Hindawi
Computational Intelligence and Neuroscience
Volume 2021, Article ID 6379469, 21 pages
https://doi.org/10.1155/2021/6379469

Research Article

A Hybrid SSA and SMA with Mutation Opposition-Based Learning for Constrained Engineering Problems

Shuang Wang,[1] **Qingxin Liu**,[2] **Yuxiang Liu**,[3] **Heming Jia**,[1] **Laith Abualigah**,[4,5] **Rong Zheng**,[1] **and Di Wu**[6]

[1]*School of Information Engineering, Sanming University, Sanming 365004, China*
[2]*School of Computer Science and Technology, Hainan University, Haikou 570228, China*
[3]*College of Physics and Information Engineering, Fuzhou University, Fuzhou 350108, China*
[4]*Faculty of Computer Sciences and Informatics, Amman Arab University, Amman 11953, Jordan*
[5]*School of Computer Science, Universiti Sains Malaysia, Gelugor, Penang, Malaysia*
[6]*School of Education and Music, Sanming University, Sanming 365004, China*

Correspondence should be addressed to Heming Jia; jiahemінglucky99@126.com

Received 25 July 2021; Accepted 19 August 2021; Published 7 September 2021

Academic Editor: Mario Versaci

Based on Salp Swarm Algorithm (SSA) and Slime Mould Algorithm (SMA), a novel hybrid optimization algorithm, named Hybrid Slime Mould Salp Swarm Algorithm (HSMSSA), is proposed to solve constrained engineering problems. SSA can obtain good results in solving some optimization problems. However, it is easy to suffer from local minima and lower density of population. SMA specializes in global exploration and good robustness, but its convergence rate is too slow to find satisfactory solutions efficiently. Thus, in this paper, considering the characteristics and advantages of both the above optimization algorithms, SMA is integrated into the leader position updating equations of SSA, which can share helpful information so that the proposed algorithm can utilize these two algorithms' advantages to enhance global optimization performance. Furthermore, Levy flight is utilized to enhance the exploration ability. It is worth noting that a novel strategy called mutation opposition-based learning is proposed to enhance the performance of the hybrid optimization algorithm on premature convergence avoidance, balance between exploration and exploitation phases, and finding satisfactory global optimum. To evaluate the efficiency of the proposed algorithm, HSMSSA is applied to 23 different benchmark functions of the unimodal and multimodal types. Additionally, five classical constrained engineering problems are utilized to evaluate the proposed technique's practicable abilities. The simulation results show that the HSMSSA method is more competitive and presents more engineering effectiveness for real-world constrained problems than SMA, SSA, and other comparative algorithms. In the end, we also provide some potential areas for future studies such as feature selection and multilevel threshold image segmentation.

1. Introduction

In recent years, metaheuristic algorithms have been widely concerned by a large number of scholars. Compared with other traditional optimization algorithms, the concept of metaheuristic algorithms is simple. Besides, they are flexible and can bypass local optima. Thus, metaheuristics have been successfully applied in different fields to solve various complex optimization problems in the real world [1–3].

Metaheuristic algorithms include three main categories: evolution-based, physics-based, and swarm-based techniques. The inspirations of evolutionary-based methods are the laws of evolution in nature. The most popular evolution-based algorithms include Genetic Algorithm (GA) [4], Differential Evolution Algorithm (DE) [5], and Biogeography-Based Optimizer (BBO) [6]. Physics-based algorithms mimic the physical rules in the universe. There are some representative algorithms such as Simulated Annealing (SA) [7], Gravity Search Algorithm (GSA) [8], Black Hole (BH)

algorithm [9], Multiverse Optimizer (MVO) [10], Artificial Chemical Reaction Optimization Algorithm (ACROA) [11], Ray Optimization (RO) [12], Curved Space Optimization (CSO) [13], Sine Cosine Algorithm (SCA) [14], Arithmetic Optimization Algorithm (AOA) [15], and Heat Transfer Relation-based Optimization Algorithm (HTOA) [16]. The third category algorithm is swarm-based techniques, which simulate the social behavior of creatures in nature. Some optimization techniques of this class include Particle Swarm Optimization (PSO) [17] Ant Colony Optimization Algorithm (ACO) [18], Firefly Algorithm (FA) [19], Grey Wolf Optimizer (GWO) [20], Cuckoo Search (CS) Algorithm [21], Whale Optimization Algorithm (WOA) [22], Bald Eagle Search (BES) algorithm [23], and Aquila Optimizer (AO) [24].

It is worth noting that the most widely used swarm-based optimization algorithm is PSO [25]. PSO simulates the behavior of birds flying together in flocks. During the search, they all follow the best solutions in their paths. Cacciola et al. [26] discussed the problem of corrosion profile reconstruction starting from electrical data, in which PSO was utilized to obtain the image of the reconstructed corrosion profile. The result shows that PSO can obtain the optimal solution compared with LSM, and it takes the least time. This allows us to recognize the huge potential of the optimization algorithm.

Salp Swarm Algorithm (SSA) [27] is a swarm-based algorithm proposed in 2017. SSA is inspired by swarm behavior, navigation, and foraging of salps in the ocean. Since SSA has fewer parameters and is easier to be realized in a program than other algorithms, SSA has been applied to many optimization problems, such as feature selection, image segmentation, and constrained engineering problems. However, like other metaheuristic algorithms, SSA may be easy to trap into local minima and lower population density. Therefore, many improved researches have been proposed to enhance the performance of SSA in many fields. Tubishat et al. [28] presented a Dynamic SSA (DSSA), which shows better accuracy than SSA in feature selection. Salgotra et al. [29] proposed a self-adaptive SSA to enhance exploitation ability and convergence speed. Neggaz et al. [30] proposed an improved leader in SSA using Sine Cosine Algorithm and disrupt operator for feature selection. Jia and Lang [31] presented an enhanced SSA with a crossover scheme and Levy flight to improve the movement patterns of salp leaders and followers. There are also other attempts on the hybrid algorithm of SSA. Saafan and El-gendy [32] proposed a hybrid improved Whale Optimization Salp Swarm Algorithm (IWOSSA). The IWOSSA achieves a better balance between exploration and exploitation phases and avoids premature convergence effectively. Singh et al. [33] developed a hybrid SSA with PSO, which integrated the advantages of SSA and PSO to eliminate trapping in local optima and unbalanced exploitation. Abadi et al. [34] proposed a hybrid approach by combining SSA with GA, which could obtain good results in solving some optimization problems.

Slime Mould Algorithm (SMA) [35] is the latest swarm intelligence algorithm proposed in 2020. This algorithm simulates the oscillation mode and the foraging of Slime Mould in nature. SMA has a unique search mode, which keeps the algorithm from falling into local optima, and has superior global exploration capability. The approach has been applied in real-world optimization problems like feature selection [36], parameters optimization of the fuzzy system [37], multilevel threshold image segmentation [38], control scheme [39], and parallel connected multistacks fuel cells [40].

Therefore, based on the capabilities of both above algorithms, we try to do a hybrid operation to improve the performance of SMA or SSA and then propose a new hybrid optimization algorithm (HSMSSA) to speed up the convergence rate and enhance the overall optimization performance. The specific method is that we integrate SMA as the leader role into SSA and retain the exploitation phase of SSA. At the same time, inspired by the significant performance of opposition-based learning and quasiopposition-based learning, we propose a new strategy named mutation opposition-based learning (MOBL), which switches the algorithm between opposition-based learning and quasiopposition-based learning through a mutation rate to increase the diversity of the population and speed up the convergence rate. In addition, Levy flight is utilized to improve SMA's exploration capability and balance the exploration and exploitation phases of the algorithm. The proposed HSMSSA algorithm can improve both the exploration and exploitation abilities. The proposed HSMSSA is tested on 23 different benchmark functions and compared with other optimization algorithms. Furthermore, five constrained engineering problems are also utilized to evaluate HSMSSA's capability on real-world optimization problems. The experimental results illustrate that the HSMSSA possesses the superior capability to search the global minimum and achieve less cost engineering design results than other state-of-the-art metaheuristic algorithms.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of SSA, SMA, Levy flight, and mutation opposition-based learning strategy. Section 3 describes the proposed hybrid algorithm in detail. In Section 4, the details of benchmark functions, parameter settings of the selected algorithms, simulation experiments, and results analysis are introduced. Conclusions and prospects are given in Section 5.

2. Preliminaries

2.1. Salp Swarm Algorithm. In the deep sea, salps live in groups and form a salp chain to move and forage. In the salp chain, there are leaders and followers. The leader moves towards the food and guides the followers. In the process of moving, leaders explore globally, while followers thoroughly search locally [27]. The shapes of a salp and salp chain are shown in Figure 1.

2.1.1. Leader Salps. The front salp of the chain is called the leader, so the following equation is used to perform this action to the salp leader:

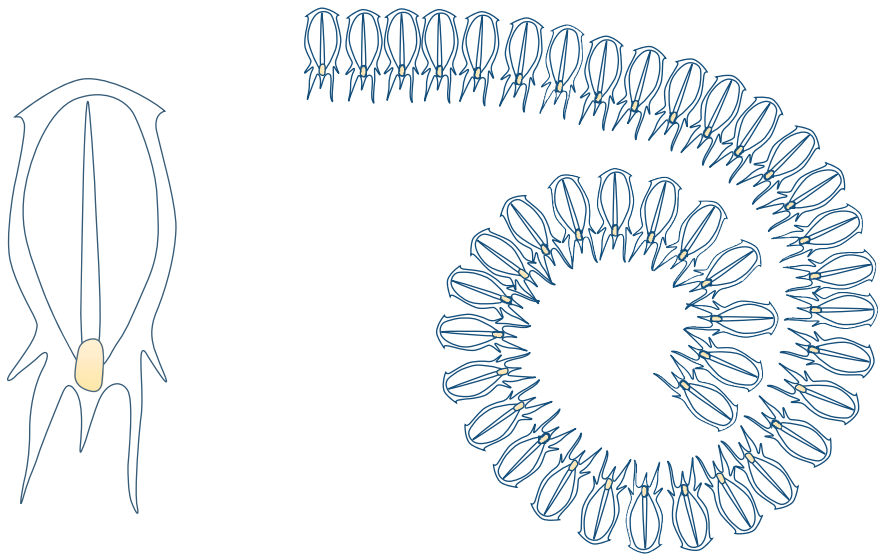

Figure 1: Individual salp and the swarm of salp [25].

$$X_j^1 = \begin{cases} F_j + c_1\left((\mathrm{UB} - \mathrm{LB})r_1 + \mathrm{LB}\right), & r_2 < 0.5, \\ F_j - c_1\left((\mathrm{UB} - \mathrm{LB})r_1 + \mathrm{LB}\right), & r_2 \geq 0.5, \end{cases} \tag{1}$$

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

where X_j^1 and F_j represent the new position of the leader and food source in the jth dimension and r_1 and r_2 are randomly generated numbers in the interval [0, 1]. It is worth noting that c_1 is essential for SSA because it balances exploration and exploitation during the search process. t is the current iteration and T is the max iteration.

