symmetry

MDPI

Article

A Hybrid Multi-Step Probability Selection Particle Swarm Optimization with Dynamic Chaotic Inertial Weight and Acceleration Coefficients for Numerical Function Optimization

Yuji Du * and Fanfan Xu

School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai 264209, China; 201700800753@mail.sdu.edu.cn

* Correspondence: 201700800702@mail.sdu.edu.cn; Tel.: +86-178-5262-1086

Received: 23 March 2020; Accepted: 6 May 2020; Published: 2 June 2020

Abstract: As a meta-heuristic algoriTthm, particle swarm optimization (PSO) has the advantages of having a simple principle, few required parameters, easy realization and strong adaptability. However, it is easy to fall into a local optimum in the early stage of iteration. Aiming at this shortcoming, this paper presents a hybrid multi-step probability selection particle swarm optimization with sine chaotic inertial weight and symmetric tangent chaotic acceleration coefficients (MPSPSO-ST), which can strengthen the overall performance of PSO to a large extent. Firstly, we propose a hybrid multi-step probability selection update mechanism (MPSPSO), which skillfully uses a multi-step process and roulette wheel selection to improve the performance. In order to achieve a good balance between global search capability and local search capability to further enhance the performance of the method, we also design sine chaotic inertial weight and symmetric tangent chaotic acceleration coefficients inspired by chaos mechanism and trigonometric functions, which are integrated into the MPSPSO-ST algorithm. This strategy enables the diversity of the swarm to be preserved to discourage premature convergence. To evaluate the effectiveness of the MPSPSO-ST algorithm, we conducted extensive experiments with 20 classic benchmark functions. The experimental results show that the MPSPSO-ST algorithm has faster convergence speed, higher optimization accuracy and better robustness, which is competitive in solving numerical optimization problems and outperforms a lot of classical PSO variants and well-known optimization algorithms.

Keywords: particle swarm optimization; multi-step; roulette wheel selection; chaotic inertial weight; symmetric chaotic acceleration coefficients; numerical function optimization

1. Introduction

With the development of scientific research, engineering technology and social economy, optimization issues have gradually become high dimensionality, high level and great difficulty. Conventional optimization has become increasingly unsuitable for dealing with these optimization problems. In order to adapt to the upgrading of optimization problems better, people have a very urgent need for the upgrading of optimization technology. In recent years, the development of optimization algorithms has been a very prevalent research direction. In optimization algorithms, meta-heuristic optimization methods play a vital role, such as particle swarm optimization (PSO) [1] grey wolf optimizer (GWO) [2] the whale optimization algorithm (WOA) [3], differential evolution (DE) [4] gravitational search algorithm (GSA) [5] moth-flame optimization (MFO) [6] biogeography-based optimization (BBO) [7] sine cosine algorithm (SCA) [8] krill herd algorithm (KH) [9] artificial bee colony (ABC) [10] ant lion optimizer (ALO) [11]

Among these well-known optimization algorithms, the PSO algorithm has attracted considerable attention as a classic swarm intelligence algorithm. The PSO algorithm is inspired by the predation behavior of birds. Since its introduction in 1995, it has undergone many improvements to form many PSO variants, which are now widely used in various fields of science and society, such as feature selection [12] artificial intelligence [13] wireless sensor network [14] energy management system [15] public resource construction [16] and so on.

The PSO algorithm has the advantages of having a simple principle, few required parameters, fast convergence and easy realization. However, the search direction of PSO is to approach the global optimum, which makes the information exchange in the group in a single direction, so that the particles quickly gather in a small search area, resulting in the poor swarm diversity [17], which makes it easy to fall into local extremes and get poor convergence accuracy. The swarm diversity information can reflect the distribution of the entire particle swarm. The lack of swarm diversity will cause the swarm to converge prematurely to some local optimums. At present, the two most widely used measures of swarm diversity are population distribution entropy [18] and population average particle distance [19] The algorithm performs cluster analysis on the individuals in the population, and then the population distribution entropy can be obtained, which reflects the particle distribution of the population in each area of the search space. For each evolutionary generation of the population, the average particle distance of all individuals in the entire population must be calculated, which expresses the dispersion degree of the individual in the population.

To overcome the above shortages, researchers have conducted a lot of research and made improvements on PSO to optimize the algorithm performance. The first direction is an updated mechanism improvement, such as comprehensive learning PSO (CLPSO) [20] orthogonal multi-swarm cooperative PSO [21] enhanced particle swarm optimization with levy flight (PSOLF) [22] hybrid chaotic quantum behaved particle swarm optimization algorithm (HCQPSO) [23] Qin et al. [24] proposed a deep-learning-driven PSO algorithm (DLD-PSO). It introduced the matrix of weights which are extracted from the deep learning prediction model into the PSO update mechanism. These improved mechanisms can effectively speed up the convergence speed and improve the search ability of the method. The second direction is parameter improvement, that is, improving the inertia weight and acceleration coefficients. If the global searching capability is strong, the convergence speed would be fast, and it is less likely to be limited by the local minimum, but the convergence accuracy is low. The strong local search capability is the opposite [25] To find a balance between global search and local search, the values of inertia weight and acceleration coefficient are particularly important. In [26] Tian et al. introduced sigmoid-based acceleration coefficients and established an appropriate ratio between exploration and exploitation, which successfully achieved a balance between global search and local search. In [27] Arasomwan et al. introduced chaos mechanism and adaptive strategy to the inertia weight, which used the regularity, randomness and ergodicity of chaos and avoided premature convergence. In [28] Taherkhani et al. obtained different adaptive inertial weight based on the optimal position and distance in particle history, which improved the convergence accuracy and speed of the algorithm. The third direction is to combine PSO with other algorithms. Zhang et al. [29] introduced the DE-PSO algorithm, which combined PSO and DE to realize the algorithm using a differential operator and enrich the swarm diversity. Garg [30] proposed a hybrid PSO-GA algorithm by incorporating genetic operators into PSO. The balance between exploration and exploitation abilities have been further improved. Javidrad et al. [31] used the simulated annealing algorithm (SA) as a local search mechanism, which improves the convergence behavior of PSO, forming a PSO-SA hybrid algorithm.

Although these PSO variants inherit the advantages of PSO and overcome some shortcomings of PSO, which makes them increasingly have advantages over conventional optimization, the deficiencies of being easily trapped in a local optimal solution and lacking swarm diversity still exist [32] which leads to the unsatisfactory in addressing complex optimization problems with different characteristics. The above methods only improve one aspect of PSO, like convergence speed, premature convergence

and the balance between exploration and exploitation. Therefore, these methods are unable to deal with more complicated optimization problems.

In this paper, a hybrid multi-step probability selection particle swarm optimization with sine chaotic inertial weight and symmetric tangent chaotic acceleration coefficients called MPSPSO-ST is proposed. MPSPSO-ST is a comprehensively improved algorithm that can modify many aspects of PSO for numerical function optimization as follows:

1. The multi-step probability selection process can enhance the search ability of particles and avoid premature convergence, which also has a positive effect on convergence speed.
2. The sine chaotic ω and symmetric tangent chaotic c_1, c_2 enrich the swarm diversity and achieve a better balance between the exploration and exploitation ability, which offers higher convergence accuracy.

The remainder of this paper is organized as follows: Section 2 presents the related theory about PSO. Section 3 describes the proposed MPSPSO-ST algorithm in detail. In Section 4, several well-known optimization methods and PSO variants are adopted to verify the performance of MPSPSO-ST in numerical optimization. Finally, Section 5 consists of the conclusions drawn in this paper and the summary of future work.

2. Related Theory about PSO

PSO is a kind of swarm intelligence optimization algorithm, which is derived from the study of bird swarm predation behavior. Each particle in the particle swarm represents a possible solution to a problem. Through the simple behavior of individual particles and the interaction of information within the group, the intelligence of solving problems is achieved.

PSO is a population-based search algorithm. An individual in the bird swarm is abstracted into a particle. Each particle searches for the optimal solution in the search space separately. The optimal solution that each particle can currently search for is recorded as the current individual optimal value, and then the individual optimal value is shared with other particles in the entire particle swarm. The optimal individual extreme value is found as the current global maximum of the whole particle swarm. All particles in the particle swarm continually update their positions and speeds based on the current individual extremes found by themselves and the current global optimal solution shared by the entire particle swarm. The particle will gradually approach the optimal position and finally find the global optimum during the iterative process.

When searching in D-dimensional target search space, each particle has two parameters, position and velocity. The position of the ith particle is represented as $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, $i = (1, 2, \ldots, N)$, where N is the number of population. The velocity of the ith particle is represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, $i = (1, 2, \ldots, N)$. The velocity and position of the particles are iterated according to the following iterative equations:

$$v_i^{(t+1)} = v_i^t + c_1 \cdot r_1 \cdot (pbest_i^t - x_i^t) + c_2 \cdot r_2 \cdot (gbest^t - x_i^t) \quad (1)$$

$$x_i^{(t+1)} = x_i^t + v_i^{(t+1)} \quad (2)$$

x_i^t indicates the position of the tth iteration of the ith particle. v_i^t denotes the speed of the tth iteration of the ith particle.c_1 and c_2 are two positive constants, generally $c_1 = c_2 = 2$; r_1 and r_2 are random numbers in the interval [0, 1]. Each particle is evaluated for fitness value $f(x_i)$ by objective function f. Each particle position represents a solution to the problem, which has a fitness value $f(x_i)$, given by the objective function f. During each iteration process, the particles memorize the position with the best fitness value by comparing the fitness value of each position and update $pbest$, $gbest$. $pbest_i^t$ is the current individual optimal position of the particle and $gbest^t$ is the current global optimal position of the entire particle swarm. Self-cognition component term $c_1 \cdot r_1 \cdot (pbest_i^t - x_i^t)$ pushes the particles to move toward their own best positions found so far. Social cognitive component term $c_2 \cdot r_2 \cdot (gbest^t - x_i^t)$

encourages the particles to move toward the global best position found currently. $v_i \in [-v_{\max}, v_{\max}]$, and $v_{\max}$ is a constant. $v_{\max}$ often set to four in practice [33]. When the particle velocity exceeds the interval, let the particle velocity v_i be equal to the upper or lower bound of the interval ($-v_{\max}$ or $v_{\max}$) so that the particle velocity is controlled within a reasonable range. Equations (1) and (2) are called the basic PSO algorithm.

In [34] Shi and Eberhart introduced inertial weight ω into the memory term in Equation (1) and found that it can balance the global search ability and local search ability better. The improved iterative equations are formulated as follows:

$$v_i^{(t+1)} = \omega \cdot v_i^t + c_1 \cdot r_1 \cdot (pbest_i^t - x_i^t) + c_2 \cdot r_2 \cdot (gbest^t - x_i^t) \tag{3}$$

$$x_i^{(t+1)} = x_i^t + v_i^{(t+1)} \tag{4}$$

where ω is a linear decreasing weight strategy defined as follows:

$$\omega = \omega_{\max} - \frac{M_j}{M_{\max}} \cdot (\omega_{\max} - \omega_{\min}) \tag{5}$$

where $\omega_{\min}$ and $\omega_{\max}$ are the initial and final values of the inertial weight, respectively, M_j is the current iteration number and $M_{\max}$ is the maximum iteration number.

Since the inertial weight ω improves the performance significantly, scholars take it as the standard PSO algorithm, which has become the basis of the current research and improvement of PSO. The pseudo-code of the standard PSO algorithm is shown in Figure 1. Inspired by this linear decreasing ω, some articles have proposed different strategies on inertial weight [28,35].

```
Initialize the parameters (N, D, Vmin, Vmax, Max_iter, Xmin, Xmax)
Randomly generate N initial positions X_i(i=1,2,...,N) of N particles within the maximum range
Randomly generate N initial velocities V_i(i=1,2,...,N) of N particles within the maximum range
Calculate the fitness value of each particle using the objective function f
Set pbest and gbest in the swarm
While iter < Max_iter do
    Using Equation (5) to update the inertia weight ω
    for i =1:N(number of particles) do
        Update the velocity V_i of the i th particle using Equation (3)
        Update the position X_i of the i th particle using Equation (4)
        Calculate the fitness values of the new particle X_i using the objective function f
        if X_i is superior to pbest_i
            Set X_i to be pbest_i
        End if
        if X_i is superior to gbest
            Set X_i to be gbest
        End if
    End for i
    iter=iter+1
End While
```

Figure 1. Pseudo-code of the standard particle swarm optimization (PSO) algorithm.

3. Hybrid Multi-Step Probability Selection Particle Swarm Optimization with Sine Chaotic Inertial Weight and Symmetric Tangent Chaotic Acceleration Coefficients (MPSPSO-ST)

The PSO algorithm currently has shown great effectiveness on various optimization problems, but it still has a drawback as lacking diversity in the search process. Due to premature convergence, the search is prone to stagnate and fall into local optimal solutions [36,37]. Based on the in-depth study of PSO, this paper proposes three modifications to improve the performance of PSO.

3.1. Hybrid Multi-Step Probability Selection Particle Swarm Optimization (MPSPSO)

According to Equation (3), the particle velocity updated in the PSO algorithm is determined by three terms: inertia term v_i^t, self-cognition component term $c_1 \cdot r_1 \cdot (pbest_i^t - x_i^t)$ and social cognitive component term $c_2 \cdot r_2 \cdot (gbest^t - x_i^t)$. Inspired by human behavior, when humans do things, they sometimes perform habitually (the velocity update only includes inertia term v_i^t, as shown in Equation (6)), sometimes they sum up their own experience (the velocity update includes inertia term v_i^t and self-cognition component term $c_1 \cdot r_1 \cdot (pbest_i^t - x_i^t)$, as shown in Equation (7)), and sometimes they summarize all the information and then do things (the basic PSO algorithm is this way, as shown in Equation (1)) [38].

