Journal of Network Intelligence ©2023 ISSN 2414-8105 (Online)
Taiwan Ubiquitous Information Volume **8**, Number **2**, May **2023**

An Improved Selfish Herd Optimization Algorithm Based on Nonlinear Inertia Weight

Xinxin Zhou*

School of Computer Science
Northeast Electric Power University
Jilin, Jilin, China
zxx51@qq.com

Xueting Yi

School of Computer Science
Northeast Electric Power University
Jilin, Jilin, China
2089654374@qq.com

*Corresponding author: Xinxin Zhou

Received July 15, 2022; revised September 31, 2022; accepted November 3, 2022

Abstract. *The Selfish Herd Optimizer (SHO) is a novel swarm intelligence algorithm with excellent performance. However, it's accuracy is low and convergence speed is slow. In order to deal with the problems, this paper puts forward an improved Selfish Herd Optimization algorithm (NWSHO). First, aiming at the problem of uneven distribution and overlapping position of the population, a good point set is utilized to initialize population. This strategy improves the algorithm's stability; Second, the nonlinear inertia weight is adopted to update the position of SHO algorithm. The strategy not only balances the global search and local development of the algorithm, but also accelerates the convergence speed and improves the solution accuracy. Finally, the performance of the proposed algorithm is compared with the standard SHO and other well-known swarm intelligence algorithms on two suites of benchmark functions. The results of experiment show the algorithm proposed in this paper is superior to other algorithms in precision and convergence speed.*

Keywords: Selfish herd optimizer, Nonlinear inertia weight, Solution accuracy, Convergence rate

1. Introduction.

The intelligent collective behavior of many species of animals have attracted the attention of researches for many years. Many animal species such as birds, ants, and fishes exhibit aggregative conducts widely known as swarm behavior. Such collective phenomenon has been studied to model the behavior of many biological swarms. Computer science researchers have studied and adapted these models to solve complex real-world problems. As a result, many swarm intelligence optimization algorithms simulate the collective behavior of animals or insects in nature, such as Genetic Algorithm (GA) [1], Particle Swarm Optimization (PSO) [2, 3, 4], Ant Colony Optimization (ACO) [5, 6, 7, 8], Simulated Annealing Algorithm (SA) [9] and Grey Wolf Optimization (GWO) [10], etc. Swarm intelligence algorithms is widely utilized to tackle various problems of optimization such as the backpack problem [11], task scheduling [12, 13], and image segmentation [14], etc. The Selfish Herd Optimizer (SHO) is a meta-heuristic algorithm proposed by Fausto in 2017 [15]. The selfish herd theory which is proposed by Hamilton [16] is the basic idea for SHO. The theory imitates the hunting relationship of predators and prey. Most algorithms have only a single entity with almost the same behavior. For the SHO, it has two search factors with different behavior, which makes the population diverse.

However, in SHO, the leader of herd has an important task. It should choose a route and other herds follow the leader. While the leader gets trapped into locally optimal value, other herds also get trapped into the local optimum value. It influences the algorithm's precision and convergence. Therefore, the paper puts forward a selfish herd optimization algorithm with nonlinear inertia weight. The strategy not only balances the global search and local development of the algorithm, but also accelerates the convergence speed and improves the solution accuracy

In this paper, the main contributions are summarized as follows:

(1) The good point set is applied to make an initial population, which solves the uneven distribution of population and improves algorithm's ergodicity;

(2) A nonlinear inertia weight is proposed to adjust the step size of position update and accelerates the convergence algorithm's speed;

(3) The proposed method is applied on two suites of benchmark functions to verify its effectiveness;

(4) The algorithm is compared with the standard SHO and some other swarm intelligence algorithms. Mean, standard deviation and best fitness are utilized as evaluation measures.

Other parts of the article are arranged as follows: the related work in detail is described in section 2; A description of the standard SHO is introduced in section 3; Section 4 proposes an improved selfish herd optimization algorithm (NWSHO); For section 5, the benchmark functions are applied to test the effectiveness and convergence of this proposed algorithm. And the results of experiment are analyzed. Finally, in section 6, the paper makes conclusions and states the future work.

2. **Related Works.**

Recently, with the rapid development of various swarm intelligence algorithms, they have been utilized widely in various fields. However, the shortcomings have also aroused many scholars' attention. For example, the algorithm is prone to slow convergence and stagnation when dealing with complex optimization problems. Thus, it gets trapped into local optima easily. Currently, there are many strategies of improvement for the algorithm to make a balance between the global search and local development ability. The location update method of discoverers and participants in the Bird Swarm Algorithm (BSA) is applied in the Sparrow Search Algorithm (SSA) by Lv et al. [17]. Mao and Zhang [18] applied adaptive weight, chaotic mapping, reverse learning strategy and Cauchy mutation to Sparrow Search Algorithm (SSA). It improves the algorithm's convergence accuracy and globally optimal capability. In order to enhance the capacity of global exploration, a method of random walk with Levy distribution was adopted by Amirsadri et al. [19] on Gray Wolf Optimization (GWO). Then it is applied to optimize Back propagation neural network. Arora and Anand [20] puts forward a new Grasshopper Optimization Algorithm (GOA) which applies the chaotic theory on accelerating global convergence speed. Chaotic mapping is utilized to keep a balance between the exploration and development ability of algorithm. The reverse learning strategy was utilized to improve the algorithm's convergence by Verma et al. [21]. And the method based on dimension was used to update the position, so as to look for a better globally optimal value. The adaptive elite mutation strategy was introduced to Particle Swarm Optimization (PSO) by Kang et al. [22]. The strategy can make the algorithm jump out of the local optimum. It is applied to select optimal hyper-parameters of gaussian process regression (GPR). Similar to other algorithms, SHO has the problems of slow convergence rate and entrapment in local optima. The scholars have improved the SHO from different aspects. Anand and Arora [23] introduced chaotic search to searching process of SHO. The method enhances global convergence speed. Levy-flight distribution strategy was applied to improve global search ability and precision by Zhao [24].

For the optimization algorithm, the inertia weight is one of the foremost parameters. The larger inertia weight is helpful to enhance the algorithm's ability of global exploration; On the contrary, the smaller inertia weight is more conducive to enhance local development ability. To a certain degree, the change of inertia weight affects the optimization problem [25]. Normally, the strategy of linear decreasing weight is used to keep the best balance between local exploitation and global search [26, 27, 28]. On the basis of existing linear decreasing weight, nonlinear decreasing weight based on the open parabola was proposed [29]. It enhances the algorithm's performance. For standard Particle Swarm Optimization (PSO), the value of inertia weight affects the algorithm's search ability. Therefore, researchers have improved the algorithm's inertia weight [30, 31, 32, 33, 34, 35, 36, 37]. Some scholars have applied the strategy of linear decreasing inertia weight to particle swarm optimization (PSO) [38]. Subsequently, the concept of weight is introduced into other swarm intelligence algorithms to enhance the performance. For instance, on the basis of weighted distance, Malik et al. [39] proposes an improved Gray Wolf Optimization algorithm. For the algorithm, the weighted sum of the best positions rather than a simple average value is utilized to improve position update strategy. It can obtain a better optimal solution. Hu et al. [40] introduced the inertia weight into Whale Optimization Algorithm (WOA). The improved algorithm's performance is superior to the standard algorithm. Rani et al. [41] used linear weight coefficient on Cuckoo Search (CS) to keep from getting trapped into locally optimal value.

The above literature has improved the algorithm for different problems. These strategies have improved the algorithm's performance. For the original SHO, the main problems are that convergence rate is slow and solution accuracy is low. To deal with these problems, a Selfish Herd Optimization algorithm based on nonlinear inertia weight is proposed in this paper.

3. Selfish Herd Optimizer.

3.1. Inspiration Analysis of SHO. The selfish herd theory proposed by Hamilton [16] inspires the SHO. The theory is used to imitate the hunting relationship of prey (selfish herd) and predator. The population consists of predators and prey. During a predator attack, the prey in the group produces aggregation behavior. Then, in order to have a greater chance of survival, each prey will move to the direction (the center of the group). The marginal individuals of the group are more vulnerable to be attacked. Therefore, the marginal individuals will flee from the group to have the chance of survival.

3.2. Mathematical Model of SHO. (1) The population is defined as S and it consists of two groups: H and $P(S = H \cup P)$. $H()$ is a group of the herd. $P()$ is a group of predators. The formula is as follows:

$$N_h = floor\,(N \cdot rand\,(0.7, 0.9)) \tag{1}$$

$$N_P = N - N_h \tag{2}$$

In the formula, N_h means the number of herds, Np is the number of predators, rand represent a random number between 0.7 and 0.9, *floor* () is a function that converts a real number to an integer.

(2) Each individual (s_i) in the population (S) is assigned a value of survival SV_{si}. It is calculated as follows:

$$SV_{si} = \frac{f(s_i) - f_{best}}{f_{best} - f_{worst}} \tag{3}$$

Here, f_{best} and f_{worst} represent that the objective function finds the best fitness values and worst fitness values. $f()$ denotes objective function.

(3) The herd with greatest value of survival is defined as the herd's leader h_L. The formula of leader h_L is as follows:

$$h_L = \left(SV_{h_i} = \max_{i \in \{1,2,\ldots,N_h\}} SV_{h_i}) \right) \tag{4}$$

$$h_L = \begin{cases} h_L + 2 \cdot \alpha \cdot \varphi_{h_L,P_M} \cdot (P_M - h_L), SV_{h_L} = 1 \\ h_L + 2 \cdot \alpha \cdot \varphi_{h_L,x_{best}} \cdot (x_{best} - h_L), SV_{h_L} < 1 \end{cases} \tag{5}$$

Where φ is the attractive force between individuals.α is a random number within the interval [0,1]. x_{best} denotes the best position of current iteration population. P_M is the position in which the herd is easy to be caught. φ and P_M are defined as:

$$\varphi_{h_i,h_j} = SV_{h_j} \cdot e^{-||h_i - h_j||^2} \tag{6}$$

$$P_M = \frac{\sum_{i=1}^{N_P} SV_{p_i} \cdot p_i}{\sum_{j=1}^{N_P} SV_{p_j}} \tag{7}$$

In the formula, $||h_i - h_j||$ denotes Euclidean distance.

