symmetry

MDPI

Article

Improved Salp Swarm Algorithm with Simulated Annealing for Solving Engineering Optimization Problems

Qing Duan, Lu Wang, Hongwei Kang *, Yong Shen, Xingping Sun and Qingyi Chen

School of Software, Yunnan University, Kunming 650000, China; qduan@ynu.edu.cn (Q.D.); wl@mail.ynu.edu.cn (L.W.); sheny@ynu.edu.cn (Y.S.); sunxp@ynu.edu.cn (X.S.); devas9@ynu.edu.cn (Q.C.)
* Correspondence: hwkang@ynu.edu.cn

Abstract: Swarm-based algorithm can successfully avoid the local optimal constraints, thus achieving a smooth balance between exploration and exploitation. Salp swarm algorithm (SSA), as a swarm-based algorithm on account of the predation behavior of the salp, can solve complex daily life optimization problems in nature. SSA also has the problems of local stagnation and slow convergence rate. This paper introduces an improved salp swarm algorithm, which improve the SSA by using the chaotic sequence initialization strategy and symmetric adaptive population division. Moreover, a simulated annealing mechanism based on symmetric perturbation is introduced to enhance the local jumping ability of the algorithm. The improved algorithm is referred to SASSA. The CEC standard benchmark functions are used to evaluate the efficiency of the SASSA and the results demonstrate that the SASSA has better global search capability. SASSA is also applied to solve engineering optimization problems. The experimental results demonstrate that the exploratory and exploitative proclivities of the proposed algorithm and its convergence patterns are vividly improved.

Keywords: swarm-based algorithm; salp swarm algorithm; single objective optimization; symmetric perturbation; simulated annealing; engineering optimization problems

Citation: Duan, Q.; Wang, L.; Kang, H.; Shen, Y.; Sun, X.; Chen, Q. Improved Salp Swarm Algorithm with Simulated Annealing for Solving Engineering Optimization Problems. *Symmetry* **2021**, *13*, 1092. https://doi.org/10.3390/sym13061092

Academic Editors: Jan Awrejcewicz, Aviv Gibali, Kok Lay Teo and Yonghong Wu

Received: 1 May 2021
Accepted: 17 June 2021
Published: 20 June 2021

1. Introduction

The purpose of optimization is to find all possible results in a search space and to select the optimal solution according to conditions and parameters. Optimization has been pre-applied to engineering and scientific disciplines, such as chemistry [1], engineering design [2] and information systems [3]. The problems related to these fields are complex in nature and difficult to optimize, which is the basis for developing different meta-heuristic algorithms to find the optimal solution.

There are two kinds of meta-heuristic algorithms: algorithms based on a single solution and algorithms based on swarm solution. The algorithm based on a single solution selects a candidate solution from all possible solution sets, and the selected candidate solution is evaluated repeatedly until the desired optimization result is achieved. The advantage of this approach is that it is faster to execute because of its lower complexity. However, its disadvantage is that it may get stuck in the local area, which results in the failure to obtain the global optimal solution. Popular methods belonging to this category include the mountain climbing algorithm [4], tabu search [5], etc. In contrast, swarm-based algorithms consider all possible solutions rather than a single candidate solution. Algorithms based on a swarm solution are divided into two categories, evolutionary algorithm and swarm intelligence algorithm. Evolutionary algorithms follow a mechanism inspired by biological evolution, including four operators—random selection, reproduction, recombination and mutation—and include the genetic algorithm [6], differential evolution (DE) [7], etc. Swarm intelligence algorithm is a kind of algorithm based on population, which is evolved from social behavior. It realizes the swarm behavior of all kinds of creatures in nature, such as birds, ants, gray wolves, and bees. This approach has been welcomed by researchers for its wide range of applications, ease of understanding and

implementation, and ability to solve many complex optimization problems in real life. Widely used swarm intelligence algorithms include particle swarm optimization (PSO) [8], ant colony optimization (ACO) [9], whale optimization algorithm (WOA) [10], grey wolf optimization (GWO) [11], and artificial bee colony algorithm (ABC) [12]. Some scholars have improved such algorithms and applied them to practical optimization problems. Sun Xingping [13] proposed an improved NSGA-III that combines multi-group coevolution and natural selection. Liu Yang [14] improved the particle swarm optimization algorithm and applied it to the mine water reuse system. Shen Yong [15] improved the JSO algorithm and applied it to solve the constraint problem.

The engineering optimization problem is a constrained optimization problem, and it is one of the most important challenges in practical problems. The main purpose is to solve the problems with constraints in real life and optimize its economic indicators or various parameters. In real life, many engineering optimization problems have complex constraints, and it turns out that they simply add constraints on the basis of functional problems, but they are very difficult in actual operations. Some of these constraint conditions are simple intervals, but more are composed of linear equations, which make the solution space very complicated. Traditional classical algorithms, such as Newton's method, elimination method, and constraint variable rotation method, statically make dynamic problems and can handle these constraint problems to a certain extent. However, due to the complexity of the objective function of many practical constraint optimization problems, these traditional algorithms often do not work well. In recent years, experimental research has found that the swarm intelligence algorithm has unique advantages, so many scholars apply it to solving engineering optimization problems.

Salp swarm algorithm (SSA) [16] is a new meta-heuristic intelligent algorithm proposed by S. Mirjalili in 2017. In algorithm iteration, leaders lead followers and move towards food in a chain behavior. In the process of movement, the leaders are guided by the food source (i.e., the current global optimal solution) to make global exploration, while the followers make full local exploration, which greatly reduces the situation of getting stuck in the local area. Because of its simple structure, fast convergence speed and few control parameters, many scholars have studied and improved it and applied it to different fields. Sayed [17] proposed an SSA algorithm based on chaos theory to solve SSA algorithm's disadvantage that it is prone to fall into local optimal and slow convergence. Ibrahim [18] used the global convergence of PSO to propose a hybrid optimization algorithm based on SSA and PSO. Faris [19] used crossover operators to replace average operators and proposed a binary SSA algorithm with crossover. Liu Jingsen [20] proposed a leader–follower adaptive SSA algorithm and applied it to engineering optimization problems. Nibedan Panda [21] proposed an SSA algorithm based on space transformation search and applied it to the training of neural networks.

In order to improve the optimization ability of SSA, extending the application of algorithm of space, this paper proposes an improved salp swarm algorithm(SASSA) based on the simulated annealing (SA) [22]. First, logistic mapping was used to initialize the population to enhance the diversity of the initial population. Secondly, the symmetric adaptive division of the population was carried out to balance the development and exploration ability of the algorithm. Finally, the simulated annealing mechanism based on symmetric perturbation was introduced into the salp swarm algorithm to improve the performance of the existing algorithm. The performance of the above algorithms was evaluated on the benchmark function, and the new algorithm was compared with the original salp swarm algorithm and other popular meta-heuristic algorithms. The main work we did is as follows:

1. We proposed an improved salp swarm algorithm based on the idea of a simulated annealing algorithm.
2. We tested the improved algorithm on the benchmark function.

3. The advantages of the improved algorithm were verified, and the results evaluated by the original salp swarm algorithm and other meta-heuristic algorithms on benchmark functions such as GWO and WOA are compared.
4. The improved algorithm was applied to solve engineering optimization problems to prove its ability and effectiveness in solving practical problems.

The following sections are organized as follows: Section 2 introduces the background and principle of salp swarm algorithm; Section 3 introduces the improvement process and steps of the algorithm in detail. Section 4 describes the experimental equipment, environment, reference function and required parameters and gives the experimental results and statistical comparison with other algorithms; Section 5 introduces the application of the algorithm in solving engineering optimization Problems. The last section summarizes the conclusion of this paper and gives the future research direction.

2. Salp Swarm Algorithm

2.1. Principle of Bionics

Salps are sea creatures with transparent, pail-shaped bodies. Their body structure are highly similar to those of jellyfish. During movement, salps provide a reverse thrust by drawing water from their surroundings through their barrel-shaped bodies. The body tissues of salp are so fragile that it is difficult for them to survive in the experimental environment. Therefore, it is not until recent years that some breakthroughs have been made in the study of this species, among which the most interesting one is the group behavior of salp.

The group behavior of salp is not distributed in a "group" mode but is often connected end to end to form a "chain" that moves sequentially, as shown in Figure 1. The salp chain also has a leader, which has the optimal judgment on the environment, often staying at the head of the chain. But unlike other groups, the leader no longer directly affects the movement of the whole group, but only directly affects the movement of the second salp next to him, and the second salp directly affects the third salp, and so on. This method is similar to a more rigorous and detailed hierarchy, each individual is only affected by the "directly leader" and cannot be over-managed. Therefore, the influence of the leader on the lower salps is sharply reduced layer by layer. The lower salps can easily retain their diversity rather than blindly move towards the leader. Since the salps follow the movement pattern in succession, the salps other than the leaders are collectively referred to as followers in this paper.

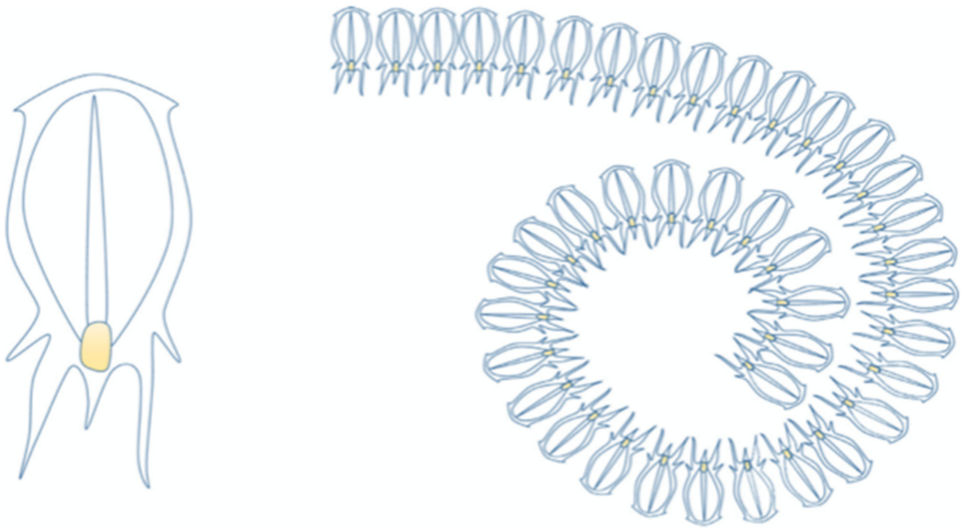

Figure 1. Salp group.