Previous studies have shown that the length and direction of v_i^t are coupled with $pbest_i^t$ and $gbest^t$, resulting in a slow update of $pbest_i^t$ [39]. Gao et al. [38] proposed a variant named PSO-MP to decompose the single-step velocity update formula in the standard PSO into the three-steps update and then choose the best one. This method that updating step-by-step first and then selecting the optimal value to update can achieve decoupling of v_i^t, $pbest_i^t$ and $gbest^t$, respectively. Thereby, the search ability of this algorithm is enhanced.

In order to increase the diversity and randomness of the particle population, this paper proposes a PSO variant using hybrid multi-step probability selection, namely MPSPSO. The main idea of this variant is to decompose the single-step velocity update formula in the standard PSO into four velocity update formulas. In the early iterations, we directly select the one with the best fitness value of four velocity update formulas. While in the later iterations, we probabilistically select one of the four velocity update formulas using the roulette wheel selection mechanism according to the fitness value. Finally, the optimal solution is obtained. This algorithm can not only achieve the decoupling of v_i^t, $pbest_i^t$ and $gbest^t$ but also enhance the randomness and diversity of optimization, thereby effectively avoiding falling into local optimums, so that it can improve the search accuracy and efficiency. The detailed introduction about MPSPSO is presented as follows.

1. Calculate the particle velocity and position by Equations (3) and (4), where x_i^{t+1} is the vector sum with x_i^t, $\omega \cdot v_i^t$, $c_1 \cdot r_1 \cdot (pbest_i^t - x_i^t)$ and $c_2 \cdot r_2 \cdot (gbest^t - x_i^t)$, as shown in Figure 2.

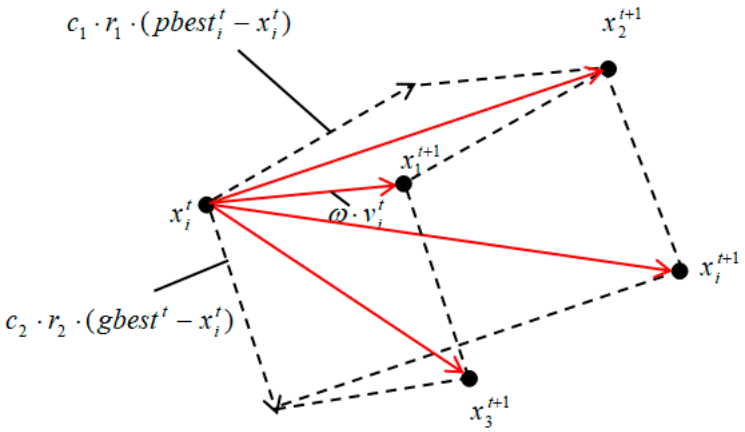

Figure 2. Position updating of the particle.

As can be seen from Figure 2, in the standard PSO, only x_i^t and x_i^{t+1} are used to update the particle position. x_1^{t+1}, x_2^{t+1} and x_3^{t+1} are used as temporary points only for the generation of x_i^{t+1}. These three points may be better than others, but they are ignored. Therefore, the velocity update Equation (3) is divided into four steps in this paper, and three temporary points are iterative objects which can be selected. The equations are described as follows:

$$v_1^{t+1} = \omega \cdot v_i^t, x_1^{t+1} = x_i^t + v_1^{t+1} \tag{6}$$

$$v_2^{t+1} = v_1^{t+1} + c_1 \cdot r_1 \cdot (pbest_i^t - x_i^t), x_2^{t+1} = x_i^t + v_2^{t+1} \tag{7}$$

$$v_3^{t+1} = v_1^{t+1} + c_2 \cdot r_2 \cdot (gbest^t - x_i^t), x_3^{t+1} = x_i^t + v_3^{t+1} \tag{8}$$

$$v_4^{t+1} = v_1^{t+1} + c_1 \cdot r_1 \cdot (pbest_i^t - x_i^t) + c_2 \cdot r_2 \cdot (gbest^t - x_i^t), x_4^{t+1} = x_i^t + v_4^{t+1} \tag{9}$$

2. If iteration $< 0.9 \times t_{\max}$($t_{\max}$ is the maximum iteration number), choose the position with the best fitness given by the objective function f (in this paper, we all solve $\min f(x)$ problem, so the smaller $f(x)$ is, the better the fitness is) among the above equations as the final position x_i^{t+1}. The corresponding velocity is taken as the final velocity v_i^{t+1}.

The process is described as follows:

$$v_i^{t+1} = \begin{cases} v_1^{t+1}, & if \min\left\{f(x_1^{t+1}), f(x_2^{t+1}), f(x_3^{t+1}), f(x_4^{t+1})\right\} = f(x_1^{t+1}) \\ v_2^{t+1}, & if \min\left\{f(x_1^{t+1}), f(x_2^{t+1}), f(x_3^{t+1}), f(x_4^{t+1})\right\} = f(x_2^{t+1}) \\ v_3^{t+1}, & if \min\left\{f(x_1^{t+1}), f(x_2^{t+1}), f(x_3^{t+1}), f(x_4^{t+1})\right\} = f(x_3^{t+1}) \\ v_4^{t+1}, & otherwise \end{cases} \tag{10}$$

$$x_i^{t+1} = \begin{cases} x_1^{t+1}, & if \min\left\{f(x_1^{t+1}), f(x_2^{t+1}), f(x_3^{t+1}), f(x_4^{t+1})\right\} = f(x_1^{t+1}) \\ x_2^{t+1} & if \min\left\{f(x_1^{t+1}), f(x_2^{t+1}), f(x_3^{t+1}), f(x_4^{t+1})\right\} = f(x_2^{t+1}) \\ x_3^{t+1} & if \min\left\{f(x_1^{t+1}), f(x_2^{t+1}), f(x_3^{t+1}), f(x_4^{t+1})\right\} = f(x_3^{t+1}) \\ x_4^{t+1} & otherwise \end{cases} \tag{11}$$

The rationality and effectiveness of this step-by-step process are reflected in the refinement of the particle search trajectory. Firstly, the particles move by Equation (6) according to the speed and inertia of the previous step. If the search direction in the previous step is a direction that is closer to the best solution, then this position may be a better point. On the basis of x_1^{t+1}, move along the individual extreme value direction to x_2^{t+1} by Equation (7) to refine the local search; move to x_3^{t+1} along the global extreme value direction by Equation (8) to refine the global search; or comprehensively considering the individual extreme value and the global extreme value, move to x_4^{t+1} along the individual extreme value direction and the global extreme value direction by Equation (9) as the standard PSO. Such a step-by-step search operation can take the temporary points that were initially ignored as optional objects and achieve greater use of potential information so that the algorithm has multi-selected opportunities to find the optimal solution more efficiently.

3. After iterating to 90% of $t_{\max}$, according to the fitness of the four positions from Equations (6)–(9), given by the objective function f, we probabilistically select one of the four positions as the update particle position x_i^{t+1} of this generation using a roulette wheel selection mechanism. The particle velocity corresponding to this position is taken as the update particle velocity v_i^{t+1}. Roulette wheel selection [40] is widely used in improving various population-based algorithms such as GA [41] DE [42] which all achieve satisfactory results. Inspired by this, we propose the above idea.

Roulette wheel selection mechanism, also known as a proportional selection method, the basic idea of which is that the probability of an individual being selected is proportional to its fitness, that is, the size of the objective function value. The smaller the objective function value of the particle position

is, the better the particle fitness is, the higher the probability of being selected is. The specific operations are as follows:

- Calculate the objective function value $f(i = 1, 2, 3, 4)$ of each of the four particle positions.
- Due to it that the smaller $f(x)$ is, the better the fitness of the position is, so the probability of being selected should be inversely proportional to the function value of the position. From this, the probability of each position being selected is calculated as follows:

$$P(x_i) = 1 - \frac{f(x_i)}{\sum_{j=1}^{4} f(x_j)} \tag{12}$$

- Calculate the cumulative probability of each position:

$$q_i = \sum_{j=1}^{i} P(x_j) \tag{13}$$

The cumulative probability diagram is in Figure 3 as follows:

- Randomly generate a uniformly distributed random number r in the interval [0, 1].
- When r satisfies $r < q_1$, select position x_1^{t+1}; when r satisfies $q_{k-1} < r \leq q_k$, select position x_k^{t+1}.

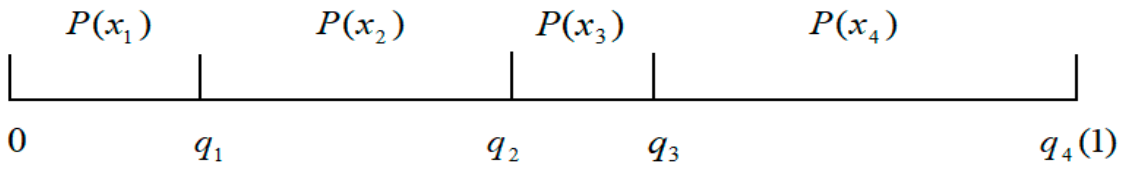

Figure 3. The diagram of roulette wheel selection cumulative probability.

After iterating to 90% of $t_{\max}$, the reason for using the roulette wheel selection mechanism is that the whole swarm are now close to the global optimum, but the swarm diversity in the later iterations is poor, and the point with the best fitness in this generation may not always approach the best solution quickly after iterative calculations, so the roulette wheel selection mechanism is used at the later stage of the iteration to select the final update point x_i^{t+1}. Possibility of selecting good particle positions for this generation is high but not absolutely and other particle positions may also be selected, which makes the choice of search direction more diverse. This operation enriches the swarm diversity and ensures that the ability to jump out of local extreme value is effectively modified.

Similar to the next iteration, each particle will modify its speed and position within each iteration until the optimal solution is found or all iterations are terminated. MPSPSO successfully avoids premature convergence and greatly increases the possibility of finding the optimal solution.

3.2. Sine Chaotic Inertia Weight ω

The inertia weight plays a vital role in the convergence behavior of PSO [34,43]. Chaos is a nonlinear dynamic phenomenon, which has regularity, randomness and ergodicity. Due to these characteristics, chaos has become a popular optimization method today. Optimizing the search by chaos mechanism is superior to stochastic search [44,45]. It has the ability not to repeatedly go through all states in a certain range, which can meet the needs of ω for different functions to the greatest extent [46]. The authors of [47] compared 15 classic methods of inertial weight, and the result indicates that chaotic inertial weight is the best strategy to improve accuracy.

Previous research has shown that the sine function plays a great role in the adjustment of ω [48,49]. Combining the sine function and chaos mechanism, this paper uses a sine iterator to directly generate the sequence to achieve the sine chaotic inertial weight.

A sine iterator is a type of chaotic sequence generator. Mathematically, its basic form is as follows:

$$X_{n+1} = ax_n^2 \sin(\pi x_n) \tag{14}$$

When $a = 2.3$, $X_0 = 0.7$, its simple form is as follows:

$$X_{n+1} = \sin(\pi x_n) \tag{15}$$

It generates chaotic sequences in (0, 1).

Therefore, this paper proposes the following update form of ω:

$$\omega_{t+1} = \phi \times \sin(\pi \omega_t) + \tau \tag{16}$$

The initial value ω_1 is an arbitrary number in [0, 1], and ϕ, τ are constants.

In this paper, we take $\phi = 0.9$, $\tau = 0$. With the progress of the iterative process, ω can traverse almost all values in the interval of [0.2, 0.9] approximately. The proposal of sine chaotic inertial weight can enhance the global exploration of the algorithm and strengthen the ability of avoiding local optimum.

3.3. Symmetric Tangent Chaotic Acceleration Coefficients c_1, c_2

The cognitive component c_1 and social component c_2 are very important for the algorithm to find the optimal solution accurately and efficiently. Usually, the cognitive component and social component are given equal weight $c_1 = c_2 = 2$. With a large cognitive component and a small social component at the beginning, particles are allowed to move around the search space during the early stages. On the other hand, a small cognitive component and a large social component allow the particles to converge to the global optimum in the latter part of the optimization process [50]. Logistic mapping in the chaos mechanism has attracted much attention owing to better optimization performance, such as randomness and ergodicity. Based on the above two points, this paper proposes symmetric tangent chaotic acceleration coefficients.

The formula of Logistic mapping is $z = \mu \times z \times (1 - z)$, and chaos performance is best when $\mu = 4$. This paper firstly proposes the symmetric tangent acceleration coefficient, and then introduces the chaotic term. The equations of the symmetric tangent acceleration coefficients are as follows:

$$c_1 = -\partial \times m^2 \times \tan[\frac{\pi}{8} \times (1 + m^2)] + \theta \tag{17}$$

$$c_2 = -\partial \times (1 - m)^2 \times \tan[\frac{\pi}{8} \times (1 + (1 - m)^2)] + \theta \tag{18}$$

where $m = \frac{t}{t_{\max}}$, t is the current number of iterations, $t_{\max}$ is the maximum number of iteration, and ∂, θ are constants.

Then add chaotic terms to Equations (17) and (18): firstly, take a random number in (0,1), then generate the Logistic mapping sequence. We propose the symmetric tangent chaotic acceleration coefficients, which are defined by Equations (19) and (20).

$$c_1 = -\partial \times m^2 \times \tan[\frac{\pi}{8} \times (1 + m^2)] + \theta + \rho \times z \tag{19}$$

$$c_2 = -\partial \times (1 - m)^2 \times \tan[\frac{\pi}{8} \times (1 + (1 - m)^2)] + \theta + \rho \times z \tag{20}$$

where ρ is a constant.