(4) The herds consist of the following herd (H_F) and deserting herd (H_D). The definition formula is as follows:

$$H_F = \{h_i \neq h_L \mid SV_{h_i} \geq rand(0,1)\} \tag{8}$$

$$H_D = \{h_i \neq h_L \mid SV_{h_i} < rand(0,1)\} \tag{9}$$

The formula of position update for the following herd is as follows:

$$h_i = \left\{ \begin{array}{c} h_i + 2 \cdot (\beta \cdot \varphi_{h_i,h_L} \cdot (h_L - h_i) + \gamma \cdot \varphi_{h_i,h_{c_i}} \cdot (h_{c_i} - h_i)), h_i \in H_d \\ h_i + 2 \cdot \delta \cdot \varphi_{h_i,h_M} \cdot (h_M - h_i), h_i \in H_S \end{array} \right\} \tag{10}$$

Where φ is the attractive force between individuals. And the specific definition is shown in Formula (6).β , γ and δ are any numbers within the interval [0,1]. h_{c_i} donates locally optimal individual, h_M donates relative safe location of herd. The definition formula is as follows:

$$h_{c_i} = \left(\begin{array}{c} (h_j \in H, h_j \neq [h_i, h_L] \mid SV_{h_j} > SV_{h_i}) \\ r_{i,j} = \min_{j \in \{1,2,\ldots,N_h\}} h_i - h_j \end{array} \right) \tag{11}$$

$$h_M = \frac{\sum_{i=1}^{N_h} SV_{h_i} \cdot h_i}{\sum_{j=1}^{N_h} SV_{h_j}} \tag{12}$$

Here, $r_{i,j}$ donates Euclidean distance.

The formula of position update for deserting herd is as follows:

$$h_i = h_i + 2 \cdot (\beta \cdot \varphi_{h_i,x_{best}} \cdot (x_{best} - h_i) + \gamma \cdot (1 - SV_{h_i}) \cdot \varepsilon) \tag{13}$$

Where ε is a random direction of the space of search. x_{best} denotes global optima. β and γ are random numbers within the interval [0,1].

(5) The movement of predators is calculated as:

$$p_i = p_i + 2 \cdot \rho \cdot (h_r - p_i) \tag{14}$$

Where ρ is any numbers within the interval [0,1]. h_r denotes a random selection between herd, and it bases on the probability of predation θ_{p_i,h_j}. $\theta_{p_i,h_j}$is defined as:

$$\theta_{p_i,h_j} = \frac{\omega_{p_i,h_j}}{\sum_{m=1}^{N_h} \omega_{p_i,h_m}} \tag{15}$$

$$\omega_{p_i,h_j} = \left(1 - SV_{h_j}\right) \cdot e^{-||p_i - h_j||^2} \tag{16}$$

Where ω_{p_i,h_j}indicates the attraction of predator p_i and herd h_j. $||p_i - h_j||$ denotes Euclidean distance between predator p_i and herd h_j.

(6) The predation phase: every prey has a dangerous area. It is usually a circle whose radius R is defined as:

$$R = \frac{\sum_{j=1}^{n} \left| x_j^{low} - x_j^{high} \right|}{2 \cdot n} \tag{17}$$

Here x_j^{low} and x_j^{high} are the initial the lower boundaries and upper boundaries.

The set of herds in danger is defined as follows:

$$T_{p_i} = \left\{ h_j \in H | SV_{h_j} < SV_{p_i}, ||p_i - h_j|| \le R \right\} \tag{18}$$

Where SV_{h_j} and SV_{p_i} denote the survival value of h_{j} and p_i. $||p_i - h_j||$ denotes Euclidean distance.

In dangerous areas, herd has the probability to be hunted. The probability μ_{p_i,h_j} is as follows:

$$\mu_{p_i,h_j} = \frac{\omega_{p_i,h_j}}{\sum_{h_m \in M} \omega_{p_i,h_m}} \tag{19}$$

Where ω_{p_i,h_j}indicates the attraction of predatorp_i and herd h_j. The specific definition is shown in Formula (16).

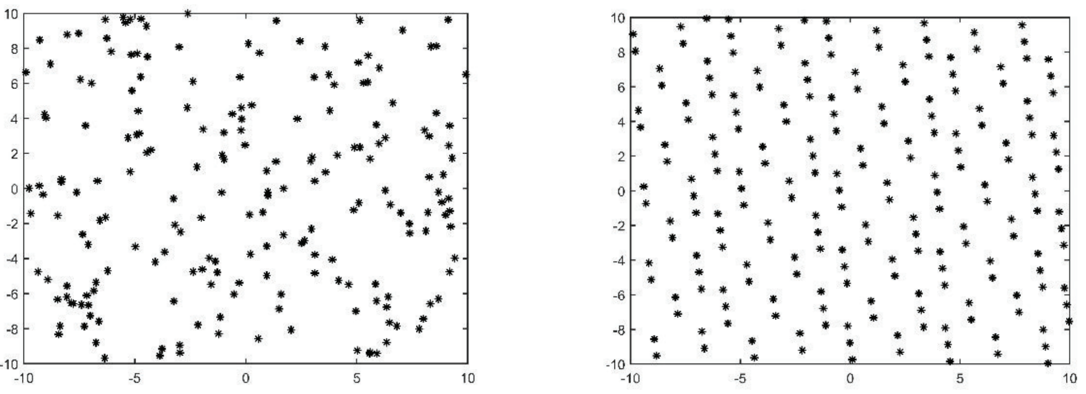
(7) The restoration phase: In SHO, mating operation is used to produce new herd. Then the new herd will replace the herd which is hunted by predators. Mating operation based on mating probability is to select prey to mate. It is specifically defined as:

$$\vartheta_{h_j} = \frac{SV_{h_j}}{\sum_{h_m \in M} SV_{h_m}}, h_j \in M \tag{20}$$

$$h_{new} = mix\left([h_{r_1,1}, h_{r_2,2}, \ldots, h_{r_n,n}]\right) \tag{21}$$

In the formula, M is a set of survival herds. $mix()$ is used to select dimensional components of different individuals $(s = r_1, r_2, ..., r_n)$.

4. **The Improved Algorithm (NWSHO).** Hamilton illustrated his theory by modeling circular pond. The pond has a population of frogs (prey) and a water snakes (predator). When the water snakes appear, the frogs will scatter to the rim of pond. The water snake will certainly attack the nearest one to it. Hamilton suggests that the predation risk of each frog is related not only to how close they are from the attacking predator, but also with the relative position of other frogs in the pond. Therefore, frogs attempt to reduce their predation risk by jumping to smaller gasps between other neighboring frogs which are used as shield. Hamilton suggests that the theory of marginal predation is basic to reduce an individual's domain of danger. The theory states that predators attack the closest prey at the aggregation. Therefore, Hamilton suggests there should be a strong movement of individuals to the center of aggregation in the face of predation. Based on the above theory, the population consists of predators and herds in SHO. Among them, the herd is divided into the leader of herd and others. When the herd is attacked by predators, they choose to follow the leader or escape. Therefore, the location update of herd's leader affects the location updates of other prey followers and predators. When the leader of herd gets trapped into local optimization, the algorithm is prone to slow



(a) initialization with random method (b) initialization with good point set strategy

Figure 1. Initialized population distribution diagram

convergence and low accuracy. Thus, an improved Selfish Herd Optimization algorithm is proposed.

4.1. **Population Initialization Based on a Good Point Set.** The distribution of population's initial position in swarm intelligence algorithm will affect the capacity of global search and global optimal solution. The population with uniform distribution and diversity is helpful for improving the algorithm's optimization performance.

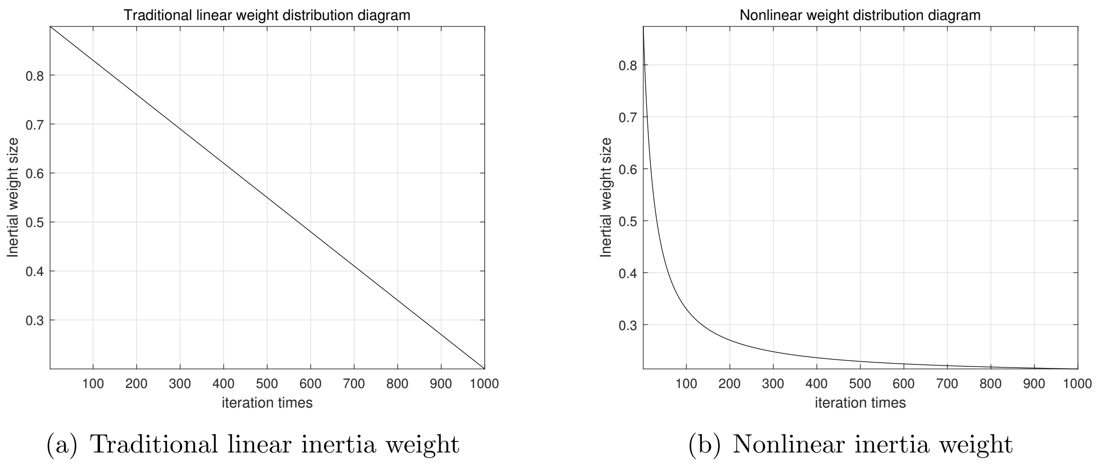
However, the standard SHO uses a random strategy for population initialization. The strategy does not guarantee the population with the diversity of the initial position. And it causes initial position of the individuals to gather near the local optimum, which affects the population to find the global optimum. Hence, this paper puts forward a good point set to initialize population. With the method, the population distributed in the whole solution space evenly. The strategy is conducive to finding the global optimum.

The good point set was proposed by Chinese mathematician Luogeng Hua et al. [42]. It is defined as: Gs is unit cube in the S-dimensional European space, if $r \in G_S$,then $P_n(k) = \left\{ (\left\{ r_1^{(n)} * k \right\},\ \left\{ r_2^{(n)} * k \right\}, \ldots, \left\{ r_s^{(n)} * k \right\}), 1 \leq k \leq n \right\}$. The deviation $\varphi(n)$ is defined as: $\varphi(n) = C(r, \varepsilon) * n^{-1+\varepsilon}$. $C(r, \varepsilon)$ is a constant related to r, ε; ε is an arbitrary constant in the computer. $\left\{ r_s^{(n)} * k \right\}$ represents decimal part. Then $P_n(k)$ is the set of good points, r is the best point. The value of r is $\{2 * \cos(2\pi k/p), 1 \leq k \leq s\}$. Among them, p satisfies $(p-3)/2 \geq s$. It is mapped to the space of search as:

$$x_{i(j)} = (ub_j - lb_j) \cdot \left\{ r_j^{(i)} * k \right\} + lb_j \tag{22}$$

In this paper, random initialization and good point set are utilized to make an initial population in a two-dimension space with the size of 200, as shown in Figure 1. Figure 1 (a) and (b) shows the good point set is more evenly distributed than the random method in the same population size. Therefore, this paper used the good point set to initialize the population. It generates a uniformly distributed, rich and diverse population in the solution space, which is conducive to avoiding getting trap into the locally optimal value. The method enhances the population's ergodicity and stability. The algorithm's performance is improved.

