon rank sum test results display that the proposed algorithm owns the strongest searching efficiency and the greatest finding mechanism around the best solution, which further proves that PWSSA has the wonderful searching performance.

4.4. Iteration Curves Discussion. Iteration is the activity of repeating a feedback procedure with the purpose of finding the desired goal. Each repetition of all procedures in an algorithm is called one iteration, and the result of each iteration provides the initial value for the next iteration. To exhibit the convergence speed and global search ability of all algorithms more intuitively, the average convergence curves of all

Table 3: Comparison of testing results for functions ($D = 100$).

Function	Index	Algorithm				
		PWSSA	SSA	SA	LWOA	LSSA
$f_{9(D=100)}$	Best	$8.88178E-16$	0.02270	21.26791	$2.07645E-05$	0.00985
	Worst	$4.44089E-15$	0.22181	21.32175	0.002352706	0.04736
	Med	$8.88178E-16$	0.07466	21.30594	$4.72716E-04$	0.01632
	Std	$1.12347E-15$	0.06114	0.02102	$9.\ 50222E-04$	0.01411
$f_{10(D=100)}$	Best	$3.55168E-08$	$2.65636E-04$	12367.37563	7026.80204	$1.65828E-06$
	Worst	$3.00205E-04$	0.08699	15156.57782	19124.12214	0.00753
	Med	$6.80021E-06$	0.02705	13920.22547	15014.00730	0.00335
	Std	$9.50020E-05$	0.03267	863.08271	3611.41568	0.00251
$f_{11(D=100)}$	Best	$4.92091E-20$	$1.00288E-04$	150.17481	18.04087	$6.10563E-06$
	Worst	$2.16703E-18$	0.00917	186.65021	53.76659	$3.35757E-04$
	Med	$4.09074E-19$	0.00125	178.64669	27.58041	$5.93374E-05$
	Std	$6.98319E-19$	0.00265	10.44040	13.99187	$9.57910E-05$
$f_{12(D=100)}$	Best	0	$1.68224E-09$	$2.69963E+46$	$1.69524E-11$	$6.33227E-10$
	Worst	$8.02130E-31$	$2.22047E-06$	$7.39875E+52$	$2.85668E-07$	$1.04169E-06$
	Med	$2.28378E-33$	$1.57269E-07$	$1.66322E+50$	$5.03202E-09$	$2.29748E-08$
	Std	$2.50569E-31$	$7.15649E-07$	$2.31712E+52$	$9.25094E-08$	$3.53412E-07$

Table 4: Comparison of the Wilcoxon rank sum test.

Function	Algorithm			
	SSA	SA	LWOA	LSSA
f_1	0.011	0.004	0.004	0.002
f_2	0.002	$1.83E-04$	0.076	0.003
f_3	$1.79E-04$	$1.79E-04$	$1.79E-04$	$1.79E-04$
f_4	$5.83E-04$	0.473	0.186	0.008
f_5	$7.69E-04$	$4.40E-04$	0.273	0.001
f_6	0.011	$1.83E-04$	0.021	0.011
f_7	$1.63E-04$	$1.63E-04$	$1.63E-04$	$1.63E-04$
f_8	$1.73E-04$	$1.73E-04$	$1.73E-04$	$1.73E-04$
$f_{9(D=2)}$	$1.48E-04$	$1.48E-04$	$1.48E-04$	$1.48E-04$
$f_{10(D=2)}$	$4.40E-04$	$1.83E-04$	0.273	0.002
$f_{11(D=2)}$	$1.83E-04$	$1.83E-04$	$1.83E-04$	$1.83E-04$
$f_{12(D=2)}$	$1.63E-04$	$1.63E-04$	$1.63E-04$	$1.63E-04$
$f_{9(D=100)}$	$8.74E-05$	$8.74E-05$	$8.74E-05$	$8.74E-05$
$f_{10(D=100)}$	$2.46E-04$	$1.83E-04$	$1.83E-04$	0.001
$f_{11(D=100)}$	$1.83E-04$	$1.83E-04$	$1.83E-04$	$1.83E-04$
$f_{12(D=100)}$	$1.73E-04$	$1.73E-04$	$1.73E-04$	$1.73E-04$

algorithms applied to functions of different dimensions are displayed in Figures 2 and 3. Single logarithmic coordinates are used in this paper for a more detailed analysis. Figure 2 exhibits two-dimensional convergence curves of PWSSA and its competitors. Figure 3 demonstrates iteration graphs of algorithms at variable-dimension functions (100D), respectively. Note that all convergence curves discussed in the following subsections are the averages of ten independent executions. As the dimension increases, the optimization performance and iteration speed of all algorithms decrease, although the performance degradation of PWSSA is not severe. The proposed algorithm achieves the target value for most functions with the fastest iteration speed and highest efficiency. The LSSA has better iteration rates than SSA in most functions but still cannot outperform the proposed algorithm. It is noticeable that PWSSA gives the superior global iteration rate and accuracy in comparison with original SSA, which is easy to be trapped to the local optimal. All figures reveal that SSA will be much poorer as the dimension increases, while the proposed algorithm still can offer the distinguished searching ability and its convergence speed and precision rank number one in all functions. In other words, PWSSA can apply fewer iterations to solve problems and is more competent than other algorithms. PWSSA enormously boosts the iteration speed and searching ability of basis SSA mainly because of the introduction of the many-sided learning and local random perturbation strategies between successive followers positions.