2.2. The Flow of SSA

The optimization of the salp swarm algorithm is as follows [23,24]:

Firstly, population initialization. N is the population size of the salp and D is the spatial dimension. Food exists in space, $F = [F_1, F_2, \ldots, F_D]^T$. The upper and lower bounds of the search space are $ub = [ub_1, ub_2, \ldots, ub_D]$ and $lb = [lb_1, lb_2, \ldots, lb_D]$. Then initialize the position of the salp x_j^i in a random manner, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, D$.

$$x_j^i = rand(N, D) * (ub(j) - lb(j)) + lb(j) \tag{1}$$

The second is to update the position of the leader. The leader is responsible for finding food and directing the actions of the entire team. Therefore, the leader's position update follows the following formula:

$$x_j^1 = \begin{cases} F_j + c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right), c_3 \geq 0.5 \\ F_j - c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right), c_3 < 0.5 \end{cases} \tag{2}$$

where x_j^1 represents the leader position, F_j is the food position, ub_j and lb_j are the bound. The control parameters include c_1, c_2 and c_3, among which c_2 and c_3 are random numbers within [0, 1], c_2 controls the step size and c_3 controls the direction. c_1 is the primary control parameter, which balances the exploration and development capabilities of the algorithm during iteration. In order to make the algorithm perform a global search in the first half of iteration and accurate development in the second half, the value of the c_1 follows the following formula:

$$c_1 = 2e^{-(4l/Max\_Iteration)^2} \tag{3}$$

where l is the current iteration and *Max_Iteration* is the maximum iteration.

Last update the position of the follower. The follower's position is only related to its initial position, motion speed and acceleration in the process of motion. The motion pattern conforms to Newton's law of motion. Therefore, the moving distance R of the follower can be expressed as:

$$R = \frac{1}{2}at^2 + v_0 t \tag{4}$$

Time t is the difference value of iteration times, so $t = 1$; v_0 is the follower initial speed which is 0; a is the acceleration of the follower of that iteration, and the calculation formula is $a = (v_{final} - v_0)/t$. Since the follower only follows the movement of the preceding salp close to itself, the movement speed $vfinal = \left(x_j^{i-1} - x_j^i\right)/t$, where it is known that $t = 1$ and $v_0 = 0$, therefore:

$$R = \frac{1}{2}\left(x_j^{i-1} - x_j^i\right) \tag{5}$$

Therefore, the update of follower position follows the following formula:

$$x_j^{i'} = x_j^i + R = \frac{1}{2}\left(x_j^i + x_j^{i-1}\right) \tag{6}$$

where, $x_j^{i'}$ is the position of the i-th follower in the j-th dimensional space before the update, and x_j^i is the position of the follower after the update. The steps of salp swarm algorithm are shown in Algorithm 1:

Algorithm 1 Salp Swarm Algorithm.

begin

Set algorithm parameters: The population size is N, the dimension of the problem is D, the maximum number of iterations is $Max\_Iteration$.

Randomly initialize the population according to Equation (1). The fitness value of each salp individual is calculated, and the optimal individual is selected as the food source location.

while ($t <= Max\_Iteration$) do

for $i = 1$ to N do

if ($i <= N/2$) do

Update the position of leader according to Equation (2).

else

Update the position of follower according to Equation (6).

end if

end for

Calculate the fitness value of individual population and the food source location is updated.

$l = l + 1$.

end while

end

3. The Improvement of Salp Swarm Algorithm

3.1. Population Initialization Based on Logistic Mapping

The core of the swarm intelligence algorithm is the continuous iteration of population, so the initialization of the population has a direct impact on the final solution and also affects the optimization ability. The more abundant and diverse the initialized population is, the more favorable it will be to find the global optimal solution of the population [25]. Without the help of prior knowledge, most swarm intelligence algorithms are basically random population initialization, which is greatly affects its performance.

The chaotic sequence has the characteristics of ergodicity and randomness, and the population initialization by chaotic sequence can have better diversity. The chaotic sequences commonly used at present are iterative mapping, tent mapping, logistic mapping, etc. Through comparative study, logistic mapping is used to perform population initialization in this paper.

The logistic mapping mathematical formula [26] is:

$$y_{j+1}^{i} = p y_{j}^{i}\left(1 - y_{j}^{i}\right) \tag{7}$$

where p is an adjustable parameter, usually set to 4. $i = 1, 2, \ldots, N$ represents the population size, $j = 1, 2, \ldots, D$ represents the ordinal number of chaotic variables. After logistic mapping, the initialization formula of the population becomes:

$$x_{j}^{i} = y_{j}^{i} * (ub(j) - lb(j)) + lb(j) \tag{8}$$

3.2. Symmetric Adaptive Population Division

In the basic SSA, the number of follower and leader is half of the population of salps, which causes the algorithm to search asymmetry: in the early iteration, the number of leaders is small and the ratio is low, which leads to insufficient global search and easy to fall into local extremum. However, in the late iteration, the number of followers is small, which leads to insufficient local search and low optimization accuracy. In response to this problem, literature [18] proposed a leader-follower adaptive adjustment strategy. This paper proposes a symmetric adaptive division population according to this strategy, which adjusted the number of leaders of the salp to have an adaptive decreasing trend

as the number of iterations increases, while the number of followers shows the adaptive increasing trend. This will make the algorithm focus more on global breadth exploration in the early stage, and more in-depth mining near the optimal value in the later stage, thus improving the optimization accuracy. The improved symmetric adaptive population division calculation formula is as follows:

Introduce the control factor ω:

$$\omega = b \cdot (-k \cdot rand() + tan\left(\frac{\pi}{4} - \frac{\pi l}{4 \cdot Max\_Iteration}\right)) \tag{9}$$

where l is the current iteration number and *Max_Iteration* is the maximum iteration number, and b is the proportion coefficient, which is used to avoid the imbalance of proportion. k is the disturbance deviation factor, and the decreasing ω value is disturbed in combination with the rand function.

The modified number of leaders per iteration is $\omega \cdot N$, and the number of followers is equal to $1 - \omega \cdot N$.

3.3. Simulated Annealing Mechanism Based on Symmetric Perturbation

The simulated annealing algorithm was first proposed by Metropolis and Kirkpatrick [27]. The simulated annealing algorithm originates from the principle of solid annealing [28].

The core of simulated annealing algorithm is to generate a new solution based on the current solution in some way, and accept the new solution with a certain probability, so as to enhance the local jumping out ability of the algorithm, and keep the algorithm still has a strong diversity in the later iteration.

The generation of new solutions is particularly important in simulated annealing. Based on the simulated annealing algorithm, this paper introduces symmetric perturbation to generate new solutions. Symmetric perturbation refers to the mapping of the position of the new solution to the current optimal position in the symmetrical interval. The symmetrical interval is determined by the product of the current temperature and the random number mapped to the dimensional space.

The flow of simulated annealing mechanism based on symmetric perturbation is as follows:

(1) Initialization: set the initial temperature T, initial solution S and the maximum number of iterations *Max_Iteration*.

(2) for l = 1, 2, ..., *Max_Iteration* steps (3) to (6).

(3) Perturb the current solution S to obtain the new solution S'. The formula is as follows:

$$S' = T \times rnd(1, d) / normrnd(1, d) + S \tag{10}$$

(4) Calculate incremental $df = f(S') - f(S)$, where df is the evaluation function.

(5) According to the Metropolis criterion, the sampling formula is as follows:

$$P = \begin{cases} 1, df < 0; \\ e^{-\frac{df}{T}}, df \geq 0. \end{cases} \tag{11}$$

If $df < 0$, accept the new solution; otherwise, accept the new solution with probability $e^{-\frac{df}{T}}$.

(6) If the termination condition is satisfied, the current solution is the optimal solution and output, then stop the algorithm; otherwise, go back to step (2) after reducing the temperature. The termination condition is usually a continuous number of new solutions that have not been accepted or have reached the termination temperature.

3.4. Improved Salp Swarm Algorithm

As mentioned above, the salp swarm algorithm has issues related to slow convergence speed and low optimization accuracy. SASSA introduced a logistic chaotic map to initialize

the population, which enriched the diversity of the population. The symmetric adaptive population division strategy is introduced to balance the development and exploration ability of the algorithm. Finally, the simulated annealing mechanism based on symmetric perturbation is introduced to accept the inferior solution with a certain probability, and the hybrid operation in the genetic algorithm is used. Hybridization means that the new solution and the old solution produced by simulated annealing are hybridized in proportion to obtain the final new solution. The new solution not only retains the advantages of the old solution but also reduces the influence of perturbation error. The hybridization formula is as follows:

$$S' = \left(1 - \sqrt{c}\right) \times S + \sqrt{c} \times S' \tag{12}$$

where c is a random number between 0 and 1.