4.2. **The Position Update Strategy of Herd's Leader Based on Nonlinear Inertia Weight.** In SHO, the leader of herd has the highest survival value. When the



(a) Traditional linear inertia weight (b) Nonlinear inertia weight

FIGURE 2. Comparison chart of different weights

position is updated, the leader has a certain leadership. However, as can be seen from Formula (5) the method of update has certain randomness. The method will make the herd's leader get trapped into local optimization in the search process. The inertia weight is one of the foremost parameters. When inertia weight is larger, it is helpful to improve the ability of global search; On the contrary, it is more conducive to enhancing the algorithm's ability of the local search, which can avoid missing the optimal value. Therefore, the inertia weight was used into the SHO for the update position of herd's leader. The strategy improves search capability of herd's leader and avoids getting trapped into local optimum. Thereby the ability of search and convergence speed are improved.

Generally, the inertia weight is linear, as shown in Figure 2 (a). It will decline regularly with the iteration process, so it cannot adapt to the actual situation of the iteration process better. And the nonlinear weight can reflect the actual situation in the iterative process. Consequently, this paper introduces a nonlinear inertia weight to improve method of position update for the herd's leader. The herd's leader can learn from himself. In this strategy, the randomness of position update is reduced. Therefore, the search ability and convergence speed are improved. The specific definition of nonlinear inertia weight is as follows:

$$\omega = \omega_{min} * \left(\omega_{max}/\omega_{min}\right)^{(1/(1+20*i/it))} \tag{23}$$

Here, the minimum inertia weight ω_{min} is assigned as 0.2, the maximum inertia weight ω_{max} is assigned as 0.9. *it* indicates the maximum number of iterations; i is the number of current iteration. The output diagram of nonlinear inertia weight is shown in Figure 2 (b). At the early stage of iteration, the nonlinear weight's value is larger. It is not only helpful for the herd's leader to search globally but also avoids getting trapped into the local optimization; At the algorithm's later stage, the nonlinear weight's value decreases gradually. It is beneficial for herd's leader to search locally. The nonlinear weight which improves the algorithm's ability of search is introduced to make a balance between local search and global exploration.

For herd's leader, the formula of position update is corrected as:

$$h_L = \left\{ \begin{matrix} \omega * h_L + 2\alpha\varphi_{h_L,P_M}\left(P_M - h_L\right), \ SV_{h_L} = 1 \\ \omega * h_L + 2\alpha\varphi_{h_L,x_{best}}\left(x_{best} - h_L\right), \ SV_{h_L} < 1 \end{matrix} \right\} \tag{24}$$

Here, ω denotes the nonlinear inertia weight.

4.3. **The Process of NWSHO.**

4.3.1. *Population Initialization.* The population is initialized according to the good point set proposed in Section 4.1; Then the group is divided into herd and predator. In the nature, the amount of herds is usually more than that of predators. The method of division is shown in Formula (1) and (2);

4.3.2. *Survival value assignment.* The individuals of population are set to a survival value. The value indicates successfully kill the herd or the opportunity to live in the attack. Survival value is calculated as Formula (3);

4.3.3. *Herd movement.* Herd is divided into the leader of herd and others. When there is danger, other herd will choose to follow their leader or escape.

(1) Herd's leader movement

In SHO, as shown in Formula (4), the leader of the herd has the highest survival value. Therefore, the location of herd leader is safest. The position update method adopts the nonlinear inertia weight strategy proposed in this paper as Formula (24);

(2) The movements of following Herd and desertion Herd

For the population of herd, the position of leader is safest. And the positions of others are divided into two situations: relatively safe and more dangerous. When the danger is coming, the herd in a relatively safe position will move to the leader, that is, the following movement of the herd. The method of the following movement is shown in Formula (10). The herd in a more dangerous position will escape from the group, that is, the desertion movement. The strategy of the desertion movement is shown in Formula (13).

4.3.4. *Predator movement.* For predator, the update of location correlates with the survival value and herd's location. It is defined as Formula (14).

4.3.5. *Recalculation of survival values.* In the whole population, individual's survival value may change after the movement of herd and predator. Therefore, the value of survival is recalculated as Formula (3).

4.3.6. *Predation phase.* Each herd has a dangerous area. The area is defined as Formula (18). If the herd belongs to this field, it is likely to be killed by predators. The probability of being killed is calculated as Formula (19);

4.3.7. *Restoration phase.* The number of herd population changes dynamically with time in the nature. But in a balanced biological system, this change is often periodic. In SHO, all killed individuals are replaced with new ones, as shown in Formula (21). Mating operation may produce new herd. Herd selects mating herd by mating probability, which is defined as Formula (20).

In the process of searching, each individual is evaluated by fitness function. Then the position is updated. If the iteration times of the algorithm reach the upper limit or the candidate solution satisfying the termination conditions has been obtained, the algorithm ends. The flowsheet of NWSHO is shown in Figure 3. Table 1 shows the pseudo code.

5. Experiment and Analysis of Result.

The NWSHO is compared with other intelligent algorithms and the standard SHO to verify the performance. The experiment is implemented using MATLAB R2020b under Microsoft Windows 10 operating system. All simulations are carried out on Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz 2.30 GHz and 8.00GB memory computer.

Table 1. The pseudo code

```
Algorithm: Improved SHO Algorithm Based on Nonlinear Inertial Weight
Begin
    Step 1. Initializing parameters of population $S$
    Step 2. Utilize the good point set strategy to initialize the population by Formula (22)
    Step 3. Divide the population $S$ into predators and herds by Formula (1) and Formula (2)
    Step 4. For population $S$ do
                Calculate the survival values by Formula (3)
            End for
    Step 5. while ($t < T$)
                For each selfish herd
                  IF leader of the herd
                    Update position of herd's leader by Formula (24)
                  Else
                    The herd's following and escape movement by Formula (10) and Formula (13)
                  End IF
                End for
                For each predator
                    Update position of the predators by Formula (14)
                End for
                    Recalculate the survival values of each member by Formula (3)
                Perform predation phase by Formula (18) and Formula (19)
                Perform restoration phase by Formula (20) and Formula (21)
                $t = t + 1$
    End while
End
```

5.1. **Benchmark Functions.** In this paper, two sets of benchmark functions are used to evaluate the algorithm's performance. The first set of functions have 15 benchmark functions, which are utilized in literature [15] to investigate the standard SHO's performance. The functions in detail are given in Table 2. The functions consist of two categories. $F_1 - F_8$ are the first category. They are unimodal functions used to test the algorithm's convergence and solution accuracy. After the iteration, if the result is closer to the theoretical optimum value, the accuracy is higher. In the second category, $F_9 - F_{15}$ are multimodal functions used to verify the performance of global search and ability to avoid premature.

The second set of functions have 15 benchmark functions derived from CEC2017 [43, 44]. The information of functions is shown in Table 3. Among them, $F_{16} - F_{23}$ are unimodal functions. And $F_{24} - F_{30}$ are multimodal functions.

5.2. **Performance Metrics.** The performance evaluation indexes of the algorithm are mean value, standard deviation and best fitness. In addition, the test called Wilcoxon's rank sum test is utilized to test whether NWSHO is different from other algorithms. In order to reduce the algorithm error, each algorithm runs for 30 times independently, and the mean value is obtained as the experimental data.

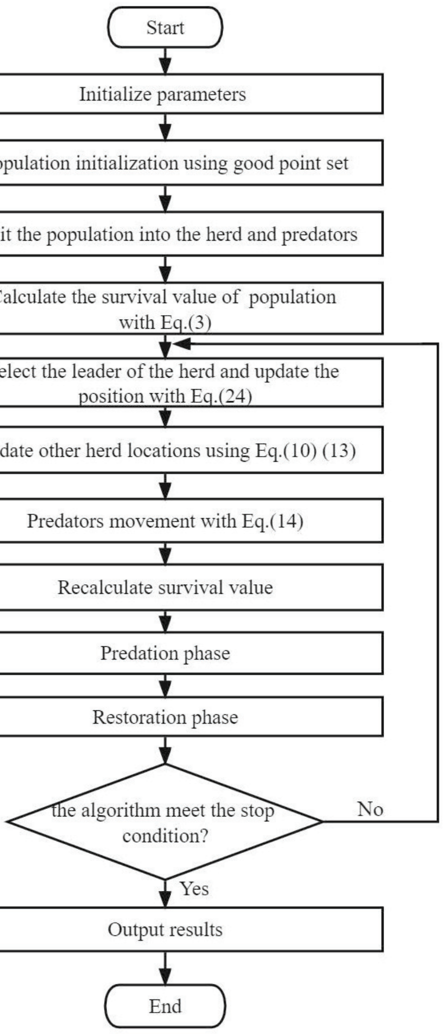

Start
Initialize parameters
Population initialization using good point set
Split the population into the herd and predators
Calculate the survival value of population with Eq.(3)
Select the leader of the herd and update the position with Eq.(24)
Update other herd locations using Eq.(10) (13)
Predators movement with Eq.(14)
Recalculate survival value
Predation phase
Restoration phase
the algorithm meet the stop condition?
No
Yes
Output results
End

Figure 3. Flowchart of NWSHO

(1) Mean value (Mean): It is the average of operation results as given in the following equation:

$$Mean = \frac{1}{N}\sum_{I=1}^{N} f_i \tag{25}$$

Where N is the running times, which is assigned to 30. f_i is the result obtained after the $i-th$ run of the test algorithm.

