


A Novel Simple Particle Swarm Optimization Algorithm for Global Optimization

Xin Zhang , Dexuan Zou *  and Xin Shen 

School of Electrical Engineering & Automation, Jiangsu Normal University, Xuzhou 221116, China; 2020160848@jsnu.edu.cn (X.Z.); 2020160838@jsnu.edu.cn (X.S.)

* Correspondence: 6020110007@jsnu.edu.cn; Tel.: +86-181-2003-0371

Received: 28 October 2018; Accepted: 19 November 2018; Published: 27 November 2018



Abstract: In order to overcome the several shortcomings of Particle Swarm Optimization (PSO) e.g., premature convergence, low accuracy and poor global searching ability, a novel Simple Particle Swarm Optimization based on Random weight and Confidence term (SPSORC) is proposed in this paper. The original two improvements of the algorithm are called Simple Particle Swarm Optimization (SPSO) and Simple Particle Swarm Optimization with Confidence term (SPSOC), respectively. The former has the characteristics of more simple structure and faster convergence speed, and the latter increases particle diversity. PSORC takes into account the advantages of both and enhances exploitation capability of algorithm. Twenty-two benchmark functions and four state-of-the-art improvement strategies are introduced so as to facilitate more fair comparison. In addition, a *t*-test is used to analyze the differences in large amounts of data. The stability and the search efficiency of algorithms are evaluated by comparing the success rates and the average iteration times obtained from 50-dimensional benchmark functions. The results show that the SPSO and its improved algorithms perform well comparing with several kinds of improved PSO algorithms according to both search time and computing accuracy. PSORC, in particular, is more competent for the optimization of complex problems. In all, it has more desirable convergence, stronger stability and higher accuracy.

Keywords: particle swarm optimization; confidence term; random weight; benchmark functions; *t*-test; success rates; average iteration times

1. Introduction

Since the 1950s, heuristic algorithms based on evolutionary algorithms (EAs) [1] have sprung up and been widely applied to the field of optimization control, such as moth search (MS) algorithm [2,3], genetic algorithm (GA) [4], ant colony optimization (ACO) algorithm [5], differential evolution (DE) algorithm [6], simulated annealing (SA) algorithm [7], krill herd (KH) algorithm, etc. [8–12]. Compared with traditional optimization methods such as golden section [13], Newton method [14,15], gradient method [16], heuristic algorithms have better biological characteristics and higher efficiency. It has been proved that heuristic algorithms perform well in some advanced existing fields e.g., grid computing [17], the superfluid management of 5G Networks [18], TCP/IP Mobile Cloud [19], IIR system identification [20], etc.

The Particle Swarm Optimization (PSO) algorithm proposed by Kennedy and Eberhart [21,22] in 1995 is also a member of the heuristic algorithm. Unlike other EAs, PSO does not require such steps as crossover, mutation, and selection, and it has fewer parameters. Its optimization process relies entirely on formula iteration, hence its calculation burden is low. The efficiency is very high, especially for continuous unimodal function model optimization. Due to these advantages, it has been widely used

in various theoretical and practical problems such as function optimization [23], Non-Deterministic Polynomial(NP) problem [24], and multi-objective optimization [25].

PSO is a typical algorithm that relies on swarm intelligence [26–31] to optimize complex problems, and it is inspired by the foraging behavior of birds. It can be imagined that a group of gold rushers find gold in a region. They all have instruments that can detect gold mines under the stratum, and they can communicate with their nearest gold rushers. Through communication, they can know whether the person next to them finds gold. At the beginning, in order to explore this area more comprehensively, they randomly select a location to explore and maintain a certain distance. As the exploration begins, if someone finds some gold, the neighboring gold rushers can choose whether to change his position based on his own experience and whether he trusts him. This constant search may make it easier to find more gold than to be alone. In this example, a group of gold rushers and the gold are, respectively, equivalent to the particles of PSO and the optima that needs to be searched.

In actual operation, it is observed that PSO is very prone to premature convergence and falls into the local optima when faced with multimodal functions, especially some ones with traps or discontinuities. Based on this observation, a huge amount of particle swarm optimization variants have been proposed to deal with these issues. From the literature, it can be clearly observed that most of the existing PSO algorithms can be roughly divided into six categories: principle study, parameter setting, topology improvement, updating formula improvement, hybrid mechanism, practical application.