2.1.2. Follower Salps. To update the position of the followers, the new concept is introduced, which is based on Newton's law of motion as in the following equation:

$$X_j^i = \frac{1}{2}gt^2 + \omega_0 t, \quad i \geq 2, \tag{3}$$

where X_j^i represents the position of ith follower salp in the jth dimension and g and ω_0 indicate the acceleration and the velocity, respectively. The updating process of followers can be expressed as in the following equation:

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

The pseudocode of SSA is presented in Algorithm 1.

2.2. Slime Mould Algorithm. The main idea of SMA is inspired by the behavior and morphological changes of Slime Mould in foraging. Slime Mould can dynamically change search mode based on the quality of food sources. If the food source has a high quality, the Slime Mould will use the region-limited search method. If the food concentration is low, the Slime Mould will explore other food sources in the search space. Furthermore, even if Slime Mould has found a high-quality food source, it still divides some individuals to explore another area in the region [35]. The behavior of Slime Mould can be mathematically described as follows:

$$\overrightarrow{X(t+1)} = \begin{cases} r_4 \times (\mathrm{UB} - \mathrm{LB}) + \mathrm{LB}, & r_3 < z, \\ \overrightarrow{X_b(t)} + \vec{vb} \times \left(\vec{W} \cdot \overrightarrow{X_A(t)} - \overrightarrow{X_B(t)}\right), & r_5 < p, \\ \vec{vc} \times \overrightarrow{X(t)}, & r_5 \geq p, \end{cases} \tag{5}$$

where parameters r_3, r_4, and r_5 are random values in the range of 0 to 1. UB and LB indicate the upper and lower bound of search space. z is a constant. $\overrightarrow{X_b(t)}$ represents the best position obtained in all iterations, $\overrightarrow{X_A(t)}$ and $\overrightarrow{X_B(t)}$ represent two individuals selected randomly from the population, and $\overrightarrow{X(t)}$ represents the location of Slime Mould. $\vec{vc}$ decreases linearly from one to zero, and $\vec{vb}$ is an oscillation parameter in the range $[-a, a]$, in which a is calculated as follows:

$$a = \arctan h\left(-\left(\frac{t}{T}\right) + 1\right). \tag{6}$$

The coefficient $\vec{W}$ is a very important parameter, which simulates the oscillation frequency of different food concentrations so that Slime Mould can approach food more quickly when they find high-quality food. The formula of $\vec{W}$ is listed as follows:

```
(1) Initialize the population size N and max iteration T;
(2) Initialize the positions of salp X_i (i = 1, 2, ..., N)
(3) While (t ≤ T)
(4)    Calculate fitness of each salp;
(5)    Denote the best solution as F
(6)    update c_1 by equation (2);
(7)    For i = 1 to N do
(8)       if (i = = 1) then
(9)          update position of leader salp by equation (1)
(10)      Else
(11)         update position of follower salp by equation (4)
(12)      End if
(13)   End for
(14)   t = t + 1;
(15) End While
(16) Return the best solution F;
```

Algorithm 1: Pseudocode of Salp Swarm Algorithm.

$$\overrightarrow{W(\text{SmellIndex}(i))} = \begin{cases} 1 + r_6 \times \log\left(\dfrac{bF - S(i)}{bF - wF} + 1\right), & \text{condition}, \\ 1 - r_6 \times \log\left(\dfrac{bF - S(i)}{bF - wF} + 1\right), & \text{others}, \end{cases} \tag{7}$$

$$\text{SmellIndex} = \text{sort}(S), \tag{8}$$

where $i \in 1, 2, \ldots, N$ and $S(i)$ represents the fitness of X. *condition* indicates that $S(i)$ ranks first half of the Slime Mould, and r_6 are random numbers uniformly generated in the interval of [0, 1]. bF represents the optimal fitness obtained in the current iterative process, wF represents the worst fitness value obtained in the iterative process currently, and *SmellIndex* denotes the sequence of fitness values sorted (ascends in the minimum value problem).

The p parameter can be described as follows:

$$p = \tanh|S(i) - \text{DF}|, \tag{9}$$

where DF represents the best fitness over all iterations. Figure 2 visualizes the general logic of SMA.

The pseudocode of SMA is presented in Algorithm 2.

2.3. Levy Flight. Levy flight is an effective strategy for metaheuristic algorithms, successfully designed in many algorithms [41–44]. Levy flight is a class of non-Gaussian random processes that follows Levy distribution. It alternates between short-distance and occasionally long-distance walking, which can be inferred from Figure 3. The formula of Levy flight is as follows:

$$\text{Levy} = 0.01 \times \frac{r_7 \times \sigma}{|r_8|^{(1/\beta)}}, \qquad \sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\pi\beta/2)}{\Gamma((1+\beta)/2) \times \beta \times 2^{((\beta-1)/2)}}\right)^{(1/\beta)}, \tag{10}$$

where r_7 and r_8 are random values in the range of [0, 1] and β is a constant equal to 1.5.

2.4. Mutation Opposition-Based Learning. Opposition-based learning (OBL) was proposed by Tizhoosh in 2005 [45]. The essence of OBL is selecting the best solution to the next iteration by comparing the current solution and its opposition-based learning solution. The OBL strategy has been successfully used in varieties of metaheuristic algorithms [46–51] to improve the ability of local optima stagnation avoidance, and the mathematical expression is as follows:

$$X_{\text{OBL}}(t+1) = \text{LB} + \text{UB} - X(t). \tag{11}$$

Quasiopposition-based learning (QOBL) [52] is an improved version from OBL, which applies quasiopposite points instead of opposite points. These points produced through QOBL have more likelihood of being unknown solutions than the points created by OBL. The mathematical formula of QOBL is as follows:

$$X_{\text{QOBL}}(t+1) = \begin{cases} \text{CS} + r_9 \times (\text{MP} - \text{CS}), & \text{if MP} > \text{CS}, \\ \text{MP} + r_9 \times (\text{CS} - \text{MP}), & \text{otherwise}, \end{cases} \qquad \text{CS} = \frac{\text{LB} + \text{UB}}{2}, \qquad \text{MP} = \text{LB} + \text{UB} - X(t). \tag{12}$$

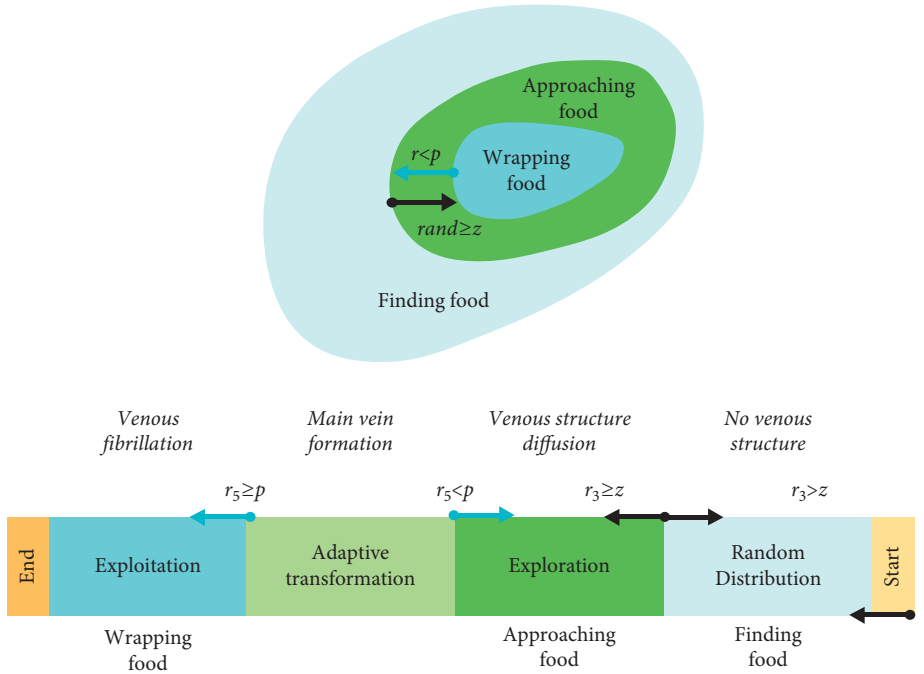

Figure 2: The steps of SMA.

(1) Initialize the population size N and max iteration T;
(2) Initialize the positions of Slime Mould X_i ($i = 1, 2, \ldots, N$)
(3) **While** ($t \leq T$)
(4) Calculate fitness of each Slime Mould;
(5) update bF, wF, and X_b;
(6) Calculate W by equation (3);
(7) **For** $i = 1$ to N **do**
(8) update p, vb, and vc;
(9) update positions by equation (1)
(10) **End For**
(11) t = t + 1;
(12) **End While**
(13) Return X_b;

Algorithm 2: Pseudocode of Slime Mould Algorithm.

Considering the superior performance of the two kinds of opposition-based learning, we propose mutation opposition-based learning (MOBL) by combining the mutation rate with these two opposition-based learning. By selecting the mutation rate, we can give full play to the characteristics of the OBL and QOBL and effectively enhance the ability of the algorithm to jump out of the local optima. Figure 4 is an MOBL example, in which Figure 4(a) shows an objective function and Figure 4(b) displays three candidate solutions and their OBL solutions or QOBL solutions. The mathematical formula is as follows:

$$X(t+1) = \begin{cases} X_{\mathrm{OBL}}(t), & if\ r_{10} < \mathrm{rate}, \\ X_{\mathrm{QOBL}}(t), & \mathrm{else}, \end{cases} \tag{13}$$

where *rate* is mutation rate, and we set it to 0.1.