(2) Standard deviation (Std): It is arithmetic square root of variance in N times of operation results. It is mainly used to reflect the dispersion degree of operation results. The larger value represents that the dispersion degree of results is greater. On the contrary, the dispersion degree is smaller. The standard deviation is defined as:

$$Std = \sqrt{\frac{1}{N-1}\sum_{i-1}^{N}(f_i - Mean)^2} \tag{26}$$

Table 2. Traditional benchmark function

Function	Range	$f_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^{n} ix_i^2$	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^{n/4} [(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} + 10x_{4i})^2$ $+ (x_{4i-2} + 10x_{4i-1})^4 + 10(x_{4i-1} + 10x_{4i})^4]$	$[-4, 5]$	0
$F_4(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-5, 10]$	0
$F_5(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]$	0
$F_6(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$	$[-10, 10]$	0
$F_7(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	$[-65.5, 65.5]$	0
$F_8(x) = \sum_{i=1}^{n} (ix_i)^4 + rand[01]$	$[-1.28, 1.28]$	0
$F_9(x) = -20\exp\left(-0.2\sqrt{1/n \sum_{i=1}^{n} x_i^2}\right)$ $- \exp\left(1/n \sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	$[-32.8, 32.8]$	0
$F_{10}(x) = \sin^2(\pi\omega_1) + \sum_{i=1}^{n-1} (\omega_i - 1)^2[1 + 10\sin^2(\pi\omega_i + 1)]$ $+ (\omega_n - 1)^2[1 + \sin^2(2\pi\omega_n)]$ $\omega_i = 1 + (x_i + 1)/4$	$[-10, 10]$	0
$F_{11}(x) = 418.983n - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]$	0
$F_{12}(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$	$[-n^2, n^2]$	-4930
$F_{13}(x) = \sum_{i=1}^{n} (x_i^2 - i)^2$	$[-500, 500]$	0
$F_{14}(x) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	$[-100, 100]$	0
$F_{15}(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$	$[-5, 10]$	0

(3) Best fitness (Best): It is the minimum value obtained by the objective function in the iteration process. The formula is as:

$$Best = min f_i \tag{27}$$

(4) Wilcoxon's rank sum test [45]: The test is nonparametric. It mainly tests whether two algorithms have difference. When h-value is equal to 1 or p-value is less than 5%, it indicates there are obvious differences between the two algorithms.

5.3. **Parameter Setting.** To test the algorithm's effectiveness, this paper compares the NWSHO with other excellent algorithms, such as standard SHO [15], OPIO [46], PSO [47], GA [48] and BBO [49]. For all algorithms, the relevant parameters are set in Table 4.

5.4. **Comparison Experiment on the First Suites of Benchmark Functions.** Firstly, the NWSHO and SHO algorithms are tested on 15 benchmark functions from the literature [15]. Table 2 describes these functions in detail. In the algorithms, parameter settings are the same as that in literature [15]. The iteration's maximum number is set as 1000. The size of population is set to 50. The mean value is got when each algorithm runs 30 times independently. The best fitness values, standard deviation and mean are recorded. Table 5 shows the results of experiment.

Table 3. CEC2017 benchmark functions

Function	Range	$f_{\min}$
$F_{16}(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]$	0
$F_{17}(x) = \max\{\lvert x_i \rvert, 1 \le i \le n\}$	$[-100, 100]$	0
$F_{18}(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	$[-100, 100]$	0
$F_{19}(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{(i+1)}$	$[-1, 1]$	0
$F_{20}(x) = \sum_{i=1}^{n} i x_i^4$	$[-1.28, 1.28]$	0
$F_{21}(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^6$	$[-100, 100]$	0
$F_{22}(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$	$[-100, 100]$	0
$F_{23}(x) = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} x_i^2$	$[-100, 100]$	0
$F_{24}(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]$	-418.98*dim
$F_{25}(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	0
$F_{26}(x) = 1/4000 \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(x_i/\sqrt{i}\right) + 1$	$[-600, 600]$	0
$F_{27}(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2 \left[1 + \sin^2(2\pi x_n)\right]$	$[-50, 50]$	0
$F_{28} = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	$[-10, 10]$	0
$F_{29} = \sum_{i=1}^{n} x_i{}^6 (2 + \sin(1/x_i))$	$[-1, 1]$	0
$F_{30}(x) = \lvert \sum_{i=1}^{n} x_i^2 - n \rvert^{1/4} + (0.5 \sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_i)/n + 0.5$	$[-10, 10]$	0

Table 4. The parameter settings

Algorithm	Related parameters
NWSHO	$N = 50, M = 1000, Nh = 0.7N - 0.9N, w_{\min} = 0.2, w_{\max} = 0.9;$
SHO [15]	$N = 50, M = 1000, Nh = 0.7N - 0.9N;$
OPIO [46]	$N = 50, Vmax = 0.5; R = 0.2;$
PSO [47]	$N = 50, w = [0.2, 0.9]$
GA [48]	$N = 50, M = 0.1$
BBO [49]	$N = 50, I = 1;$

As shown in Table 5, for $F_1, F_2, F_3, F_5, F_{15}$, the mean value, standard deviation and best fitness values of NWSHO algorithm proposed in the paper are all 0. The corresponding function's theoretical optimum value is found. However, for the original SHO, the better results are obtained only on two functions; For most functions, the mean value, standard deviation and best fitness values of NWSHO algorithm are closer to the theoretical optimum value. The optimization accuracy of NESHO is higher than that of SHO algorithm. It indicates NWSHO algorithm is superior to SHO algorithm in optimization accuracy and stability. It is proved that the strategy with good point set and nonlinear inertia weight introduced in this paper is effective.

The convergence diagrams of partial functions are shown in Figure 4. For unimodal function F_1, multimodal functions F_9 and F_{14}, the convergence diagrams show that NWSHO algorithm's convergence speed is significantly faster than standard SHO algorithm's convergence speed. For the unimodal function F_1, NWSHO algorithm finds the optimal value when the algorithm iterates to 300. However, for the standard SHO

Table 5. The experimental results of $F_1 - F_{15}$ under 30 dimensions

Function		NWSHO	SHO	f_{min}
F_1	Mean	**0**	1.9288e-05	0
	Std	**0**	3.0923e-06	
	Best	**0**	1.3364e-05	
F_2	Mean	**0**	0.0279	0
	Std	**0**	0.0053	
	Best	**0**	0.0201	
F_3	Mean	**0**	5.3517	0
	Std	**0**	1.1277	
	Best	**0**	2.8160	
F_4	Mean	**28.5046**	73.3426	0
	Std	**0.4468**	49.3539	
	Best	27.4262	**1.4460**	
F_5	Mean	**0**	1.2067	0
	Std	**0**	2.7720	
	Best	**0**	0.0136	
F_6	Mean	8.8421e-07	**6.2400e-07**	0
	Std	1.1607e-06	**7.3686e-07**	
	Best	2.0377e-08	**4.9069e-10**	
F_7	Mean	-3.0455e+04	**-3.0499e+04**	0
	Std	**11.3488**	47.8153	
	Best	**-3.0457e+04**	-3.0457e+04	
F_8	Mean	**1.2550e+11**	1.2550e+11	0
	Std	0.0191	**0.0136**	
	Best	1.2550e+11	1.2550e+11	
F_9	Mean	**8.8818e-16**	2.9589	0
	Std	**0**	1.3180	
	Best	**8.8818e-16**	0.0134	
F_{10}	Mean	**0.3953**	1.4648	0
	Std	**0.9530**	1.9226	
	Best	**7.3272e-04**	7.5576e-04	
F_{11}	Mean	5.4305e+03	**3.2449e+03**	0
	Std	**651.1905**	1.3779e+03	
	Best	4.0474e+03	**1.1741e+03**	
F_{12}	Mean	**-1.8017e+07**	-1.6848e+07	0
	Std	**1.8879e+06**	1.9192e+06	
	Best	**-2.1285e+07**	-2.0250e+07	
F_{13}	Mean	1.1053e-12	**4.4659e-13**	0
	Std	1.4255e-12	**5.7896e-13**	
	Best	3.3399e-16	**2.8434e-17**	
F_{14}	Mean	**0.0999**	1.3612	0
	Std	**2.7565e-13**	0.8781	
	Best	**0.0999**	0.2999	
F_{15}	Mean	**0**	0.0074	0
	Std	**0**	0.0011	
	Best	**0**	0.0054	

algorithm, most of the individuals have gathered at the local optimum value. These individuals eventually cannot flee from the local optimum value. For multimodal functions F_9 and F_{14}, NWSHO algorithm has fast speed of convergence and strong ability of global search. When the population iterates to 100, the optimal value has been found. But the standard SHO algorithm falls into local optimization for many times on function F_{14}. For unimodal function F_6, when the population iterations to 100, the accuracy of NWSHO algorithm is slightly higher. Nevertheless, the standard SHO algorithm flee from the local optimum value for many times after the 900th iteration. For function F_{12}, the convergence of NWSHO algorithm is fast. Then it finds the theoretical optimal value.

5.5. **Comparison Experiment on the Second Suite of Benchmark Functions.** To further test the performance of NWSHO, 15 CEC2017 benchmark functions are selected to simulate experiments. This experiment compares NWSHO algorithm with SHO, OPIO, PSO, GA and BBO algorithms. For the experiment, the maximum iteration is 1000. The dimensions are set as 30, 50 and 100 respectively. Each experiment is run 30 times independently to get mean value. The functions $F_{16} - F_{30}$ are used to evaluate the algorithm. The best function value, standard deviation and mean are recorded in Table 6.

The mean value, standard deviation and optimal fitness values of NWSHO on the most unimodal benchmark functions (except for function F_{18}) are all 0 in Table 6. And the theoretical optimal values of the corresponding functions have been found. In the multimodal function, the optimal fitness value of F_{26}, F_{28}, F_{29} is 0. The mean and standard deviation of F_{29} are 0. It shows that NWSHO is superior to standard SHO and other standard algorithms on stability and accuracy of optimization.

The convergence diagram of partial functions on 30 dimensions is shown in Figure 5. We can see from Figure 5 the convergence for NWSHO is fairly higher than that of SHO and other optimization algorithms. From the convergence diagram of the unimodal function F_{20}, we can see the NWSHO converges quickly and finds the optimal value when iterating to 100. However, for other algorithms, they get trapped into locally optimal value. For unimodal function F_{18}, NWSHO algorithm has slightly higher accuracy when the population iteration reaches 100 times. Then it converges to the optima when the population iteration reaches 300 times. However, for the standard SHO, the convergence speed decreases when iterating to 200. Then the optimal value is found after the 300th iteration. For multimodal function F_{30}, the convergence speed of NWSHO is quick. When the population iterations reach 200 times, the algorithm gradually finds the optimal value and does not fall into local optimization. For the other algorithm, except for the OPIO, other algorithms cannot flee from locally optimum value when the population iterates to 100. To sum up, the nonlinear inertia weight strategy introduced in this paper enhances algorithm's accuracy of solution and convergence speed. Otherwise, it can keep the algorithm from getting trapped into locally optimal value to some extent.