4.5. Box-Plot Charts Discussion. Box-plot charts are used to show dispersion information about a set of data. They have the advantages of detecting abnormal values and data skewness and are widely used to distinguish algorithm capabilities in terms of data symmetry and data dispersion. There are six parameters in a typical box-plot chart, namely, the maximum value, minimum value, median, upper quartile, lower quartile, and outliers. A set of data can be evaluated using five of these parameters. Figures 4 and 5 show box-plot charts of all algorithms after calculating a different function. There are many local optima in high-dimensional functions, so the aggregation degree of solutions is a crucial index for evaluating algorithm performance. If an algorithm becomes trapped around a local extremum, it can result in premature convergence. PWSSA produces the narrowest box-plot charts and fewest outliers for all functions. The median and upper/lower quartiles computed by PWSSA are lower than those given by the other algorithms, demonstrating that the collaborative random search strengthens the capability for individual diversity and avoids premature convergence. It is apparent that the proposed algorithm tends to obtain the best performance in precision on most functions as the dimension increases, which is mainly contributed by followers' random positions generated, and SA and LWOA have the worst performances. SA has the largest variance in all algorithms. All box-plot

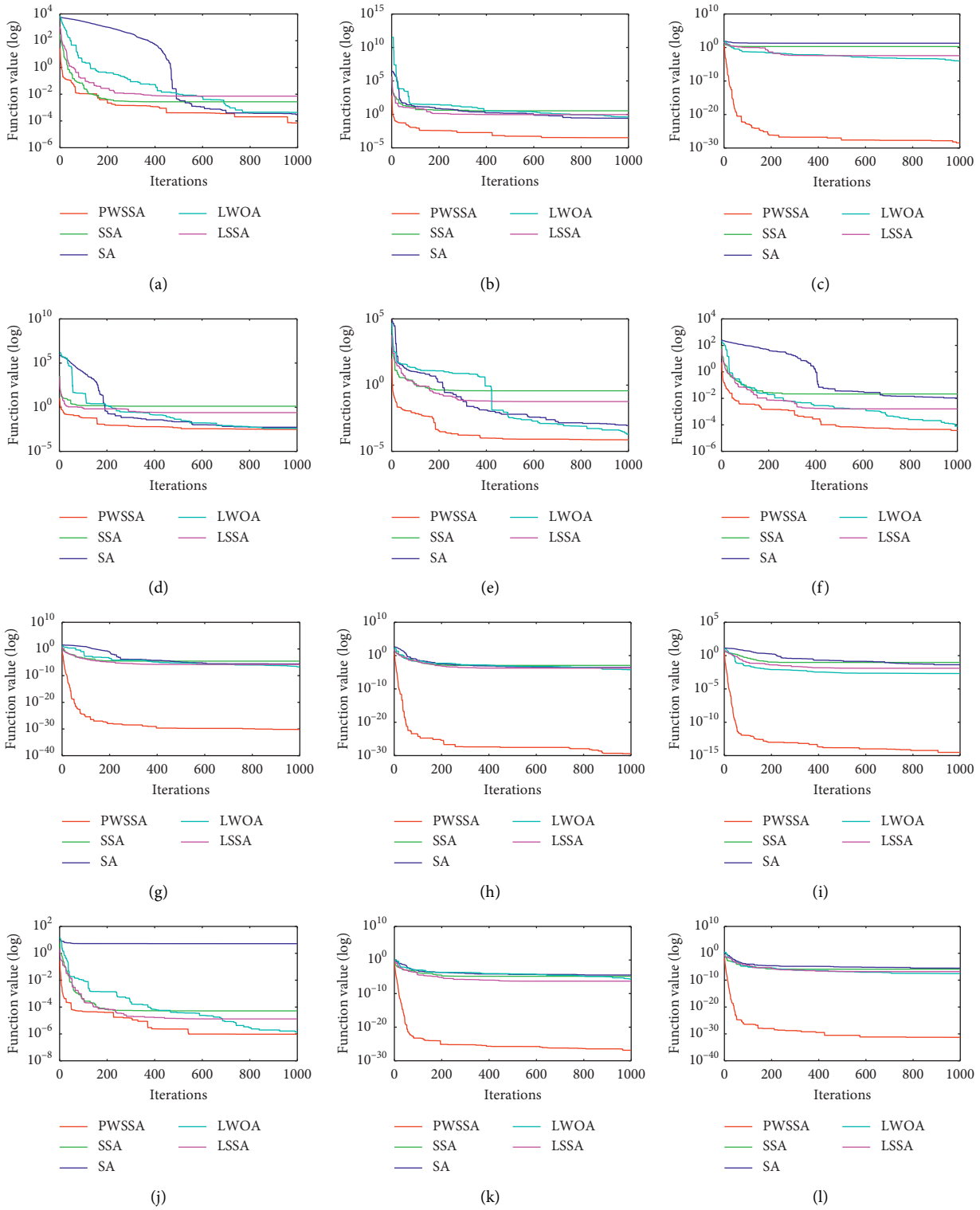

Figure 2: Convergence curves for functions ($D = 2$). (a) f_1. (b) f_2. (c) f_3. (d) f_4. (e) f_5. (f) f_6. (g) f_7. (h) f_8. (i) $f_{9(D=2)}$. (j) $f_{10(D=2)}$. (k) $f_{11(D=2)}$. (l) $f_{12(D=2)}$.