3. The Proposed Algorithm

3.1. Details of HSMSSA. In SSA, the population is divided into leader salps and follower salps: leader salps are the first half of salps in the chain, and follower salps follow the leader. However, the leader salp has poor randomness and is easy to fall into local optima. For the SMA algorithm, Slime Mould selects different search modes according to the positive and negative feedback of the current food concentration and has a certain probability of isolating some individuals to explore other regions in search space. These mechanisms increase the randomness of Slime Mould and enhance the ability to explore. The *vb* parameter is utilized to realize the oscillation mode of Slime Mould, which is in the range of [−a, a]. However, *vb* has the drawback of low randomness, which cannot effectively simulate the process of Slime Mould

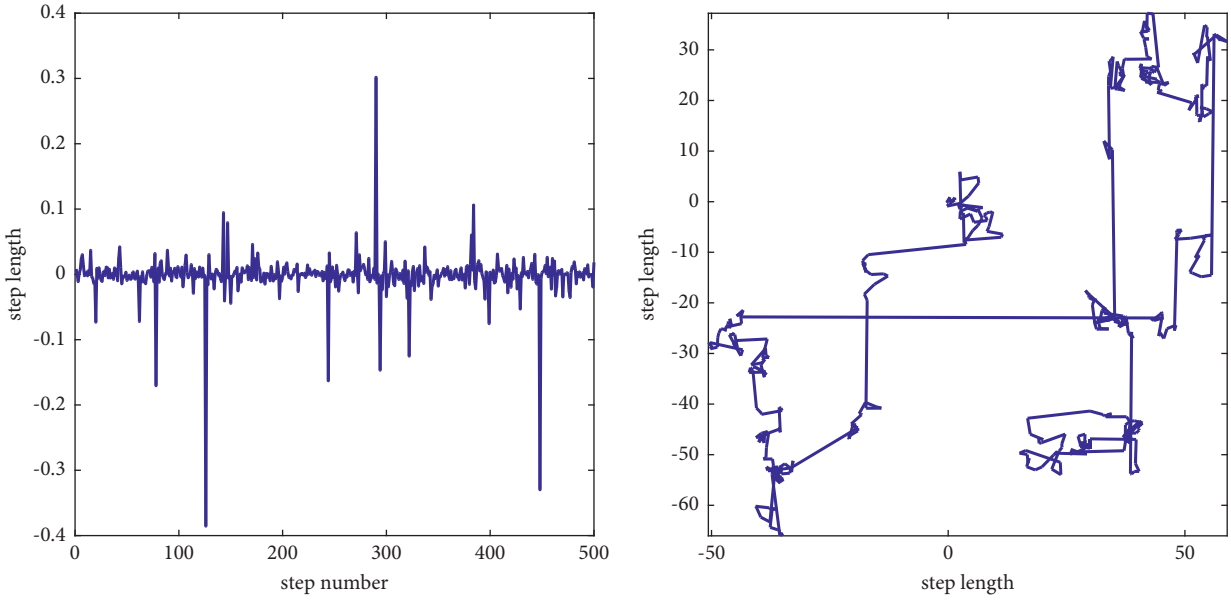

Figure 3: Levy distribution and 2D Levy trajectory.

looking for food sources. Therefore, we introduce Levy flight into the exploration phase to further enhance the exploration ability. Next, we integrate SMA into SSA, change the position update method of leader salps, and further improve the randomness of the algorithm through Levy flight. For followers, we propose a mutation opposition-based learning to enhance its population diversity and increase the ability of the algorithm to jump out of the local optima. The mathematical formula of leader salps is as follows:

$$\overrightarrow{X(t+1)} = \begin{cases} r_4 \times (\mathrm{UB} - \mathrm{LB}) + \mathrm{LB}, & r_3 < z, \\ X_b + vb \times \left(W \times X_A - X_B\right) \times \mathrm{Levy}, & r_5 < p, \\ vc \times X_i, & r_5 \geq p. \end{cases} \tag{14}$$

The pseudocode of HSMSSA is given in Algorithm 3, and the summarized flowchart is displayed in Figure 5. As shown in Algorithm 3, the position of the population is initially generated randomly. Then, each individual's fitness will be calculated. For the entire population in each iteration, parameter W is calculated using equation (7). The search agents of population size N are assigned to the two algorithms, which can utilize the advantages of SSA and SMA, and realize the sharing of helpful information to achieve global optimization. If the search agent belongs to the first half of the population, the position will be updated using equation (14) in SMA with Levy flight. Otherwise, the position is determined using equation (4) and MOBL. Finally, if the termination criteria are satisfied, the algorithm returns the best solution found so far; else the previous steps are repeated.

3.2. Computational Complexity Analysis. HSMSSA mainly consists of three components: initialization, fitness evaluation, and position updating. In the initialization phase, the computational complexity of positions generated is $O(N \times D)$, where D is the dimension size of the problem. Then, the computational complexity of fitness evaluation for the solution is O(N) during the iteration process. Finally, we utilize mutation opposition-based learning to keep the algorithm from falling into local optima; thus, the computational complexities of position updating of HSMSSA are $O(2 \times N \times D)$. Therefore, the total computational complexity of the proposed HSMSSA algorithm is $O(N \times D + N + 2 \times N \times D)$.

4. Experimental Results and Discussion

This section compared the HSMSSA with some state-of-the-art metaheuristics algorithms on 23 benchmark functions to validate its performance. Moreover, five engineering design problems are employed as examples for real-world applications. The experimentations ran on Windows 10 with 24 GB RAM and Intel (R) i5-9500. All simulations were carried out using MATLAB R2020b.

4.1. Definition of 23 Benchmark Functions. To assess HSMSSA's ability of exploration, exploitation, and escaping from local optima, 23 benchmark functions, including unimodal and multimodal functions, are tested [27]. The unimodal benchmark functions (F1–F7) are utilized to examine the exploitation ability of HSMSSA. The description of the unimodal benchmark function is shown in Table 1. The multimodal and fixed-dimension multimodal benchmark functions (F8–F23) shown in Tables 2 and 3 are used to test the exploration ability of HSMSSA.

In order to show the experimental results more representative, the HSMSSA is compared with the basic SMA [35]

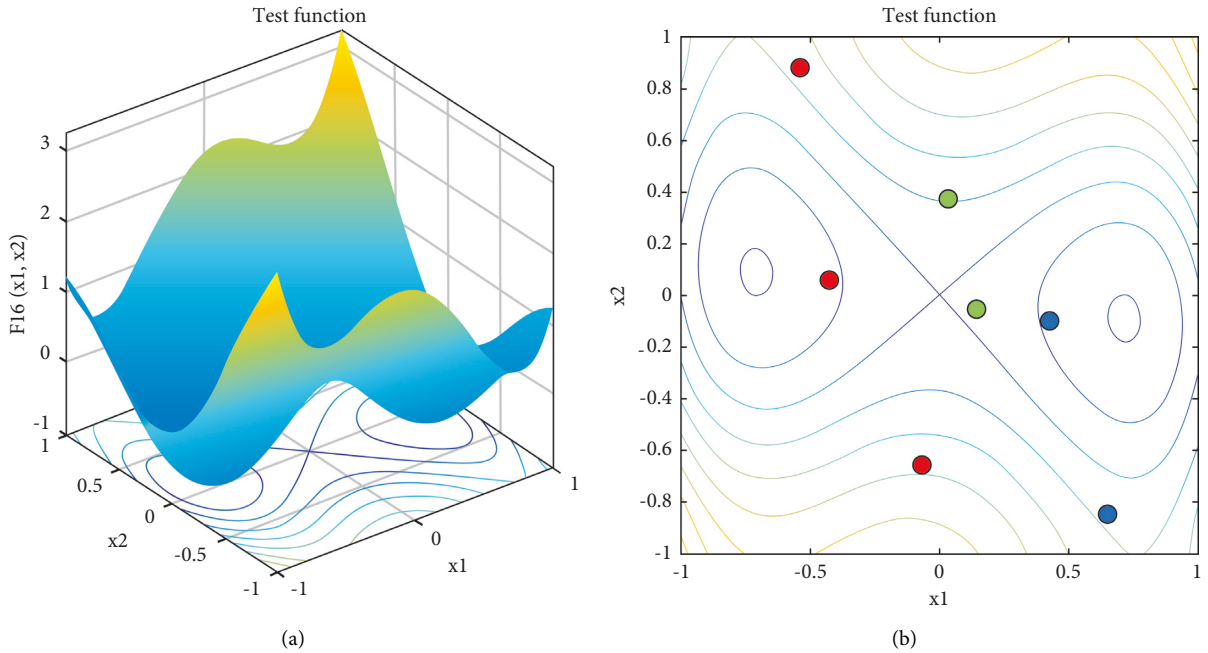

Figure 4: MOBL example: (a) objective function and (b) candidate solutions (red point), its OBL solution (blue point), and its QOBL solution (green point).

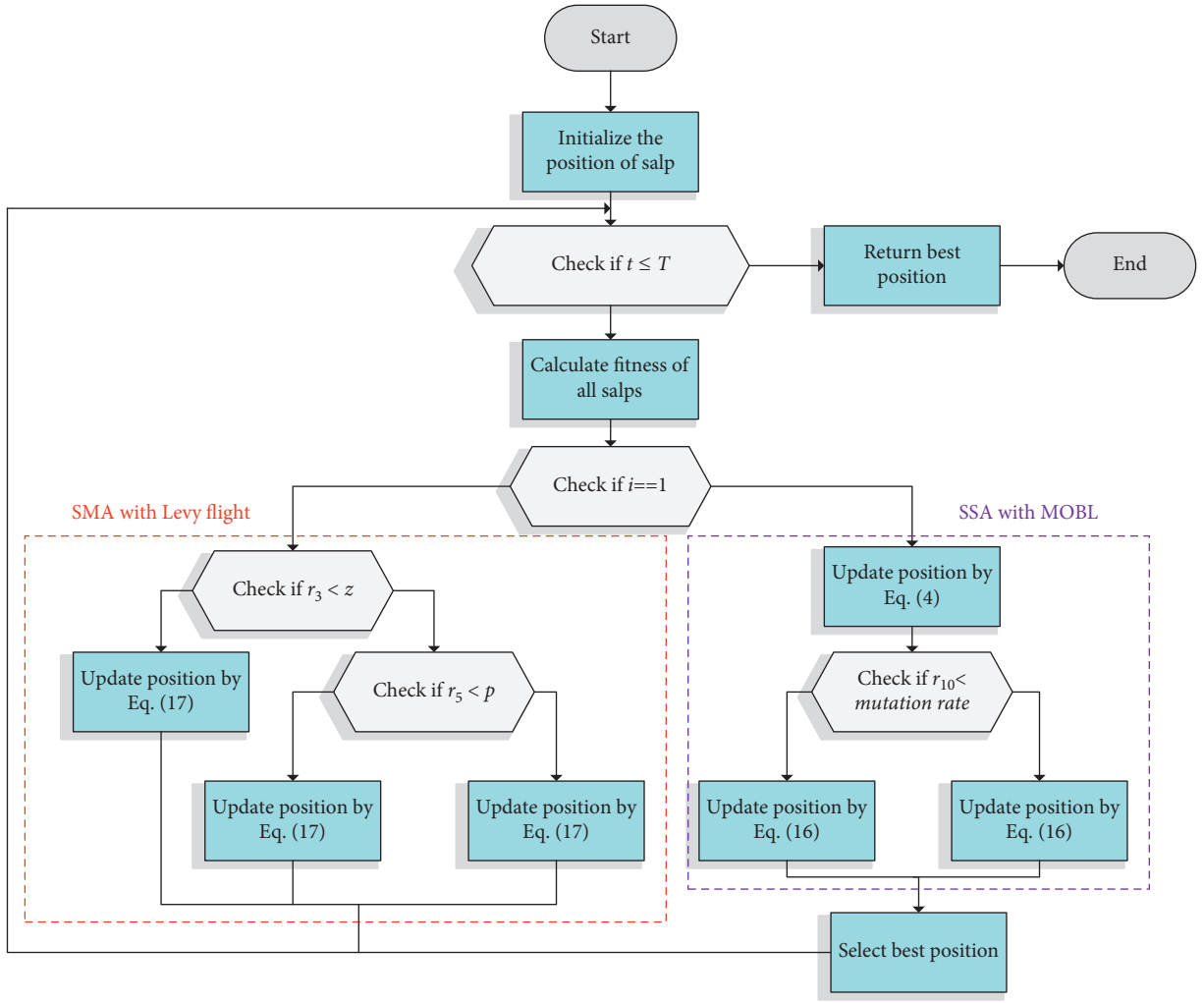

Figure 5: Flowchart of HSMSSA.