To validate the algorithm's performance under different dimensions, the dimensions are set to 30, 50 and 100. Table 7 shows the results of experiment. For the benchmark function $F_{16}, F_{17}, F_{19}, F_{20}, F_{21}, F_{22}, F_{23}$, and F_{29}, even if the dimension increases, the mean value, standard deviation and optimal fitness values of NWSHO algorithm are still zero. It indicates NWSHO has the better stability and ability of global search, although the dimension is increasing. For other algorithms, average value, standard deviation and optimal fitness value of multiple benchmark functions will change while the dimension increases. For example, for the unimodal function F_{18}, the mean value, standard deviation and optimal fitness value of the standard SHO change by about 10 orders of magnitude; For the function F_1, the average value and standard deviation of GA algorithm and BBO

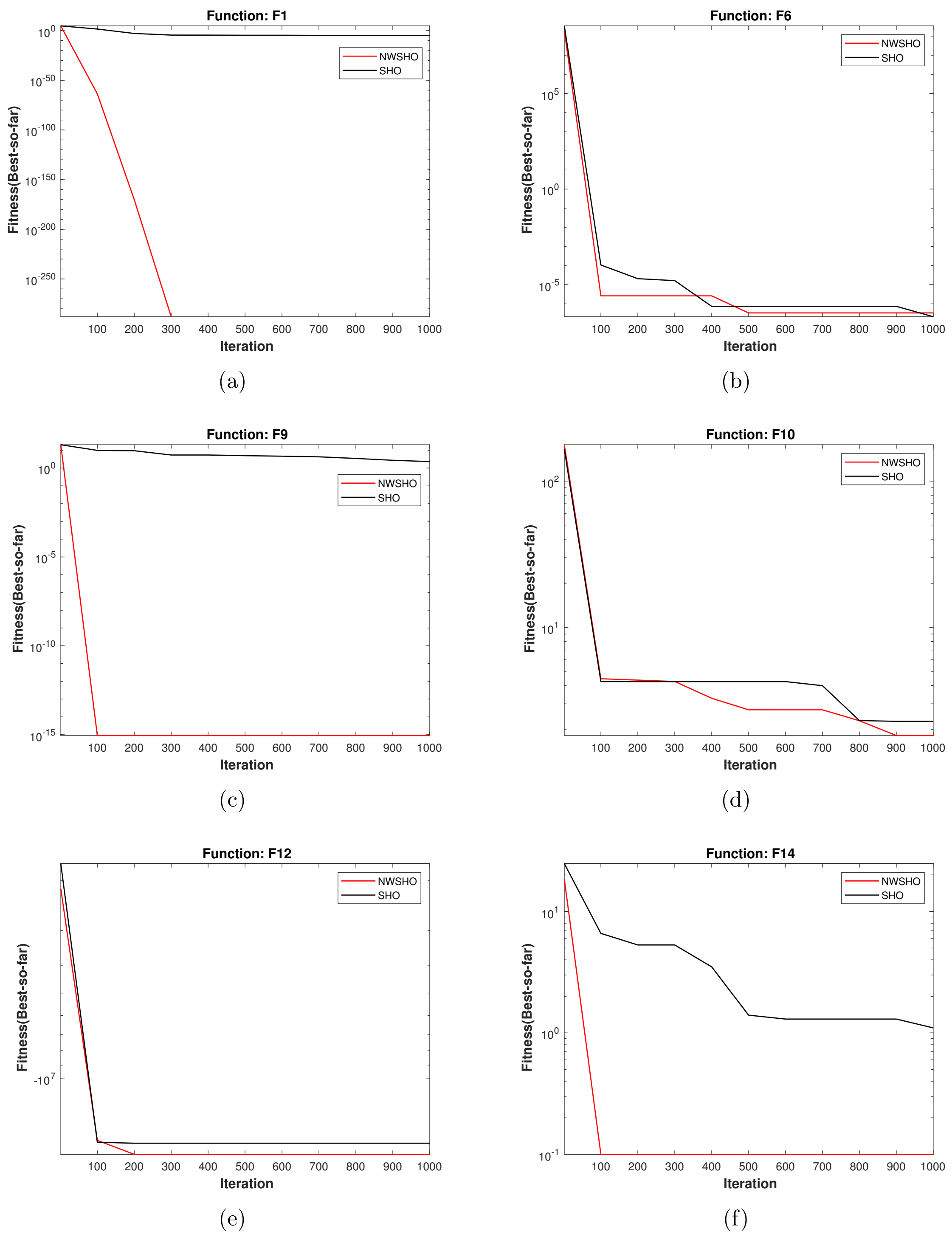


FIGURE 4. The convergence diagram of partial functions

algorithm change by about 30 orders of magnitude. In general, the stability and accuracy of NWSHO are higher than other algorithms.

Table 6. Experimental results of each algorithm in 30 dimensions

Function		NWSHO	SHO	OPIO	PSO	GA	BBO
F_{16}	Mean	**0**	0.1770	0.0164	51.7667	1.1285e+11	7.8504e+03
	Std	**0**	0.0113	0.0104	30.9590	2.5690e+11	4.2371e+04
	Best	**0**	0.1554	0.0040	1.0000	2.1726e+05	80.0836
F_{17}	Mean	**0**	0.1770	0.6739	52.8000	84.0896	85.0549
	Std	**0**	0.0113	0.5678	28.6734	4.6251	2.8996
	Best	**0**	0.0035	0.0050	2	69.1760	79.1849
F_{18}	Mean	1.8782e-05	**1.7058e-05**	7.2990	49.6333	6.1470e+04	5.9781e+04
	Std	2.3957e-06	**2.3244e-06**	0.3099	29.7443	6.1964e+03	5.3470e+03
	Best	**1.1785e-05**	1.2971e-05	6.0804	7.0000	5.0344e+04	5.1630e+04
F_{19}	Mean	**0**	1.1224e-05	7.2659e-12	57.0333	0.5073	0.3914
	Std	**0**	7.3383e-06	2.4264e-11	29.3169	0.1953	0.1579
	Best	**0**	3.0383e-06	1.2878e-15	5.0000	0.2141	0.1166
F_{20}	Mean	**0**	0.0108	6.8376e-12	60.6333	97.1215	94.2383
	Std	**0**	0.0028	1.4791e-11	27.1363	20.5806	25.1356
	Best	**0**	0.0060	1.2971e-14	4.0000	52.6845	39.0759
F_{21}	Mean	**0**	27.5925	4.6713e-06	42.6667	1.9873e+18	1.1983e+18
	Std	**0**	61.6582	1.1657e-05	30.1540	5.0770e+17	3.3603e+17
	Best	**0**	3.5450e-10	1.0525e-08	3.0000	1.1773e+18	5.4977e+17
F_{22}	Mean	**0**	5.7814e-04	2.9040	49.6333	2.1574e+12	1.2908e+12
	Std	**0**	3.3579e-04	9.4463	31.0722	7.2380e+11	3.3873e+11
	Best	**0**	5.6843e-05	3.0756e-05	4.000	4.6220e+11	7.1534e+11
F_{23}	Mean	**0**	7.3502	6.8361	44.9000	1.8884e+09	1.9152e+09
	Std	**0**	3.0310	13.4727	33.3481	5.5248e+08	5.3878e+08
	Best	**0**	2.6241	0.0477	1.0000	7.1553e+08	7.4270e+08
F_{24}	Mean	-7.3048e+03	-9.5117e+03	-1.3819e+04	50.2333	-2.4977e+03	-2.966e+03
	Std	688.9777	1.4295e+03	4.6141e+03	**34.1242**	398.7892	338.4343
	Best	-9.0345e+03	-1.2569e+04	-2.4669e+04	**1.0000**	-3.3533e+03	-3.5924e+03
F_{25}	Mean	53.1053	55.4092	**0.0198**	45.3000	429.6691	401.7535
	Std	16.2463	12.2011	**0.0279**	28.3806	23.1441	23.5300
	Best	24.1663	31.2853	**0.0012**	4.0000	380.2632	359.9953
F_{26}	Mean	**2.4654e-04**	0.0095	0.0030	49.7333	552.9029	547.4418
	Std	**0.0014**	0.0170	0.0034	27.2155	57.7185	40.0277
	Best	**0**	1.7385e-08	1.0808e-04	11.0000	390.0026	475.3812
F_{27}	Mean	9.9812e-06	**8.4675e-06**	3.0136	47.9333	9.0508e+08	9.3250e+08
	Std	1.3680e-06	**1.2177e-06**	0.0215	32.7087	2.2534e+08	1.6751e+08
	Best	6.5496e-06	**5.8024e-06**	2.9978	1.0000	2.8751e+08	5.4472e+08
F_{28}	Mean	0.2173	0.8713	**0.0035**	55.6000	58.4314	57.8071
	Std	0.2679	0.8423	**0.0018**	31.9899	4.9286	6.0275
	Best	**0**	0.1833	5.0436e-04	2.0000	43.6637	44.9686
F_{29}	Mean	**0**	8.0227e-05	2.3338e-19	49.8000	2.7706	2.2901
	Std	**0**	3.0004e-05	3.7993e-19	30.6520	0.6838	0.7110
	Best	**0**	1.7243e-05	9.0140e-23	1.0000	1.2097	0.7454
F_{30}	Mean	9.3177e-04	**4.9577e-04**	0.7448	49.4667	154.8529	150.9546
	Std	3.3831e-04	**1.2680e-04**	0.1629	29.6773	13.0742	12.4249
	Best	3.3240e-04	**2.3021e-04**	0.3824	1.0000	131.5327	127.7931