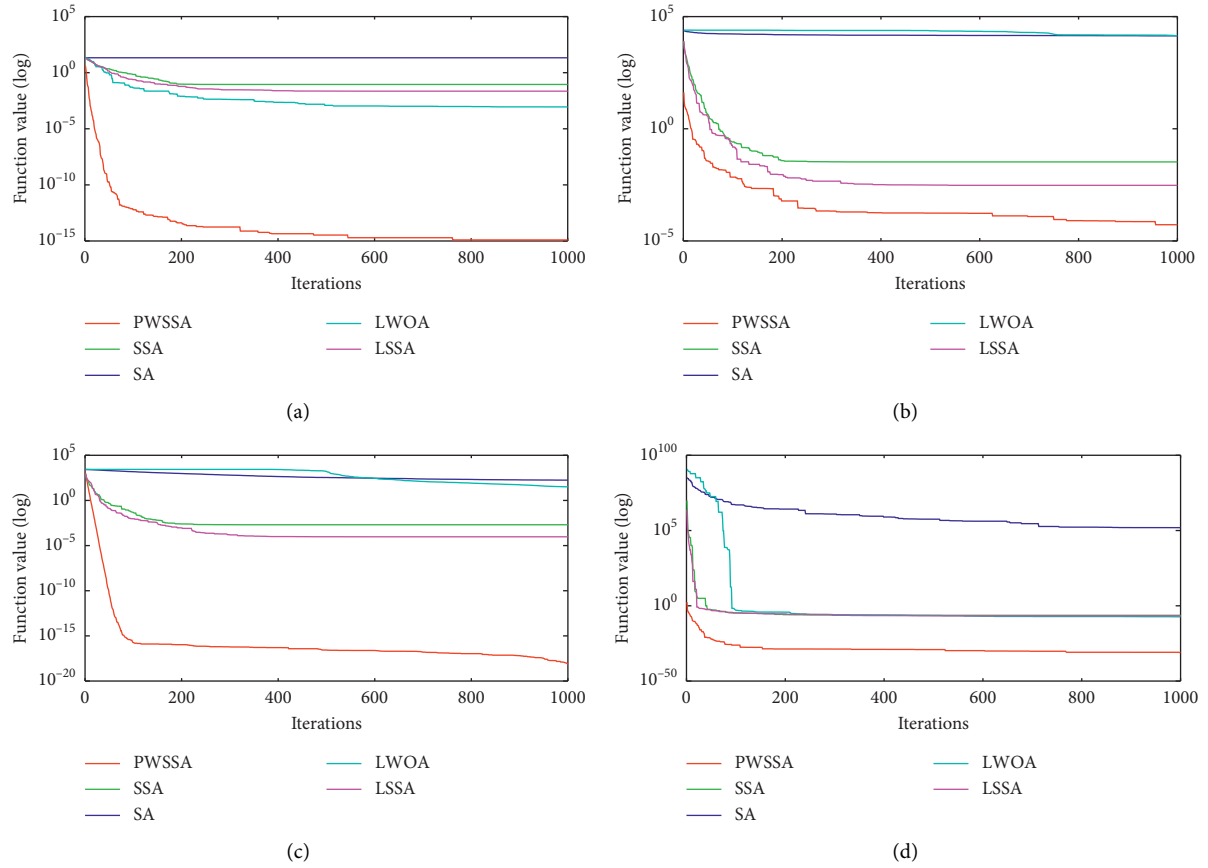

Figure 3: Convergence curves for functions ($D = 100$). (a) $f_{9(D=100)}$. (b) $f_{10(D=100)}$. (c) $f_{11(D=100)}$. (d) $f_{12(D=100)}$.

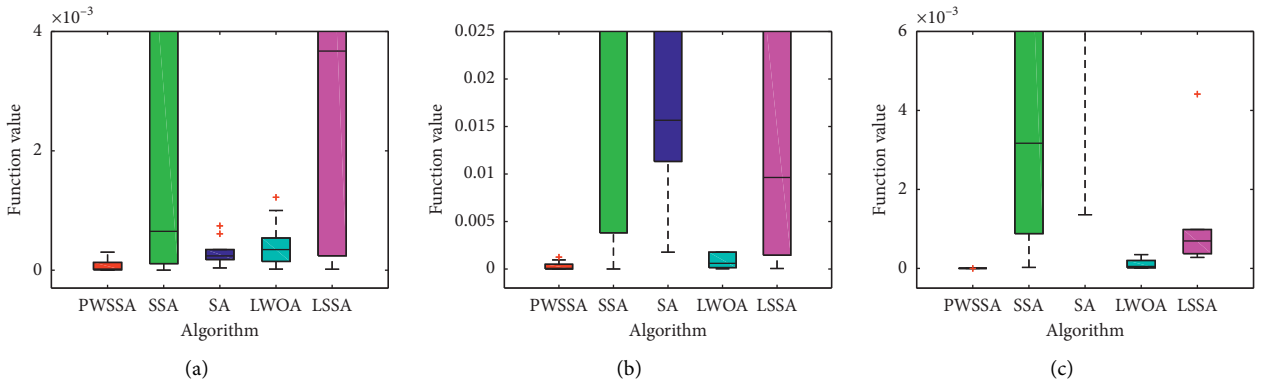

Figure 4: Continued.