(1) Set the initial values of the population size N and the maximum number of iterations T
(2) Initialize positions of the population X
(3) **While** $t \leq T$
(4) Check if the position goes out of the search space boundary and bring it back.
(5) Calculate the fitness of each search agent
(6) Update W, bF, and wF
(7) **For** $i = 1$ to N
(8) **If** $i = = 1$
(9) **If** $r_3 < z$
(10) Update position by equation (14)
(11) **Else**
(12) **IF** $r_5 < p$
(13) Update position by equation (14)
(14) **Else**
(15) Update position by equation (14)
(16) **End if**
(17) **End if**
(18) **Else**
(19) Update position by equation (4)
(20) **If** $r_{10} < 0.1$
(21) Update the position of MOBL using equation (13)
(22) **Else**
(23) Update the position of MOBL using equation (13)
(24) **End if**
(25) **End if**
(26) Check if the position goes out of the search space boundary and bring it back.
(27) select the best position into the next iteration.
(28) t = t + 1
(29) **End for**
(30) **End while**
(31) Return X_{best}

Algorithm 3: Pseudocode of HSMSSA.

and SSA [27], AO [24], AOA [15], WOA [22], SCA [14], and MVO [10]. For all tests, we set the population size $N = 30$, dimension size $D = 30$, and maximum iteration $T = 500$, respectively, for all algorithms with 30 independent runs. The parameter settings of each algorithm are shown in Table 4. After all, average results and standard deviations are employed to evaluate the results. Note that the best results will be bolded.

4.1.1. Evaluation of Exploitation Capability (F1–F7). As we can see, unimodal benchmark functions have only one global optimum. These functions are allowed to evaluate the exploitation ability of the metaheuristic algorithms. It can be seen from Table 5 that HSMSSA is very competitive with SMA, SSA, and other metaheuristic algorithms. In particular, HSMSSA can achieve much better results than other metaheuristic algorithms except F6. For F1–F4, HSMSSA can find the theoretical optimum. For all unimodal functions except F5, HSMSSA gets the smallest average values and standard deviations compared to other algorithms, which indicate the best accuracy and stability. Hence, the exploitation capability of the proposed HSMSSA algorithm is excellent.

4.1.2. Evaluation of Exploration Capability (F8–F23). Unlike unimodal functions, multimodal functions have many local optima. Thus, this kind of test problem turns very useful to evaluate the exploration capability of an optimization algorithm. The results shown in Table 5 for functions F8–F23 indicate that HSMSSA also has an excellent exploration capability. In fact, we can see that HSMSSA can find the theoretical optimum in F9, F11, F16–F17, and F19–F23. These results reveal that HSMSSA can also provide superior exploration capability.

4.1.3. Analysis of Convergence Behavior. The convergence curves of some functions are selected and shown in Figure 6, which show the convergence rate of algorithms. It can be seen that HSMSSA shows competitive performance compared to other state-of-the-art algorithms. The HSMSSA presents a faster convergence speed than all other algorithms in F7–F13, F15, and F19–F23. For other benchmark functions, HSMSSA shows a better capability of local optima avoidance than other comparison algorithms in F5 and F6.

4.1.4. Qualitative Results and Analysis. Furthermore, Figure 7 shows the results of several representative test functions on search history, trajectory, average fitness, and

Table 1: Unimodal benchmark functions.

Function	Dim	Range	$F_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	$[-100, 100]$	0
$F_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^{n} (x_i + 5)^2$	30	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	30	$[-1.28, 1.28]$	0

Table 2: Multimodal benchmark functions.

Function	Dim	Range	$F_{\min}$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]$	$-418.9829 \times \text{Dim}$
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	
$F_{10}(x) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}) - \exp((1/n)\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0
$F_{11}(x) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$	30	$[-600, 600]$	0
$F_{12}(x) = (\pi/n)\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\}$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, where $y_i = 1 + (x_i + 1)/4$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a < x_i < a, \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	$[-50, 50]$	0
$F_{13}(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)]$ $+ (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0

Table 3: Fixed-dimension multimodal benchmark functions.

Function	Dim	Range	$F_{\min}$
$F_{14}(x) = ((1/500) + \sum_{j=1}^{25} 1/(j + \sum_{i=1}^{2} (x_i - a_{ij})^6))^{-1}$	2	$[-65, 65]$	0.998
$F_{15}(x) = \sum_{i=1}^{11} [a_i - (x_1(b_i^2 + b_i x_2)/(b_i^2 + b_i x_3 + x_4))]^2$	4	$[-5, 5]$	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1 x_2 - 4x_2^2 + x_2^4$	2	$[-5, 5]$	-1.0316
$F_{17}(x) = (x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - (1/8\pi))\cos x_1 + 10$	2	$[-5, 5]$	0.398
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)]$ $\times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_2 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)]$	2	$[-2, 2]$	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	$[-1, 2]$	-3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	$[0, 1]$	-3.32
$F_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.1532
$F_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.4028
$F_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.5363

Table 4: Parameter settings for the comparative algorithms.

Algorithm	Parameters
SMA [35]	$z = 0.03$
SSA [27]	$c_1 = [1, 0]$; $c_2 \in [0, 1]$; $c_3 \in [0, 1]$
AO [24]	$U = 0.00565$; $r_1 = 10$; $\omega = 0.005$; $\alpha = 0.1$; $\delta = 0.1$; $G_1 \in [-1, 1]$; $G_2 = [2, 0]$
AOA [15]	$\alpha = 5$; $\mu = 0.5$;
WOA [22]	$a_1 = [2, 0]$; $a_2 = [-1, -2]$; $b = 1$
SCA [14]	$a = [2, 0]$
MVO [10]	$WEP \in [0.2, 1]$; $TDR \in [0, 1]$; $r_1, r_2, r_3 \in [0, 1]$

convergence curve. From search history maps, we can see the search agent's distribution of the HSMSSA while exploring and exploiting the search space. Because of the fast convergence, the vast majority of search agents are concentrated near the global optimum. Inspecting trajectory figure in Figure 5, the first search agent constantly oscillates in the first dimension of the search space, which suggests that the search agent investigates the most promising areas and better solutions widely. This powerful search capability is likely to come from the Levy flight and MOBL strategies. The average fitness presents if exploration and exploitation are conducive to improve the first random population, and an accurate approximation of the global optimum can be found in the end.

Similarly, it can be noticed that the average fitness oscillates in the early iterations and then decreases abruptly and begins to level off. The average fitness maps also show the significant improvement of the first random population and the final global optimal, accurate approximation acquisition. At last, the convergence curves reveal the best fitness value found by search agents after each iteration. By observing this, the HSMSSA shows breakneck convergence speed.

4.1.5. Wilcoxon Signed-Rank Test. Because the algorithm results are random, we need to carry out statistical tests to prove that the results have statistical significance. We use Wilcoxon signed-ranks (WSR) test results to evaluate the statistical significance of the two algorithms at 5% significance level [53]. The WSR is a statistical test that is applied to two different results for searching the significantly different. As is well-known, a p-value less than 0.05 indicates that it is significantly superior to other algorithms. Otherwise, the obtained results are not statistically significant. The calculated results of the Wilcoxon signed-rank test between HSMSSA and other algorithms for each benchmark function are listed in Table 6. HSMSSA outperforms all other algorithms in varying degrees. This superiority is statistically significant on unimodal functions F2 and F4–F7, which indicates that HSMSSA possesses high exploitation. HSMSSA also shows better results on multimodal function F8–F23, suggesting that HSMSSA has a high capability of exploration. To sum up, HSMSSA can provide better results for almost all benchmark functions than other comparative algorithms.

4.2. Experiments on Engineering Design Problems. In this section, HSMSSA is evaluated to solve five classical engineering design problems: pressure vessel design problem, tension spring design problem, three-bar truss design problem, speed reducer problem, and cantilever beam design. To address these problems, we set the population size $N = 30$ and maximum iteration $T = 500$. The results of HSMSSA are compared to various state-of-the-art algorithms in the literature. The parameter settings are the same as previous numerical experiments.

4.2.1. Pressure Vessel Design Problem. The pressure vessel design problem [53] is to minimize the total cost of cylindrical pressure vessel to match pressure requirements and form the pressure vessel shown in Figure 8. Four parameters in this problem need to be minimized, including the thickness of the shell (Ts), the thickness of head (Th), inner radius (R), and the length of the cylindrical section without the head (L), as shown in Figure 8. The constraints and equation are as follows.

Consider

$$\vec{x} = [\, x_1 \;\; x_2 \;\; x_3 \;\; x_4 \,] = [\, T_s \;\; T_h \;\; R \;\; L \,]. \tag{15}$$

Minimize

$$f(\vec{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3, \tag{16}$$

subject to

$$\begin{cases} g_1(\vec{x}) = -x_1 + 0.0193 x_3 \le 0, \\ g_2(\vec{x}) = -x_3 + 0.00954 x_3 \le 0, \\ g_3(\vec{x}) = -\pi x_3^2 x_4 - \dfrac{4}{3}\pi x_3^3 + 1296000 \le 0, \\ g_4(\vec{x}) = x_4 - 240 \le 0. \end{cases} \tag{17}$$

Variable range is

$$\begin{cases} 0 \le x_1 \le 99, \\ 0 \le x_2 \le 99, \\ 10 \le x_3 \le 200, \\ 10 \le x_4 \le 200. \end{cases} \tag{18}$$

Table 5: Results of algorithms on 23 benchmark functions.