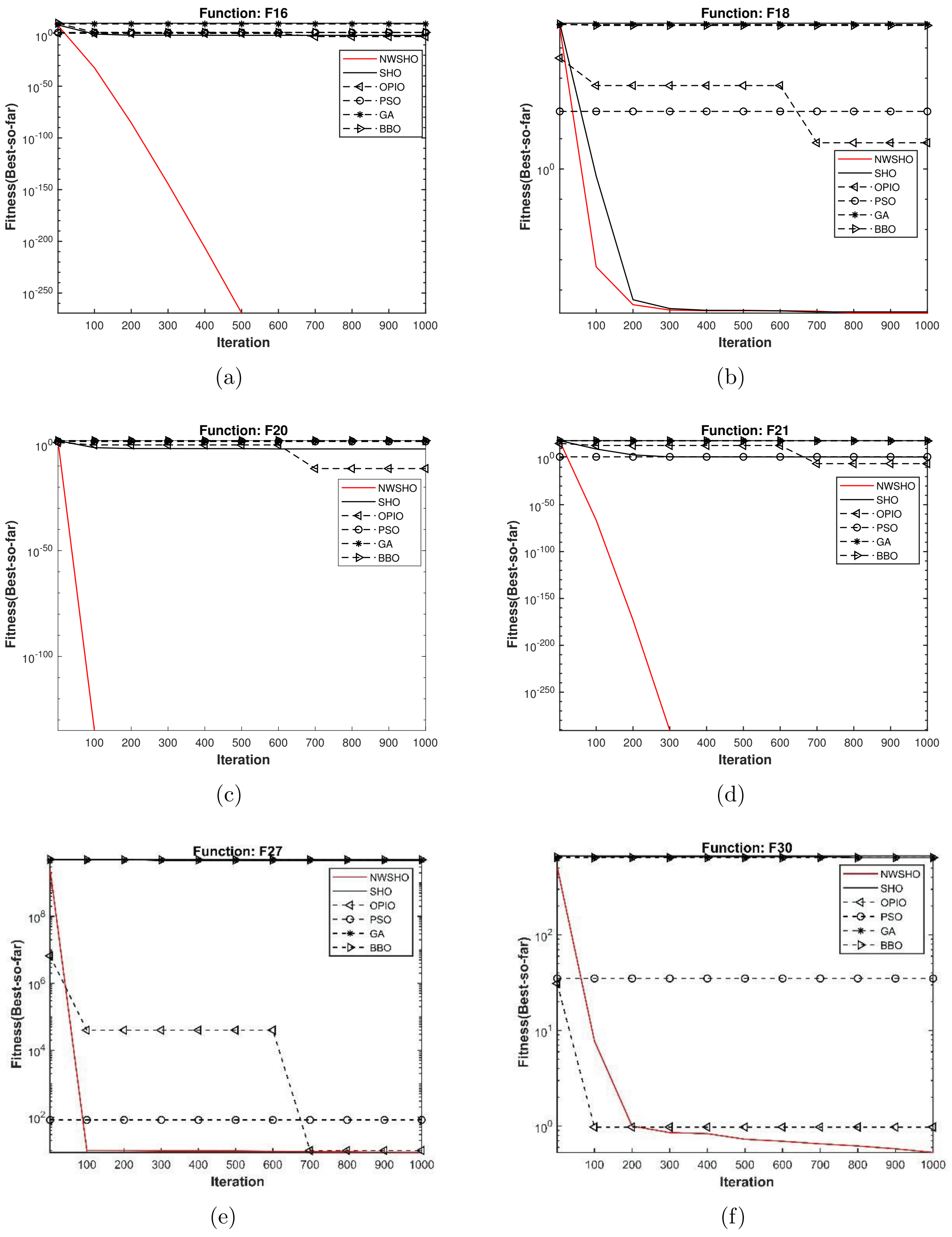


Figure 5. The convergence diagram of partial functions on 30 dimensions

Table 7. The results of different algorithms under different dimensions

fun		NWSHO			SHO			OPIO			PSO			GA			BBO		
	dim	30	50	100	30	50	100	30	50	100	30	50	100	30	50	100	30	50	100
	Mean	**0**	**0**	**0**	0.1770	3.90e+05	8.71e+36	0.0164	0.0308	0.0740	51.7667	45.4667	46.7000	1.1e+11	4.01e+21	831e+47	7.85e+03	1.7e+06	1.33e+41
F_{16}	Std	**0**	**0**	**0**	0.0113	1.80e+06	2.84e+37	0.0104	0.0212	0.0268	30.9590	28.4505	27.8520	2.6e+11	1.1e+22	2.83e+48	4.24e+04	9.1e+06	7.29e+41
	Best	**0**	**0**	**0**	0.1554	0.3180	1.06e+26	0.0040	0.0079	0.0297	1.0000	4.0000	2.0986	2.2e+05	2.60e+18	1.96e+37	80.0836	174.6749	398.5095
	Mean	**0**	**0**	**0**	0.1770	11.4771	94.6699	0.6739	0.0190	0.0231	52.8000	46.7000	53.0667	84.0896	90.3707	95.2196	85.0549	89.7975	94.5207
F_{17}	Std	**0**	**0**	**0**	0.0113	6.1829	1.4706	0.5678	0.0111	0.0137	28.6734	31.1138	30.7727	4.6251	1.6824	0.9593	2.8996	3.0318	1.5050
	Best	**0**	**0**	**0**	0.0035	2.9907	89.2248	0.0050	0.0028	0.0098	2.0000	2.0000	10.5834	69.1760	85.8184	92.7163	79.1849	79.7703	90.2217
	Mean	1.88e-05	3.72e-05	1.24e-04	**1.71e-05**	3.58e-05	2.44e+05	7.2990	12.3905	24.9426	49.6333	49.3333	47.4667	6.1e+04	1.17e+05	2.63e+05	5.98e+04	1.11e+05	2.58e+05
F_{18}	Std	2.40e-06	3.64e-06	1.54e-05	**2.32e-06**	3.56e-06	1.95e+04	0.3099	0.2619	0.1443	29.7443	28.3930	26.2491	6.2e+03	7.04e+03	1.35e+04	5.35e+03	8.18e+03	1.34e+04
	Best	**1.18e-05**	3.06e-05	**9.74e-05**	1.30e-05	**2.85e-05**	1.64e+05	6.0804	11.6251	24.6365	7.0000	2.0000	7.2634	5.0e+04	1.01e+05	2.34e+05	5.16e+04	9.26e+04	2.22e+05
	Mean	**0**	**0**	**0**	1.12e-05	1.83e-05	0.6584	7.27e-12	2.44e-11	2.43e-10	57.0333	47.0000	57.9667	0.5073	0.7866	1.2144	0.3914	0.5916	0.7833
F_{19}	Std	**0**	**0**	**0**	7.34e-06	2.18e-05	0.1773	2.43e-11	7.07e-11	1.06e-09	29.3169	26.1745	29.1352	0.1953	0.2105	0.2970	0.1579	0.2120	0.2892
	Best	**0**	**0**	**0**	3.04e-06	2.74e-06	0.3121	1.29e-15	1.05e-17	9.24e-16	5.0000	6.0000	4.1489	0.2141	0.3618	0.5109	0.1166	0.2035	0.3337
	Mean	**0**	**0**	**0**	0.0108	0.0501	1.45e+03	6.84e-12	3.08e-11	1.37e-10	60.6333	47.6000	42.7000	97.1215	351.0299	1.76e+03	94.2383	352.5841	1.73e+03
F_{20}	Std	**0**	**0**	**0**	0.0028	0.0098	82.6457	1.48e-11	8.24e-11	1.67e-10	27.1363	28.6833	26.8330	20.5806	40.0370	128.0223	25.1356	42.1604	145.6030
	Best	**0**	**0**	**0**	0.0060	0.0353	1.26e+03	1.30e-14	2.72e-13	8.54e-13	4.0000	2.0000	2.6478	52.6845	242.5807	1.49e+03	39.0759	265.6255	1.37e+03
	Mean	**0**	**0**	**0**	27.593	30.1678	1.04e+19	4.67e-06	9.52e-05	1.41e-04	42.6667	55.9000	51.9667	2.0e+18	4.20e+18	1.15e+19	1.20e+18	3.13e+18	8.77e+18
F_{21}	Std	**0**	**0**	**0**	61.658	30.9669	1.02e+18	1.17e-05	2.58e-04	4.51e-04	30.1540	27.2831	27.2719	5.1e+17	7.56e+17	1.19e+18	3.36e+17	5.17e+17	8.74e+17
	Best	**0**	**0**	**0**	3.55e-10	2.60e-04	8.35e+18	1.05e-08	1.87e-09	4.27e-10	3.0000	1.0000	3.0000	1.2e+18	2.72e+18	9.10e+18	5.50e+17	1.90e+18	6.16e+18
	Mean	**0**	**0**	**0**	5.78e-04	0.0523	1.04e+13	2.9040	2.1789	1.5202	49.6333	46.8000	56.6667	2.2e+12	4.78e+12	1.06e+13	1.29e+12	3.13e+12	8.09e+12
F_{22}	Std	**0**	**0**	**0**	3.36e-04	0.1177	1.28e+12	9.4463	4.6905	3.8670	31.0722	25.0593	27.0763	7.2e+11	8.29e+11	1.32e+12	3.39e+11	6.06e+11	1.16e+12
	Best	**0**	**0**	**0**	5.68e-05	0.0019	8.49e+12	3.08e-05	0.0065	0.0045	4.000	4.0000	4.0000	4.6e+11	3.29e+12	8.45e+12	7.15e+11	1.73e+12	6.32e+12
	Mean	**0**	**0**	**0**	7.3502	34.9409	6.35e+05	6.8361	20.6234	106.6078	44.9000	49.9667	55.0333	1.9e+09	4.35e+09	1.24e+10	1.92e+09	4.67e+09	1.20e+10
F_{23}	Std	**0**	**0**	**0**	3.0310	18.2611	1.15e+09	13.4727	32.3738	283.6485	33.3481	27.1020	24.1711	5.5e+08	9.05e+08	1.72e+09	5.39e+08	8.48e+08	1.38e+09
	Best	**0**	**0**	**0**	2.6241	8.5878	4.15e+09	0.0477	0.1686	0.7108	1.0000	1.0004	8.0000	7.2e+08	2.88e+09	8.21e+09	7.43e+08	3.01e+09	8.44e+09
	Mean	-7.3e+3	-1.2e+04	**-7.5e+3**	-9.5e+3	-1.2e+04	-7.7e+3	-1.4e+4	-1.8e+04	-2.4e+04	50.2333	51.4333	51.2333	-2.5e+3	-3.4e+3	-4.8e+3	-3.0e+3	-3.7e+3	-5.2e+3
F_{24}	Std	688.978	864.1874	3.28e+03	1.4e+03	1.04e+03	663.3805	4.6e+03	6.96e+03	9.52e+03	34.1242	27.8271	**27.6564**	398.789	519.1752	784.2262	338.4343	660.1227	747.5053
	Best	-9.0e+3	-1.3e+04	-2.0e+04	-1.3e+4	-1.4e+04	-9.5e+3	-2.5e+4	-3.9e+04	-4.7e+04	1.0000	2.0000	4.0000	-3.4e+3	-4.8e+3	-6.5e+3	-3.6e+3	-5.0e+3	-7.2e+3
	Mean	53.1053	42.8653	**5.5849**	55.4092	145.7277	1.47e+03	0.0198	**0.0144**	0.0347	45.3000	51.6333	46.1000	429.669	760.5189	1.61e+03	401.7535	724.2319	1.57e+03
F_{25}	Std	16.2463	27.3546	30.5899	12.2011	32.1354	34.9524	0.0279	**0.0168**	0.0706	28.3806	29.4027	30.8817	23.1441	23.9390	32.2804	23.5300	32.8638	41.8353
	Best	**24.166**	**0**	**0**	31.2853	96.4649	1.38e+03	0.0012	0.0011	9.58e-04	4.0000	5.0006	1.0000	380.263	698.4247	1.53e+03	359.9953	637.3594	1.47e+03
	Mean	**2.47e-04**	**0**	**0**	0.0095	0.0040	2.19e+03	0.0030	0.0070	0.0118	49.7333	49.2000	47.3333	552.903	1.04e+03	2.37e+03	547.4418	1.03e+03	2.28e+03
F_{26}	Std	**0.0014**	**0**	**0**	0.0170	0.0085	107.1183	0.0034	0.0083	0.0129	27.2155	30.2511	31.5041	57.7185	76.7655	116.4482	40.0277	74.3003	137.5406
	Best	**0**	**0**	**0**	1.74e-08	2.54e-08	1.96e+03	1.08e-04	1.65e-04	3.22e-04	11.0000	1.0023	1.0000	390.003	910.0219	1.99e+03	475.3812	852.9620	1.94e+03
	Mean	9.98e-06	**0.1663**	**8.4507**	**8.47e-06**	3.1158	4.80e+09	3.0136	5.0142	10.0389	47.9333	49.8000	50.3000	9.1e+08	1.96e+09	4.83e+09	9.33e+08	1.88e+09	4.72e+09
F_{27}	Std	1.37e-06	**0.9025**	**0.5179**	1.22e-06	10.5150	5.80e+08	0.0215	0.0146	0.0502	32.7087	26.7626	31.4535	2.3e+08	2.73e+08	3.78e+08	1.68e+08	2.50e+08	4.10e+08
	Best	6.55e-06	1.61e-05	7.3399	5.80e-06	1.81e-05	2.61e+09	2.9978	4.9960	10.0025	1.0000	9.0156	2.0000	2.9e+08	1.40e+09	3.85e+09	5.45e+08	1.41e+09	3.80e+09
	Mean	**0.2173**	**0**	**0**	0.8713	6.2341	217.4529	0.0035	0.0065	0.0111	55.6000	49.9000	49.7667	58.4314	109.7408	244.9257	57.8071	107.0340	239.0229
F_{28}	Std	**0.2679**	**0**	**0**	0.8423	3.8194	11.6871	0.0018	0.0039	0.0068	31.9899	30.4284	30.9557	4.9286	5.3860	10.3221	6.0275	4.3744	9.3928
	Best	**0**	**0**	**0**	0.1833	0.9170	187.0470	5.04e-04	0.0018	0.0042	2.0000	1.0236	5.0000	43.6637	95.6545	221.5706	44.9686	97.6478	216.9315
	Mean	**0**	**0**	**0**	8.02e-05	3.20e-04	13.7524	2.33e-19	2.16e-17	8.96e-18	49.8000	48.4667	46.3000	2.7706	6.5041	16.5689	2.2901	5.9499	16.1513
F_{29}	Std	**0**	**0**	**0**	3.00e-05	1.24e-04	1.1678	3.80e-19	9.43e-17	2.60e-17	30.6520	31.3212	23.0474	0.6838	0.9021	1.4769	0.7110	0.9304	1.8898
	Best	**0**	**0**	**0**	1.72e-05	1.37e-04	11.3272	9.01e-23	6.22e-22	1.74e-21	1.0000	2.0689	1.0000	1.2097	4.8185	12.5471	0.7454	3.9522	12.0985
	Mean	9.32e-04	**0.0085**	0.6002	**4.96e-04**	0.0101	595.7277	0.7448	0.9248	1.0179	49.4667	62.8000	47.3667	154.853	290.2766	640.0197	150.9546	282.3473	627.7390
F_{30}	Std	3.38e-04	**0.0027**	0.0786	**1.27e-04**	0.0040	38.8617	0.1629	0.1066	0.1422	29.6773	27.8028	31.5611	13.0742	19.1373	35.4937	12.4249	21.4470	30.5029
	Best	3.32e-04	0.0044	0.4235	**2.30e-04**	0.0038	427.1223	0.3824	0.7790	0.7212	1.0000	14.2569	1.0029	131.533	247.2713	568.5226	127.7931	233.7856	551.1242