This paper takes $\partial = 0.2$, $\theta = 1.5$ and $\rho = 0.1$. The proposal of the symmetric tangent acceleration factor enables the algorithm to balance the global search in the early stage of iteration and the local search in the later stage of iteration better. Meanwhile, the proposal of chaotic terms makes the acceleration coefficients have chaotic characteristics while maintaining the original change trend.

Summarizing the above description, the pseudo-code of MPSPSO-ST proposed in this paper is shown in Figure 4:

```
Initialize the parameters (N, D, Vmin, Vmax, Max_iter, Xmin, Xmax)
Randomly generate N initial positions X_i(i=1,2,⋯,N) of N particles within the maximum range
Randomly generate N initial velocities V_i(i=1,2,⋯,N) of N particles within the maximum range
Calculate the fitness value of each particle using the objective function f
Set pbest and gbest in the swarm
While iter < Max_iter do
    Using Equations (19) and (20) to update acceleration coefficients c_1 and c_2
    Using Equation (16) to update the inertia weight ω
    if iter < Max_iter× 90%
        for i =1:N(number of particles) do
            Update the velocity V_i of the i th particle using Equation (10)
            Update the position X_i of the i th particle using Equation (11)
        End for i
    else
        for i =1:N(number of particles) do
            Choose one of Equations (6) (7) (8) (9) to update the velocity V_i and the position X_i
            of the i th particle using roulette selection algorithm
        End for i
    End if
    for i =1:N(number of particles) do
        Calculate the fitness values of the new particle X_i using the objective function f
        if X_i is superior to pbest_i
            Set X_i to be pbest_i
        End if
        if X_i is superior to gbest
            Set X_i to be gbest
        End if
    End for i
    iter=iter+1
End While
```

Figure 4. Pseudo-code of the multi-step probability selection particle swarm optimization with sine chaotic inertial weight and symmetric tangent chaotic acceleration coefficients (MPSPSO-ST) algorithm.

4. Experimental Results and Discussion

In this study, we used 20 well-known multi-dimensional classical benchmark functions, i.e., objective function f, to perform a large number of experiments to evaluate the performance of MPSPSO-ST proposed in this paper. These test functions were adopted by extensive literature [51,52]. We conducted two groups of experiments: in the first group, we compared the search performance of

MPSPSO-ST with standard PSO, basic PSO, MPSPSO. In the second group, we compared MPSPSO-ST with three PSO variants (chaos particle swarm optimization (CPSO), particle swarm optimization with the nonlinear dynamic acceleration coefficients (PSO-NDAC), adaptive inertia weight and acceleration coefficients PSO (AIWCPSO)) and three well-known similar optimization algorithms (DE, MFO, SCA).

The Appendix A lists 20 classic benchmark functions adopted for this experiment to test the performance of MPSPSO-ST. These test functions are divided into two types. The first type contains eight unimodal functions and the second type contains 12 multimodal functions. There are many well-known functions, like f_{10}(Michalewicz),f_{11}(Griewank),f_{13}(levy) and f_{19}(Zakharov).f_{10}(Michalewicz) has **d!** local minima. The parameter **m** defines the steepness of the valleys and ridges and a larger **m** leads to a more difficult search. f_{11}(Griewank) has many widespread local minima, which causes more difficulties in searching for the global optimum. f_{13}(levy) and f_{19}(Zakharov) are both widely used in the optimization field because they are classical. Dim, Range and $\boldsymbol{f_{\min}}$ represent the space dimensions of solution, the range of function variation and the minimum value of the function, that is, the optimal solutions, respectively.

4.1. Comparison of MPSPSO-ST with Standard PSO, Basic PSO and MPSPSO

The parameter settings for MPSPSO-ST, Standard PSO, Basic PSO and MPSPSO are shown in Table 1. The experimental results obtained by testing the 20 benchmark functions in the Appendix A are shown in Table 2. The convergence diagrams of the four algorithms are shown in Figures 5–7. The values in the diagrams are the mean best results of 20 runs of the whole swarm so far.

Table 1. Parameter settings for MPSPSO-ST and Standard PSO, Basic PSO and MPSPSO.

Algorithm	Population Size	Iteration	Run Times	Parameter Settings
Standard PSO	40	500	20	$c_1 = c_2 = 2, \omega = 0.9 \sim 0.4, V_{\max} = 6$
Basic PSO	40	500	20	$c_1 = c_2 = 2, \omega = 1, V_{\max} = 6$
MPSPSO	40	500	20	$c_1 = c_2 = 2, \omega = 0.9 \sim 0.4, V_{\max} = 6$
MPSPSO-ST	40	500	20	$c_1 = -\partial \times m^2 \times \tan[\frac{\pi}{8} \times (1+m^2)] + \theta + \rho \times z, c_2 = -\partial \times (1-m)^2 \times \tan[\frac{\pi}{8} \times (1+(1-m)^2)] + \theta + \rho \times z, \omega_{t+1} = \phi \times \sin(\pi\omega_t) + \tau, V_{\max} = 6$

Table 2. Experimental results for MPSPSO-ST, Standard PSO and Basic PSO. MPSPSO on 20 classical test functions.

Function	Algorithm	The Best	The Worst	Mean	S.d.
f_1	Standard PSO	4.5678×10^{-4}	2.1867×10^{-2}	5.0366×10^{-3}	2.0787×10^{-2}
	Basic PSO	9.0029×10^{1}	1.7626×10^{2}	1.3408×10^{2}	8.0903×10^{1}
	MPSPSO	1.2001×10^{-8}	9.9103×10^{-7}	2.6957×10^{-7}	1.1904×10^{-6}
	MPSPSO-ST	$\mathbf{5.6834 \times 10^{-16}}$	$\mathbf{1.9825 \times 10^{-13}}$	$\mathbf{2.0811 \times 10^{-14}}$	$\mathbf{1.9440 \times 10^{-13}}$
f_2	Standard PSO	0.1209	0.7743	0.2986	0.6429
	Basic PSO	56.7585	142.5399	102.9728	104.4540
	MPSPSO	0.0306	0.1479	0.0640	0.1217
	MPSPSO-ST	**0.0191**	**0.0732**	**0.0396**	**0.0644**
f_3	Standard PSO	4.3416×10^{1}	1.1851×10^{2}	7.6565×10^{1}	9.3955×10^{1}
	Basic PSO	2.7751×10^{2}	8.5694×10^{2}	4.8829×10^{2}	5.5224×10^{2}
	MPSPSO	1.8069×10^{0}	1.4074×10^{1}	5.7546×10^{0}	1.5430×10^{1}
	MPSPSO-ST	$\mathbf{7.9038 \times 10^{-3}}$	$\mathbf{2.3292 \times 10^{-1}}$	$\mathbf{7.0004 \times 10^{-2}}$	$\mathbf{2.5658 \times 10^{-1}}$
f_4	Standard PSO	1.9400×10^{-7}	4.0724×10^{-3}	4.4258×10^{-4}	4.1605×10^{-3}
	Basic PSO	7.3786×10^{-2}	1.2261×10^{0}	6.5420×10^{-1}	1.3869×10^{0}
	MPSPSO	4.6486×10^{-21}	1.7877×10^{-16}	1.2995×10^{-17}	1.7277×10^{-16}
	MPSPSO-ST	$\mathbf{1.8112 \times 10^{-49}}$	$\mathbf{2.6982 \times 10^{-40}}$	$\mathbf{1.9660 \times 10^{-41}}$	$\mathbf{2.6120 \times 10^{-40}}$

Table 2. *Cont.*

Function	Algorithm	The Best	The Worst	Mean	S.d.
f_5	Standard PSO	1.02591	12.0564	4.08932	12.80900
	Basic PSO	5.2064×10^{4}	1.2287×10^{5}	8.5260×10^{4}	9.1677×10^{4}
	MPSPSO	0.66672	3.14810	1.26960	3.71450
	MPSPSO-ST	**0.66667**	**2.03490**	**0.79497**	**1.70870**
f_6	Standard PSO	2.5432×10^{-3}	2.4399×10^{-2}	7.9220×10^{-3}	2.4535×10^{-2}
	Basic PSO	8.2531×10^{1}	1.6702×10^{2}	1.3159×10^{2}	8.8401×10^{1}
	MPSPSO	1.5414×10^{-8}	1.0729×10^{-6}	2.3913×10^{-7}	1.1474×10^{-6}
	MPSPSO-ST	$\mathbf{6.1306 \times 10^{-16}}$	$\mathbf{5.1809 \times 10^{-13}}$	$\mathbf{5.0002 \times 10^{-14}}$	$\mathbf{5.2461 \times 10^{-13}}$
f_7	Standard PSO	3.1183×10^{-2}	2.5205×10^{-1}	1.0527×10^{-1}	2.7090×10^{-1}
	Basic PSO	1.3802×10^{3}	2.3578×10^{3}	1.8650×10^{3}	1.2075×10^{3}
	MPSPSO	5.7551×10^{-7}	7.9424×10^{-5}	8.8537×10^{-6}	7.6769×10^{-5}
	MPSPSO-ST	$\mathbf{5.7456 \times 10^{-16}}$	$\mathbf{7.9810 \times 10^{-14}}$	$\mathbf{2.7045 \times 10^{-14}}$	$\mathbf{1.0692 \times 10^{-13}}$
f_8	Standard PSO	4.7811×10^{-4}	4.7318×10^{-2}	7.0241×10^{-3}	4.8574×10^{-2}
	Basic PSO	6.0663×10^{1}	1.3860×10^{2}	1.0412×10^{2}	1.0270×10^{2}
	MPSPSO	2.9684×10^{-13}	3.2834×10^{-10}	5.1454×10^{-11}	3.7774×10^{-10}
	MPSPSO-ST	$\mathbf{7.5201 \times 10^{-31}}$	$\mathbf{5.7745 \times 10^{-26}}$	$\mathbf{7.5066 \times 10^{-27}}$	$\mathbf{6.5486 \times 10^{-26}}$
f_9	Standard PSO	0.29987	0.49987	0.43487	0.25593
	Basic PSO	1.10070	1.5999	1.34680	0.57279
	MPSPSO	0.29987	0.49987	0.42487	0.27839
	MPSPSO-ST	**0.29987**	**0.49987**	**0.36518**	**0.25683**
f_{10}	Standard PSO	-1.7207×10^{-9}	-9.1788×10^{-10}	-1.2473×10^{-9}	9.7966×10^{-10}
	Basic PSO	-9.3338×10^{-10}	-6.8522×10^{-10}	-8.0715×10^{-10}	3.4338×10^{-10}
	MPSPSO	-2.3176×10^{-9}	-1.1251×10^{-9}	-1.8146×10^{-9}	1.4152×10^{-9}
	MPSPSO-ST	$\mathbf{-3.0052 \times 10^{-9}}$	$\mathbf{-2.4403 \times 10^{-9}}$	$\mathbf{-2.7811 \times 10^{-9}}$	$\mathbf{7.4879 \times 10^{-10}}$
f_{11}	Standard PSO	1.4317×10^{-5}	3.2392×10^{-2}	9.4530×10^{-3}	3.6992×10^{-2}
	Basic PSO	1.0250×10^{0}	1.0392×10^{0}	1.0335×10^{0}	1.9501×10^{-2}
	MPSPSO	$\mathbf{2.0319 \times 10^{-8}}$	$\mathbf{3.6910 \times 10^{-2}}$	$\mathbf{1.0222 \times 10^{-2}}$	$\mathbf{4.3936 \times 10^{-2}}$
	MPSPSO-ST	3.9968×10^{-15}	8.0685×10^{-2}	2.0720×10^{-2}	1.1602×10^{-1}
f_{12}	**Standard PSO**	1.1716×10^{-5}	$\mathbf{1.0372 \times 10^{-1}}$	$\mathbf{5.2801 \times 10^{-3}}$	$\mathbf{1.0100 \times 10^{-1}}$
	Basic PSO	4.1217×10^{0}	6.0335×10^{0}	5.2257×10^{0}	2.4996×10^{0}
	MPSPSO	1.4159×10^{-9}	1.0367×10^{-1}	1.5550×10^{-2}	1.6555×10^{-1}
	MPSPSO-ST	$\mathbf{9.1102 \times 10^{-17}}$	3.9616×10^{0}	8.3068×10^{-1}	4.6458×10^{0}
f_{13}	Standard PSO	3.4293	6.0435	4.5710	3.6773
	Basic PSO	124.6619	182.6454	150.8625	74.4685
	MPSPSO	**2.3077**	**5.9331**	**3.7067**	**3.5467**
	MPSPSO-ST	2.4172	11.536	5.3201	11.0171
f_{14}	Standard PSO	7.409×10^{-6}	4.3215×10^{-4}	8.4032×10^{-5}	4.1803×10^{-4}
	Basic PSO	1.6260×10^{0}	4.1097×10^{0}	2.4549×10^{0}	2.8544×10^{0}
	MPSPSO	6.8133×10^{-11}	$\mathbf{2.3492 \times 10^{-8}}$	$\mathbf{2.4795 \times 10^{-9}}$	$\mathbf{2.4440 \times 10^{-8}}$
	MPSPSO-ST	$\mathbf{7.6029 \times 10^{-17}}$	2.8983×10^{-3}	1.6939×10^{-4}	2.8349×10^{-3}
f_{15}	Standard PSO	2.5207×10^{-2}	2.0130×10^{-1}	7.7555×10^{-2}	2.3124×10^{-1}
	Basic PSO	2.7387×10^{1}	3.6274×10^{1}	3.1328×10^{1}	1.0927×10^{1}
	MPSPSO	4.5570×10^{-5}	4.0095×10^{-3}	1.1036×10^{-3}	4.9777×10^{-3}
	MPSPSO-ST	$\mathbf{6.5547 \times 10^{-8}}$	$\mathbf{3.0083 \times 10^{-4}}$	$\mathbf{3.6657 \times 10^{-5}}$	$\mathbf{3.3579 \times 10^{-4}}$
f_{16}	Standard PSO	3.0000	3.0000	3.0000	6.7349×10^{-15}
	Basic PSO	3.0099	3.4032	3.1322	5.5096×10^{-1}
	MPSPSO	3.0000	3.0000	3.0000	1.9357×10^{-15}
	MPSPSO-ST	**3.0000**	**3.0000**	**3.0000**	**0**
f_{17}	Standard PSO	0.9980	7.8740	2.6270	9.4631
	Basic PSO	0.9980	7.8740	2.0372	6.5904
	MPSPSO	0.9980	2.9821	1.4449	2.9697
	MPSPSO-ST	**0.9980**	**0.9980**	**0.9980**	**0**