The flow chart of SASSA algorithm is shown in Figure 2:

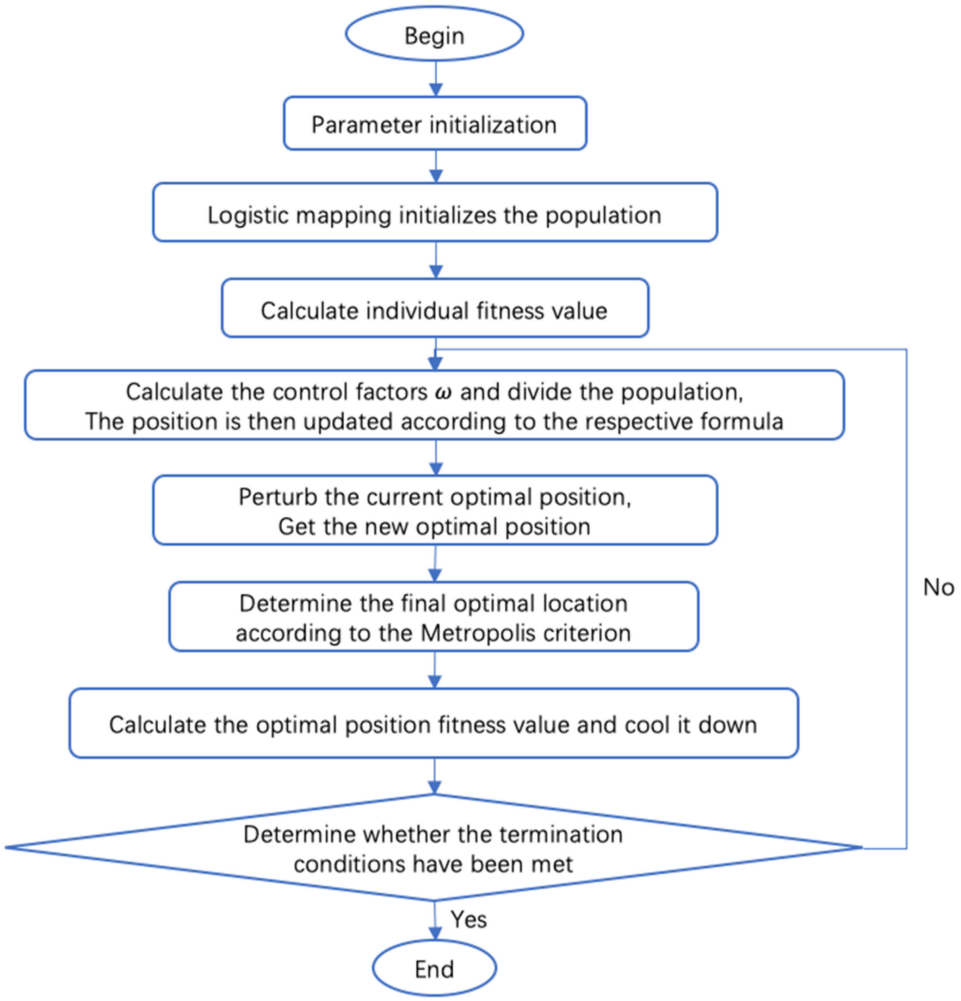

Figure 2. Flow chart of improved salp swarm algorithm based on the simulated annealing (SASSA).

The steps of SASSA are shown in Algorithm 2:

Algorithm 2 SASSA.

begin

Set algorithm parameters: the population size is N, the dimension of the problem is D, the maximum number of iterations is *Max_Iteration*, the initial temperature is T, and the cooling rate is Q.

According to Equation (8), Logistic chaotic map is used to initialize the population. The fitness value of each salp individual is calculated, and the optimal individual is selected as the food source location.

while (t < = *Max_Iteration*) do
ω is calculated by formula (9)
for i = 1 to N do
if (i< = $\omega \cdot N$) do
Update the position of leader according to Equation (2).
else
Update the position of follower according to Equation (6).
end if
end for
Disturbing the current optimal salp's position S
S' = Mutate (S)
Calculate increment df = $f(S')$ − $f(S)$
if ($df < 0$) do
$S \leftarrow S'$
else
$P = exp(-df/T)$
if (*rand* < = P)
S' = Crossover (S, S')
$S \leftarrow S'$
$T = T^*q$
end if
end if
Calculated the fitness value of individual population and the food source location is updated.
$l = l + 1$.
end while
end

3.5. Complexity Analysis

According to Algorithm 2, in each iteration, the population initialization, leader position update, follower position update and food source position update of SASSA algorithm are all serial. The population, dimension and iteration number are N, D and M respectively, then the time complexity of SASSA algorithm is as follows:

(1) Leader position initialization, follower position initialization and salp position correction based on the upper and lower bounds were performed with a complexity of $O(N^*D)$;
(2) During the leader position Update, the number of leaders is $\omega \cdot N$, so the complexity is $O(\omega^*N^*D)$;
(3) During the follower position update, the number of followers is $(1 - \omega) \cdot N$, thus the complexity is $O((1 - \omega)^*N^*D)$;
(4) In the simulated annealing stage, the time complexity is $O(k^*N^*D)$, where k is the number of times that the algorithm perturbed the solution in the simulated annealing mechanism.

The time complexity of SASSA algorithm is $O(N*D) + O(\omega*N*D) + O((1-\omega)*N*D) + O(k*N*D) = O(C*N*D)$. The total time complexity is $O(C*N*D*M)$, and C is constant.

Similarly, the time complexity of the basic SSA is the same. Therefore, the algorithm proposed in this paper is equivalent to the original algorithm in time complexity, and the execution efficiency does not decrease.

4. Benchmark Function Experiments

In this section, we will test the algorithm's performance through 21 benchmark functions and compare the results with other algorithms.

4.1. Benchmark Function

We used the reference function selected from the literature [29,30] to test the performance of the algorithm. The function equation is shown in Tables 1–3, where *Dim* represents the dimension of the function, *Range* is the upper and lower bound and *fmin* represents the optimal value. In general, we use these test functions to minimize. These functions can be divided into unimodal benchmark functions, multimodal benchmark functions and fixed-dimension multimodal benchmark functions. Unimodal function can evaluate the ability of algorithm development. Multimodal benchmark functions can test the exploration ability of the algorithm and the ability to jump out of the local optimum. Fixed-dimension multimodal benchmark functions can evaluate the comprehensive ability of the algorithm. Therefore, if we select these to test the algorithm, we can satisfy different types of problems and comprehensively evaluate the performance of the optimized algorithm.

Table 1. Unimodal benchmark functions.

Function	Dim	Range	f_{min}
$F1(x) = \sum_{i=1}^{n} x_i^2$	N	$[-100, 100]$	0
$F2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	N	$[-10, 10]$	0
$F3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	N	$[-100, 100]$	0
$F4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	N	$[-100, 100]$	0
$F5(x) = \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	N	$[-30, 30]$	0
$F6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	N	$[-100, 100]$	0
$F7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1]$	N	$[-128, 128]$	0

4.2. Experimental Settings

All the tests were carried out under the same conditions. The population size was 30, and the maximum number of iterations was set to 500. Each benchmark function was run independently 30 times to mitigate the effect of randomness on the test results. The experiment was conducted on an Intel I7 processor and a computer with 16G memory. The system was macOS Catalina, and the test software was MATLAB R2020a. Based on the mean and standard deviation (Std) of fitness, values without loss of generality were used to evaluate performance.

4.3. Results Analysis

In this section, the test results are displayed in tables and images in an intuitive manner. The improved algorithm is compared with the SSA and several recently successful meta-heuristic algorithms, namely the moth flame optimization (MFO) [31], GWO and WOA. As can be seen from Table 4, in the unimodal benchmark function $f1$–$f7$, except for $f5$, SASSA achieved good results. Among them, in $f1$, $f2$, $f3$ and $f4$, SASSA has obvious advantages in mean value, Std and lowest value. In the $f7$ function, the results of GWO are

very close to the improved algorithm but inferior to the improved algorithm in terms of lowest value. In terms of $f6$ function, the performance of the improved algorithm is only moderate. As for the $f5$ function, the result of the improved algorithm is worse than that of the GWO and the WOA but it obtains the best result in terms of the lowest value.

Table 2. Multimodal benchmark functions.

Function	Dim	Range	f_{min}
$F8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	N	$[-500, 500]$	-418.9829×5
$F9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	N	$[-5.12, 5.12]$	0
$F10(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	N	$[-32, 32]$	0
$F11(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	N	$[-600, 600]$	0
$F12(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 [1 + 10\sin^2(\pi y_i + 1)] + (y_n - 1)^2\} + \sum_{i=1}^{n}\mu(x_i, 10,100,4)$ $y_l = 1 + \frac{x_i+1}{4}$ $\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	N	$[-50, 50]$	0
$F13(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)\} + \sum_{i=1}^{n}\mu(x_i, 5,100,4)$	N	$[-50, 50]$	0

Table 3. Fixed-dimension multimodal benchmark functions.

Function	Dim	Range	f_{min}
$F14(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	$[-65, 65]$	1
$F15(x) = 4x_1^2 - 2.1x_1^2 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	-1.0316
$F16(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]$	3
$F17(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	3	$[1, 3]$	-3.86
$F18(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	6	$[0, 1]$	-3.32
$F19(x) = -\sum_{i=1}^{5}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.1532
$F20(x) = -\sum_{i=1}^{7}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	10.4028
$F21(x) = -\sum_{i=1}^{10}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.5363

In terms of multimodal benchmark functions, it can be seen from Table 5 that the SASSA has better results in $f8$, $f9$, $f10$ and $f11$. In $f9$ and $f11$, the mean value and Std can be directly set to 0, and the Std in $f10$ can also be set to 0, which indicates that the algorithm has relatively strong stability. However, in $f12$ and $f13$, the performance of the SASSA is poor.

As for the fixed-dimension multimodal benchmark functions, it can be seen from Table 6 that the SASSA has obvious advantages in $f18$, $f19$, $f20$ and $f21$ functions and

achieves good results in terms of mean value and lowest value. In $f14$ and $f17$, the performance of the SASSA is poor, but in $f15$ and $f16$, the Std of the SASSA is better than that of other algorithms in the case that the mean value of all algorithms can obtain the lowest value, which also provides the data basis for its better stability.

Table 4. Results of unimodal benchmark functions.