Function		HSMSSA	SMA	SSA	AO	AOA	WOA	SCA	MVO
F1	Mean	**0.0000E + 00**	3.3100$E-303$	2.1200$E-07$	3.4900$E-113$	4.9600$E-06$	3.0200$E-74$	2.2039$E+01$	1.2581$E+00$
	Std	**0.0000E + 00**	**0.0000E + 00**	2.3600$E-07$	1.9100$E-112$	2.1600$E-06$	1.6500$E-73$	3.4334$E+01$	4.4824$E-01$
F2	Mean	**0.0000E + 00**	8.7800$E-161$	1.9590$E+00$	5.0900$E-59$	1.9805$E-03$	3.1700$E-50$	2.6402$E-02$	1.2927$E+00$
	Std	**0.0000E + 00**	4.8000$E-160$	1.4157$E+00$	2.7900$E-58$	1.9063$E-03$	1.5800$E-49$	2.9329$E-02$	1.1310$E+00$
F3	Mean	**0.0000E + 00**	1.4159e − 313	2.1110$E+03$	3.7000$E-101$	9.6539$E-04$	4.3306$E+04$	8.0776$E+03$	2.3785$E+02$
	Std	**0.0000E + 00**	**0.0000E + 00**	1.5514$E+03$	2.0200$E-100$	8.1952$E-04$	1.6945$E+04$	4.7486$E+03$	1.2902$E+02$
F4	Mean	**0.0000E + 00**	1.4800$E-150$	1.1761$E+01$	1.7300$E-55$	1.7772$E-02$	5.2341$E+01$	3.4937$E+01$	1.9521$E+00$
	Std	**0.0000E + 00**	8.1300$E-150$	4.0647$E+00$	9.5000$E-55$	1.1446$E-02$	2.8502$E+01$	1.3547$E+01$	7.5824$E-01$
F5	Mean	8.0753$E-02$	5.0378$E+00$	1.0827$E+02$	**5.2141E − 03**	2.8034$E+01$	2.8032$E+01$	7.1843$E+04$	4.7661$E+02$
	Std	2.7660$E-01$	9.1628$E+00$	1.4901$E+02$	**7.2117E − 03**	2.0336$E-01$	4.1099$E-01$	1.2854$E+05$	7.8524$E+02$
F6	Mean	**8.2900E − 07**	5.8093$E-03$	1.3800$E-07$	1.3859$E-04$	3.1018$E+00$	3.6069$E-01$	1.1633$E+01$	1.1004$E+00$
	Std	**4.9600E − 07**	2.4730$E-03$	1.5300$E-07$	2.1774$E-04$	2.2067$E-01$	2.2358$E-01$	9.7843$E+00$	3.6012$E-01$
F7	Mean	**6.4400E − 05**	1.6357$E-04$	1.6394$E-01$	7.8800$E-05$	1.0695$E-04$	3.0691$E-03$	1.3682$E-01$	3.6527$E-02$
	Std	**6.2600E − 05**	1.3341$E-04$	6.1309$E-02$	6.5100$E-05$	1.0095$E-04$	2.8218$E-03$	2.0698$E-01$	1.6353$E-02$
F8	Mean	**−1.2569E + 04**	−1.2569$E+04$	−7.1073$E+03$	−9.3742$E+03$	−5.6142$E+03$	−1.0050$E+04$	−3.7684$E+03$	−7.6853$E+03$
	Std	**2.0511E − 02**	5.2108$E-01$	9.0982$E+02$	3.7928$E+03$	3.9485$E+02$	1.6846$E+03$	2.8948$E+02$	6.3161$E+02$
F9	Mean	**0.0000E + 00**	**0.0000E + 00**	5.3927$E+01$	**0.0000E + 00**	1.5400$E-06$	**0.0000E + 00**	3.6516$E+01$	1.2767$E+02$
	Std	**0.0000E + 00**	**0.0000E + 00**	1.6886$E+01$	**0.0000E + 00**	1.0800$E-06$	**0.0000E + 00**	3.5214$E+01$	3.5868$E+01$
F10	Mean	**8.8800E − 16**	**8.8800E − 16**	2.5258$E+00$	2.1600$E-14$	4.2571$E-04$	4.3200$E-15$	1.4566$E+01$	2.5576$E+00$
	Std	**0.0000E + 00**	**0.0000E + 00**	6.8528$E-01$	1.1400$E-13$	1.8869$E-04$	2.7200$E-15$	8.3427$E+00$	3.3443$E+00$
F11	Mean	**0.0000E + 00**	**0.0000E + 00**	2.0677$E-02$	**0.0000E + 00**	5.1653$E-04$	1.9620$E-02$	9.3797$E-01$	8.7497$E-01$
	Std	**0.0000E + 00**	**0.0000E + 00**	1.4808$E-02$	**0.0000E + 00**	2.6954$E-03$	5.4806$E-02$	3.3652$E-01$	7.4092$E-02$
F12	Mean	2.4700$E-05$	5.9758$E-03$	7.1903$E+00$	3.3600$E-06$	7.3782$E-01$	1.7457$E-02$	1.4353$E+03$	2.1818$E+00$
	Std	5.1900$E-05$	6.9167$E-03$	2.7304$E+00$	3.9400$E-06$	2.7918$E-02$	1.2428$E-02$	7.0295$E+03$	1.3736$E+00$
F13	Mean	**4.6300E − 07**	5.4042$E-03$	1.2368$E+01$	1.4900$E-05$	2.9623$E+00$	5.1556$E-01$	1.8722$E+05$	1.4761$E-01$
	Std	**1.4800E − 07**	7.9865$E-03$	1.2274$E+01$	2.0600$E-05$	2.0518$E-02$	2.3195$E-01$	4.3886$E+05$	6.2682$E-02$
F14	Mean	**9.9800E − 01**	**9.9800E − 01**	1.2294$E+00$	2.4408$E+00$	1.0093$E+01$	2.9945$E+00$	1.7920$E+00$	**9.9800E − 01**
	Std	**5.0400E − 16**	3.5900$E-13$	6.7284$E-01$	2.6435$E+00$	3.9607$E+00$	3.2280$E+00$	9.8831$E-01$	4.2300$E-11$
F15	Mean	**3.9234E − 04**	5.4854$E-04$	2.8300$E-03$	4.8202$E-04$	7.5731$E-03$	8.4669$E-04$	1.0574$E-03$	5.7644$E-03$
	Std	**1.0289E − 04**	2.5100$E-04$	5.9501$E-03$	1.0743$E-04$	1.9781$E-02$	6.5602$E-04$	3.8453$E-04$	1.2715$E-02$
F16	Mean	**−1.0316E + 00**	**−1.0316E + 00**	**−1.0316E + 00**	−1.0311$E+00$	**−1.0316E + 00**	**−1.0316E + 00**	**−1.0316E + 00**	**−1.0316E + 00**
	Std	**1.4300E − 13**	6.6000$E-10$	2.3300$E-12$	4.6089$E-04$	2.7600$E-11$	1.0400$E-09$	4.2000$E-05$	3.8000$E-07$
F17	Mean	**3.9789E − 01**	**3.9789E − 01**	**3.9789E − 01**	3.9808$E-01$	4.0051$E-01$	**3.9789E − 01**	3.9921$E-01$	**3.9789E − 01**
	Std	**2.6700E − 12**	3.0000$E-08$	1.3400$E-10$	2.2131$E-04$	1.4391$E-02$	6.9400$E-06$	1.2200$E-03$	1.3300$E-07$
F18	Mean	3.9000$E+00$	3.0000$E+00$	**3.0000E + 00**	3.0365$E+00$	1.2901$E+01$	3.0003$E+00$	3.0001$E+00$	8.4000$E+00$
	Std	4.9295$E+00$	1.8000$E-10$	**1.5000E − 13**	3.6597$E-02$	2.1835$E+01$	1.5988$E-03$	9.1200$E-05$	2.0550$E+01$
F19	Mean	**−3.8628E + 00**	**−3.8628E + 00**	**−3.8628E + 00**	−3.8560$E+00$	−3.7416$E+00$	−3.8575$E+00$	−3.8512$E+00$	**−3.8628E + 00**
	Std	**4.2800E − 11**	8.4700$E-07$	1.4600$E-10$	7.4968$E-03$	5.3652$E-01$	5.7697$E-03$	8.9408$E-03$	1.9300$E-06$
F20	Mean	**−3.2823E + 00**	−3.2426$E+00$	−3.2259$E+00$	−3.2075$E+00$	−3.2903$E+00$	−3.1996$E+00$	−2.9186$E+00$	−3.2655$E+00$
	Std	**5.7048E − 02**	5.7100$E-02$	5.9965$E-02$	7.2462$E-02$	6.3492$E-02$	1.2887$E-01$	3.7689$E-01$	6.1457$E-02$

Table 5: Continued.

Function		HSMSSA	SMA	SSA	AO	AOA	WOA	SCA	MVO
F21	Mean	**−1.0153E + 01**	$-1.0153E+01$	$-6.3092E+00$	$-1.0142E+01$	$-7.4694E+00$	$-8.1869E+00$	$-2.4287E+00$	$-7.0425E+00$
	Std	**1.0300E − 07**	$1.9553E-04$	$3.5233E+00$	$2.6961E-02$	$3.0153E+00$	$2.6299E+00$	$2.0334E+00$	$3.0750E+00$
F22	Mean	**−1.0403E + 01**	$-1.0403E+01$	$-7.8362E+00$	$-1.0392E+01$	$-7.3418E+00$	$-7.6682E+00$	$-3.4206E+00$	$-8.4065E+00$
	Std	**1.6600E − 07**	$3.1185E-04$	$3.4923E+00$	$1.8669E-02$	$3.2433E+00$	$3.4333E+00$	$1.7953E+00$	$2.9406E+00$
F23	Mean	**−1.0536E + 01**	$-1.0536E+01$	$-9.4298E+00$	$-1.0525E+01$	$-7.9685E+00$	$-7.2480E+00$	$-3.6694E+00$	$-9.6618E+00$
	Std	**9.5800E − 08**	$2.5920E-04$	$2.5607E+00$	$1.6743E-02$	$3.5174E+00$	$3.2297E+00$	$1.8159E+00$	$2.3084E+00$