Table 2. *Cont.*

Function	Algorithm	The Best	The Worst	Mean	S.d.
f_{18}	Standard PSO	4.3426×10^{-4}	1.1096×10^{-3}	8.6577×10^{-4}	7.7543×10^{-4}
	Basic PSO	5.8557×10^{-4}	1.9988×10^{-3}	1.2394×10^{-3}	1.6428×10^{-3}
	MPSPSO	3.0749×10^{-4}	1.0349×10^{-3}	6.5378×10^{-4}	1.4993×10^{-3}
	MPSPSO-ST	$\mathbf{3.0749 \times 10^{-4}}$	$\mathbf{1.0383 \times 10^{-3}}$	$\mathbf{4.0910 \times 10^{-4}}$	$\mathbf{1.0838 \times 10^{-3}}$
f_{19}	Standard PSO	1.8802×10^{-17}	5.3438×10^{-15}	9.9794×10^{-16}	7.0982×10^{-15}
	Basic PSO	1.6320×10^{0}	4.5516×10^{0}	2.8228×10^{0}	3.6775×10^{0}
	MPSPSO	8.1831×10^{-38}	3.5528×10^{-33}	2.2073×10^{-34}	3.4617×10^{-33}
	MPSPSO-ST	$\mathbf{3.2993 \times 10^{-58}}$	$\mathbf{2.8784 \times 10^{-55}}$	$\mathbf{5.5092 \times 10^{-56}}$	$\mathbf{3.2355 \times 10^{-55}}$
f_{20}	Standard PSO	−3.3220	−3.2031	−3.2744	0.2605
	Basic PSO	−3.1587	−2.5910	−2.9131	0.8005
	MPSPSO	−3.3220	−3.2031	−3.2566	0.2645
	MPSPSO-ST	**−3.3220**	**−3.2031**	**−3.2982**	**0.2126**

Figure 5 clearly shows that compared to the other three algorithms, MPSPSO-ST has very prominent advantages, which offers the fastest convergence speed and the strongest optimization accuracy. From Figure 6a,d,e,f, we observe that for multimodal functions f_{10}, f_{15}, f_{17}, f_{18}, MPSPSO-ST owns faster convergence speed and better optimization performance compared with the other three algorithms, especially these advantages are more outstanding for f_{10}, f_{15}, f_{17}. However, for multimodal functions f_{11}, f_{13} in Figure 6b,c, the performance of MPSPSO-ST is not as good as that of MPSPSO and Standard PSO. Nevertheless, MPSPSO-ST can maintain a faster convergence rate when the final optimization results of MPSPSO-ST and MPSPSO, Standard PSO are quite close. As seen from Figure 7, the convergence speed and global search capability of MPSPSO-ST are the most competitive for multimodal functions f_{19}, f_{20}.

The information in Table 2 can reflect the above conclusions more specifically and accurately. The indicators in Table 2 are the best values (the best), the worst value (the worst), mean value (mean), standard deviation (S.D.) of MPSPSO-ST and Standard PSO and Basic PSO, MPSPSO in 20 independent running experiments. Among all unimodal functions $f_1 \sim f_8$ and multimodal functions f_9, f_{10}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19} and f_{20}, MPSPSO-ST overtakes all other methods in terms of convergence rate and optimization accuracy. Especially for multimodal functions f_{16} and f_{17}, MPSPSO-ST can all reach the theoretical optimal value in a few iterations and the S.D. of MPSPSO-ST can reach 0. It means that MPSPSO-ST can achieve theoretical optimal value in each of 20 independent experiments, which shows MPSPSO-ST has steadier search ability than the other three algorithms during the search process. For multimodal functions f_{11}, f_{12} and f_{14}, the optimization performance of MPSPSO-ST is not optimal, but it is the best at The best, indicating that MPSPSO-ST has obtained the best optimization results in the optimization of these test functions but the overall has not been maintained. Meanwhile, it can be seen that the standard deviation of MPSPSO-ST is smaller than the other three algorithms in a majority of test functions, illustrating that it has stronger robustness and stability.

As seen from Table 2, Figures 5–7, we can conclude that the proposed MPSPSO-ST owns better optimization capability compared with Standard PSO, Basic PSO and MPSPSO, which indicates that MPSPSO-ST has excellent ability in solving numerical optimization problems.

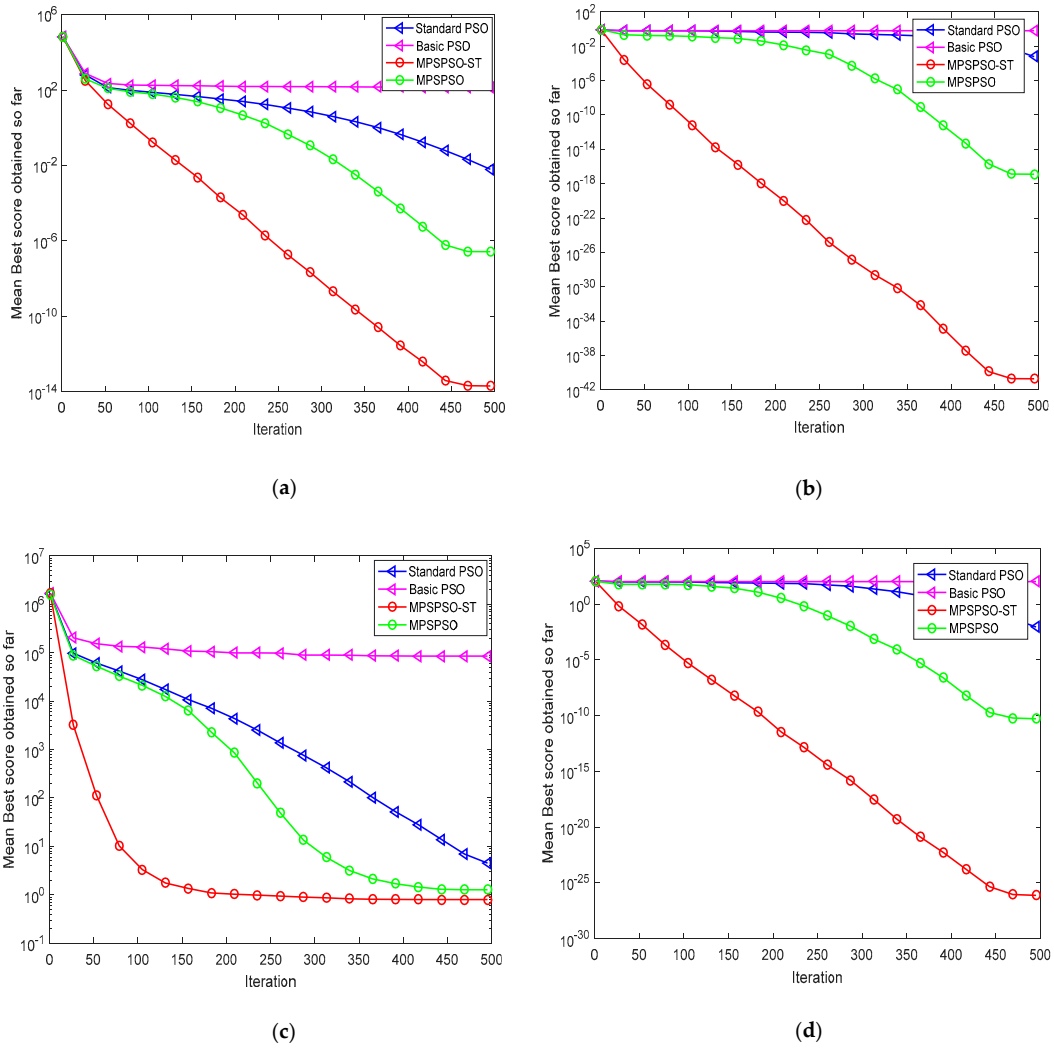

Figure 5. The convergence curves of four algorithms for the functions: (**a**) f_1; (**b**) f_4; (**c**) f_5; (**d**) f_8.

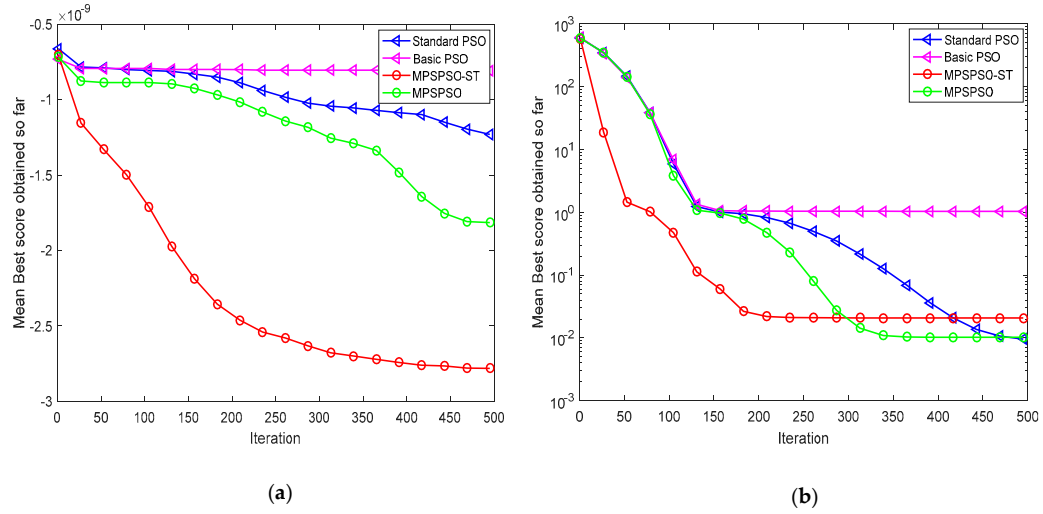

Figure 6. *Cont*.

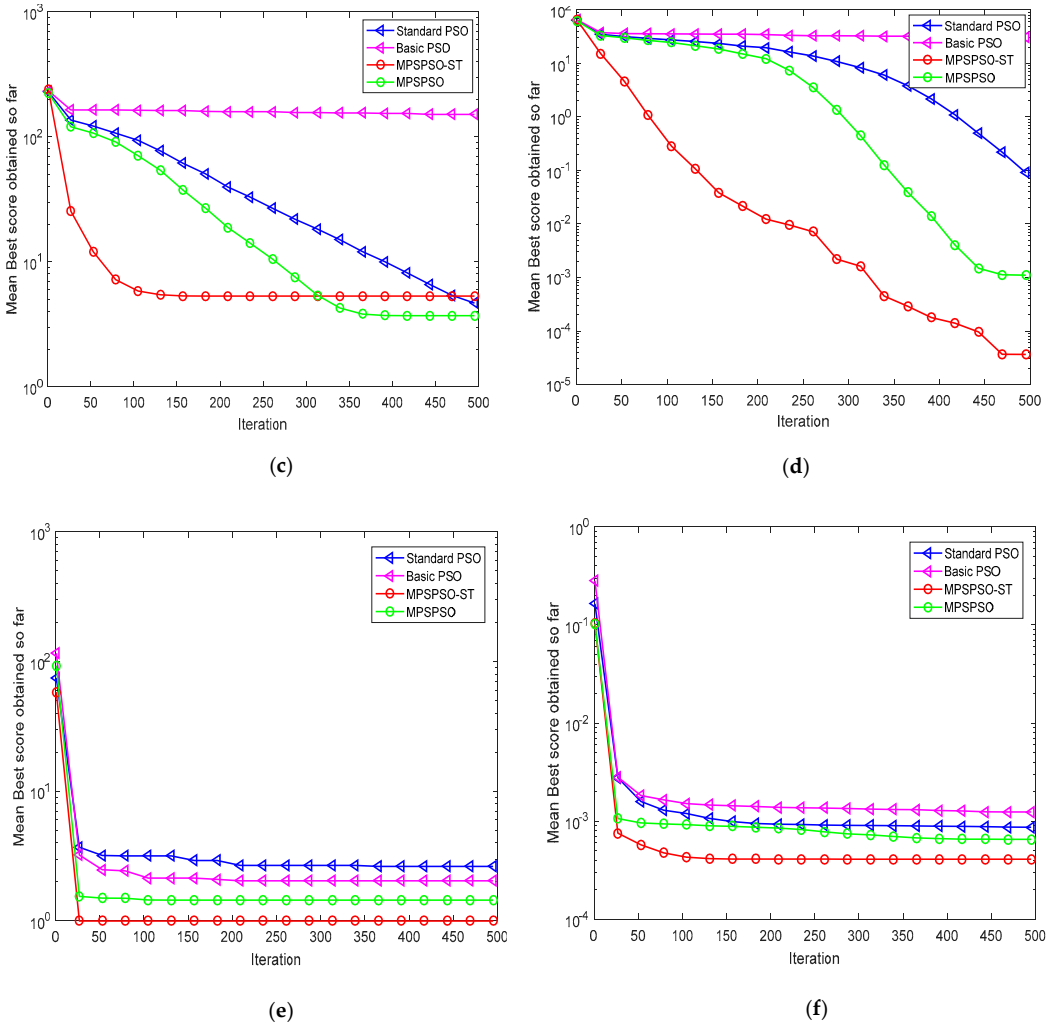

Figure 6. The convergence curves of four algorithms for the functions: (**a**) f_{10}; (**b**) f_{11}; (**c**) f_{13}; (**d**) f_{15}; (**e**) f_{17}; (**f**) f_{18}.