Function	Index	SASSA	SSA	MFO	GWO	WOA
$f1$	Mean	1.63×10^{-127}	1.70×10^{-7}	1.71×10^{3}	1.25×10^{-27}	7.63×10^{-73}
	Std	6.23×10^{-127}	2.82×10^{-7}	3.77×10^{3}	1.77×10^{-27}	2.64×10^{-72}
	Lowest	1.36×10^{-142}	2.64×10^{-8}	0.8602	3.91×10^{-29}	5.76×10^{-84}
$f2$	Mean	2.54×10^{-64}	0.0280	1.3333	3.42×10^{-33}	1.49×10^{-51}
	Std	1.01×10^{-63}	0.1176	3.4575	4.02×10^{-33}	7.71×10^{-51}
	Lowest	8.30×10^{-70}	5.99×10^{-6}	5.13×10^{-10}	5.63×10^{-35}	1.76×10^{-60}
$f3$	Mean	7.25×10^{-126}	1.46×10^{3}	2.30×10^{4}	1.14×10^{-5}	4.57×10^{4}
	Std	3.56×10^{-125}	676.0907	1.33×10^{4}	3.34×10^{-5}	1.55×10^{4}
	Lowest	2.66×10^{-139}	291.4835	2.85×10^{3}	2.20×10^{-9}	2.51×10^{4}
$f4$	Mean	5.26×10^{-66}	2.12×10^{-5}	2.9135	2.72×10^{-18}	4.2870
	Std	2.40×10^{-65}	7.82×10^{-6}	5.1788	4.23×10^{-18}	8.9829
	Lowest	1.69×10^{-72}	1.10×10^{-5}	0.0034	2.58×10^{-20}	2.59×10^{-4}
$f5$	Mean	8.2487	348.0704	6.47×10^{3}	6.3292	6.9341
	Std	1.7800	623.0314	2.27×10^{4}	0.8115	0.4261
	Lowest	2.57×10^{-4}	0.2785	0.0147	3.7198	6.1502
$f6$	Mean	2.32×10^{-7}	8.26×10^{-10}	3.82×10^{-13}	0.0084	0.0013
	Std	1.50×10^{-7}	2.87×10^{-10}	9.08×10^{-13}	0.0461	0.0014
	Lowest	8.03×10^{-8}	3.56×10^{-10}	4.29×10^{-15}	1.28×10^{-6}	2.24×10^{-4}
$f7$	Mean	1.03×10^{-4}	0.0120	0.0084	7.93×10^{-4}	0.0026
	Std	9.58×10^{-5}	0.0077	0.0074	5.73×10^{-4}	0.0030
	Lowest	2.97×10^{-7}	0.0015	0.0019	1.02×10^{-4}	1.21×10^{-4}

Table 5. Results of multimodal benchmark functions.

Function	Index	SASSA	SSA	MFO	GWO	WOA
$f8$	Mean	-5.74×10^{4}	-2.64×10^{3}	-3.28×10^{3}	-2.65×10^{3}	-3.21×10^{3}
	Std	1.57×10^{4}	312.5324	313.8152	396.5429	540.8742
	Lowest	-9.54×10^{4}	-3.54×10^{3}	-3.73×10^{3}	-3.61×10^{3}	-4.19×10^{3}
$f9$	Mean	0	18.4730	20.8680	1.0448	1.1907
	Std	0	7.4073	12.7232	2.2098	6.5219
	Lowest	0	6.9647	7.9597	0	0
$f10$	Mean	8.88×10^{-16}	0.5367	0.0938	7.40×10^{-15}	4.09×10^{-15}
	Std	0	0.8212	0.5137	1.64×10^{-15}	2.35×10^{-15}
	Lowest	8.88×10^{-16}	5.06×10^{-6}	5.74×10^{-8}	4.44×10^{-15}	8.78×10^{-16}
$f11$	Mean	0	0.1993	0.1185	0.0163	0.0906
	Std	0	0.1202	0.0526	0.0172	0.1492
	Lowest	0	0.0443	0.0295	0	0
$f12$	Mean	0.0136	0.6016	0.0622	0.0052	0.0108
	Std	0.0171	0.8066	0.1713	0.0115	0.0195
	Lowest	5.44×10^{-4}	1.93×10^{-11}	9.29×10^{-16}	3.43×10^{-7}	2.17×10^{-4}
$f13$	Mean	0.0087	0.0029	0.0026	0.0169	0.0420
	Std	0.0104	0.0049	0.0047	0.0383	0.0517
	Lowest	1.18×10^{-4}	3.39×10^{-11}	2.52×10^{-16}	2.72×10^{-6}	0.0027

The Friedman test [32,33] were obtained by using the mean value of each algorithm on all 21 test functions in Tables 4–6, and are shown in Table 7. Table 8 shows the related statistical values of the Friedman test. If the chi-square statistic was greater than the critical value, the null hypothesis was rejected. p represents the probability of the null hypothesis obtaining. The null hypothesis here was that there is no significant difference in performance among the five algorithms considered here.

Table 6. Results of fix-dimension multimodal benchmark functions.

Function	Index	SASSA	SSA	MFO	GWO	WOA
$f14$	Mean	1.9213	1.2298	2.6093	5.6902	2.4068
	Std	1.5345	0.5005	2.2834	4.6976	2.5646
	Lowest	1	1	1	1	1
$f15$	Mean	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
	Std	3.48×10^{-12}	4.15×10^{-11}	6.78×10^{-11}	3.17×10^{-8}	3.80×10^{-10}
	Lowest	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
$f16$	Mean	3.0000	3.0000	3.0000	3.0000	3.0001
	Std	3.06×10^{-17}	2.74×10^{-13}	2.23×10^{-15}	2.96×10^{-5}	2.28×10^{-4}
	Lowest	3.0000	3.0000	3.0000	3.0000	3.0000
$f17$	Mean	−3.8616	−3.8628	−3.8625	−3.8609	−3.8550
	Std	0.0446	1.14×10^{-10}	0.0014	0.0030	0.0112
	Lowest	−3.8628	−3.8628	−3.8628	−3.8628	−3.8627
$f18$	Mean	−3.2760	−3.2213	−3.2197	−3.2662	−3.2238
	Std	0.0862	0.0629	0.0557	0.0660	0.1044
	Lowest	−3.3220	−3.3220	−3.3220	−3.3220	−3.3219
$f19$	Mean	−10.1530	−6.8070	−7.8876	−9.2651	−8.7734
	Std	4.45×10^{-9}	3.3092	3.1199	2.3302	2.2810
	Lowest	−10.1532	−10.1532	−10.1532	−10.1528	−10.1532
$f20$	Mean	−10.4027	−8.3941	−7.6667	−9.9292	−7.6605
	Std	5.08×10^{-9}	3.2041	3.6731	1.8008	3.2626
	Lowest	−10.4028	−10.4028	−10.4028	−10.4027	−10.4011
$f21$	Mean	−10.3561	−8.8569	−7.8695	−10.0856	−6.8170
	Std	0.9873	2.8998	3.5977	1.7470	3.0241
	Lowest	−10.5363	−10.5363	−10.5363	−10.5361	−10.5318

Table 7. The Friedman ranks (benchmark functions).

Rank	Name	F-Rank
0	SASSA	1.69
1	GWO	2.69
2	WOA	3.48
3	SSA	3.5
4	MFO	3.64

Table 8. Related statistical values (benchmark functions).

Chi-Sq′	Prob > Chi-Sq′ (p)	Critical Value
24.43076923	6.55×10^{-5}	9.49

According to the Friedman ranking in Table 7, SASSA can get better rank than the original algorithms and other compared algorithm. Table 8 shows that the null hypothesis was rejected, and thus the Friedman ranking was correct. On the whole, the SASSA obtained better results in contrast to the SSA as well as the other compared algorithm.

In order to better verify the algorithm's capability, we extracted some convergent images from 21 test functions, as shown in Figure 3. According to the convergence curve in the figure, we can observe that SASSA has a better convergence speed in realizing functions F3, F4, F7, F9 and F11, and other algorithms fall into local optima too early. In terms of F5 function, although the improved algorithm does not get the best result, its initial convergence speed is the fastest among all algorithms. As for F1 and F2, although SASSA cannot match the GWO in the convergence speed at the beginning, its convergence speed is greatly improved in the later stage, and better results can be explored.