1. **Principle study:** The inertia weight factor, which adjusts the ability of PSO algorithm in local and global search was introduced by Shi and Eberhart [32], effectively avoiding falling into local optimum for PSO. Shi and Eberhart provided a way of thinking for future improvement. In 2001, Parsopoulos and Vrahatis [33]’s research showed that basic PSO can work effectively and stably in noisy environments, and in many cases, the presence of noise can also help PSO avoid falling into local optimum. The basic PSO was introduced for continuous nonlinear function [21,22]. However, because the basic PSO is easy to fall into the local optima, local PSO(LPSO) [34] was introduced in 2002. Clerc and Kennedy [35] proposed a constriction factor to enhance the explosion, stability, and convergence in a multidimensional complex space. Xu and Yu [36] used the super-martingale convergence theorem to analyze the convergence of the particle swarm optimization algorithm. The results showed that the particle swarm optimization algorithm achieves the global optima in probability and the quantum-behaved particle swarm optimization (QPSO) [37] has also been proved to have global convergence.
2. **Parameter setting:** A particle swarm optimization with fitness adjustment parameters (PSOFAB) [38], based on the fitness performance, was proposed in order to converge to an approximate optimal solution. The experimental results were analyzed by the Wilcoxon signed rank test, and its analysis showed that PSOFAP [38] is effective in increasing the convergence speed and the solution quality. It accurately adapts the parameter value without performing parametric sensitivity analysis. The inertia weight of the hybrid particle swarm optimization incorporating fuzzy reasoning and weighted particle (HPSOFW) [39] is changed based on defuzzification output. The chaotic binary PSO with time-varying acceleration coefficients (CBPSOTVAC) [40] using 116 benchmark problems from the OR-Library to test has time-varying acceleration coefficients for the multidimensional knapsack problem. A self-organizing hierarchical PSO [41] also uses time-varying acceleration coefficients.
3. **Topology improvement:** In 2006, Kennedy and Mendes [42] explained the neighborhood topologies in fully informed and best-of-neighborhood particle swarms in detail. A dynamic multiswarm particle swarm optimizer (DMSPSO) [43] was proposed, and it adopts a neighborhood topology including a random selection of small swarms with small neighborhood. Moreover, the regrouped group is dynamic and randomly assigned. In 2014, FNNPSO [44] use Fluid Neural Networks (FNNs) to create a dynamic neighborhood mechanism. The results showed that FNNPSO can outperform both the standard PSO algorithm and PCGT-PSO. Sun and Li proposed a two-swarm cooperation particle swarm optimization (TCPSO) [45] that used the slave swarm and

the master swarm to exchange the information, which is beneficial for enhancing the convergence speed and maintaining the swarm diversity in TCP SO, and particles update the next particle with information from its neighboring particles, rather than its own history best solution and current velocity. This strategy makes the particles of the subordinate group more inclined to local optimization, thus accelerating convergence. Inspired by the cluster reaction of the starlings, Netjinda et al. [46] used the collective response mechanism to influence the velocity and position of the current particle by seven adjacent ones, thereby increasing the diversity of the particles. A nonparametric particle swarm optimization (NP-PSO) [47] combines local and global topologies with two quadratic interpolation operations to enhance the PSO capability without tuning any algorithmic parameter.