In order to evaluate the improvement in the algorithm's performance more comprehensively, Wilcoxon rank sum test is utilized to verify whether NWSHO has significant difference with other algorithms under the significant level of $p = 5\%$. Table 8 describes the results of experiment. When h-value is equal to 1 or the p-value is less than 5%, it indicates it is significantly different between NWSHO and the comparison algorithm. On the contrary, it is not different between the two algorithms.

In Table 8, NWSHO is significantly different from SHO and PSO on the most benchmark functions (except for F_{25}); Except for F_{28}, NWSHO and OPIO have obvious differences in the remaining functions; In general, NWSHO is very different from SHO, OPIO and PSO in most functions. And NWSHO is different from GA.

Table 8. The results of Wilcoxon rank sum test under 30 dimensions

fun		*SHO*	*OPIO*	*PSO*	*GA*	*BBO*
F_{16}	p	1.2118e-12	1.2118e-12	1.2108e-12	1.2118e-12	1.2118e-12
	h	1	1	1	1	1
F_{17}	p	1.2118e-12	1.2118e-12	1.2068e-12	1.2118e-12	1.2118e-12
	h	1	1	1	1	1
F_{18}	p	0.0051	3.0199e-11	3.0085e-11	3.0199e-11	3.0199e-11
	h	1	1	1	1	1
F_{19}	p	1.2118e-12	1.2118e-12	1.2098e-12	1.2118e-12	1.2118e-12
	h	1	1	1	1	1
F_{20}	p	1.2118e-12	1.2118e-12	1.2039e-12	1.2118e-12	1.2118e-12
	h	1	1	1	1	1
F_{21}	p	1.2118e-12	1.2118e-12	1.2078e-12	1.2118e-12	1.2118e-12
	h	1	1	1	1	1
F_{22}	p	1.2118e-12	1.2118e-12	1.2078e-12	1.2118e-12	1.2118e-12
	h	1	1	1	1	1
F_{23}	p	1.2118e-12	1.2118e-12	1.2098e-12	1.2118e-12	1.2118e-12
	h	1	1	1	1	1
F_{24}	p	2.6695e-09	2.3897e-08	3.0123e-11	3.0199e-11	3.0199e-11
	h	1	1	1	1	1
F_{25}	p	0.4290	3.0199e-11	0.2458	3.0199e-11	3.0199e-11
	h	0	1	0	1	1
F_{26}	p	1.4065e-11	3.3192e-11	1.7134e-12	1.7203e-12	1.7203e-12
	h	1	1	1	1	1
F_{27}	p	4.3531e-05	3.0199e-11	3.0066e-11	3.0199e-11	3.0199e-11
	h	1	1	1	1	1
F_{28}	p	1.5739e-06	0.6607	2.2507e-11	2.2623e-11	2.2623e-11
	h	1	0	1	1	1
F_{29}	p	1.2118e-12	1.2118e-12	1.2088e-12	1.2118e-12	1.2118e-12
	h	1	1	1	1	1
F_{30}	p	1.0277e-06	3.0199e-11	3.0142e-11	3.0199e-11	3.0199e-11
	h	1	1	1	1	1

6. **Conclusion.** For the standard SHO, the accuracy is low and the convergence is slow. To overcome the problems, a selfish herd optimization algorithm based on nonlinear inertia weight is proposed in this paper. First, the good point set was used instead of the randomization to initialize the population. Therefore, the problem of uneven distribution is solved; Second, the nonlinear inertia weight is applied into the formula of updating

position. While making a balance between global search and algorithm's ability of local development, the accuracy of solution and convergence are improved. This paper applies two sets of benchmark functions to carry out simulation experiments. The NWSHO is compared with other standard algorithms and the standard SHO. The results of experiment indicate the NWSHO is better than others in terms of solution accuracy and convergence.

In the future, the algorithm is mainly utilized to deal with practical issues, such as optimal parameter selection problem, multi-objective optimization problem, knapsack problem and image threshold segmentation problem.

Funding Statement: This work was supported by Jilin Science and Technology Innovation Development Program Projects (No. 20190302202).

REFERENCES

[1] H.-B. Dong, T. Li, R. Ding, and J. Sun, "A novel hybrid genetic algorithm with granular information for feature selection and optimization," *Applied Soft Computing*, vol. 65, pp. 33-46, 2018.

[2] Q.-M. Yan, R.-Q. Ma, Y.-X Ma, and J.-J. Wang, "An adaptive simulated annealing particle swarm optimization algorithm," *Journal of Xi'an University of Electronic Science and technology*, vol. 48, no. 4, pp. 120-127, 2021.

[3] Y. Li, W.-S. Mu, X.-Q. Chu, and Z.-T. Fu, "K-means clustering algorithm based on improved quantum particle swarm optimization and its application," *Control and Decision*, vol. 37, no. 4, pp. 839-850, 2022.

[4] T. Bai, Y.-B. Kan, J.-X. Chang, Q. Huang, and F.-J. Chang, "Fusing feasible search space into PSO for multi-objective cascade reservoir optimization," *Applied Soft Computing*, vol. 51, no. 2, pp. 328-340, 2018.

[5] H.-E. Tseng, C.-C. Chang, S.-C. Lee, and Y.-M. Huang, "Hybrid bidirectional ant colony optimization (hybrid BACO): An algorithm for disassembly sequence planning," *Engineering Applications of Artificial Intelligence*, vol. 83, pp. 45-56, 2019.