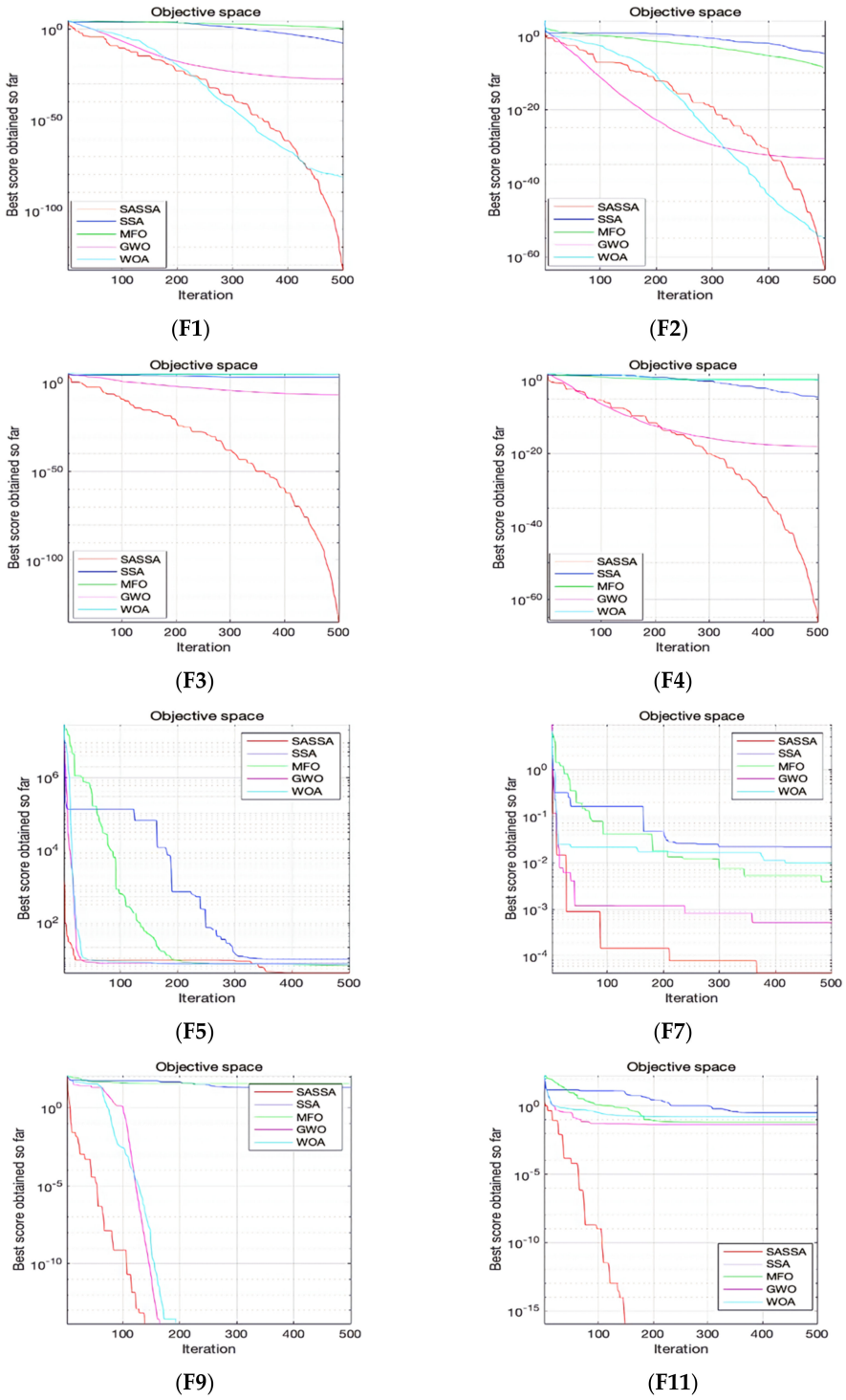

Figure 3. Convergence curve of benchmark functions.

In general, the improved algorithm SASSA can achieve good results in three kinds of test functions. Although some test functions do not yield better solutions, they also show convergence and stability. Therefore, it is of great significance to improve the classical salp swarm algorithm.

5. Applications in Solving Engineering Optimization Problems

5.1. Problem Description

The engineering optimization problem is a kind of constrained optimization problem. The constrained optimization problem is a very common planning problem in the science and engineering fields. The corresponding description of the constrained optimization problem is as follows [34]:

$$minf(x)s.t.\ g_i(x) \le 0, i = 1, 2, \ldots, m, h_i(x) = 0, i = m+1, m+2, \ldots, nl_i \le x_i \le u_i, i = 1, 2, \ldots, n \quad (13)$$

Among them, the objective functions $f(x)$, g_1,g_2, ... ,g_m and h_{m+1},h_{m+2}, ... , h_n are real valued functions in the domain, $g_i(x) \le 0, (i = 1, 2, \ldots, m)$ means inequality constraint, $h_i(x) = 0, (i = m+1, m+2, \ldots, n)$ represents equality constraints. The decision variables are x, $x = (x_1, x_2, \ldots, x_n) \in R^n$.

The core of the constrained optimization problem is to find a feasible solution in the feasible region. If $f(x^*) \le f(x)$ holds for each feasible solution x, then x^* is the optimal solution of the constrained optimization problem under the given constraints. If the function of the optimization problem is linear, the optimization problem is a linear constrained optimization problem, otherwise it is a nonlinear constrained optimization problem. The engineering optimization problems used in this paper are all nonlinear single-objective constrained optimization problems.

5.2. Constraint Handling

The processing of constraint conditions is the key to solving constraint optimization problems. In function problems, these constraints determine the value range of decision variables. In actual engineering optimization problems, these constraints are the various objective factors that must be met to solve the target problem. In order to deal with these constrained optimization problems, commonly used methods include rejection method, repair method, penalty function method, etc. Continuous-time solvers is also an effective method to deal with optimization problems with nonlinear constraints. In this method, a virtual dynamical system along with the main system evolves and estimates the optimal solution of the problem [35,36]. When dealing with engineering optimization problems with constraints, the most common and simplest method is the penalty function method [37]. Its idea is to add a "penalty term" to the objective function of the problem and define the constraint as a whole, it satisfies the constraints without affecting the solution, and the problem with constraints is thus transformed into an unconstrained optimization problem. The formula of the penalty function is as follows:

$$F(x) = f(x) + k_1 \times \sum_{i=1}^{m} g_i(x) \times b_1 + k_2 \times \sum_{i=m+1}^{n} h_i(x) \times b_2 \quad (14)$$

The objective function is $f(x)$, the inequality penalty coefficient is k_1, the equality penalty coefficient is k_2, $g_i(x)$ is the inequality constraint, $h_i(x)$ is the equality constraint, b_1 and b_2 are defined as follows:

$$b_1 = \begin{cases} 0, if\ g_i(x) \le 0 \\ 1,\ else \end{cases} \quad b_2 = \begin{cases} 0, if\ h_i(x) = 0 \\ 1,\ else \end{cases} \quad (15)$$

A penalty term is added to the objective function of the constrained optimization problem, and the constrained optimization problem is transformed into a general optimization problem to be solved.

5.3. Experimental Settings

In order to verify the feasibility of the SASSA, four classic engineering optimization problems were selected for simulation to verify the performance of the algorithm in solving constraint optimization problems. In this paper, weight minimization of a speed reducer, gear train design problem, optimal operation of alkylation unit and welded beam design were selected as research objects [38]. Among them, weight minimization of a speed reducer, gear train design problem and welded beam design are problems in the field of physical engineering design. Weight minimization of a speed reducer is to minimize the weight, gear train design problem is to make the design more in line with requirements, and welded beam design is to minimize the production cost. And optimal operation of alkylation unit is a problem in the field of chemical engineering, aiming to make production more efficient. The four questions selected cover the two fields of physics and chemistry, and involve weight, cost, efficiency, etc., which can provide a good evaluation of the performance of the improved algorithm. In order to objectively evaluate the performance of SASSA, we selected SSA, GWO [39], DE [40], BBO [41], ACO and PSO for a comparison experiment. The SSA is the basic salp swarm algorithm. By comparing with it, we can comprehensively evaluate the optimization ability, the degree of improvement and the field of adaptation of the improved algorithm, so as to provide a better theoretical basis for the improvement strategy. The GWO originated from the predation of gray wolves, it is a highly developed algorithm. The DE is a convenient and easy algorithm, and its effectiveness has long been proven. The BBO is an evolutionary algorithm, literature proves that the BBO algorithm is an excellent algorithm for solving engineering problems, so comparing it with it can improve the credibility of the improved algorithm. The ACO is a probabilistic algorithm and has a wide range of applications, comparing it with the improved algorithm can be more beneficial to evaluate the improvement ability of the improved algorithm.

The environment used in this experiment is: the operating system is MacOS Catalina, the processor is I7, the memory is 16G, and the software is MATLAB R2020A. The population size is 30, and the maximum number of iterations is 1000. The experiment was repeated 50 times to reduce the influence of randomness on the test results. Without loss of generality, the performance was evaluated according to the Mean and standard deviation of fitness values.

5.4. Results

5.4.1. Weight Minimization of a Speed Reducer

Weight minimization of a speed reducer is a typical engineering optimization problem, and its optimization purpose is to minimize the total weight of the reducer [42]. The design of the reducer is subject to some constraints, including the surface stress of the gear teeth, bending stress, shaft deflection and shaft stress. Reflected in the design drawings are gear width (b), gear modulus (m), the number of teeth on the pinion (z), the length of the first shaft between the bearings (l_1), and the length of the second shaft between the bearings (l_2)), the diameter of the first shaft d_1, and the diameter of the second shaft d_2, as shown in Figure 4:

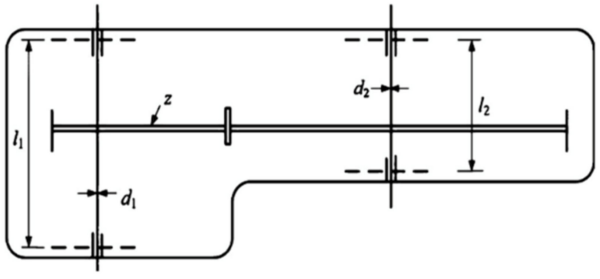

Figure 4. Weight Minimization of a Speed Reducer.

Use x_1–x_7 to represent the above seven variables, and the mathematical description of the problem of weight minimization of a speed reducer is as follows:

Minimize:

$$f(x) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right)$$

Subject to:

$$\begin{aligned}
g_1(x) &= -x_1x_2^2x_3 + 27 \le 0,\\
g_2(x) &= -x_1x_2^2x_3^2 + 397.5 \le 0,\\
g_3(x) &= -x_2x_6^4x_3x_4^{-3} + 1.93 \le 0,\\
g_4(x) &= -x_2x_7^4x_3x_5^{-3} + 1.93 \le 0,\\
g_5(x) &= 10x_6^{-3}\sqrt{16.91 \times 10^6 + \left(745x_4x_2^{-1}x_3^{-1}\right)^2} - 1100 \le 0,\\
g_6(x) &= 10x_7^{-3}\sqrt{157.5 \times 10^6 + \left(745x_5x_2^{-1}x_3^{-1}\right)^2} - 850 \le 0,\\
g_7(x) &= x_2x_3 - 40 \le 0,\\
g_8(x) &= -x_1x_2^{-1} + 5 \le 0,\\
g_9(x) &= x_1x_2^{-1} - 12 \le 0,\\
g_{10}(x) &= 1.5x_6 - x_4 + 1.9 \le 0,\\
g_{11}(x) &= 1.1x_7 - x_5 + 1.9 \le 0,
\end{aligned}$$

With bounds:

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

The experimental results are shown in Table 9. SASSA has the best performance on the best value and mean value, indicating that after its optimization, the weight of the reducer is the least, but the std is slightly inferior to GWO, indicating that its stability needs to be improved. Figure 5 shows that the iteration curves of SASSA, SSA and GWO are relatively close. Combining with the data, it can be seen that the convergence accuracy of SASSA is the best. In terms of convergence speed, it can be seen directly that SASSA is in the leading position. Therefore, the convergence speed and performance of SASSA are better than other comparison algorithms.