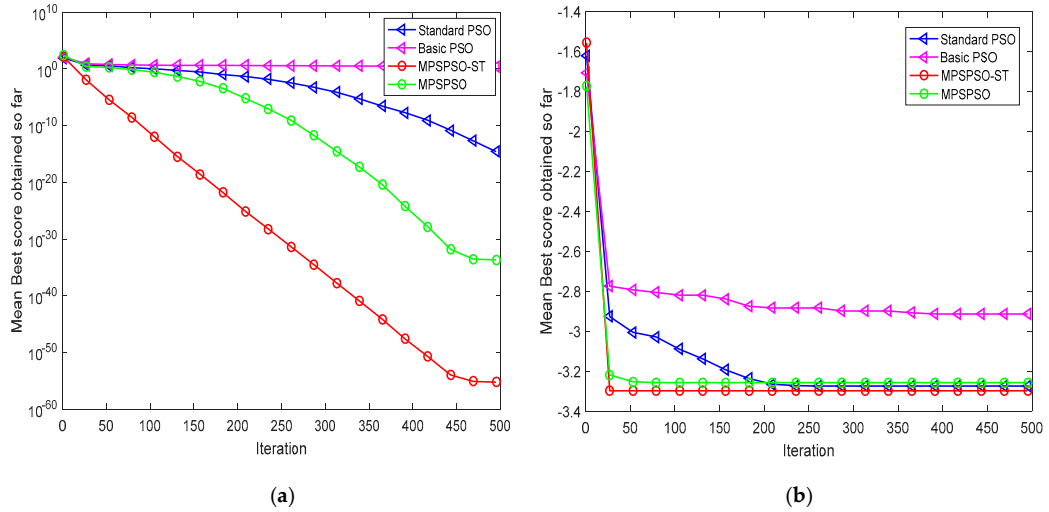

Figure 7. The convergence curves of four algorithms for the functions: (**a**) f_{19}; (**b**) f_{20}.

4.2. Comparison of MPSPSO-ST with CPSO, PSO-NDAC, AIWCPSO, DE, MFO and SCA

In this section, we have designed comparisons of MPSPSO-ST with several classic improved PSO variants (chaos particle swarm optimization (CPSO) [53] particle swarm optimization with the nonlinear

dynamic acceleration coefficients (PSO-NDAC) [32] AIWCPSO [54]) and other well-known classic optimization algorithms (differential evolution algorithm (DE), moth-flame optimization algorithm (MFO) and sine cosine algorithm (SCA)). CPSO realizes chaotic searching on the current global best individual using chaos mechanism. PSO-NDAC proposes the nonlinear dynamic acceleration coefficients to add to PSO. AIWCPSO designs an inertia weight strategy that the adaptive inertia weight is adjusted according to the updated particles of the previous generation. DE is a group-based adaptive global optimization algorithm, which belongs to an evolutionary algorithm. MFO is a swarm intelligence optimization algorithm, the main inspiration of which is the navigation method of moths in nature called transverse orientation. SCA creates multiple initial random candidate solutions and requires them to fluctuate outwards or towards the best solution based on sine and cosine functions. The parameter settings for each algorithm are shown in Table 3.

Table 3. Parameter settings for MPSPSO-ST and particle swarm optimization with the nonlinear dynamic acceleration coefficients (PSO-NDAC), chaos particle swarm optimization (CPSO), AIWCPSO, moth-flame optimization (MFO), sine cosine algorithm (SCA) and differential evolution (DE).

Algorithm	Population Size	Iteration	Run Times	Parameter Settings
PSO-NDAC	40	500	20	$c_1 = -2 \times m^2 + 2.5, c_2 = 0.5 \times (1-m)^2 + 2.5 \times m$ $(m = \frac{t}{t_{\max}}), \omega = 0.9 \sim 0.4, V_{\max} = 6$
CPSO	40	500	20	$c_1 = c_2 = 2, \omega = 0.9 \sim 0.4, \mu = 4, V_{\max} = 6$
AIWCPSO	40	500	20	$c_1 = c_2 = 2, \omega = 0.9 \sim 0.4, V_{\max} = 6$
MFO	40	500	20	t is random number in the range$[-2, 1]$
SCA	40	500	20	$r_1 = 4 \sim 0$, r_2 is a random number in the range $[0, 2\pi]$, r_3 is a random number in the range $[0, 2]$, r_4 is a random number in the range $[0, 1]$
DE	40	500	20	F=0.3, CR=0.5
MPSPSO-ST	40	500	20	$c_1 = -\partial \times m^2 \times \tan[\frac{\pi}{8} \times (1+m^2)] + \theta + \rho \times z, c_2 =$ $-\partial \times (1-m)^2 \times \tan[\frac{\pi}{8} \times (1+(1-m)^2)] + \theta + \rho \times$ $z, \omega_{t+1} = \phi \times \sin(\pi \omega_t) + \tau, V_{\max} = 6$

The experimental results obtained by the test functions listed in the Appendix A are shown in Table 4. The convergence diagrams of the seven algorithms are shown in Figures 8–10, and the values in the diagrams are the mean best results in 20 runs in the whole swarm so far.

As seen from Figure 8, MPSPSO-ST has an undeniable advantage in terms of convergence rate and optimization accuracy for unimodal functions f_1, f_3, f_4, f_6 compared with the other six algorithms. Figure 9a–c,f show that MPSPSO-ST also has the best convergence rate and solution accuracy for multimodal functions f_7, f_8, f_{10}, f_{15}. Through Figure 9d,e, we find that the search performance of MPSPSO-ST is not optimal for multimodal functions f_{11}, f_{14}. For multimodal function f_{11}, MPSPSO-ST has the best performance except for AIWCPSO. Furthermore, in the case where the optimization effect of MPSPSO-ST is very close to the optimization effect of AIWCPSO, MPSPSO-ST has a faster convergence speed to the global optimal solution than AIWCPSO. For multimodal functions f_{17}, f_{18}, f_{19}, f_{20} in Figure 10, MPSPSO-ST outperforms the other six methods.

As shown in Table 4, for multimodal functions f_{11} and f_{12}, MPSPSO-ST is not the best but The best it can find is the greatest compared to other algorithms, which shows that MPSPSO -ST has the ability to find the best results in 20 independent experiments but it has not been maintained in general. Besides, for multimodal functions f_{16}, f_{17}, the theoretical optimal value can be successfully found each time by MPSPSO-ST in 20 independent experiments, which indicates that the search performance of MPSPSO-ST is very stable. Therefore, we can see that MPSPSO-ST has prominent advantages in robustness and stability.

Table 4. Experimental results for MPSPSO-ST and PSO-NDAC, CPSO, AIWCPSO, MFO, SCA and DE on 20 classical test functions.

Function	Algorithm	The Best	The Worst	Mean	S.D.
f_1	PSO-NDAC	5.1300×10^{-8}	3.3867×10^{-5}	2.6932×10^{-6}	3.2888×10^{-5}
	CPSO	2.5381×10^{-1}	3.3147×10^{1}	7.9537×10^{0}	4.6559×10^{1}
	AIWCPSO	1.6492×10^{-6}	2.8529×10^{-5}	8.0015×10^{-6}	2.8947×10^{-5}
	MFO	5.3151×10^{-1}	1.0000×10^{4}	5.0147×10^{2}	9.7458×10^{3}
	SCA	1.2638×10^{-15}	2.2184×10^{2}	1.1287×10^{1}	2.1604×10^{2}
	DE	6.1421×10^{-3}	8.6630×10^{1}	1.3950×10^{1}	9.8238×10^{1}
	MPSPSO-ST	$\mathbf{8.7871 \times 10^{-16}}$	$\mathbf{2.5463 \times 10^{-13}}$	$\mathbf{8.1821 \times 10^{-14}}$	$\mathbf{3.3863 \times 10^{-13}}$
f_2	PSO-NDAC	0.02018	0.09354	0.05649	0.08375
	CPSO	0.05965	2.81610	1.25650	3.50140
	AIWCPSO	0.03554	0.08707	0.05680	0.05934
	MFO	0.05698	29.70930	3.39012	31.62590
	SCA	0.00815	0.31364	0.06917	0.37656
	DE	0.03722	0.20296	0.10343	0.19898
	MPSPSO-ST	**0.01611**	**0.05752**	**0.03274**	**0.04851**
f_3	PSO-NDAC	8.8027×10^{0}	8.9787×10^{1}	2.7801×10^{1}	8.7341×10^{1}
	CPSO	2.4701×10^{1}	1.0485×10^{2}	6.0563×10^{1}	9.0587×10^{1}
	AIWCPSO	1.9160×10^{1}	1.1344×10^{2}	4.5443×10^{1}	9.0026×10^{1}
	MFO	1.7315×10^{3}	5.3692×10^{4}	1.9764×10^{4}	5.0273×10^{4}
	SCA	2.1712×10^{0}	1.2644×10^{4}	1.9731×10^{3}	1.8014×10^{4}
	DE	5.8248×10^{3}	2.5593×10^{4}	1.2039×10^{4}	2.2101×10^{4}
	MPSPSO-ST	4.1427×10^{-2}	5.9152×10^{-1}	2.1145×10^{-1}	7.7382×10^{-1}
f_4	PSO-NDAC	4.0015×10^{-25}	2.6450×10^{-19}	3.5505×10^{-20}	3.5317×10^{-19}
	CPSO	2.5551×10^{-4}	8.0566×10^{-3}	2.2690×10^{-3}	8.7752×10^{-3}
	AIWCPSO	3.0713×10^{-18}	3.1036×10^{-12}	2.2484×10^{-13}	3.1787×10^{-12}
	MFO	8.7526×10^{-14}	2.4125×10^{-7}	1.2382×10^{-8}	2.3484×10^{-7}
	SCA	1.7091×10^{-56}	1.3950×10^{-3}	6.9753×10^{-5}	1.3597×10^{-3}
	DE	1.0221×10^{-18}	3.9932×10^{-6}	2.0004×10^{-7}	3.8918×10^{-6}
	MPSPSO-ST	$\mathbf{1.8781 \times 10^{-49}}$	$\mathbf{6.6562 \times 10^{-41}}$	$\mathbf{1.1225 \times 10^{-41}}$	$\mathbf{9.4197 \times 10^{-41}}$
f_5	PSO-NDAC	6.6668×10^{-1}	6.5562×10^{0}	1.6663×10^{0}	7.0782×10^{0}
	CPSO	1.1582×10^{0}	3.4957×10^{4}	1.0897×10^{4}	5.4315×10^{4}
	AIWCPSO	6.6749×10^{-1}	4.7360×10^{0}	1.3134×10^{0}	5.3422×10^{0}
	MFO	1.9549×10^{0}	7.2585×10^{4}	1.0531×10^{4}	1.1194×10^{5}
	SCA	6.6713×10^{-1}	7.4345×10^{1}	4.5179×10^{0}	7.1707×10^{1}
	DE	3.0014×10^{0}	1.5603×10^{3}	1.3819×10^{2}	1.6222×10^{3}
	MPSPSO-ST	$\mathbf{6.6667 \times 10^{-1}}$	$\mathbf{3.5021 \times 10^{0}}$	$\mathbf{1.0923 \times 10^{0}}$	$\mathbf{3.7675 \times 10^{0}}$
f_6	PSO-NDAC	1.7100×10^{-8}	1.4269×10^{-5}	2.1761×10^{-6}	1.4714×10^{-5}
	CPSO	1.4766×10^{0}	3.7322×10^{1}	1.1397×10^{1}	3.7236×10^{1}
	AIWCPSO	1.2828×10^{-6}	1.7912×10^{-5}	6.8441×10^{-6}	2.1667×10^{-5}
	MFO	5.0435×10^{-1}	9.9013×10^{3}	1.0440×10^{3}	1.3243×10^{4}
	SCA	4.3620×10^{0}	1.4545×10^{1}	5.5499×10^{0}	9.8103×10^{0}
	DE	6.2442×10^{-10}	1.5620×10^{2}	2.3732×10^{1}	1.9729×10^{2}
	MPSPSO-ST	$\mathbf{1.0678 \times 10^{-15}}$	$\mathbf{4.8348 \times 10^{-13}}$	$\mathbf{7.7464 \times 10^{-14}}$	$\mathbf{5.0284 \times 10^{-13}}$
f_7	PSO-NDAC	4.4619×10^{-7}	6.0106×10^{-5}	8.5332×10^{-6}	6.1709×10^{-5}
	CPSO	5.7821×10^{1}	1.1138×10^{3}	5.0476×10^{2}	1.3492×10^{3}
	AIWCPSO	3.5964×10^{-6}	7.4185×10^{-5}	2.1497×10^{-5}	8.3628×10^{-5}
	MFO	3.0001×10^{-2}	1.5001×10^{3}	4.5025×10^{2}	2.0993×10^{3}
	SCA	$\mathbf{1.8083 \times 10^{-22}}$	1.9105×10^{1}	1.0553×10^{0}	1.8618×10^{1}
	DE	7.3789×10^{-5}	1.4108×10^{1}	1.3408×10^{0}	1.4138×10^{1}
	MPSPSO-ST	4.6508×10^{-16}	$\mathbf{1.1855 \times 10^{-12}}$	$\mathbf{1.8800 \times 10^{-14}}$	$\mathbf{1.5397 \times 10^{-12}}$
f_8	PSO-NDAC	7.8413×10^{-14}	1.4895×10^{-9}	1.4322×10^{-10}	1.6302×10^{-9}
	CPSO	1.2651×10^{-2}	2.7074×10^{0}	8.0086×10^{-1}	3.2157×10^{0}
	AIWCPSO	6.6641×10^{-11}	1.6291×10^{-7}	1.1648×10^{-8}	1.6120×10^{-7}
	MFO	6.5346×10^{-6}	1.3422×10^{1}	1.3437×10^{0}	1.3419×10^{1}
	SCA	6.8560×10^{-28}	1.4863×10^{0}	7.8368×10^{-2}	1.4465×10^{0}
	DE	2.9936×10^{-7}	3.3894×10^{-2}	2.9902×10^{-3}	3.2996×10^{-2}
	MPSPSO-ST	$\mathbf{2.2820 \times 10^{-31}}$	$\mathbf{2.5041 \times 10^{-26}}$	$\mathbf{3.3096 \times 10^{-27}}$	$\mathbf{2.8730 \times 10^{-26}}$