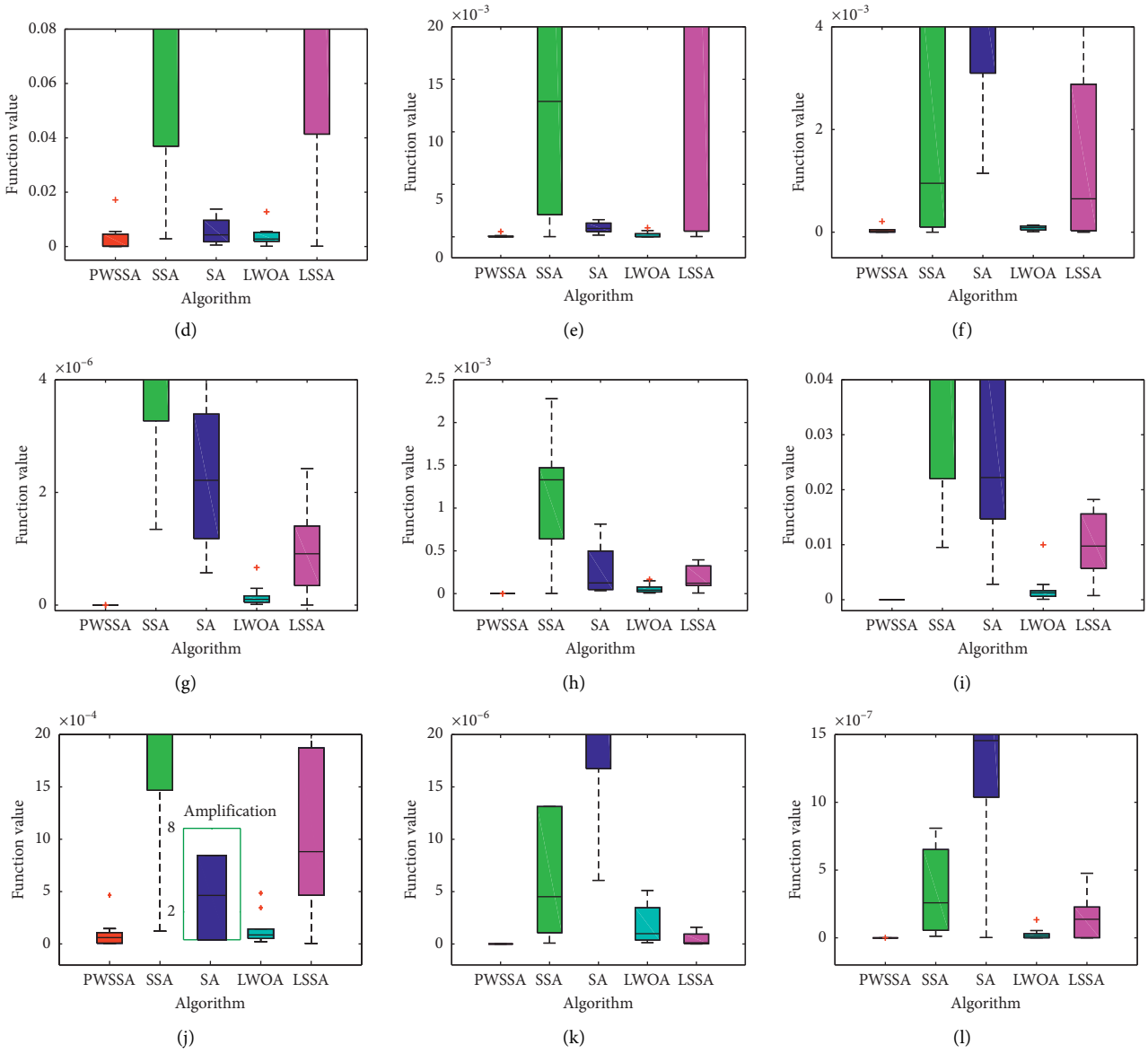

Figure 4: Box-plot charts of functions ($D = 2$). (a) f_1. (b) f_2. (c) f_3. (d) f_4. (e) f_5. (f) f_6. (g) f_7. (h) f_8. (i) $f_{9(D=2)}$. (j) $f_{10(D=2)}$. (k) $f_{11(D=2)}$. (l) $f_{12(D=2)}$.

charts demonstrate that the proposed algorithm has large robust and big stability in comparison with other algorithms, and the figures can show that PWSSA can avoid local extremum.

4.6. Searching Paths Discussion. To further discuss the powerful searching capability and optimization performance in PWSSA, Figure 6 gives the optimal PWSSA search path, the optimal SSA search path, and the contour plot of each function in the two-dimension plane.

Searching path figures can examine whether the algorithm will fall into the local optimal solution on complex functions. Through comparison of searching paths with the traditional SSA algorithm, all searching paths of PWSSA are shorter than SSA, which demonstrate the efficiency of PWSSA in function problems. PWSSA also applies the finding mechanism of tightening from the neighborhood to the extreme point due to average optimality and constrained average optimality. From Figure 6, we can find that the PWSSA searching path is much less than the SSA searching path; SSA has many repeat-invalid short-distance searching paths and occasional long-distance searching paths. Two sets of searching paths display that compared with the basic SSA, PWSSA can explore a wider range and is less affected by iterations, so PWSSA owns better general-purpose optimization abilities. PWSSA also is more flexible and can not only completely avoid collisions with obstacles but also provide numerous feasible solutions. The proposed algorithm can balance the searching speed and accuracy and provide a brilliant and satisfactory solution as much as possible in the case of meeting variable-demand requirements. All searching