Table 9. Weight Minimization of a Speed Reducer.

	Best	Mean	Worst	Std
SASSA	2.9949×10^3	3.0025×10^3	3.1017×10^3	18.0827
SSA	2.9975×10^3	3.0190×10^3	3.0759×10^3	22.0886
GWO	3.0091×10^3	3.0519×10^3	3.0162×10^3	5.1551
DE	3.6746×10^5	3.6746×10^5	3.6746×10^5	1.1944×10^{-10}
BBO	3.6746×10^5	3.6746×10^5	3.6746×10^5	9.6747×10^{-7}
ACO	6.9015×10^5	6.9015×10^5	6.9015×10^5	0
PSO	3.6746×10^5	3.6746×10^5	3.6746×10^5	1.2312×10^{-10}

The convergence curve of weight minimization of a speed reducer is shown in Figure 5:

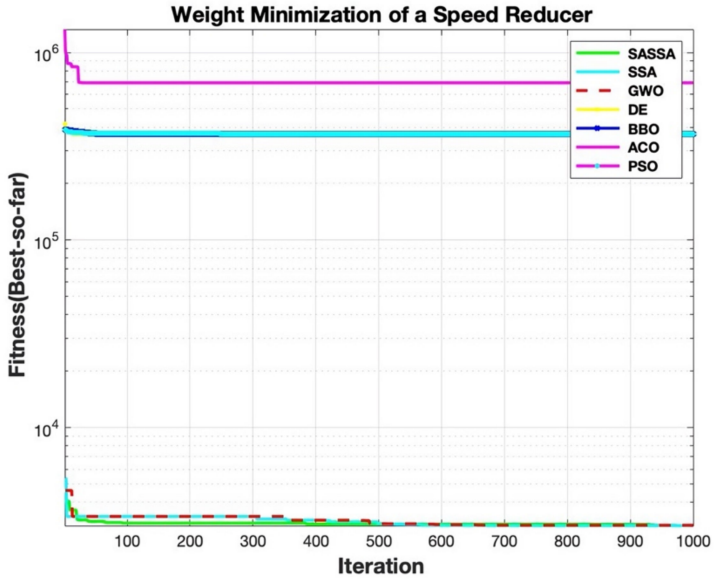

Figure 5. Convergence Curve of Weight Minimization of a Speed Reducer.

5.4.2. Gear Train Design Problem

Gear design problem is also a popular engineering optimization problem. Figure 6 shows the gear train design problem model [43]. When designing compound gears, the gear ratio between the drive shaft and the driven shaft should be considered. The gear ratio is defined as the ratio of the angular velocity of the output shaft to the angular velocity of the input shaft. Our goal is to make the gear ratio as close as possible to 1/6.931. For each gear, the number of gears is between 12 and 60. The variables *Ta*, *Tb*, *Td* and *Tf* are the number of teeth of gears A, B, D, and F, and the number of teeth must be an integer.

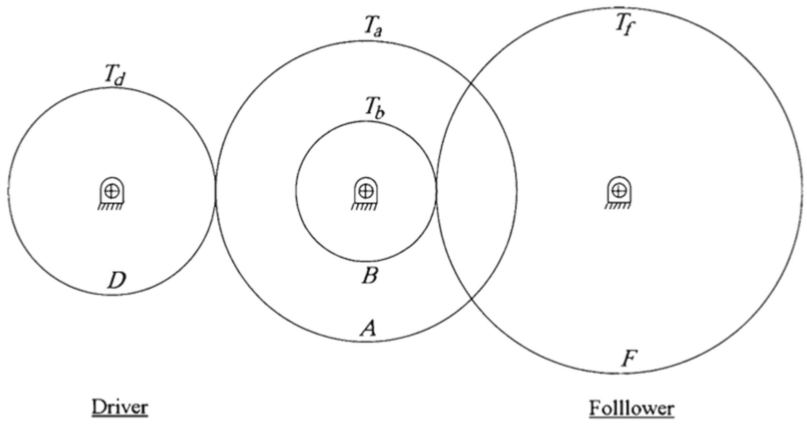

Figure 6. Gear train design Problem.

Use x_1–x_4 to represent the above four variables, and the mathematical description of the problem is as follows:

Minimize:

$$f(x) = \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}\right)^2$$

Subject to:

$$12 \leq x_i \leq 60, i = 1, 2, 3, 4$$

Table 10 shows the experimental results of the gear train design problem. As can be seen from the table, both SASSA and PSO can get 1/6.931, but the mean value and variance of SASSA are the smallest, indicating that its overall performance and stability are better.

Figure 7 is the convergence curve. It can be seen that compared with other algorithms, SASSA has significant advantages in terms of convergence accuracy and speed.

Table 10. Gear train design Problem.

	Best	Mean	Worst	Std
SASSA	0	2.5461×10^{-32}	2.7810×10^{-31}	7.7458×10^{-32}
SSA	1.4130×10^{-23}	1.3639×10^{-20}	7.1462×10^{-20}	2.2508×10^{-20}
GWO	1.1932×10^{-17}	6.3166×10^{-13}	2.3137×10^{-12}	6.8706×10^{-13}
DE	1.9891×10^{-14}	1.9877×10^{-11}	2.1586×10^{-10}	4.8930×10^{-11}
BBO	8.0696×10^{-22}	3.8308×10^{-18}	1.8866×10^{-17}	5.9474×10^{-18}
ACO	5.0119×10^{-4}	5.0119×10^{-4}	5.0119×10^{-4}	2.2247×10^{-19}
PSO	0	1.8183×10^{-24}	1.9587×10^{-23}	5.0490×10^{-24}

The convergence curve of gear train design problem is shown in Figure 7:

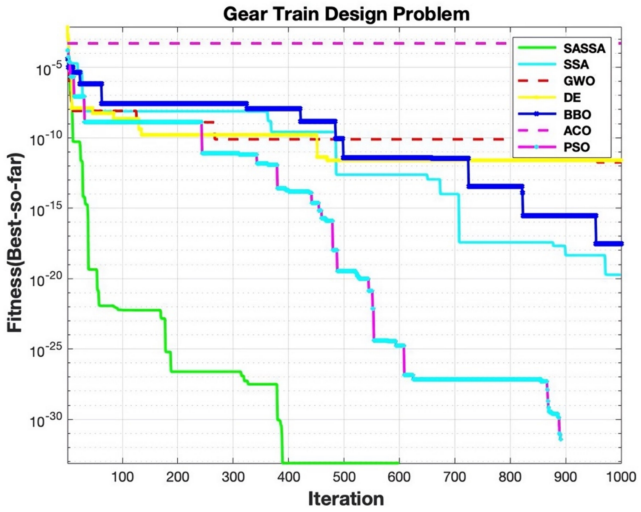

Figure 7. Convergence Curve of Gear train design Problem.

5.4.3. Optimal Operation of Alkylation Unit

The optimal operation of alkylation unit is a very common in the petroleum industry. Figure 8 shows a simplified alkylation process flow [44]. As shown in the figure, the olefin feedstock (100% butane), pure isobutane recycle and 100% isobutene supplement are introduced into the reactor together with the acid catalyst, and then the reactor product is passed through a fractionator, where isobutene and alkane are separated. The base product, spent acid is also removed from the reactor.

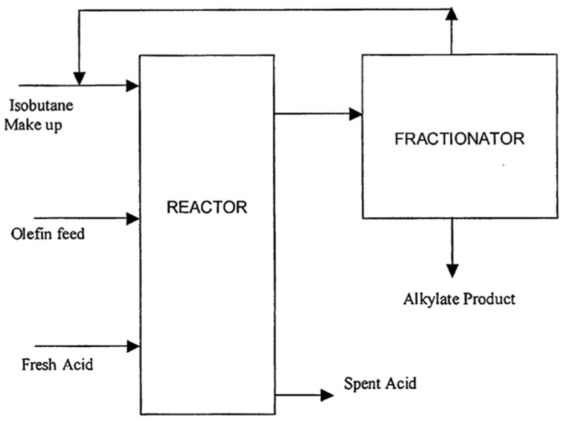

Figure 8. Optimal Operation of Alkylation Unit.