[6] J. Li, Y. Xia, B. Li, and Z. Zeng, "A pseudo-dynamic search ant colony optimization algorithm with improved negative feedback mechanism," *Cognitive Systems Research*, vol. 62, pp. 1-9, 2020.

[7] M.G.H. Omran and S. Al-Sharhan, "An improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 818-829, 2019.

[8] K. Anjaria and A. Mishra, "Thread scheduling using ant colony optimization: An intelligent scheduling approach towards minimal information leakage," *Karbala International Journal of Modern Science*, vol. 3, no. 4 pp. 241-258, 2017.

[9] Z.-Y. Liu, Z.-S. Liu, Z.-P. Zhu, Y.-D. Shen, and J.-W. Dong, "Simulated annealing for a multi-level nurse rostering problem in hemodialysis service," *Applied Soft Computing*, vol. 64, no. 3, pp. 148-160, 2018.

[10] M.A. Al-Betar, M.A. Awadallah, H. Faris, I. Aljarah, and A.I. Hammouri, "Natural selection methods for Grey Wolf Optimizer," *Expert Systems with Applications*, vol. 113, pp. 481-498, 2018.

[11] U. Mohan, S. Ramani, and S. Mishra, "Constant factor approximation algorithm for TSP satisfying a biased triangle inequality," *Theoretical Computer Science*, vol. 657, pp. 111-126, 2017.

[12] Y.-M. Xu, K.-L. Li, J.-T. Hu, and K.-Q. Li, "A Genetic Algorithm for Task Scheduling on Heterogeneous Computing Systems Using Multiple Priority Queues," *Information Sciences*, vol. 270, pp. 255-287, 2014.

[13] Y.-M. Xu, K.-L. Li, L.-G. He, L.-X. Zhang, and K.-Q. Li, "A Hybrid Chemical Reaction Optimization Scheme for Task Scheduling on Heterogeneous Computing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3208-3222, 2015.

[14] J.-S. Baxter, E. Gibson, R. Eagleson, and T.-M. Peters, "The semiotics of medical image Segmentation," *Medical Image Analysis*, vol. 44, pp. 54-71, 2018.

[15] F. Fausto, E. Cuevas, and Valdivia A, "A global optimization algorithm inspired in the behavior of selfish herds," *BioSystems*, vol. 160, pp. 39-55, 2017.

[16] W.-D. Hamilton, "Geometry for the selfish herd," *Journal of Theoretical Biology*, vol. 31, no. 2, pp. 295-311, 1971.

[17] X. Lv, X.-D. Mu, and J. Zhang, "Multi-threshold image segmentation based on improved sparrow search algorithm," *System Engineering and Electronics*, vol. 43, no. 2, pp. 318-327, 2021.

[18] Q.-H. Mao, and Q. Zhang, "Improved Sparrow Algorithm Combining Cauchy mutation and opposition-based learning," *Journal of Frontiers of Computer Science and Technology*, vol. 15, no. 6, pp. 1155-1164, 2021.

[19] S. Amirsadri, S.-J. Mousavirad, and H. Ebrahimpour-Komleh, "A Levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training," *Neural Computing and Applications*, vol. 30, no. 12, pp. 3707-3720, 2018.

[20] S. Arora, and P. Anand, "Chaotic grasshopper optimization algorithm for global optimization," *Neural Computing and Applications*, vol. 31, no. 8, pp. 4385-4405, 2019.

[21] O.-P. Verma, D. Aggarwal, and T. Patodi, "Opposition and dimensional based modified firefly algorithm," *Expert Systems with Applications*, vol. 44, pp. 168-176, 2016.

[22] L.-L. Kang, R.-S. Chen, N.-X. Xiong, Y.-C. Chen, Y.-X. Hu, and C.-M. Chen, "Selecting Hyper-Parameters of Gaussian Process Regression Based on Non-Inertial Particle Swarm Optimization in Internet of Things", *IEEE Access*, vol. 7, pp. 59504-59513, 2019.

[23] P. Anand and S. Arora, "A novel chaotic selfish herd optimizer for global optimization and feature selection," *Artificial Intelligence Review*, vol. 53, pp. 1441-1486, 2020.

[24] R. Zhao, Y. Wang, and C. Liu, "Selfish herd optimizer with levy-flight distribution strategy for global optimization problem," *Physica A: Statistical Mechanics and its Applications*, vol. 538, 122687, 2020.

[25] Y. B. Tian and R.-J. Zhu, "Research advances on inertia weight in particle swarm optimization," *Computer Engineering and Application*, vol. 44, no. 23, pp. 39-41, 2008.

[26] A. Mazahery, M.-O. Shabani, M. Alizadeh and A.-A. Tofigh, "Concurrent fitness evaluations in searching for the optimal process conditions of AI matrix nanocomposites by linearly decreasing weight," *Journal of Composite Materials*, vol. 47, no. 14, pp. 1765-1772, 2013.

[27] M.O. Shabani and A. Mazahery, "Application of a linearly decreasing weight particle swarm to optimize the process conditions of al matrix nanocomposites," *Metallurgist*, vol. 56, no. 5, pp. 414-422,2012.

[28] C.-H. Yang, C.-J. Hsiao and L.-Y. Chuang, "Linearly Decreasing Weight Particle Swarm Optimization with Accelerated Strategy for Data Clustering," *IAENG International Journal of Computer Science*, vol. 37, no. 3, pp. 234-241,2010.

[29] G.M. Chen, J.Y. Jia and Q. Han, "Study on the Strategy of Decreasing Inertia Weight in Particle Swarm Optimization Algorithm," *Journal-Xian Jiaotong University*, vol. 40, no. 1, pp. 53-56, 2006.

[30] X.-J. Lei, A. Fu, and J.-J. Sun, "Performance analysis and research of improved PSO algorithm," *Computer Application Research*, vol. 27, no. 2, pp. 453-458, 2010.

[31] Z.-J. Teng, J.-L. Lv, and L.-W. Guo, "Research on particle swarm optimization algorithm based on dynamic acceleration factor," *Microelectronics and Computer*, vol. 34, no. 12, pp. 125-129, 2017.

[32] L.-S. Li, and X.-J. Zhang, "A new chaos PSO algorithm with adaptive inertia weight," *Computer Engineering and Application*, vol. 54, no. 9, pp. 139-144, 2018.

[33] Z.-G. Zhao, S.-Y. Huang, and W.-Q. Wang, "A simplified particle swarm optimization algorithm based on random inertia weight," *Computer application Research*, vol. 31, no. 2, pp. 361-363, 2014.

[34] Z.-H. Gao, L. Mei, and Y.-J. Zhu, "Particle swarm optimization algorithm based on compound strategy inertia weight," *Computer Application*, vol. 32, no. 8, pp. 2216-2218, 2012.

[35] P. Chauhan, K. Deep and M. Pant, "Novel inertia weight strategies for particle swarm optimization," *Memetic Computing*, vol. 5, no. 3, pp. 229-251, 2013.

[36] H.-B. Dong, D.-J. Li, and X.-P. Zhang, "A particle swarm optimization algorithm for dynamically adjusting inertia weight," *Computer Science*, vol. 45, no. 2, pp. 98-139, 2018.

[37] X.-F. Yang and S.-L. Liu, "Dynamic adjustment strategies of inertia weight in particle swarm optimization algorithm," *International Journal of Control and Automation*, vol. 7, no. 5, pp. 353-364, 2014.

[38] Y.-C. Ao, Y.-B. Shi, and W. Zhang, "Improved particle swarm optimization with adaptive inertia weight," *Journal of University of Electronic Science and technology*, vol. 43, no. 6, pp. 874-880, 2014.

[39] M.-S. Malik, E.-R. Mohideen, and L. Ali, "Weighted distance Grey wolf optimizer for global optimization problems," *IEEE International Conference on Computational Intelligence and Computing Research(ICCIC)*, pp. 1-6, 2015.

[40] H. Hu, Y. Bai, and T. Xu, "Improved whale optimization algorithms based on inertia weights and theirs applications," *International Journal of Circuits, Systems and Signal Processing*, vol. 11, pp. 12-26, 2017.

[41] K.-A. Rani, W.-F. Hoon, M.-A. Malek, N.-M. Affendi, L. Mohamed, and N. Saudin, "Modified cuckoo search algorithm in weighted sum optimization for linear antenna array synthesis", *IEEE Symposium on Wireless Technology Applications (ISWTA)*, pp. 210-215, 2012.

[42] L.-G. Hua, and Y. Wang, *Application of number theory in modern analysis*, Science Press, Beijing, 1978.

[43] W. Long, J. Jiao, X. Liang, and M. Tang, "An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization," *Engineering Applications of Artificial Intelligence*, vol. 68, pp. 63-80, 2018.

[44] Y. Sun, T. Yang, and Z. Liu, "A whale optimization algorithm based on quadratic interpolation for high-dimensional global optimization problems," *Applied Soft Computing*, vol. 85, 105744, 2019.

[45] R.-S.-M. de Barros, J.-I.-G. Hidalgo, and D.-R. de Lima Cabral, "Wilcoxon Rank Sum Test Drift Detector," *Neurocomputing*, vol. 275, pp. 1954-1963, 2018.

[46] W. Long, J. Jiao, X. Liang, T. Wu, M. Xu, and S. Cai, "Pinhole-imaging-based learning butterfly optimization algorithm for global optimization and feature selection," *Applied Soft Computing*, vol. 103, 107146, 2021.

[47] H.-X. Lu, S.-Y. Yin, G.-L. Gong, Y. Liu, and G. Chen, "Particle swarm optimization algorithm based on depth deterministic strategy gradient," *Journal of University of Electronic Science and Technology*, vol. 50, no. 2, pp. 199-206, 2021.

[48] H.-P. Pan, H.-B. Ding, M.-Z. Lei, L.Q. Wang, "Maximum power tracking control of direct drive ocean wave power generation system based on genetic algorithm," *Journal of Solar Energy*, vol. 42, no. 3, pp. 221-227, 2021.

[49] X.-Q. Zhang, A.M. Gong, X.Q. Xu, C.C. Luo, F. Xie, "Research on searching sliding surface of homogeneous soil slope based on improved biogeographic algorithm," *Henan Science*, vol. 39, no. 2, pp. 276-281, 2021.