Table 4. *Cont.*

Function	Algorithm	The Best	The Worst	Mean	S.D.
f_9	PSO-NDAC	0.29987	0.59987	0.41987	0.36332
	CPSO	0.20245	0.89988	0.59128	0.87943
	AIWCPSO	0.19987	0.59987	0.41994	0.38963
	MFO	1.29987	12.19990	5.56991	17.25210
	SCA	0.09988	4.55470	0.61124	4.18660
	DE	0.29987	1.39990	0.50236	1.23470
	MPSPSO-ST	**0.29987**	**0.49987**	**0.38987**	**0.27928**
f_{10}	PSO-NDAC	-2.9410×10^{-9}	-1.8034×10^{-9}	-2.3857×10^{-9}	1.1856×10^{-9}
	CPSO	-1.1181×10^{-9}	-8.6693×10^{-10}	-9.9583×10^{-10}	2.7155×10^{-10}
	AIWCPSO	-2.8340×10^{-9}	-1.6520×10^{-9}	-2.4220×10^{-9}	1.1758×10^{-9}
	MFO	-2.6658×10^{-9}	-2.0384×10^{-9}	-2.3715×10^{-9}	6.6030×10^{-10}
	SCA	-7.1394×10^{-10}	-1.1393×10^{-10}	-2.6959×10^{-10}	7.6044×10^{-10}
	DE	-2.1077×10^{-9}	-1.6630×10^{-9}	-1.8151×10^{-9}	4.4156×10^{-10}
	MPSPSO-ST	$\mathbf{-3.0193 \times 10^{-9}}$	$\mathbf{-2.3895 \times 10^{-9}}$	$\mathbf{-2.7955 \times 10^{-9}}$	$\mathbf{6.9722 \times 10^{-10}}$
f_{11}	PSO-NDAC	1.8235×10^{-8}	1.1349×10^{0}	5.8798×10^{-1}	2.1714×10^{0}
	CPSO	1.2241×10^{-2}	7.0711×10^{-1}	2.4554×10^{-1}	1.0575×10^{0}
	AIWCPSO	4.8976×10^{-6}	$\mathbf{2.7095 \times 10^{-2}}$	$\mathbf{8.7549 \times 10^{-3}}$	$\mathbf{3.7409 \times 10^{-2}}$
	MFO	5.0123×10^{-1}	9.1002×10^{1}	9.9449×10^{0}	1.2071×10^{2}
	SCA	7.8826×10^{-15}	2.5680×10^{0}	4.5262×10^{-1}	2.7232×10^{0}
	DE	9.1420×10^{-4}	2.0907×10^{0}	6.3608×10^{-1}	2.9071×10^{0}
	MPSPSO-ST	$\mathbf{6.6613 \times 10^{-16}}$	9.3347×10^{-2}	1.9044×10^{-2}	9.7708×10^{-2}
f_{12}	**PSO-NDAC**	5.7978×10^{-11}	$\mathbf{1.0370 \times 10^{-1}}$	$\mathbf{1.0369 \times 10^{-2}}$	$\mathbf{1.3911 \times 10^{-1}}$
	CPSO	2.9995×10^{-1}	2.0577×10^{0}	7.9385×10^{-1}	2.0733×10^{0}
	AIWCPSO	5.5656×10^{-8}	3.1096×10^{-1}	5.1833×10^{-2}	3.7374×10^{-1}
	MFO	2.0323×10^{0}	1.4052×10^{1}	7.0648×10^{0}	1.6366×10^{1}
	SCA	5.4712×10^{-1}	2.0995×10^{1}	1.8558×10^{0}	1.9818×10^{1}
	DE	1.4196×10^{-1}	3.7376×10^{4}	1.8701×10^{3}	3.6428×10^{4}
	MPSPSO-ST	$\mathbf{6.2512 \times 10^{-17}}$	3.9495×10^{0}	1.1213×10^{0}	5.1966×10^{0}
f_{13}	PSO-NDAC	2.3083	5.7135	3.5316	3.5264
	CPSO	10.3682	56.6080	24.7796	48.5283
	AIWCPSO	1.6613	4.6190	3.0787	4.1172
	MFO	2.8600	56.0777	18.1767	66.0427
	SCA	26.0094	34.1472	29.7011	10.6788
	DE	**0.0038**	**0.8927**	**0.4323**	**1.0218**
	MPSPSO-ST	2.4172	8.6693	5.3316	9.3605
f_{14}	PSO-NDAC	1.1023×10^{-9}	2.4166×10^{-6}	2.7192×10^{-7}	2.7198×10^{-6}
	CPSO	3.6970×10^{-2}	1.1233×10^{0}	3.4541×10^{-1}	1.1816×10^{0}
	AIWCPSO	2.6544×10^{-8}	$\mathbf{2.3484 \times 10^{-6}}$	$\mathbf{4.0181 \times 10^{-7}}$	$\mathbf{2.6397 \times 10^{-6}}$
	MFO	6.7555×10^{-1}	1.1573×10^{2}	2.4571×10^{1}	1.2484×10^{2}
	SCA	$\mathbf{8.2348 \times 10^{-24}}$	5.5803×10^{-2}	3.0040×10^{-3}	5.4329×10^{-2}
	DE	1.7097×10^{-4}	6.8415×10^{-1}	5.6675×10^{-2}	6.5169×10^{-1}
	MPSPSO-ST	1.7084×10^{-17}	1.1430×10^{-4}	1.1812×10^{-5}	1.4598×10^{-4}
f_{15}	PSO-NDAC	5.6249×10^{-4}	3.7831×10^{-1}	2.7575×10^{-2}	3.6192×10^{-1}
	CPSO	2.9421×10^{0}	1.7870×10^{1}	1.0966×10^{1}	2.0972×10^{1}
	AIWCPSO	3.3407×10^{-4}	3.0759×10^{-3}	8.7949×10^{-4}	3.3632×10^{-3}
	MFO	1.9292×10^{-2}	2.2203×10^{1}	6.5223×10^{0}	2.8871×10^{1}
	SCA	4.9150×10^{-5}	7.1893×10^{-1}	3.7656×10^{-2}	6.9966×10^{-1}
	DE	5.4837×10^{-7}	3.3238×10^{-2}	4.0703×10^{-3}	3.2331×10^{-2}
	MPSPSO-ST	$\mathbf{1.0274 \times 10^{-8}}$	$\mathbf{9.8253 \times 10^{-4}}$	$\mathbf{1.7602 \times 10^{-4}}$	$\mathbf{1.4203 \times 10^{-3}}$
f_{16}	PSO-NDAC	3.0000	3.0000	3.0000	1.8841×10^{-15}
	CPSO	3.0008	3.4160	3.0888	5.3477×10^{-1}
	AIWCPSO	3.0000	3.0000	3.0000	1.9357×10^{-15}
	MFO	3.0000	3.0000	3.0000	8.2725×10^{-15}
	SCA	3.0000	3.0010	3.0002	1.0726×10^{-3}
	DE	3.0000	3.0000	3.0000	1.8310×10^{-15}
	MPSPSO-ST	**3.0000**	**3.0000**	**3.0000**	**0**

Table 4. *Cont.*

Function	Algorithm	The Best	The Worst	Mean	S.D.
f_{17}	PSO-NDAC	0.99800	1.99200	1.09740	1.33360
	CPSO	4.14576	28.82710	13.89500	20.24470
	AIWCPSO	0.99800	5.92880	1.78910	6.63320
	MFO	0.99800	5.92880	1.59210	5.30030
	SCA	0.99801	2.98210	1.09930	1.93180
	DE	0.99801	10.76320	2.08246	11.49000
	MPSPSO-ST	**0.99800**	**0.99800**	**0.99800**	**0**
f_{18}	PSO-NDAC	3.0749×10^{-4}	1.0028×10^{-3}	5.9124×10^{-4}	1.1285×10^{-3}
	CPSO	6.6030×10^{-4}	4.0282×10^{-2}	6.4056×10^{-3}	5.0951×10^{-2}
	AIWCPSO	3.0749×10^{-4}	1.5941×10^{-3}	7.4898×10^{-4}	1.5257×10^{-3}
	MFO	3.7221×10^{-4}	1.6554×10^{-3}	9.5160×10^{-4}	1.7842×10^{-3}
	SCA	8.1546×10^{-4}	1.6696×10^{-3}	1.3903×10^{-3}	9.2161×10^{-4}
	DE	3.1525×10^{-4}	4.6017×10^{-3}	1.1767×10^{-3}	3.9121×10^{-3}
	MPSPSO-ST	$\mathbf{3.0749 \times 10^{-4}}$	$\mathbf{1.0371 \times 10^{-3}}$	$\mathbf{3.8036 \times 10^{-4}}$	$\mathbf{9.7776 \times 10^{-4}}$
f_{19}	PSO-NDAC	1.5069×10^{-33}	7.4158×10^{-30}	6.4473×10^{-31}	7.2887×10^{-30}
	CPSO	2.1384×10^{-2}	2.2297×10^{1}	5.7052×10^{0}	3.8606×10^{1}
	AIWCPSO	2.2815×10^{-30}	1.4904×10^{-27}	2.6907×10^{-28}	1.8522×10^{-27}
	MFO	3.3701×10^{-23}	1.3478×10^{-19}	3.2070×10^{-20}	2.0313×10^{-19}
	SCA	3.3833×10^{-48}	5.0261×10^{-12}	2.5131×10^{-13}	4.8989×10^{-12}
	DE	8.7979×10^{-27}	8.2996×10^{-2}	5.9547×10^{-3}	8.4550×10^{-2}
	MPSPSO-ST	$\mathbf{3.3480 \times 10^{-58}}$	$\mathbf{4.6387 \times 10^{-55}}$	$\mathbf{3.2719 \times 10^{-56}}$	$\mathbf{4.4551 \times 10^{-55}}$
f_{20}	PSO-NDAC	−3.3220	−3.2031	−3.2804	0.2536
	CPSO	−3.2242	−2.6097	−3.0177	0.6943
	AIWCPSO	−3.3220	−3.2031	−3.2744	0.2605
	MFO	−3.3220	−3.1376	−3.2351	0.2619
	SCA	−3.0134	−2.9619	−2.9892	0.0738
	DE	−3.3220	−3.2030	−3.2799	0.2521
	MPSPSO-ST	**−3.3220**	**−3.2031**	**−3.2982**	**0.2127**

Combining Table 4 and Figures 8–10, among the 20 classical benchmark functions $f_1 \sim f_{20}$, referring to two indicators, mean (mean) and standard deviation (S.D.) in Table 4, MPSPSO-ST is significantly superior to the other six algorithms in searching better global optimal value for these 16 test functions $f_1 \sim f_{10}$, $f_{15} \sim f_{20}$. In addition, although the final optimization result of MPSPSO-ST is inferior to AIWCPSO for multimodal function f_{11}, it is extremely close to AIWCPSO in terms of optimization effect and has the fastest convergence speed, and its convergence accuracy is better than the five other algorithms except for AIWCPSO, so MPSPSO-ST is used as the most efficient optimization algorithm. For multimodal functions f_{11}, f_{14}, AIWCPSO shows the best performance. For multimodal function f_{12}, PSO-NDAC has the best performance. The performance of AIWCPSO and PSO-NDAC is similar in other test functions, second only to MPSPSO-ST, so AIWCPSO and PSO-NDAC are tied as the second most effective optimization algorithm. DE is listed as the third most effective algorithm, achieving the best solution on multimodal function f_{13}. Although SCA has not got the best solution in the optimization of any test function, from the indicator The best in Table 4, SCA has found better optimal values than other algorithms in 20 independent experiments for unimodal function f_7 and multimodal function f_{14}, but the lack of maintenance leads to a general mean. So SCA is listed as the fourth most influential optimization algorithm. For CPSO and MFO, they are unable to find a better global optimal solution than all other algorithms in all test functions.

To sum up, it can be concluded that MPSPSO-ST offers superior overall performance among all seven algorithms, followed by AIWCPSO and PSO- NDAC, then DE, then SCA, and finally CPSO and MFO.