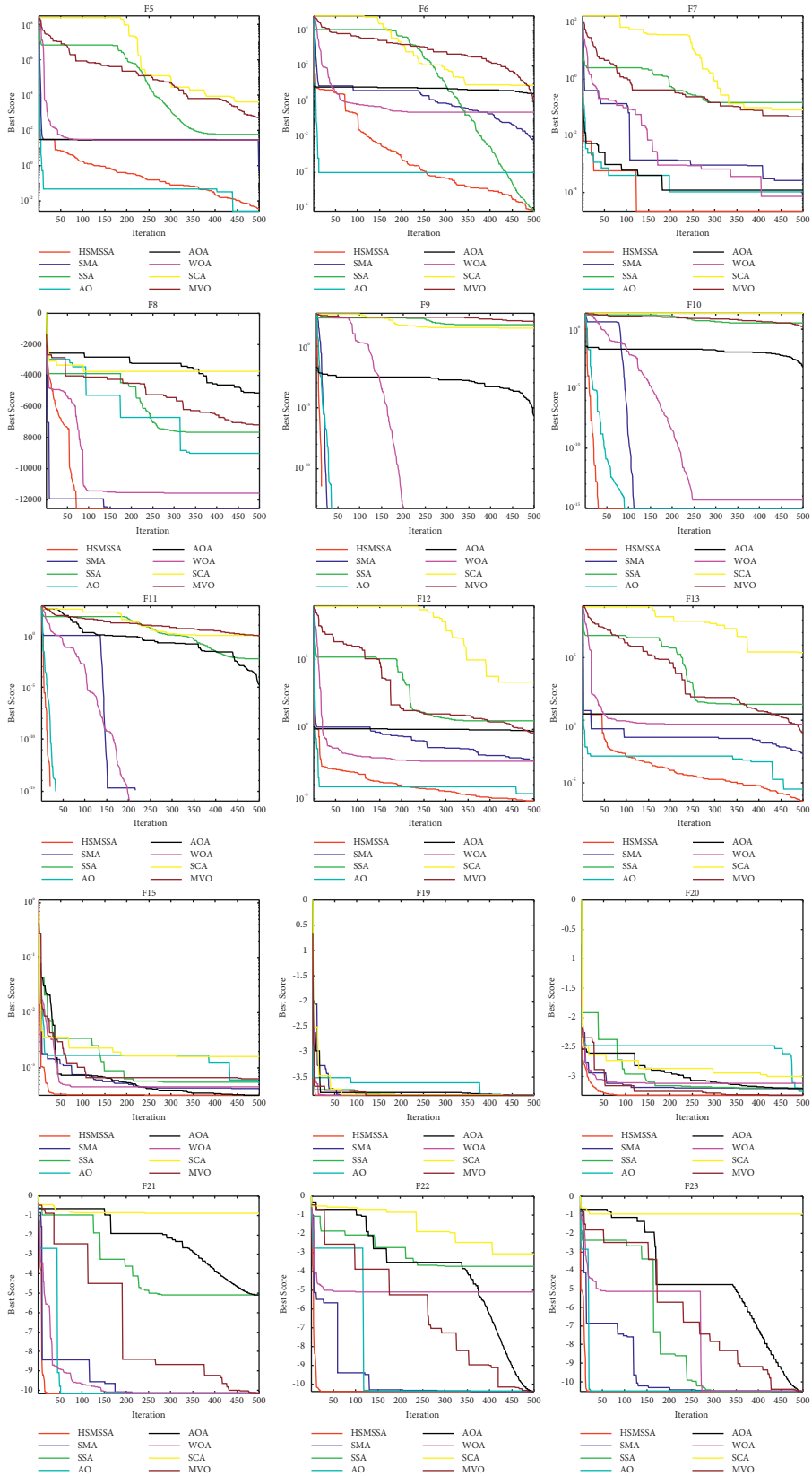

Figure 6: Convergence curves of 23 benchmark functions.

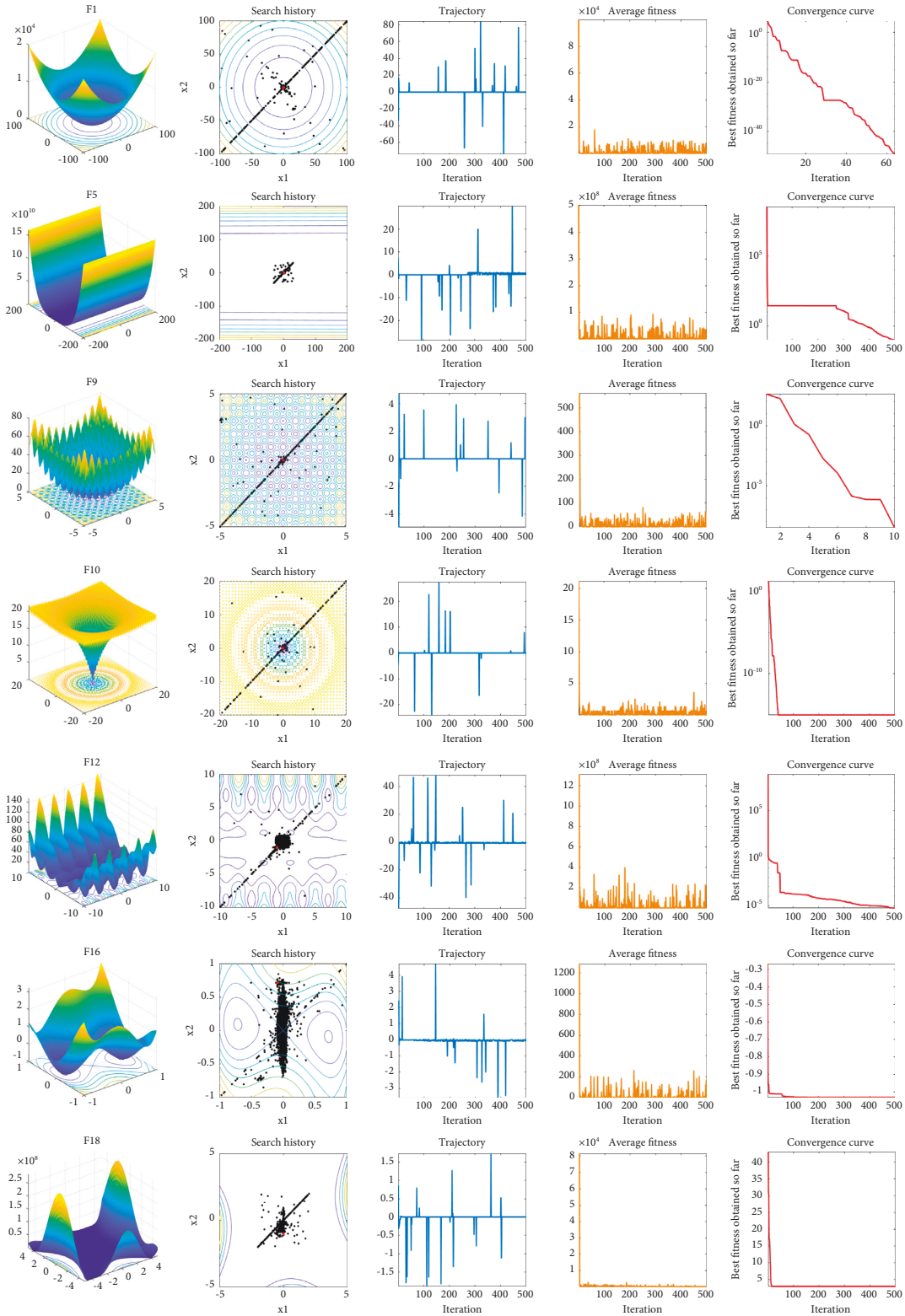

Figure 7: Parameter space, search history, trajectory, average fitness, and convergence curves of HSMSSA.

Table 6: The results of Wilcoxon's sign rank test for all functions.

Function	HSMSSA versus						
	SMA	SSA	AO	AOA	WOA	SCA	MVO
F1	**NaN**	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F2	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F3	**NaN**	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F4	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F5	6.1035$E-05$	6.1035$E-05$	4.2725$E-03$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F6	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F7	2.1545$E-02$	6.1035$E-05$	**1.6882E−01**	**7.1973E−01**	1.8311$E-04$	6.1035$E-05$	6.1035$E-05$
F8	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F9	**NaN**	6.1035$E-05$	**NaN**	6.1035$E-05$	**NaN**	6.1035$E-05$	6.1035$E-05$
F10	**NaN**	6.1035$E-05$	**NaN**	6.1035$E-05$	1.9531$E-03$	6.1035$E-05$	6.1035$E-05$
F11	**NaN**	6.1035$E-05$	**NaN**	6.1035$E-05$	**NaN**	6.1035$E-05$	6.1035$E-05$
F12	6.1035$E-05$	6.1035$E-05$	3.0518$E-04$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F13	6.1035$E-05$	6.1035$E-05$	3.0518$E-04$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F14	6.1035$E-05$	**5.0000E−01**	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F15	1.5259$E-03$	6.1035$E-05$	1.2207$E-04$	4.3721$E-02$	1.8311$E-04$	6.1035$E-05$	1.2207$E-04$
F16	6.1035$E-05$	1.3184$E-02$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F17	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	4.2725$E-04$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F18	**1.7334E−01**	6.4697$E-03$	4.3252$E-02$	3.5339$E-02$	4.3252$E-02$	4.3252$E-02$	4.3252$E-02$
F19	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F20	2.5574$E-02$	1.0254$E-02$	1.1597$E-03$	**2.2931E−01**	4.5359$E-02$	6.1035$E-05$	**3.5913E−01**
F21	6.1035$E-05$	2.5574$E-02$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F22	6.1035$E-05$	**4.2120E−01**	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
F23	6.1035$E-05$	**4.2120E−01**	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$	6.1035$E-05$
(W|L|T)	(17|1|5)	(20|3|0)	(19|1|3)	(21|2|0)	(21|0|2)	(23|0|0)	(22|1|0)

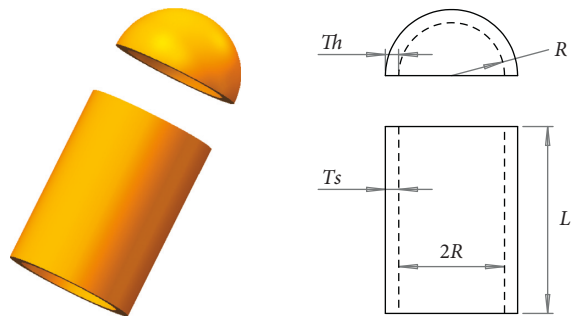

Figure 8: Pressure vessel design problem.

From the results in Table 7, we can see that HSMSSA can obtain superior optimal values compared with SMA, SSA, AO, AOA, WOA, SCA, and MVO.

4.2.2. Tension Spring Design Problem. This problem [27] tries to minimize the weight of the tension spring, and there are three parameters that need to be minimized, including the wire diameter (d), mean coil diameter (D), and the number of active coils (N). Figure 9 shows the structure of the tension spring. The mathematical of this problem can be written as follows.

Consider

$$\vec{x} = [\, x_1 \;\; x_2 \;\; x_3 \;\; x_4 \,] = [\, d \;\; D \;\; N \,]. \tag{19}$$

Minimize

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2, \tag{20}$$

subject to

$$\begin{aligned} g_1(\vec{x}) &= 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0, \\ g_2(\vec{x}) &= \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} \le 0, \\ g_3(\vec{x}) &= 1 - \frac{140.45x_1}{x_2^2x_3} \le 0, \\ g_4(\vec{x}) &= \frac{x_1 + x_2}{1.5} - 1 \le 0. \end{aligned} \tag{21}$$

Table 7: Comparison of HSMSSA results with other competitors for the pressure vessel design problem.