The main purpose of this problem is to increase the octane number of the olefin feedstock under acidic conditions, and the objective function is defined as the alkylation product. Literature [45] transformed this problem into a constrained optimization problem with 7 variables and 14 constraints. The mathematical description is as follows:

Maximize:

$$f(x) = 0.035x_1x_6 + 1.715x_1 + 10.0x_2 + 4.0565x_3 - 0.063x_3x_5$$

Subject to:

$$\begin{aligned}
g_1(x) &= 0.0059553571x_6^2x_1 + 0.88392857x_3 - 0.1175625x_6x_1 - x_1 \leq 0,\\
g_2(x) &= 1.1088x_1 + 0.1303533x_1x_6 - 0.0066033x_1x_6^2 - x_3 \leq 0,\\
g_3(x) &= 6.66173269x_6^2 - 56.596669x_4 + 172.39878x_5 - 10000 - 191.20592x_6 \leq 0,\\
g_4(x) &= 1.08702x_6 - 0.03762x_6^2 + 0.32175x_4 + 56.85075 - x_5 \leq 0,\\
g_5(x) &= 0.006198x_7x_4x_3 + 2462.3121x_2 - 25.125634x_2x_4 - x_3x_4 \leq 0,\\
g_6(x) &= 161.18996x_3x_4 + 5000.0x_2x_4 - 489510.0x_2 - x_3x_4x_7 \leq 0,\\
g_7(x) &= 0.33x_7 + 44.333333 - x_5 \leq 0,\\
g_8(x) &= 0.022556x_5 - 1.0 - 0.007595x_7 \leq 0,\\
g_9(x) &= 0.00061x_3 - 1.0 - 0.0005x_1 \leq 0,\\
g_{10}(x) &= 0.819672x_1 - x_3 + 0.819672 \leq 0,\\
g_{11}(x) &= 24500.0x_2 - 250.0x_2x_4 - x_3x_4 \leq 0,\\
g_{12}(x) &= 1020.4082x_4x_2 + 1.2244898x_3x_4 - 100000x_2 \leq 0,\\
g_{13}(x) &= 6.25x_1x_6 + 6.25x_1 - 7.625x_3 - 100000 \leq 0,\\
g_{14}(x) &= 1.22x_3 - x_6x_1 - x_1 + 1.0 \leq 0,
\end{aligned}$$

With bounds:

$$\begin{aligned}
&1000 \leq x_1 \leq 2000,\\
&0 \leq x_2 \leq 100,\\
&2000 \leq x_3 \leq 4000,\\
&0 \leq x_4 \leq 100,\\
&0 \leq x_5 \leq 100,\\
&0 \leq x_6 \leq 20,\\
&0 \leq x_7 \leq 200.
\end{aligned}$$

We first converted the maximization problem into the minimization problem to solve it. Table 11 shows that SASSA performs best in all standards, indicating that it can maximize the alkylation product value and has a good effect on the optimization of the alkylation process. Figure 9 is the convergence curve of the optimal operation of alkylation unit. It can be seen from the figure that the convergence performance of the improved algorithm is not optimal at the beginning. But at the later stage of the iteration, after a long period of local stagnation, the improved algorithm still has the ability to get rid of the current local best points and continue to explore so that the overall optimization performance is further enhanced. This shows that the improvement strategy mentioned in the previous article for the basic algorithm that tends to fall into a partial stagnation in the later iteration of the iteration has played a role.

Table 11. Optimal Operation of Alkylation Unit.

	Best	**Mean**	**Worst**	**Std**
SASSA	−452.8468	1.5985×10^3	7.1265×10^3	3.2772×10^3
SSA	−443.2917	3.3583×10^4	7.4636×10^5	1.6777×10^5
GWO	−431.5267	5.4967×10^6	5.2486×10^7	1.6104×10^7
DE	−423.8519	7.7895×10^3	1.6719×10^4	6.7751×10^3
BBO	−439.6189	1.4217×10^5	2.8428×10^6	6.3567×10^5
ACO	5.2070×10^{12}	5.2070×10^{12}	5.2070×10^{12}	0
PSO	−310.1030	2.0528×10^6	1.1166×10^7	3.4116×10^6

The convergence curve of optimal operation of alkylation unit is shown in Figure 9:

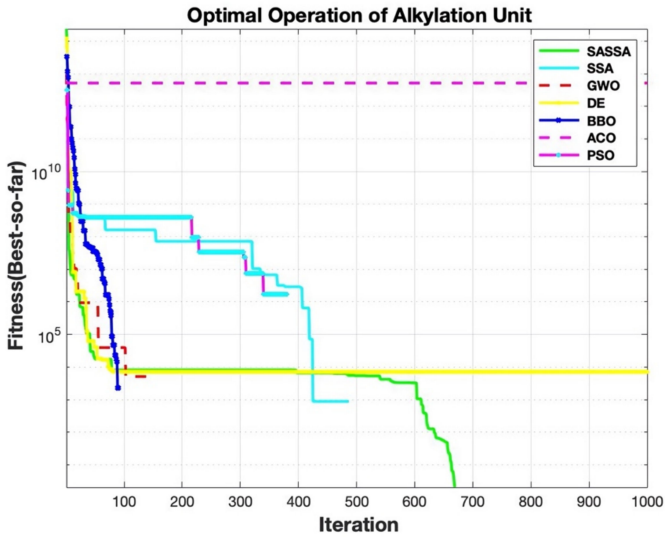

Figure 9. Convergence Curve of Optimal Operation of Alkylation Unit.

5.4.4. Welded Beam Design

The problem of welded beam design can be described as: under the constraints such as shear stress, bending stress of beam, bending load on bar, deflection of beam end and boundary conditions, the optimal design variables h, l, t and b are sought to minimize the cost of manufacturing welded beam [46], as shown in Figure 10:

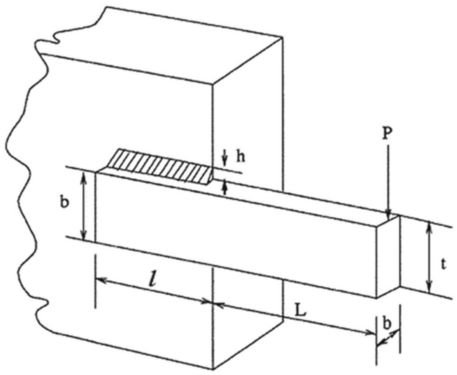

Figure 10. Welded Beam Design.

Use x_1–x_4 to represent the above four variables, the mathematical description is as follows:

Minimize:

$$f(x) = 0.04811x_3x_4(x_2 + 14) + 1.10471x_1^2x_2$$

Subject to:

$$\begin{gathered} g_1(x) = x_1 - x_4 \leq 0, \\ g_2(x) = \delta(\overline{x}) - \delta_{max} \leq 0, \\ g_3(x) = P \leq P_c(\overline{x}), \\ g_4(x) = \tau_{max} \geq \tau(\overline{x}), \\ g_5(x) = \sigma(\overline{x}) - \sigma_{max} \leq 0, \end{gathered}$$

where:

$$\tau = \sqrt{\tau'^2 + \tau''^2 + 2\tau'\tau''\frac{x_2}{2R}}, \tau'' = \frac{RM}{J}, \tau' = \frac{P}{\sqrt{2}x_2x_1}, M = P\left(\frac{x_2}{2} + L\right),$$
$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}, J = 2\left(\left(\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2\right)\sqrt{2}x_2x_1,\right.$$
$$\sigma(x) = \frac{6PL}{x_4x_3^2}, \sigma(\overline{x}) = \frac{6PL^3}{Ex_3^2x_4}, P_c(\overline{x}) = \frac{4.013Ex_3x_4^3}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$
$$L = 14in, P = 6000lb, E = 30.10^6psi, \sigma_{max} = 30,000psi,$$
$$\tau_{max} = 13,600psi, G = 12.10^6psi, \delta_{max} = 0.25in,$$

With bounds:

$$0.125 \leq x_1 \leq 2,$$
$$0.1 \leq x_2, x_3 \leq 10,$$
$$0.1 \leq x_4 \leq 2.$$

The experimental results are shown in Table 12. It can be seen that the improved algorithm can get the best cost value among all the algorithms. Figure 11 shows that the improved algorithm has the best iteration speed, indicating that it has the fastest speed when it obtains the best value.

Table 12. Welded Beam Design.

	Best	Mean	Worst	Std
SASSA	1.7208	1.7232	1.7319	0.0016
SSA	1.7225	1.7675	1.9785	0.0817
GWO	1.7555	1.9321	2.3010	0.1561
DE	1.0890×10^{14}	1.0890×10^{14}	1.0890×10^{14}	0.0321
BBO	1.0890×10^{14}	1.0890×10^{14}	1.0890×10^{14}	0.1447
ACO	1.6916×10^{5}	1.6916×10^{5}	1.6916×10^{5}	5.9720×10^{-11}
PSO	1.0890×10^{14}	1.0890×10^{14}	1.0890×10^{14}	0.0321

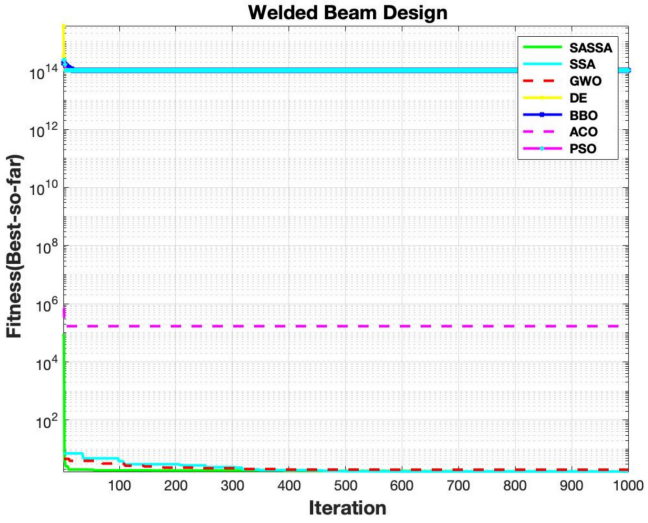

Figure 11. Convergence Curve of Welded Beam Design.

The convergence curve of optimal operation of alkylation unit is shown in Figure 11:

Use the data of the improved algorithm in Tables 9–12 on the four engineering optimization problems to get the ranking of Friedman test. As shown in Table 13, compared with other successful meta-heuristic algorithms, SASSA can also get the best ranking in engineering optimization problems. The results in Table 14 also show that the null hypothesis is rejected so the Friedman ranking is correct.

Table 13. The Friedman ranks(engineering optimization problems).

Rank	Name	F-Rank
0	SASSA	1
1	SSA	2.67
2	PSO	4
3	DE	4.33
4	BBO	4.33
5	GWO	4.67
6	ACO	7

Table 14. Related statistical values (engineering optimization problems).