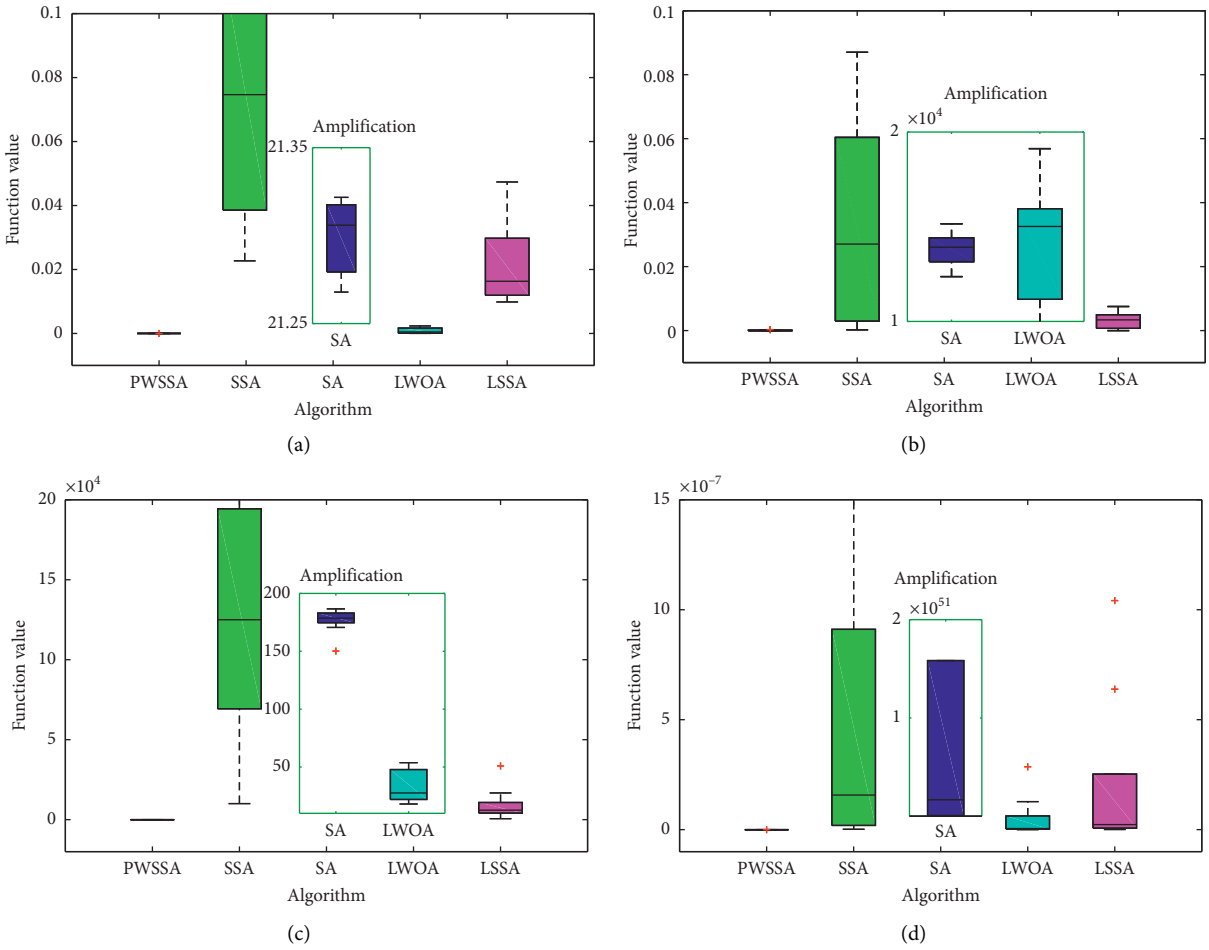

Figure 5: Box-plot charts of functions ($D = 100$). (a) $f_{9(D=100)}$. (b) $f_{10(D=100)}$. (c) $f_{11(D=100)}$. (d) $f_{12(D=100)}$.

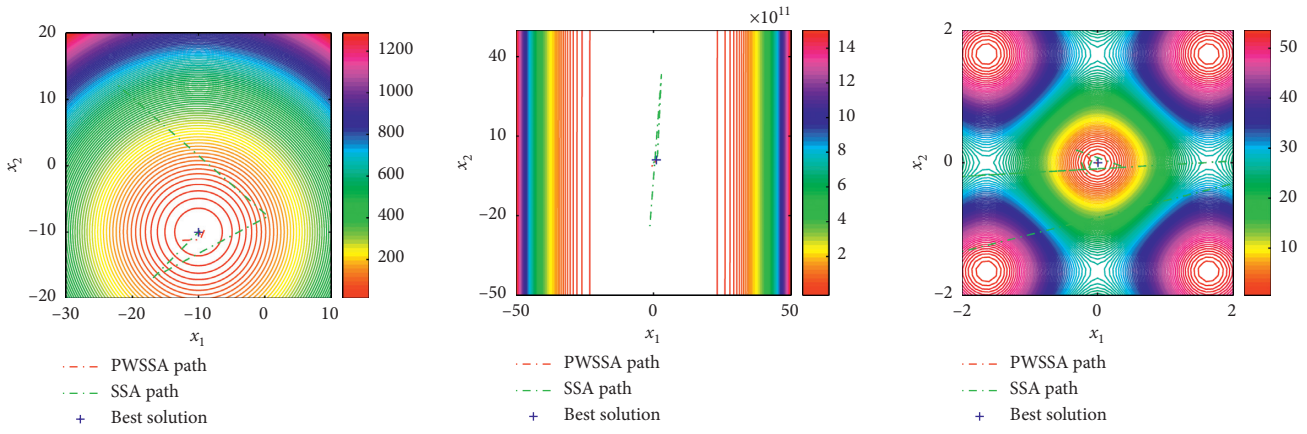

Figure 6: Continued.