In general, each category used to assess a specified behavior in the optimization algorithm. The unimodal functions are used to assess the convergence rate of the algorithm since they contain a single extreme solution in the search domain ($f_1 \sim f_8$). Meanwhile, the multimodal functions are used to evaluate the ability of the algorithm to avoid the local point and reach to global solution since they contain more than an extreme solution ($f_9 \sim f_{20}$) 52In the above two experiment sections, among 20 test functions applied in this paper, MPSPSO-ST is significantly superior to other comparison algorithms for all unimodal functions and most multimodal functions, which reflects that MPSPSO-ST has an excellent convergence speed and the ability to get rid of local extreme values.

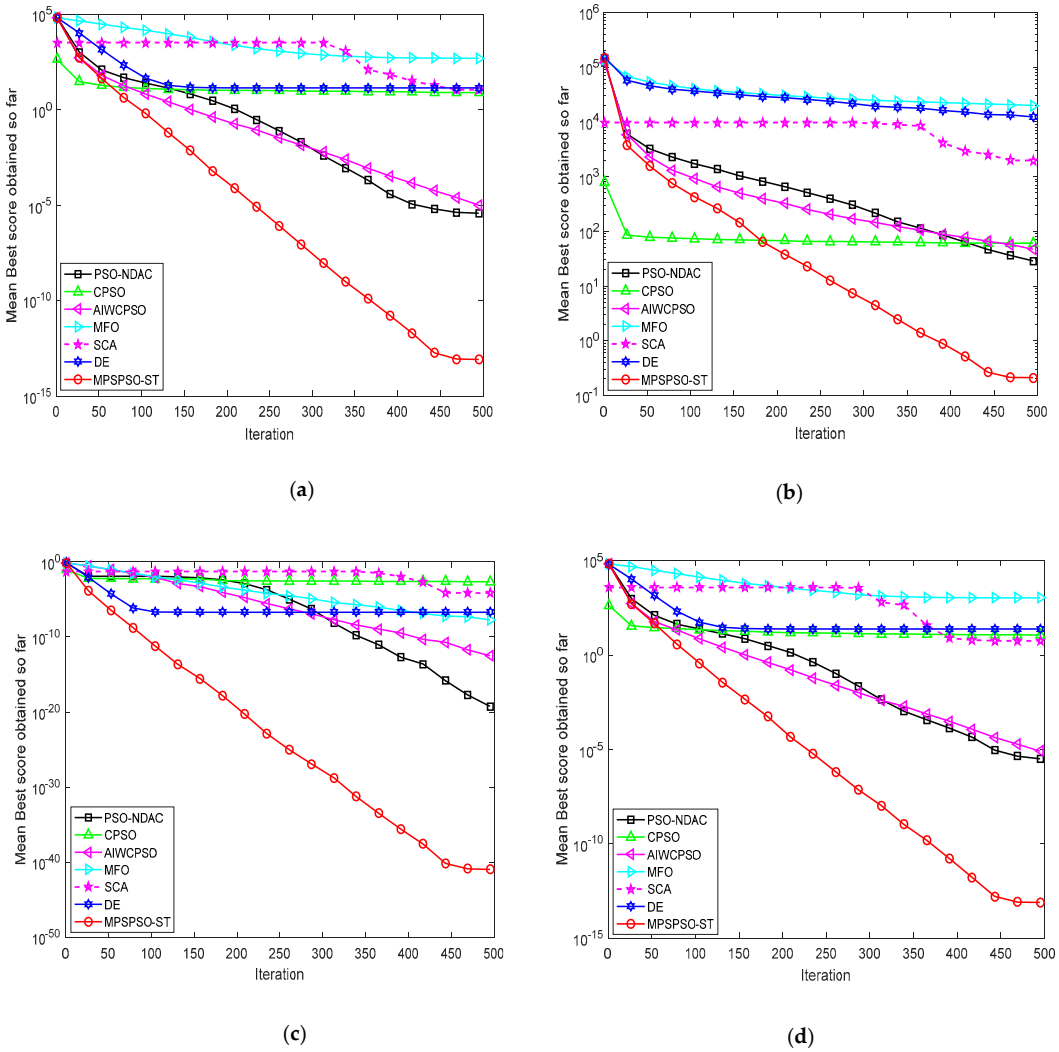

Figure 8. The convergence curves of seven algorithms for the functions: (**a**) f_1; (**b**) f_3; (**c**) f_4; (**d**) f_6.

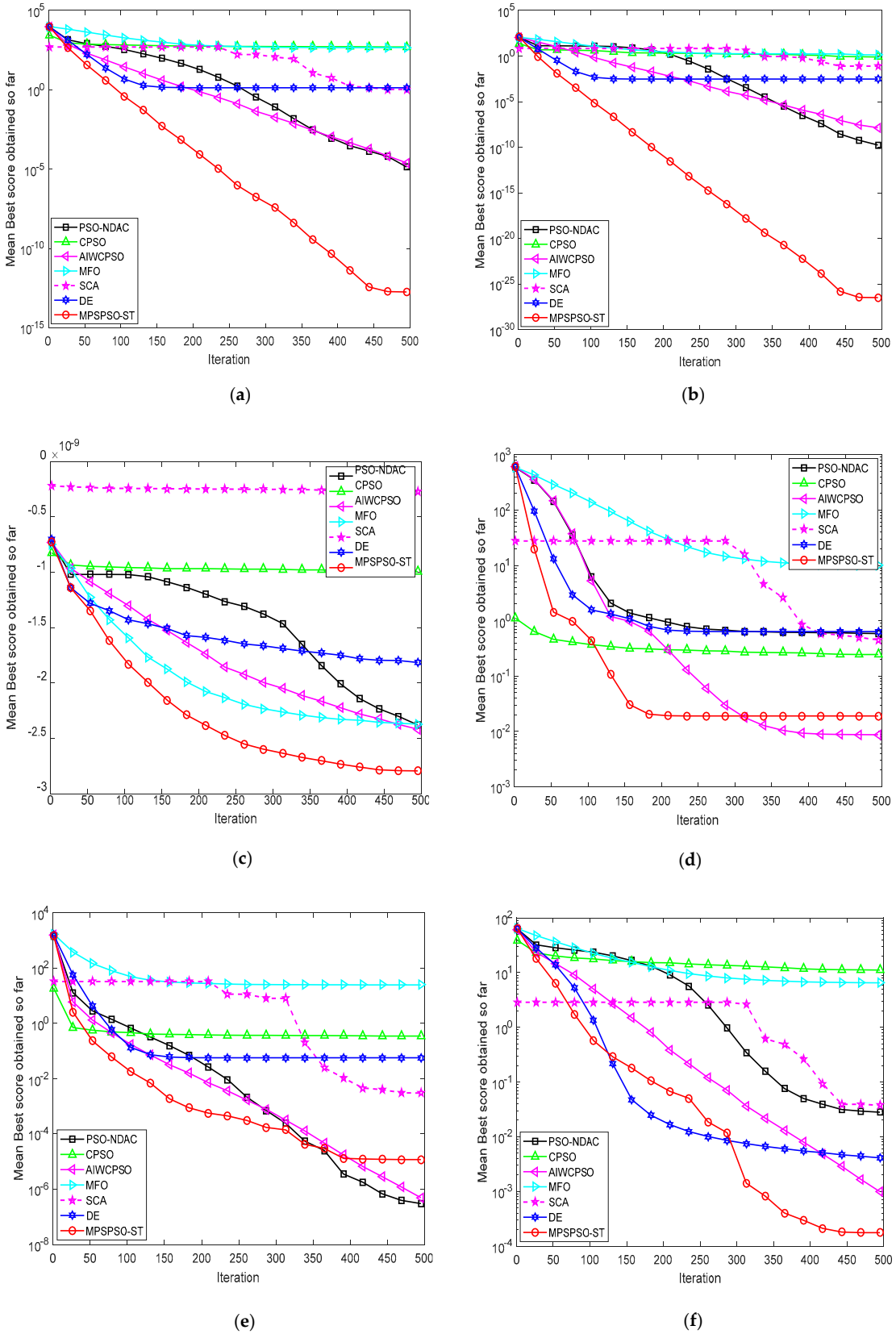

Figure 9. The convergence curves of seven algorithms for the functions: (**a**) f_7; (**b**) f_8; (**c**) f_{10}; (**d**) f_{11}; (**e**) f_{14}; (**f**) f_{15}.

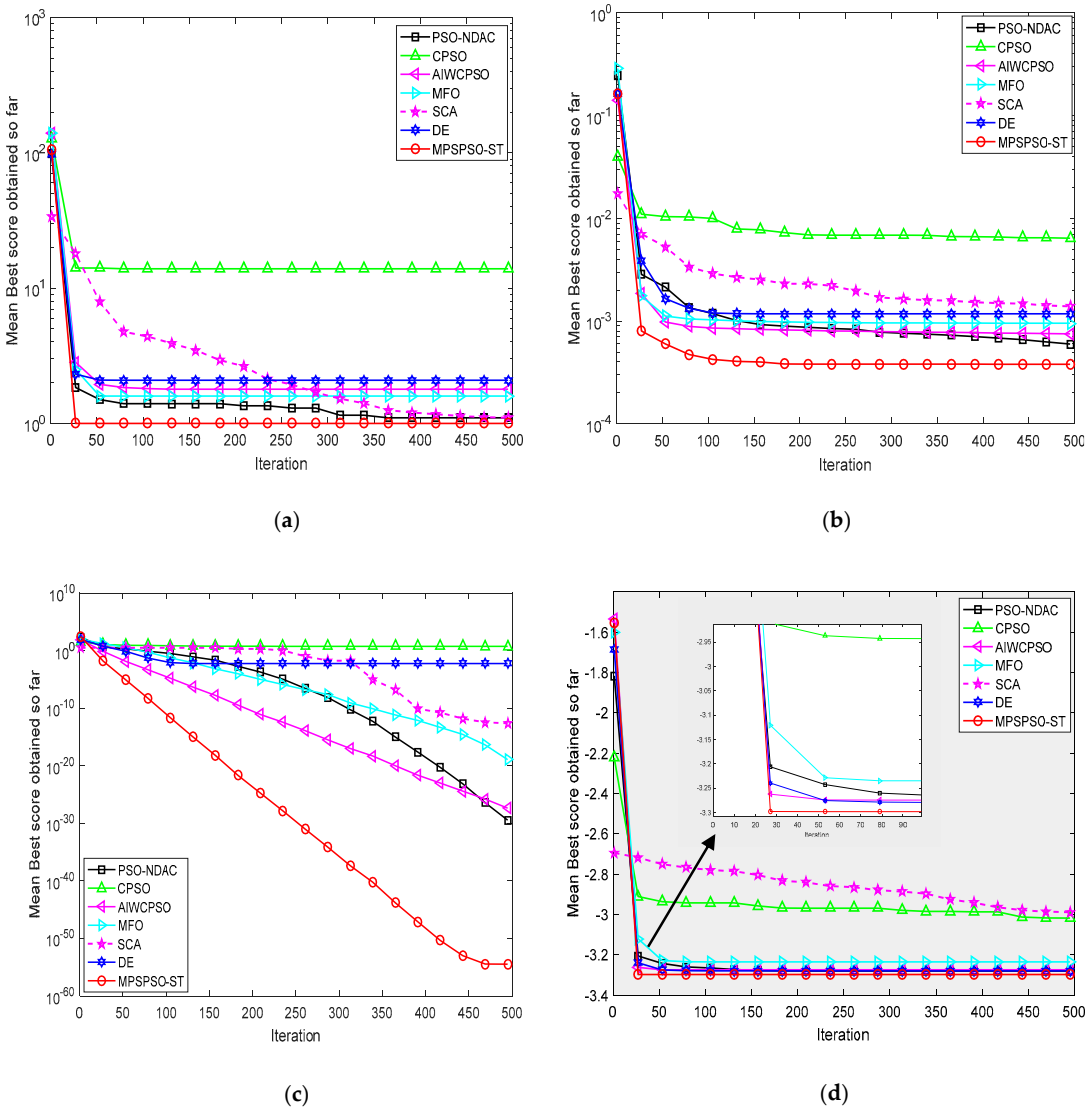

Figure 10. The convergence curves of seven algorithms for the functions: (**a**) f_{17}; (**b**) f_{18}; (**c**) f_{19}; (**d**) f_{20}.

5. Conclusions

In this paper, we propose the MPSPSO-ST algorithm, which can facilitate the algorithm performance of the traditional PSO. Firstly, we propose a hybrid multi-step probability selection update mechanism, i.e., the MPSPSO algorithm. Secondly, based on the MPSPSO algorithm, in order to achieve a good balance between exploration and exploitation to optimize the performance of the algorithm, we design the sine chaotic inertial weight and symmetric tangent chaotic acceleration coefficients, which are integrated into the MPSPSO-ST algorithm. Finally, to appraise the effectiveness of the MPSPSO-ST algorithm, we conduct extensive experiments with 20 classic benchmark functions and the experiments consisted of two groups. The first group of experiments is the comparisons of MPSPSO-ST with the Standard PSO, Basic PSO and MPSPSO. The experimental results show that the hybrid multi-step probability selection update mechanism proposed in this paper is very effective. The sine chaotic inertial weight and the symmetric tangent chaotic acceleration coefficients proposed on this basis can effectively maintain swarm diversity in the search process, which enhances the performance of MPSPSO-ST significantly. The second group of experiments is the comparisons of the MPSPSO-ST algorithm with three classical improved PSO variants (CPSO, PSO-NDAC, AIWCPSO) and three well-known classic optimization algorithms (DE, MFO, SCA). The experimental results show that the MPSPSO-ST algorithm surpasses all other six algorithms obviously on the majority of the classic

benchmark functions, with faster convergence speed, higher optimization accuracy, and greater stability robustness, which is successful in solving numerical optimization tasks. MPSPSO-ST has the basic characteristics of PSO, simple, easy to realize and other advantages while taking the search accuracy and search efficiency into account. It can further avoid premature convergence to a certain extent. Besides, MPSPSO-ST does not introduce any new parameters, making it more versatile and easier to operate and realize. Indeed, the proposed modification implies increasing complexity of the algorithm, but it is worthwhile in order to solve complex numerical optimization problems more efficiently.