Algorithm	Optimum variables				Optimum cost
	T_s	T_h	R	L	
HSMSSA	0.8533992	0.4183956	45.8059	135.5385	5961.5318
SMA	0.8744808	0.4273835	46.83887	125.5848	6008.4161
SSA	0.8934774	0.4387327	47.53594	119.2156	6047.6543
AO	0.8828092	0.4312524	47.26462	121.6449	6028.1585
AOA	0.807105	0.4426515	44.63354	147.5659	6052.8917
WOA	0.8310091	0.3646671	44.00895	154.3182	6047.0417
SCA	0.8820038	0.4335084	47.24144	125.7922	6139.5293
MVO	0.8380677	0.4992101	45.82367	135.3623	6253.5397

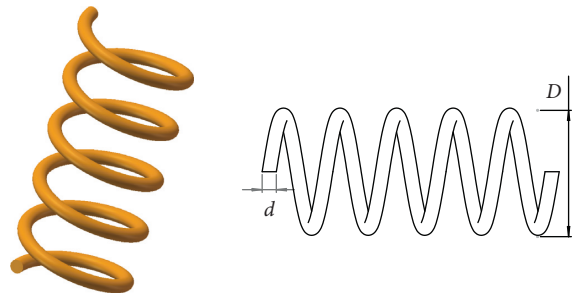

Figure 9: Tension spring design problem.

Variable range is

$$\begin{aligned} 0.05 \le x_1 \le 2.00, \\ 0.25 \le x_2 \le 1.30, \\ 2.00 \le x_3 \le 15.00. \end{aligned} \tag{22}$$

Results of HSMSSA for solving tension spring design problem are listed in Table 8, which are compared with SMA, SSA, AO, AOA, WOA, SCA, and MVO. It is evident that HSMSSA obtained the best results compared to all other algorithms.

4.2.3. Three-Bar Truss Design Problem. Three-bar truss design is a complex problem in the field of civil engineering [49]. The goal of this problem is to achieve the minimum weight in truss design. Figure 10 shows the design of this problem. The formula of this problem can be described as follows.

Consider

$$\vec{x} = [\, x_1 \;\; x_2 \,] = [\, A_1 \;\; A_2 \,]. \tag{23}$$

Minimize

$$f(\vec{x}) = \left(2\sqrt{2}\, x_1 + x_2\right) * l, \tag{24}$$

subject to

$$\begin{aligned} g_1(\vec{x}) &= \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0, \\ g_2(\vec{x}) &= \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0, \\ g_3(\vec{x}) &= \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \le 0. \end{aligned} \tag{25}$$

Variable range is

$$\begin{aligned} 0 \le x_1, \\ x_2 \le 1, \end{aligned} \tag{26}$$

where $l = 100$ cm, $P = 2$ KN/cm^2, and $\sigma = 2$ KN/cm^2.

Results of HSMSSA for solving the three-bar design problem are listed in Table 9, which are compared with SMA, SSA, AO, AOA, WOA, SCA, and MVO. It can be observed that HSMSSA has an excellent ability to solve the problem in confined space.

4.2.4. Speed Reducer Problem. In this problem [15], the total weight of the reducer is minimized by optimizing seven variables. Figure 11 shows the design of this problem, and the mathematical formula is as follows.

Minimize

Table 8: Comparison of HSMSSA results with other competitors for the tension spring design problem.

Algorithm	Optimum variables			Optimum weight
	d	D	N	
HSMSSA	**0.05**	**0.348099**	**10.6486**	**0.011007**
SMA	0.057372	0.57232	4.1494	0.011584
SSA	0.05936	0.63542	3.4741	0.012256
AO	0.056912	0.55833	4.3271	0.011442
AOA	0.052579	0.41008	8.9843	0.012453
WOA	0.055292	0.47277	6.929	0.012906
SCA	0.0588	0.6172	3.65	0.012057
MVO	0.05251	0.37602	10.33513	0.012790

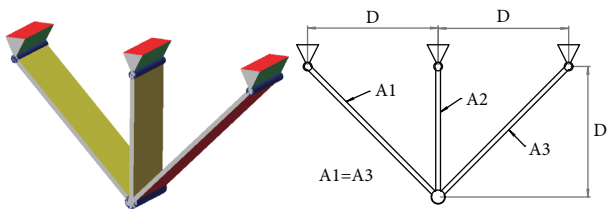

Figure 10: Three-bar truss design problem.

Table 9: Comparison of HSMSSA results with other competitors for the three-bar truss design problem.

Algorithm	Optimum variables		Optimum cost
	x_1	x_2	
HSMSSA	**0.78842**	**0.40811**	**263.8523**
SMA	0.79109	0.40022	263.8668
SSA	0.77823	0.43736	263.9363
AO	0.77899	0.43541	263.9197
AOA	0.76342	0.48382	264.3549
WOA	0.78357	0.42252	263.886
SCA	0.78058	0.43766	264.5463
MVO	0.75492	0.51216	264.7851

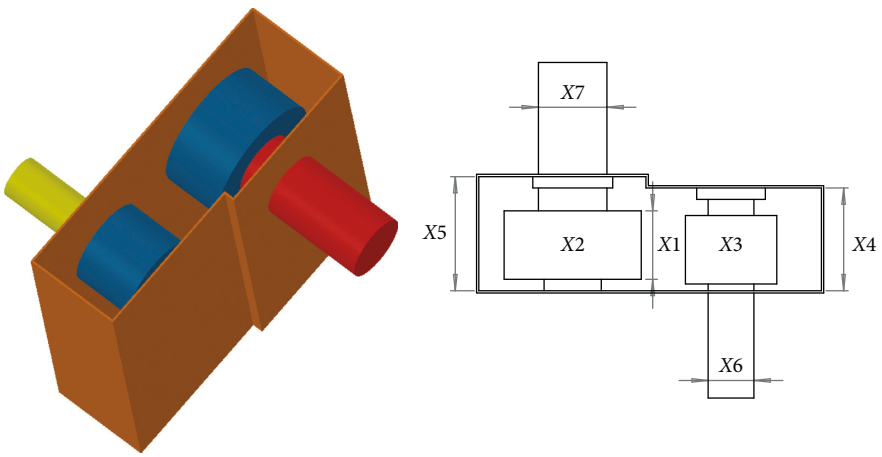

Figure 11: Speed reducer problem.

Table 10: Comparison of HSMSSA results with other competitors for the speed reducer design problem.

Algorithm	Optimum variables							Optimum weight
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
HSMSSA	**3.4976**	**0.7**	**17**	**7.3**	**7.8**	**3.35006**	**5.28553**	**2995.4374**
SMA	3.51443	0.7	17	7.32444	7.80527	3.35235	5.28494	3002.5941
SSA	3.49767	0.7	17	7.87797	8.09401	3.3943	5.28574	3018.644
AO	3.5138	0.7	17	7.41461	7.81291	3.37709	5.28457	3009.9097
AOA	3.55989	0.7	17	7.49997	8.3	3.4623	5.28512	3061.8731
WOA	3.49739	0.7	17	7.87039	8.0769	3.45309	5.2854	3034.0596
SCA	3.6	0.7	17	7.3	8.3	3.37932	5.2799	3063.9102
MVO	3.6	0.7	17	8.3	8.3	3.38276	5.36041	3111.6609

$$f(\vec{x}) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right), \tag{27}$$

subject to

$$\begin{aligned}
g_1(\vec{x}) &= \frac{27}{x_1x_2^2x_3} - 1 \le 0,\\
g_2(\vec{x}) &= \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0,\\
g_3(\vec{x}) &= \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0,\\
g_4(\vec{x}) &= \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0,\\
g_5(\vec{x}) &= \frac{\sqrt{\left(745x_4/x_2x_3\right)^2 + 16.9 \times 10^6}}{110.0x_6^3} - 1 \le 0,\\
g_6(\vec{x}) &= \frac{\sqrt{\left(745x_4/x_2x_3\right)^2 + 157.5 \times 10^6}}{85.0x_6^3} - 1 \le 0,\\
g_7(\vec{x}) &= \frac{x_2x_3}{40} - 1 \le 0,\\
g_8(\vec{x}) &= \frac{5x_2}{x_1} - 1 \le 0,\\
g_9(\vec{x}) &= \frac{x_1}{12x_2} - 1 \le 0,\\
g_{10}(\vec{x}) &= \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0,\\
g_{11}(\vec{x}) &= \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0.
\end{aligned} \tag{28}$$

Variable range is

$$\begin{aligned}
2.6 \le x_1 \le 3.6,\\
0.7 \le x_2 \le 0.8,\\
17 \le x_3 \le 28,\\
7.3 \le x_4 \le 8.3,\\
7.8 \le x_5 \le 8.3,\\
2.9 \le x_6 \le 3.9,\\
5.0 \le x_7 \le 5.5.
\end{aligned} \tag{29}$$

The comparison results are listed in Table 10, which shows the advantage of HSMSSA in realizing the minimum total weight of the problem.

4.2.5. Cantilever Beam Design. Cantilever beam design is a type of concrete engineering problem. This problem aims to determine the minimal total weight of the cantilever beam by optimizing the hollow square cross-section parameters [24]. Figure 12 illustrates the design of this problem, and the mathematical described is as follows.

Consider

$$x = [\, x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5 \,]. \tag{30}$$

Minimize

$$f(\vec{x}) = 0.6224\left(x_1 + x_2 + x_3 + x_4 + x_5\right), \tag{31}$$

subject to

$$g(\vec{x}) = \frac{60}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0. \tag{32}$$

Variable range is as follows: $0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100$.

The results are shown in Table 11. From this table, we can see that the performance of HSMSSA is better than all other algorithms and the obtained total weight is minimized.

As a summary, this section demonstrates the superiority of the proposed HSMSSA algorithm in different

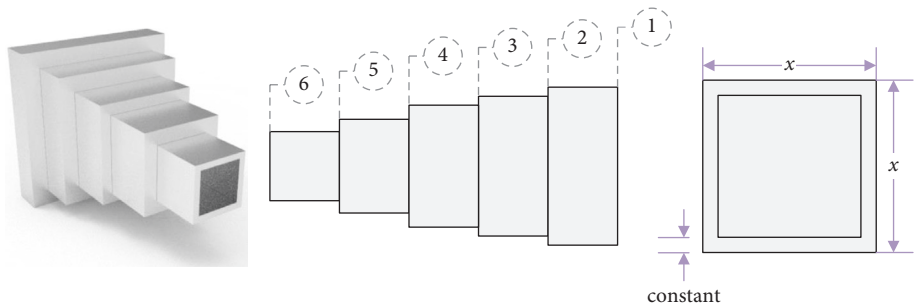

Figure 12: Cantilever beam design [24].

Table 11: Comparison of HSMSSA results with other competitors for the cantilever beam design problem.

Algorithm	Optimum variables					Optimum weight
	x_1	x_2	x_3	x_4	x_5	
HSMSSA	**5.9979**	**5.3289**	**4.4999**	**3.4817**	**2.166**	**1.3400**
SMA	6.0846	5.2309	4.5602	3.4725	2.1349	1.3405
SSA	6.1056	5.1001	4.303	3.7365	2.3183	1.3456
AO	5.8236	5.3813	4.5178	3.4768	2.3159	1.3426
AOA	5.7014	5.3722	4.6577	3.6643	2.1551	1.3448
WOA	5.8492	5.0587	4.8917	3.682	2.1564	1.3469
SCA	5.9165	5.9187	4.5133	3.3167	1.9823	1.3508
MVO	7.1088	5.0161	4.2203	3.4064	2.105	1.3647

characteristics and real case studies. HSMSSA is able to outperform the basic SMA and SSA and other well-known algorithms with very competitive results, which are derived from the robust exploration and exploitation capabilities of HSMSSA. Excellent performance in solving industrial engineering design problems indicates that HSMSSA can be widely used in real-world optimization problems.