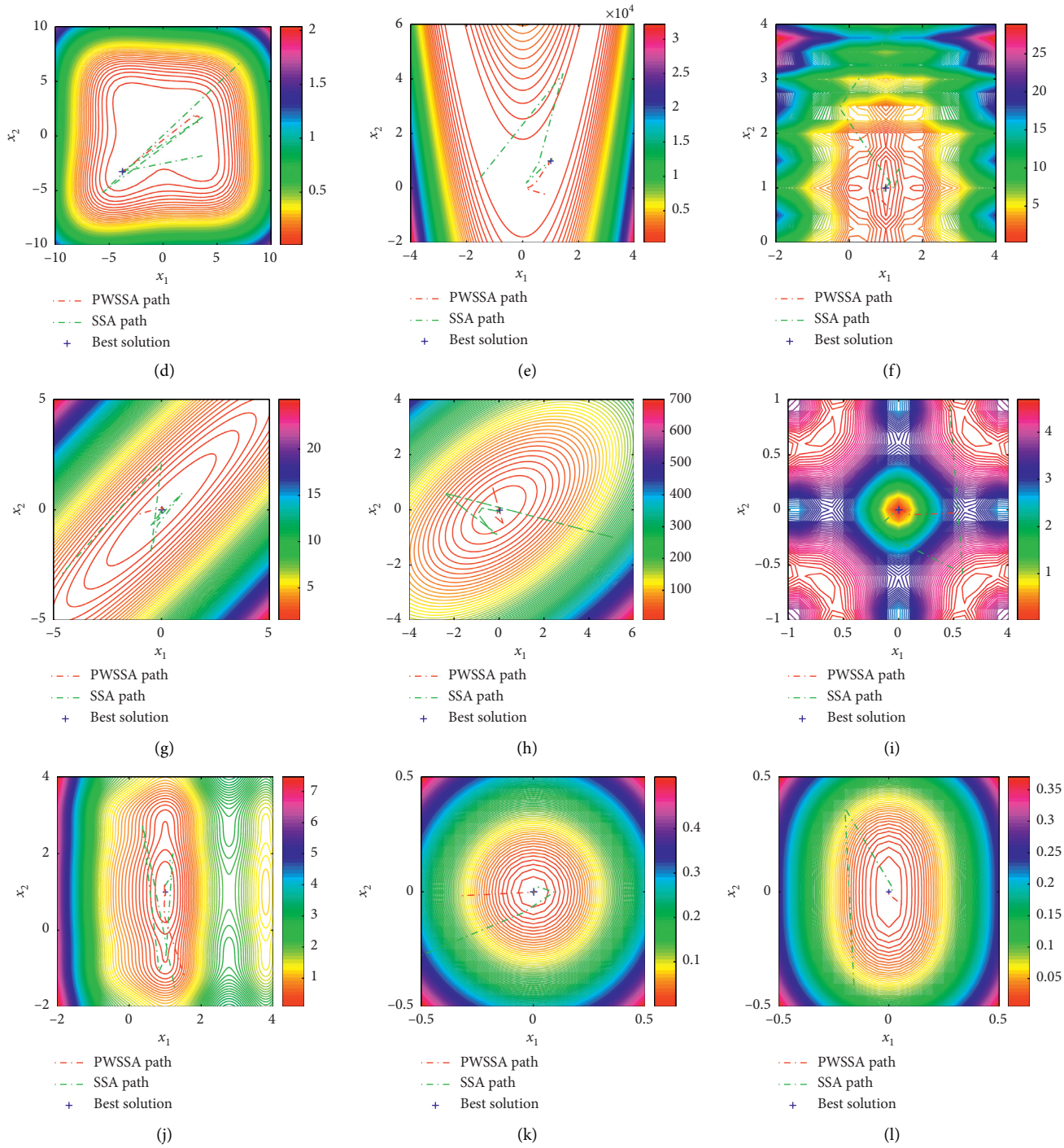

Figure 6: Comparison of searching paths. (a) f_1. (b) f_2. (c) f_3. (d) f_4. (e) f_5. (f) f_6. (g) f_7. (h) f_8. (i) $f_{9(D=2)}$. (j) $f_{10(D=2)}$. (k) $f_{11(D=2)}$. (l) $f_{12(D=2)}$.

paths results can reveal that PWSSA can quickly get the best solution and can be used to scenarios with high requirements in a real-time environment.

4.7. Time Complexity Discussion. The algorithm complexity is divided into time complexity and space complexity. The time complexity refers to the computational workload required to execute the algorithm, and the spatial complexity refers to the memory space in a computer required to execute the algorithm. In computer science, the algorithm time complexity, which is a function, qualitatively describes the algorithm running time and is usually expressed by symbol $O(f(n))$, where $f(n)$ means the mathematical function which includes n, n^2, and $\log^n$. In this way, the time complexity can be called asymptotic

Table 5: Comparison of the number of running times.

Function	Index	Algorithm				
		PWSSA	SSA	SA	LWOA	LSSA
f_1	MIN	9	38	51	31	33
	MAX	116	215	784	229	157
	AVE	37.3	116.1	539.6	112.6	88.2
f_2	MIN	2	14	35	18	7
	MAX	6	124	147	292	326
	AVE	3.3	51.1	93	109.5	118.7
f_3	MIN	1	2	38	4	1
	MAX	2	12	544	39	3
	AVE	1.4	3.7	360.9	17.7	1.6
f_4	MIN	2	5	9	13	7
	MAX	6	29	89	84	36
	AVE	3.9	16.9	46	41.3	19.1
f_5	MIN	2	5	16	10	3
	MAX	5	137	186	289	106
	AVE	2.9	43.9	103.2	147.4	33.9
f_6	MIN	5	40	29	10	12
	MAX	14	116	897	143	82
	AVE	9.7	64.1	386.1	54.4	53.5
f_7	MIN	4	39	43	12	22
	MAX	9	140	93	184	53
	AVE	6.2	92.9	350.1	69.5	34.2
f_8	MIN	2	47	65	127	52
	MAX	6	240	855	464	161
	AVE	4.6	93.9	358	255.3	102.8
f_9	MIN	2	25	32	10	18
	MAX	5	166	444	55	148
	AVE	3.2	75	207.2	23.2	57.7
f_{10}	MIN	1	1	19	1	1
	MAX	1	3	494	1	2
	AVE	1	1.8	112.6	1	1.6
f_{11}	MIN	3	19	29	87	34
	MAX	6	118	606	615	97
	AVE	4.9	56.1	256.4	384.8	61
f_{12}	MIN	2	37	55	40	18
	MAX	6	97	145	452	53
	AVE	4.3	70.9	645.3	161.6	33.9

when the input value approaches infinity. In other words, the time complexity means the linear and nonlinear mapping of the aim value and the number of testing times. Population initialization requires $O(N \times D)$, where D denotes the dimension of solution space, and N is the population size. The complexity of the proposed algorithm is $O(\max\_it \times N \times D)$, where max_it is the number of running times. To comprehensively compare the time complexity of different algorithms, this paper calculated running times of all algorithm for two-dimension functions and selected the two times of the worst searching value in all algorithms as the aim value. To comprehensively show the time complexity, this paper selected three indicators including the maximum number of running times (MAX), the minimum number of running times (MIN), and the average number of running times (AVE) for ten independent runs. All testing results are shown in Table 5. Table 5 shows that the maximum number of running times, the minimum number of running times (MIN), and the average number of running times in the proposed algorithm are smaller than those of other algorithms. And PWSSA is not easy to fall into local optimum. In addition, the results also show that PWSSA has superior searching ability because PWSSA can get results with good precision than other algorithms. In summary, the main reason is that PWSSA owns an excellent balance between global and local searching phases.