Therefore, the MPSPSO-ST algorithm proposed in this paper is an excellent choice for solving complex numerical optimization problems. In the future, we will further develop the strategies of parameter tuning and study the application of the MPSPSO-ST algorithm in practical problems.

Author Contributions: Conceptualization, Y.D.; methodology, Y.D.; software, Y.D. and F.X.; formal analysis, F.X.; data curation, Y.D. and F.X.; writing—original draft preparation, Y.D.; writing—review and editing, Y.D. and F.X. All authors have read and agreed to the published version of the manuscript.

Funding: The authors have received no funding for this work.

Acknowledgments: The authors gratefully acknowledge the support of anonymous reviewers.

Conflicts of Interest: The authors declare that there are no conflicts of interest.

Appendix A

Table A1. The 20 multi-dimensional classical benchmark functions are given below.

ID	Test Function	Dim	Range	f_{min}	Type
f_1	$f(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0	Unimodal
f_2	$f(x) = \sum_{i=1}^{n} ix_i^4 + random[0,1]$	30	$[-1.28, 1.28]$	0	Unimodal
f_3	$f(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]$	0	Unimodal
f_4	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	30	$[-1, 1]$	0	Unimodal
f_5	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$	30	$[-10, 10]$	0	Unimodal
f_6	$f(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30	$[-100, 100]$	0	Unimodal
f_7	$f(x) = \sum_{i=1}^{n} ix_i^2$	30	$[-10, 10]$	0	Unimodal
f_8	$f(x) = \sum_{i=1}^{n} ix_i^4$	30	$[-1.28, 1.28]$	0	Unimodal
f_9	$f(x) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^{n} x_i^2}) + 0.1 \sqrt{\sum_{i=1}^{n} x_i^2}$	30	$[-100, 100]$	0	Multimodal
f_{10}	$f(x) = -\sum_{i=1}^{n} \sin(x_i) \cdot (\sin(\frac{ix_i^2}{\pi}))^{2m}, m = 10$	30	$[0, \pi]$	-4.687	Multimodal
f_{11}	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0	Multimodal
f_{12}	$f(x) = \frac{\pi}{n} \{10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4), y_i = 1 + \frac{x_i+1}{4} u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	30	$[-50, 50]$	0	Multimodal
f_{13}	$f(x) = 0.1 \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 + \sin^2(3\pi x_{i+1}) + (x_n - 1)^2 (1 + \sin^2(3\pi x_n))$	30	$[-5, 5]$	0	Multimodal
f_{14}	$f(x) = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} x_i^2$	30	$[-100, 100]$	0	Multimodal

Table A1. *Cont.*

ID	Test Function	Dim	Range	f_{min}	Type
f_{15}	$f(x) = \sum_{i=1}^{n} \left\lvert x_i \sin(x_i) + 0.1x_i \right\rvert$	30	[−10,10]	0	Multimodal
f_{16}	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2) \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[−2,2]	3	Multimodal
f_{17}	$f(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6})^{-1}$	2	[−65.536,65.536]	0.998	Multimodal
f_{18}	$f(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	[−5,5]	0	Multimodal
f_{19}	$f(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$	6	[−5,10]	0	Multimodal
f_{20}	$f(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	[0,1]	−3.32	Multimodal

References

1. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
2. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
3. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
4. Das, S.; Mullick, S.S.; Suganthan, P.N. Recent advances in differential evolution—An updated survey. *Swarm Evol. Comput.* **2016**, *27*, 1–30. [CrossRef]
5. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
6. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [CrossRef]
7. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
8. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
9. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci.* **2012**, *17*, 4831–4845. [CrossRef]
10. Ozturk, C.; Hancer, E.; Karaboga, D. A novel binary artificial bee colony algorithm based on genetic operators. *Inf. Sci.* **2015**, *297*, 154–170. [CrossRef]
11. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
12. Jain, I.; Jain, V.K.; Jain, R. Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification. *Appl. Soft Comput.* **2018**, *6*, 203–215. [CrossRef]
13. Zhang, H.; Lin, W.; Chen, A. Path planning for the mobile robot: A review. *Symmetry* **2018**, *10*, 450. [CrossRef]
14. Phoemphon, S.; So-In, C.; Niyato, D.T. A hybrid model using fuzzy logic and an extreme learning machine with vector particle swarm optimization for wireless sensor network localization. *Appl. Soft Comput.* **2018**, *65*, 101–120. [CrossRef]
15. Qin, T.C.; Zeng, S.K.; Guo, J.B.; Skaf, Z. State of health estimation of li-ion batteries with regeneration phenomena: A similar rest time-based prognostic framework. *Symmetry* **2017**, *9*, 4. [CrossRef]
16. Wu, J.P.; Lin, B.L.; Wang, H.; Zhang, X.H.; Wang, Z.K. Optimizing the high-level maintenance planning problem of the electric multiple unit train using a modified particle swarm optimization algorithm. *Symmetry* **2018**, *10*, 349. [CrossRef]
17. Wang, H.; Sun, H.; Li, C.H.; Rahnamayan, S.; Pan, J.S. Diversity enhanced particle swarm optimization with neighborhood search. *Inf. Sci.* **2013**, *223*, 119–135. [CrossRef]
18. Joines, J.A.; Houck, C.R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In Proceedings of the First IEEE Conference on Evolutionary Computation, Orlando, FL, USA, 27–29 June 1994; pp. 579–584.
19. Fogel, D.B. An Introduction to Simulated Evolutionary Optimization. *IEEE Trans. Neur. Netw.* **1994**, *5*, 3–14. [CrossRef]

20. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
21. Yang, J.; Zhu, H.; Wang, Y. An orthogonal multi-swarm cooperative PSO algorithm with a particle trajectory knowledge base. *Symmetry* **2017**, *9*, 15. [CrossRef]
22. Jensi, R.; Jiji, G.W. An enhanced particle swarm optimization with levy flight for global optimization. *Appl. Soft Comput.* **2016**, *43*, 248–261. [CrossRef]
23. Turgut, O.E. Hybrid chaotic quantum behaved particle swarm optimization algorithm for thermal design of plate fin heat exchangers. *Appl. Math. Model.* **2016**, *40*, 50–69. [CrossRef]
24. Qin, J.; Liu, Y.; Grosvenor, R.; Lacan, F.; Jiang, Z.G. Deep learning-driven particle swarm optimisation for additive manufacturing energy optimisation. *J. Clean. Prod.* **2020**, *245*, 118702. [CrossRef]
25. Shi, Y. Optimization of PID parameters of hydroelectric generator based on adaptive inertia weight PSO. In Proceedings of the IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 24–26 May 2019; pp. 1854–1857.
26. Tian, D.; Zhao, X.; Shi, Z. Chaotic particle swarm optimization with sigmoid-based acceleration coefficients for numerical function optimization. *Swarm Evol. Comput* **2019**, *51*, 100573. [CrossRef]
27. Arasomwan, M.A.; Adewumi, A.O. On adaptive chaotic inertia weights in particle swarm optimization. In Proceedings of the IEEE Symposium on Swarm Intelligence (SIS), Singapore, 16–19 April 2013; pp. 72–79.
28. Taherkhani, M.; Safabakhsh, R. A novel stability-based adaptive inertia weight for particle swarm optimization. *Appl. Soft Comput.* **2016**, *38*, 281–295. [CrossRef]
29. Zhang, C.S.; Ning, J.X.; Lu, S.A.; Ouyang, D.T.; Ding, T.A. A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. *Oper. Res. Lett.* **2009**, *37*, 117–122. [CrossRef]
30. Garg, H. A hybrid PSO-GA algorithm for constrained optimization problems. *Appl. Math. Comput.* **2016**, *274*, 292–305. [CrossRef]
31. Javidrad, F.; Nazari, M.; Javidrad, H.R. Optimum stacking sequence design of laminates using a hybrid PSO-SA method. *Compos. Struct.* **2018**, *185*, 607–618. [CrossRef]
32. Chen, K.; Zhou, F.; Wang, Y.; Yin, L. An ameliorated particle swarm optimizer for solving numerical optimization problems. *Appl. Soft Comput.* **2018**, *73*, 482–496. [CrossRef]
33. Bansal, J.C.; Deep, K. A modified binary particle swarm optimization for knapsack problems. *Appl. Math. Comput.* **2012**, *218*, 11042–11061. [CrossRef]
34. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the IEEE International Conference on Evolutionary Computation, Anchorage, AK, USA, 4–8 May 1998; pp. 69–73.
35. Arumugam, M.S.; Rao, M.V.C. On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems. *Discrete Dyn. Nat. Soc.* **2006**. [CrossRef]
36. Datta, D.; Figueira, J.R. A real-integer-discrete-coded particle swarm optimization for design problems. *Appl. Soft Comput.* **2011**, *11*, 3625–3633. [CrossRef]
37. Datta, D.; Figueira, J.R. Graph partitioning by multi-objective real-valued metaheuristics: A comparative study. *Appl. Soft Comput.* **2011**, *11*, 3976–3987. [CrossRef]
38. Gao, F.; Cui, G.; Wu, Z.; Yang, X. A novel multi-step position-selectable updating particle swarm optimization algorithm. *Acta Electron. Sin.* **2009**, *37*, 529–537.
39. Ali, M.M.; Kaelo, P. Improved particle swarm algorithms for global optimization. *Appl. Math. Comput.* **2008**, *196*, 578–593. [CrossRef]
40. Lipowski, A.; Lipowska, D. Roulette-wheel selection via stochastic acceptance. *Phys. A Stat. Mech. Appl.* **2012**, *391*, 2193–2196. [CrossRef]
41. Thammano, A.; Teekeng, W. A modified genetic algorithm with fuzzy roulette wheel selection for job-shop scheduling problems. *Int. J. Gen. Syst.* **2015**, *44*, 499–518. [CrossRef]
42. Ho-Huu, V.; Nguyen-Thoi, T.; Truong-Khac, T.; Le-Anh, L.; Vo-Duy, T. An improved differential evolution based on roulette wheel selection for shape and size optimization of truss structures with frequency constraints. *Neural. Comput. Appl.* **2018**, *29*, 167–185. [CrossRef]
43. Peng, Y.; Peng, X.Y.; Liu, Z.Q. Statistic analysis on parameter efficiency of particle swarm optimization. *Acta Electron. Sin.* **2004**, *32*, 209–213.

44. Ikeguchi, T.; Sato, K.; Hasegawa, M. Chaotic Optimization for Quadratic Assignment Problems. In Proceedings of the 2002 IEEE International Symposium on Circuits and Systems, Phoenix-Scottsdale, AZ, USA, 26–29 May 2002; pp. 469–472.
45. Hayakawa, Y.; Marumoto, A.; Sawada, Y. Effects of the Chaotic Noise on the Performance of a Neural Netwok Model for Optimization Problems. *Phys. Rev. E* **1995**, *51*, 2693–2696. [CrossRef]
46. Feng, Y.; Teng, G.F.; Wang, A.X.; Yao, Y.M. Chaotic inertia weight in particle swarm optimization. In Proceedings of the 2007 Second International Conference on Innovative Computing, Information and Control, Kumamoto, Japan, 5–7 September 2007; pp. 1899–1902.
47. Bansal, J.C.; Singh, P.K.; Saraswat, M.; Verma, A.; Jadon, S.S.; Abraham, A. Inertia weight strategies in particle swarm optimization. In Proceedings of the 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 19–21 October 2011; pp. 633–640.
48. Wang, G.G.; Guo, L.H.; Gandomi, A.H.; Hao, G.S.; Wang, H.Q. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
49. Niu, P.; Chen, K.; Ma, Y.; Li, X.; Liu, A.; Li, G. Model turbine heat rate by fast learning network with tuning based on ameliorated krill herd algorithm. *Knowl. Based Syst.* **2017**, *118*, 80–92. [CrossRef]
50. Chaturvedi, K.T.; Pandit, M.; Srivastava, L. Particle swarm optimization with time varying acceleration coefficients for non-convex economic power dispatch. *Int. J. Electron. Power* **2009**, *31*, 249–257. [CrossRef]
51. Chen, K.; Zhou, F.; Liu, A. Chaotic dynamic weight particle swarm optimization for numerical function optimization. *Knowl. Based Syst.* **2018**, *139*, 23–40. [CrossRef]
52. Elaziz, M.A.; Mirjalili, S. A hyper-heuristic for improving the initial population of whale optimization algorithm. *Knowl. Based Syst.* **2019**, *172*, 42–63. [CrossRef]
53. Liu, J.; Gao, Y. Chaos particle swarm optimization algorithm. *Comput. Sci.* **2004**, *31*, 13–15. [CrossRef]
54. Li, T.H.S.; Kuo, P.H.; Ho, Y.F.; Liou, G.H. Intelligent control strategy for robotic arm by using adaptive inertia weight and acceleration coefficients particle swarm optimization. *IEEE Access* **2019**, *7*, 126929–126940. [CrossRef]