5. Conclusion

In this paper, a Hybrid Slime Mould Salp Swarm Algorithm (HSMSSA) is proposed by combining the whole SMA as leaders and the exploitation phase of SSA as followers. At the same time, two strategies, including Levy flight and mutation opposition-based learning, are incorporated to enhance the capabilities of exploration and exploitation of HSMSSA. The 23 standard benchmark functions are utilized to evaluate this algorithm for analyzing its exploration, exploitation, and local optima avoidance capabilities. The experimental results show competitive advantages compared to other state-of-the-art metaheuristic algorithms, proving that HSMSSA has better performance than others. Five engineering design problems are solved as well to verify the superiority of the algorithm further, and the results are also very competitive with other metaheuristic algorithms.

The proposed HSMSSA can produce very effective results for complex benchmark functions and constrained engineering problems. In the future, HSMSSA can be applied to real-world optimization problems such as multiobjective problems, feature selection, multithresholding image segmentation, convolution neural network, or any problem that belongs to NP-complete or NP-hard problems.

Data Availability

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

Conflicts of Interest

All authors declare that there are no conflicts of interest.

Acknowledgments

This research was funded by the Sanming University Introduces High-Level Talents to Start Scientific Research Funding Support Project (20YG01 and 20YG14), the Guiding Science and Technology Projects in Sanming City (2020-S-39, 2020-G-61, and 2021-S-8), the Educational Research Projects of Young and Middle-Aged Teachers in Fujian Province (JAT200638 and JAT200618), the Scientific Research and Development Fund of Sanming University (B202029 and B202009), the Open Research Fund of Key Laboratory of Agricultural Internet of Things in Fujian Province (ZD2101), the Ministry of Education Cooperative Education Project (202002064014), the School Level Education and Teaching Reform Project of Sanming University (J2010306 and J2010305), and the Higher Education Research Project of Sanming University (SHE2102 and SHE2013).

References

[1] H. Jia, C. Lang, D. Oliva, W. Song, and X. Peng, "Dynamic harris hawks optimization with mutation mechanism for

satellite image segmentation," *Remote Sensing*, vol. 11, no. 12, pp. 1421–1456, 2019.

[2] L. Abualigah and A. Diabat, "Advances in sine cosine algorithm: a comprehensive survey," *Artificial Intelligence Review*, vol. 54, no. 4, pp. 2567–2608, 2021.

[3] L. Abualigah and A. Diabat, "A comprehensive survey of the Grasshopper optimization algorithm: results, variants, and applications," *Neural Computing & Applications*, vol. 32, no. 19, pp. 15533–15556, 2020.

[4] J. J. Grefenstette, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2-3, pp. 95–99, 1988.

[5] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[6] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

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

[8] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[9] A. Hatamlou, "Black hole: a new heuristic optimization approach for data clustering," *Information Sciences*, vol. 222, pp. 175–184, 2013.

[10] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: a nature-inspired algorithm for global optimization," *Neural Computing & Applications*, vol. 27, no. 2, pp. 495–513, 2015.

[11] B. Alatas, "ACROA: artificial chemical reaction optimization algorithm for global optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13170–13180, 2011.

[12] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: Ray Optimization," *Computers & Structures*, vol. 112-113, pp. 283–294, 2012.

[13] F. F. Moghaddam, R. F. Moghaddam, and M. Cheriet, "Curved Space Optimization: a random search based on general relativity theory," pp. 1–16, 2012, https://arxiv.org/abs/1208.2214.

[14] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[15] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 376, pp. 113609–113647, 2021.

[16] V. Majidnezhad, M. R. F. Derakhshi, and S. Parsai, "Heat transfer relation-based optimization algorithm (HTOA)," *Soft Computing*, vol. 25, pp. 8129–8158, 2021.

[17] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, pp. 1942–1948, Nagoya, Japan, October 1995.

[18] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[19] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimization," *International Journal of Bio-Inspired Computation*, vol. 2, pp. 78–84, 2013.

[20] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[21] X.-S. Yang and S. Suash Deb, "Cuckoo search via lévy flights," in *Proceedings of the World Congress on Nature & Biologically Inspired Computing*, pp. 210–214, Coimbatore, India, December 2009.

[22] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[23] H. A. Alsattar, A. A. Zaidan, and B. B. Zaidan, "Novel meta-heuristic bald eagle search optimisation algorithm," *Artificial Intelligence Review*, vol. 53, no. 3, pp. 2237–2264, 2019.

[24] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-qaness, and A. H. Gandomi, "Aquila optimizer: a novel meta-heuristic optimization algorithm," *Computers & Industrial Engineering*, vol. 157, Article ID 107250, 2021.

[25] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," *Evolutionary Computation*, vol. 25, no. 1, pp. 1–54, 2017.

[26] M. Cacciola, S. Calcagno, F. C. Morabito, and M. Versaci, "Swarm optimization for imaging of corrosion by impedance measurements in eddy current test," *IEEE Transactions on Magnetics*, vol. 43, no. 4, pp. 1853–1856, 2007.

[27] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, and 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.

[28] M. Tubishat, S. Ja'afar, M. Alswaitti et al., "Dynamic Salp Swarm algorithm for feature selection," *Expert Systems with Applications*, vol. 164, Article ID 113873, 2021.

[29] R. Salgotra, U. Singh, S. Singh, G. Singh, and N. Mittal, "Self-adaptive salp swarm algorithm for engineering optimization problems," *Applied Mathematical Modelling*, vol. 89, pp. 188–207, 2021.

[30] 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.

[31] H. Jia and C. Lang, "Salp swarm algorithm with crossover scheme and Lévy flight for global optimization," *Journal of Intelligent and Fuzzy Systems*, vol. 40, no. 5, pp. 9277–9288, 2021.

[32] M. M. Saafan and E. M. El-gendy, "IWOSSA: an improved whale optimization salp swarm algorithm for solving optimization problems," *Expert Systems with Applications*, vol. 176, no. 2, Article ID 114901, 2021.

[33] N. Singh, S. B. Singh, and E. H. Houssein, "Hybridizing salp swarm algorithm with particle swarm optimization algorithm for recent optimization functions," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 1–34, 2020.

[34] M. Q. H. Abadi, S. Rahmati, A. Sharifi, and M. Ahmadi, "HSSAGA: designation and scheduling of nurses for taking care of COVID-19 patients using novel method of hybrid salp swarm algorithm and genetic algorithm," *Applied Soft Computing*, vol. 108, Article ID 107449, 2021.

[35] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime Mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, no. 3, pp. 300–323, 2020.

[36] M. Abdel-Basset, R. Mohamed, R. K. Chakrabortty, M. J. Ryan, and S. Mirjalili, "An efficient binary slime mould algorithm integrated with a novel attacking-feeding strategy for feature selection," *Computers & Industrial Engineering*, vol. 153, Article ID 107078, 2021.

[37] R.-E. Precup, R.-C. David, R.-C. Roman, E. M. Petriu, and A.-I. Szedlak-Stinean, "Slime mould algorithm-based tuning

of cost-effective fuzzy controllers for servo systems," *International Journal of Computational Intelligence Systems*, vol. 14, no. 1, pp. 1042–1052, 2021.

[38] S. Zhao, P. Wang, A. A. Heidari et al., "Multilevel threshold image segmentation with diffusion association Slime Mould algorithm and Renyi's entropy for chronic obstructive pulmonary disease," *Computers in Biology and Medicine*, vol. 134, Article ID 104427, 2021.

[39] D. Izci, S. Ekinci, and S. Ekinci, "Comparative performance analysis of slime mould algorithm for efficient design of proportional-integral-derivative controller," *Electrica*, vol. 21, no. 1, pp. 151–159, 2021.

[40] A. M. Othman and A. A. El-fergany, "Optimal dynamic operation and modeling of parallel connected multi-stacks fuel cells with improved slime mould algorithm," *Renewable Energy*, vol. 175, pp. 770–782, 2021.

[41] R. Jensi and G. W. Jiji, "An enhanced particle swarm optimization with levy flight for global optimization," *Applied Soft Computing*, vol. 43, pp. 248–261, 2016.

[42] G. Iacca, V. C. Santos Jr., V. Veloso de Melo, and V. V. Melo, "An improved Jaya optimization algorithm with Lévy flight," *Expert Systems with Applications*, vol. 165, Article ID 113902, 2021.

[43] Y. Liu and B. Cao, "A novel ant colony optimization algorithm with Levy flight," *IEEE Access*, vol. 8, pp. 67205–67213, 2020.

[44] X.-l. Lu and G. He, "QPSO algorithm based on lévy flight and its application in fuzzy portfolio," *Applied Soft Computing*, vol. 99, no. 1, Article ID 106894, 2021.

[45] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Proceedings of the Computational Intelligence for Modelling, Control & Automation*, pp. 695–701, Vienna, Austria, November 2005.

[46] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Sciences*, vol. 181, no. 20, pp. 4699–4714, 2011.

[47] A. A. Ewees, M. Abd Elaziz, and E. H. Houssein, "Improved grasshopper optimization algorithm using opposition-based learning," *Expert Systems with Applications*, vol. 112, no. 12, pp. 156–172, 2018.

[48] S. Gupta and K. Deep, "A hybrid self-adaptive sine cosine algorithm with opposition based learning," *Expert Systems with Applications*, vol. 119, no. 4, pp. 210–230, 2019.

[49] S. Gupta, K. Deep, A. A. Heidari, H. Moayedi, and M. Wang, "Opposition-based learning harris hawks optimization with advanced transition rules: principles and analysis," *Expert Systems with Applications*, vol. 158, Article ID 113510, 2020.

[50] Z. Guo, S. Wang, X. Yue, and H. Yang, "Global harmony search with generalized opposition-based learning," *Soft Computing*, vol. 21, no. 8, pp. 2129–2137, 2017.

[51] T. K. Sharma and M. Pant, "Opposition based learning ingrained shuffled frog-leaping algorithm," *Journal of Computational Science*, vol. 21, pp. 307–315, 2017.

[52] Q. Fan, Z. Chen, and Z. Xia, "A novel quasi-reflected harris hawks optimization algorithm for global optimization problems," *Soft Computing*, vol. 24, no. 19, pp. 14825–14843, 2020.

[53] H. Jia, X. Peng, and C. Lang, "Remora optimization algorithm," *Expert Systems with Applications*, vol. 185, Article ID 115665, 2021.