5. Conclusions

Metaheuristic optimization is a significant area, and most representative computational intelligence algorithms have permeated into almost all areas of science and engineering. SSA is a typical metaheuristic algorithm proposed in 2017. Despite SSA is success and popularity, there are some issues that need to be addressed in basic SSA. To overcome the problems of poor real-time stability and low precision of basic SSA for global function optimization problems, this paper has introduced a modified version based on the perturbation weight. PWSSA mainly relies on an asymptotic circular search strategy, which can achieve fast local searching and information orientation. PWSSA efficiently moves towards better function values and offers strong real-time performance. The proposed algorithm can effectively escape premature convergence in the early searching phase and avoids missing the global optimal solution in the later searching phase, thus achieving better search diversity and the possibility of finding a better solution. This paper has described the results of tests using eight low-dimension and four variable-dimension functions. In comparison experiments against other algorithms, PWSSA consistently obtained the best solutions and the smallest function values. This paper has described the results of tests using eight low-dimension and four variable-dimension functions. In comparison experiments against other algorithms, PWSSA consistently obtained the best solutions and the smallest function values. The proposed algorithm can give high-quality searching abilities for functions, which is reflected in the fact that PWSSA can get a more competitive precision than those of comparison algorithms. Iteration curves, box-plot charts, and search-path graphics were used to illustrate the effectiveness of the proposed algorithm, and the results show that PWSSA can generally obtain better solutions than previous algorithms. Through the analysis and comparison of the results using the different mathematical methods, the superiority of the proposed algorithm was proved. It has been proved by the no-free-lunch (NFL) [57] that metaheuristic optimization algorithms are able to solve all optimization problems. In other words, all metaheuristics perform similarly when solving all optimization problems, except the methods used in the paper; some of the most representative computational intelligence algorithms can be used to solve the problems, such as the monarch butterfly optimization (MBO) [22], earthworm optimization algorithm (EWA) [24], elephant herding optimization (EHO) [25], and moth search (MS) algorithm [27]. In future work,

we will focus on the proposed algorithm used to solve industrial application problems.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This research was funded by the International Cooperation Project (grant no. 2012DFR70840).

References

[1] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[2] A. Slowik and H. Kwasnicka, "Nature inspired methods and their industry applications-swarm intelligence algorithms," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1004–1015, 2018.

[3] Y. Zhou, R. Wang, C. Zhao, Q. Luo, and M. A. Metwally, "Discrete greedy flower pollination algorithm for spherical traveling salesman problem," *Neural Computing and Applications*, vol. 31, no. 7, pp. 2155–2170, 2017.

[4] Q. Gu, X. Li, and S. Jiang, "Hybrid genetic grey wolf algorithm for large-scale global optimization," *Complexity*, vol. 2019, Article ID 2653512, 18 pages, 2019.

[5] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, "A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms," *Soft Computing*, vol. 23, no. 9, pp. 3137–3166, 2017.

[6] T. Zheng and W. Luo, "An improved squirrel search algorithm for optimization," *Complexity*, vol. 2019, Article ID 6291968, 31 pages, 2019.

[7] S. A. Jalaee, A. GhasemiNejad, M. Lashkary, and M. Rezaee Jafari, "Forecasting Iran's energy demand using cuckoo optimization algorithm," *Mathematical Problems in Engineering*, vol. 2019, Article ID 2041756, 8 pages, 2019.

[8] Z. M. Gao and J. Zhao, "An improved grey wolf optimization algorithm with variable weights," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 2981282, 13 pages, 2019.

[9] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[10] E.-G. Talbi, "A taxonomy of hybrid metaheuristics," *Journal of Heuristics*, vol. 8, no. 5, pp. 541–564, 2002.

[11] L. Zuo, M. Yan, Y. Guo, and W. Ma, "An improved KF-RBF based estimation algorithm for coverage control with unknown density function," *Complexity*, vol. 2019, Article ID 6268127, 11 pages, 2019.

[12] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and S. Deb, "A hybrid method based on krill herd and quantum-behaved particle swarm optimization," *Neural Computing and Applications*, vol. 27, no. 4, pp. 989–1006, 2016.

[13] F. Miao, Y. Zhou, and Q. Luo, "A modified symbiotic organisms search algorithm for unmanned combat aerial vehicle route planning problem," *Journal of the Operational Research Society*, vol. 70, no. 1, pp. 21–52, 2018.

[14] G.-G. Wang, A. H. Gandomi, X. Zhao, and H. C. E. Chu, "Hybridizing harmony search algorithm with cuckoo search for global numerical optimization," *Soft Computing*, vol. 20, no. 1, pp. 273–285, 2016.

[15] X. Jiang, Z. Lin, T. He, X. Ma, S. Ma, and S. Li, "Optimal path finding with beetle antennae search algorithm by using ant colony optimization initialization and different searching strategies," *IEEE Access*, vol. 8, pp. 15459–15471, 2020.

[16] S. Qiao, Y. Zhou, Y. Zhou, and R. Wang, "A simple water cycle algorithm with percolation operator for clustering analysis," *Soft Computing*, vol. 23, no. 12, pp. 4081–4095, 2019.

[17] Y. Feng, K. Jia, Y. He, and X.-J. Zhao, "An effective hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm for 0-1 knapsack problems," *Computational Intelligence and Neuroscience*, vol. 2014, pp. 1–17, 2014.

[18] Y. Zhou, X. Yang, Y. Ling, and J. Zhang, "Meta-heuristic moth swarm algorithm for multilevel thresholding image segmentation," *Multimedia Tools and Applications*, vol. 77, no. 18, pp. 23699–23727, 2018.

[19] Y. Fan, J. Shao, G. Sun, and X. Shao, "Improved beetle antennae search algorithm-based Lévy flight for tuning of PID controller in force control system," *Mathematical Problems in Engineering*, vol. 2020, Article ID 4287315, 22 pages, 2020.

[20] A. H. Khan, S. Li, and X. Luo, "Obstacle avoidance and tracking control of redundant robotic manipulator: an RNN-based metaheuristic approach," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4670–4680, 2020.

[21] C. Han, G. Zhou, and Y. Zhou, "Binary symbiotic organism search algorithm for feature selection and analysis," *IEEE Access*, vol. 7, Article ID 166833, 2019.

[22] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Computing and Applications*, vol. 31, no. 7, pp. 1995–2014, 2019.

[23] X. Jiang and S. Li, "BAS: beetle antennae search algorithm for optimization problems," *International Journal of Robotics and Control*, vol. 1, no. 1, pp. 1–5, 2018.

[24] G. G. Wang, S. Deb, and L. D. S. Coelho, "Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems," *International Journal of Bio-Inspired Computation*, vol. 12, no. 1, pp. 1–22, 2018.