Chi-sq′	Prob > Chi-sq′ (p)	Critical Value
13.46341463	3.62×10^{-2}	12.59

6. Conclusions

The salp swarm algorithm is a meta-heuristic algorithm based on the predatory behavior of salp, which simulates the group of salp to join end-to-end in the form of a chain and move successively. The salp swarm algorithm has some disadvantages such as slow convergence speed and poor optimization ability. In this paper, the SASSA is constructed by combining chaos initialization population, symmetric adaptive population division and a simulated annealing mechanism based on symmetric perturbation with the salp swarm algorithm. In order to test the ability of the algorithm, 21 benchmark functions were introduced in this paper to evaluate from the aspects of mean value, Std and lowest value. The results show that the improved algorithm proposed in this paper can yield better results for three different types of test functions. At the same time, in order to verify the ability of the improved algorithm to solve practical problems, this paper used the improved algorithm to solve engineering optimization problems. Weight minimization of a speed reducer, gear train design problem, optimal operation of alkylation unit and welded beam design were selected for the experiment, and the experimental results were compared with SSA, GWO, DE, BBO, ACO and PSO, the experimental results prove that the algorithm proposed in this paper has better optimization ability and stability when dealing with engineering optimization problems, and the algorithm's exploratory and mining properties and convergence mode have also been significantly improved. The problems cover the fields of mechanical engineering and chemical engineering. This provides directions and ideas for the improvement of the basic salp swarm algorithm, and also provides a reference solution for solving complex engineering optimization problems in reality.

Author Contributions: Conceptualization, H.K.; methodology, X.S.; project administration, L.W. and Y.S.; software, Q.D.; validation, Q.D. and Q.C.; visualization, L.W. and Q.C.; formal analysis, H.K.; investigation, Q.C.; resources, X.S.; data curation, L.W.; writing—original draft preparation, L.W.; writing—review and editing, H.K. and Q.D.; supervision: Q.D.; funding acquisition: Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the Open Foundation of Key Laboratory in Software Engineering of Yunnan Province under Grant No. 2020SE307, 2020SE308, 2020SE309. This work has been supported by Scientific Research Foundation of Education Department of Yunnan Province under Grant No. 2021J0007.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bhaskar, V.; Gupta, S.K.; Ray, A.K. Applications of multiobjective optimization in chemical engineering. *Rev. Chem. Eng.* **2000**, *16*, 1–54. [CrossRef]
2. Sobieszczanski-Sobieski, J. Multidisciplinary Design Optimization: An Emerging New Engineering Discipline. In *Advances in Structural Optimization*; Springer: Berlin/Heidelberg, Germany, 1995; pp. 483–496.
3. Kondratenko, Y.P.; Simon, D. Structural and parametric optimization of fuzzy control and decision making systems. In *Recent Developments and the New Direction in Soft-Computing Foundations and Applications*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 273–289.
4. Tsamardinos, I.; Brown, L.E.; Aliferis, C.F. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.* **2006**, *65*, 31–78. [CrossRef]
5. Dengiz, B.; Alabas-Uslu, C.; Dengiz, O. A tabu search algorithm for the training of neural networks. *J. Oper. Res. Soc.* **2009**, *60*, 282–291. [CrossRef]
6. Goldberg, D.E.; Holland, J.H. Genetic Algorithms and Machine Learning. *Mach. Learn.* **1988**, *3*, 95–99. [CrossRef]
7. Wang, L.; Zeng, Y.; Chen, T. Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Syst. Appl.* **2015**, *42*, 855–863. [CrossRef]
8. Kennedy, J.; Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*; IEEE: Piscataway, NJ, USA, 1995; pp. 1942–1948.
9. Dorigo, M.; Di Caro, G. Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*; IEEE: Piscataway, NJ, USA, 1999; pp. 1470–1477.
10. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
11. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
12. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
13. Sun, X.; Wang, Y.; Kang, H.; Shen, Y.; Chen, Q.; Wang, D. Modified Multi-Crossover Operator NSGA-III for Solving Low Carbon Flexible Job Shop Scheduling Problem. *Processes* **2021**, *9*, 62. [CrossRef]
14. Liu, Y.; Zhang, Z.; Bo, L.; Zhu, D. Multi-Objective Optimization of a Mine Water Reuse System Based on Improved Particle Swarm Optimization. *Sensors* **2021**, *21*, 4114. [CrossRef]
15. Shen, Y.; Liang, Z.; Kang, H.; Sun, X.; Chen, Q. A Modified jSO Algorithm for Solving Constrained Engineering Problems. *Symmetry* **2020**, *13*, 63. [CrossRef]
16. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
17. Sayed, G.I.; Khoriba, G.; Haggag, M.H. A novel chaotic salp swarm algorithm for global optimization and feature selection. *Appl. Intell.* **2018**, *48*, 3462–3481. [CrossRef]
18. Ibrahim, R.A.; Ewees, A.; Oliva, D.; Elaziz, M.A.; Lu, S. Improved salp swarm algorithm based on particle swarm optimization for feature selection. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 3155–3169. [CrossRef]
19. Faris, H.; Mafarja, M.M.; Heidari, A.A.; Aljarah, I.; Al-Zoubi, A.M.; Mirjalili, S.; Fujita, H. An efficient binary Salp Swarm Algorithm with crossover scheme for feature selection problems. *Knowl.-Based Syst.* **2018**, *154*, 43–67. [CrossRef]
20. Liu, J.; Yuan, M.; Li, Y. The improved salp swarm algorithm is used to solve the engineering optimization design problem. *J. Syst. Simul.* **2021**, *4*, 854–866.
21. Panda, N.; Majhi, S.K. Improved Salp Swarm Algorithm with Space Transformation Search for Training Neural Network. *Arab. J. Sci. Eng.* **2019**, *45*, 2743–2761. [CrossRef]
22. Steinbrunn, M.; Moerkotte, G.; Kemper, A. Heuristic and randomized optimization for the join ordering problem. *VLDB J.* **1997**, *6*, 191–208. [CrossRef]
23. Li, J.; Wang, L.; Tan, X. Sustainable design and optimization of coal supply chain network under different carbon emission policies. *J. Clean. Prod.* **2020**, *250*, 119548. [CrossRef]
24. Zhang, J.; Wang, J.S. Improved Salp Swarm Algorithm Based on Levy Flight and Sine Cosine Operator. *IEEE Access* **2020**, *8*, 99740–99771. [CrossRef]
25. Haupt, R.L.; Haupt, S.E. *Practical Genetic Algorithms*; Wiley: Hoboken, NJ, USA, 2004.
26. Ma, B.; Ni, H.; Zhu, X.; Zhao, R. A Comprehensive Improved Salp Swarm Algorithm on Redundant Container Deployment Problem. *IEEE Access* **2019**, *7*, 136452–136470. [CrossRef]
27. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
28. Li, J.; Liu, F. A trifocal tensor calculation method based on simulated annealing algorithm. In Proceedings of the International Conference on Information Science and Control Engineering (ICISCE), Shenzhen, China, 7–9 December 2012.
29. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.-P.; Auger, A.; Tiwari, S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL Rep.* **2005**, *174*, 341–357.
30. Zhang, Q.; Chen, H.; Heidari, A.A.; Zhao, X.; Xu, Y.; Wang, P.; Li, Y.; Li, C. Chaos-Induced and Mutation-Driven Schemes Boosting Salp Chains-Inspired Optimizers. *IEEE Access* **2019**, *7*, 31243–31261. [CrossRef]
31. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [CrossRef]

32. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
33. Sun, X.; Jiang, L.; Shen, Y.; Kang, H.; Chen, Q. Success History-Based Adaptive Differential Evolution Using Turning-Based Mutation. *Mathematics* **2020**, *8*, 1565. [CrossRef]
34. Wang, Y.; Cai, Z.X.; Zeng, W.; Liu, H. A new evolutionary algorithm for solving constrained optimization problems. *J. Cent. South Univ. (Sci. Technol.)* **2006**, *37*, 119–123.
35. Hosseinzadeh, M.; Garone, E.; Schenato, L. A Distributed Method for Linear Programming Problems With Box Constraints and Time-Varying Inequalities. *IEEE Control. Syst. Lett.* **2018**, *3*, 404–409. [CrossRef]
36. Nicotra, M.M.; Liao-McPherson, D.; Kolmanovsky, I.V. Embedding Constrained Model Predictive Control in a Continuous-Time Dynamic Feedback. *IEEE Trans. Autom. Control.* **2018**, *64*, 1932–1946. [CrossRef]
37. Xu, X. Application of Chaotic Simulated Annealing Algorithm in Numerical Fuction Optimization. *Comput. Digit. Eng.* **2010**, *38*, 37–40, 47.
38. Kumar, A.; Wu, G.; Ali, M.Z.; Mallipeddi, R.; Suganthan, P.N.; Das, S. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evol. Comput.* **2020**, *56*, 100693. [CrossRef]
39. Mirjalili, S. How effective is the Grey Wolf optimizer in training multi-layer perceptrons. *Appl. Intell.* **2015**, *43*, 150–161. [CrossRef]
40. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
41. Juan, W.; Xianxiang, W.; Yanling, C. Multi-layer perceptron using hybrid differential evolution and biogeography-based optimization. *Appl. Res. Comput.* **2017**, *34*, 693–696.
42. Gandomi, A.H.; Yang, X.-S. Benchmark Problems in Structural Optimization. In *Computational Optimization, Methods and Algorithms*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 259–281.
43. Abdel-Basset, M.; Wang, G.-G.; Sangaiah, A.K.; Rushdy, E. Krill herd algorithm based on cuckoo search for solving engineering optimization problems. *Multimed. Tools Appl.* **2019**, *78*, 3861–3884. [CrossRef]
44. Andrei, N. *Nonlinear Optimization Applications Using the GAMS Technology*; Springer: Berlin/Heidelberg, Germany, 2013.
45. Sauer, R.; Colville, A.; Burwick, C. Computer points way to more profits. *Hydrocarb. Process.* **1964**, *84*, 2.
46. Wang, G.-G.; Guo, L.; Gandomi, A.; Hao, G.-S.; Wang, H. Chaotic Krill Herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]