[25] G. G. Wang, S. Deb, X. Z. Gao, and L. D. S. Coelho, "A new metaheuristic optimisation algorithm motivated by elephant herding behaviour," *International Journal of Bio-Inspired Computation*, vol. 8, no. 6, pp. 394–409, 2016.

[26] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm," *Computers & Structures*, vol. 169, pp. 1–12, 2016.

[27] G.-G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, no. 2, pp. 151–164, 2018.

[28] 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," *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017.

[29] H. M. Kanoosh, E. H. Houssein, and M. M. Selim, "Salp swarm algorithm for node localization in wireless sensor networks," *Journal of Computer Networks and Communications*, vol. 2019, Article ID 1028723, 12 pages, 2019.

[30] M. Qais, H. M. Hasanien, and S. Alghuwainem, "Salp swarm algorithm-based TS-FLCs for MPPT and fault ride-through

capability enhancement of wind generators," *ISA Transactions*, vol. 101, pp. 211–224, 2020.

[31] Z. M. Yaseen, H. Faris, and N. Al-Ansari, "Hybridized extreme learning machine model with salp swarm algorithm: a novel predictive model for hydrological application," *Complexity*, vol. 2020, Article ID 8206245, 14 pages, 2020.

[32] T. A. A. Ali, Z. Xiao, J. Sun, S. Mirjalili, V. Havyarimana, and H. Jiang, "Optimal design of IIR wideband digital differentiators and integrators using salp swarm algorithm," *Knowledge-Based Systems*, vol. 182, Article ID 104834, 2019.

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

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

[35] M. Khishe and H. Mohammadi, "Passive sonar target classification using multi-layer perceptron trained by salp swarm algorithm," *Ocean Engineering*, vol. 181, pp. 98–108, 2019.

[36] M. Masdari, M. Tahani, M. H. Naderi, and N. Babayan, "Optimization of airfoil based savonius wind turbine using coupled discrete vortex method and salp swarm algorithm," *Journal of Cleaner Production*, vol. 222, pp. 47–56, 2019.

[37] A. Singh and V. Sharma, "Salp swarm algorithm-based model predictive controller for frequency regulation of solar integrated power system," *Neural Computing and Applications*, vol. 31, no. 12, pp. 8859–8870, 2019.

[38] J. Zhang, Z. Wang, and X. Luo, "Parameter estimation for soil water retention curve using the salp swarm algorithm," *Water*, vol. 10, no. 6, p. 815, 2018.

[39] Y. Wan, M. Mao, L. Zhou, Q. Zhang, X. Xi, and C. Zheng, "A novel nature-inspired maximum power point tracking (MPPT) controller based on SSA-GWO algorithm for partially shaded photovoltaic systems," *Electronics*, vol. 8, no. 6, p. 680, 2019.

[40] H. Gao, Y. Hou, S. Zhang, and M. Diao, "An efficient approximation for Nakagami-m quantile function based on generalized opposition-based quantum salp swarm algorithm," *Mathematical Problems in Engineering*, vol. 2019, Article ID 8291063, 13 pages, 2019.

[41] Z. Xing and H. Jia, "Multilevel color image segmentation based on GLCM and improved salp swarm algorithm," *IEEE Access*, vol. 7, pp. 37672–37690, 2019.

[42] Y. Fan, J. Shao, G. Sun, and X. Shao, "Proportional-integral-derivative controller design using an advanced Lévy-flight salp swarm algorithm for hydraulic systems," *Energies*, vol. 13, no. 2, p. 459, 2020.

[43] S. K. Majhi, A. Mishra, and R. Pradhan, "A chaotic salp swarm algorithm based on quadratic integrate and fire neural model for function optimization," *Progress in Artificial Intelligence*, vol. 8, no. 3, pp. 343–358, 2019.

[44] 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 Systems with Applications*, vol. 145, Article ID 113103, 2020.

[45] H. S. N. Alwerfali, M. Abd Elaziz, M. A. A. Al-Qaness et al., "A multilevel image thresholding based on hybrid salp swarm algorithm and fuzzy entropy," *IEEE Access*, vol. 7, Article ID 181405, 2019.

[46] R. A. Ibrahim, A. A. Ewees, D. Oliva, M. Abd Elaziz, and S. Lu, "Improved salp swarm algorithm based on particle swarm optimization for feature selection," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3155–3169, 2018.

[47] N. Panda and S. K. Majhi, "Improved salp swarm algorithm with space transformation search for training neural network," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 2743–2761, 2019.

[48] R. M. Rizk-Allah, A. E. Hassanien, M. Elhoseny, and M. Gunasekaran, "A new binary salp swarm algorithm: development and application for optimization tasks," *Neural Computing and Applications*, vol. 31, no. 5, pp. 1641–1663, 2018.

[49] G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel chaotic salp swarm algorithm for global optimization and feature selection," *Applied Intelligence*, vol. 48, no. 10, pp. 3462–3481, 2018.

[50] J. Wu, R. Nan, and L. Chen, "Improved salp swarm algorithm based on weight factor and adaptive mutation," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 31, no. 3, pp. 493–515, 2019.

[51] Z. Xiang, Y. Zhou, Q. Luo, and C. Wen, "PSSA: polar coordinate salp swarm algorithm for curve design problems," *Neural Processing Letters*, vol. 52, no. 1, pp. 615–645, 2020.

[52] A. E. Hegazy, M. A. Makhlouf, and G. S. El-Tawel, "Improved salp swarm algorithm for feature selection," *Journal of King Saud University—Computer and Information Sciences*, vol. 32, no. 3, pp. 335–344, 2020.

[53] M. H. Qais, H. M. Hasanien, and S. Alghuwainem, "Enhanced salp swarm algorithm: application to variable speed wind generators," *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 82–96, 2019.

[54] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of Operations Research*, vol. 41, no. 4, pp. 421–451, 1993.

[55] 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.

[56] G.-G. Wang, L. Guo, A. H. Gandomi, G.-S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Information Sciences*, vol. 274, pp. 17–34, 2014.

[57] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.