4. **Updating formula improvement:** Mendes [48] changed the PSO's velocity and personal best solution updating formula and proposed a fully informed particle swarm (FIPS) algorithm to make good use of the whole entire swarm. Mendes [49] proposed a Comprehensive learning particle swarm optimizer (CLPSO) whose velocity updating formula eliminates the influence from global best solution to suit the multimodal functions, and CLPSO uses two tournament-selected particles to help particles study better case during iteration. The results showed that CLPSO performs better than other PSO variants for multimodal problems. A learning particle swarm optimization (*LPSO) algorithm [50] was proposed with a new framework that changed the velocity updating formula so as to organically hybridize PSO with another optimization technique. *LPSO is composed of two cascading layers: exemplar generation and a basic PSO algorithm updating method. A new global particle swarm optimization (NGPSO) algorithm [51] uses a new position updating equation that relies on the global best particle to guide the searching activities of all particles. In the latter part of the NGPSO search, the random distribution based on uniform distribution is used to increase the particle swarm diversity and avoid premature convergence. Kiran proposed a PSO with a distribution-based position update rule (PSOd) [52] whose position updating formula is combined with three variables.
5. **Hybrid mechanism:** In 2014, Wang et al. [53] proposed a series of chaotic particle-swarm krill herd (CPKH) algorithms for global numerical optimization. The CPKH is a hybrid Krill herd (KH) [54] algorithm with APSO [55] which has a mutation operator and chaotic theory. This hybrid algorithm, which with an appropriate chaotic map performs superiorly to the standard KH and other population-based optimization, has quick exploitation for solution. DPSO [56] is an accelerated PSO (APSO) [55] algorithm hybridized with a DE algorithm mutation operator. It has a superior performance due to combining the advantages from both APSO and DE. Wang et al., finally, studied and analyzed the effect of the DPSO parameters on convergence and performance by detailed parameter sensitivity studies. In the hybrid learning particle swarm optimizer with genetic disturbance (HLPSO-GD) [57], the genetic disturbance is used to cross the corresponding particle in the external archive, and new individuals are generated, which will improve the swarm's ability to escape from the local optima. Gong et al. proposed a genetic learning particle swarm optimization (GLPSO) algorithm that uses genetic evolution to breed promising exemplars based on *LPSO [50] enhancing the global search ability and search efficiency of PSO. PSOTD [58] namely a particle swarm optimization algorithm with two differential mutation, which has a novel structure with two swarms and two layers including bottom layer and top layer, was proposed for 44 benchmark functions. HNPPSO [59] is a novel particle swarm optimization combined with a multi-crossover operation, a vertical crossover, and an exemplar-based learning strategy. To deal with production scheduling optimization in foundry, a hybrid PSO combined the SA [7] algorithm [60] was proposed.
6. **Practical application:** Zou et al. used NGPSO [51] to solve the economic emission dispatch (EED) problems and the results showed that NGPSO is the most efficient approach for solving the economic emission dispatch (EED) problems. PS-CTPSO [61] based on the predatory search strategy was proposed to do with web service combinatorial optimization, which is an NP

problem, and it improves overall ergodicity. To improve the changeability of ship inner shell, IPSO [62] was proposed for a 50,000 DWT product oil tanker. MBPSO [63] was proposed for sensor management of LEO constellation to the problem of utilizing a low Earth orbit (LEO) infrared constellation in order to track the midcourse ballistic missile. GLNPSO [64] is for a capacitated Location-Routing Problem. The particle swarm algorithm is also applied to many other practical problems e.g., PID (Proportion Integration Differentiation) controller [65], optimal strategies of energy management integrated with transmission control for a hybrid electric vehicle [66], production scheduling optimization in foundry [60], etc.

In view of the shortcomings of PSO [21,22], three improvements are proposed in this paper. The first is Simple Particle Swarm Algorithm (SPSO). It does not use the velocity updating formula, and abandons the use of self-cognitive term. Although the speed of the algorithm has been greatly improved, some deficiencies have been found in actual tests. It is observed that the particles' difference is too small to jump out of the local optimal solution, which is not suitable for searching for multimodal problems. For this purpose, a second improvement named Simple Particle Swarm Optimization with Confidence Term (SPSOC) is proposed in this paper. That is, the confidence term is introduced in the SPSO's position updating formula. Although having a slight increase in time compared to SPSO, the results show that SPSOC is better for multi-peak function optimization. On the basis of this, the inertia weight is improved by introducing the difference between the stochastic objective function value and the worst one, and the final improvement is called Simple Particle Swarm Optimization based on Random weight and Confidence term (SPSORC). The inertial weight not only has a crucial effect on its convergence, but also plays an important role in balancing exploration and exploitation during the evolution. The strategy in this paper makes particle position movements more random. A large number of experiments suggest that all three improvements are very effective, and the combination of the three improvements has greatly improved the search efficiency of the particle swarm algorithm.

The rest of this paper is organized as follows: Section 2 introduces the basic PSO [21,22] and three recently improved PSO methods. In Section 3, three improvements are presented in detail. In Section 4, some analysis of PSO is further discussed. The experimental results are discussed and analyzed between four state-of-the-art PSOs and three improved ones proposed in this paper. Finally, this paper presents some important conclusions and the outlook of future work in Section 5.

2. Related Works

2.1. The Basic PSO

In general, the particle swarm optimization algorithm is composed of the position updating formula and the velocity updating formula. Each particle iterates with reference to its own history best solution p_{best} and the global best value g_{best} to change position and velocity information. The basic particle swarm optimization (bPSO) [21,22] algorithm iteration formula is as follows:

$$v_{in}^{t+1} = v_{in}^t + c_1 r_1 (p_{best}^t - x_{in}^t) + c_2 r_2 (g_{best}^t - x_{in}^t), \quad (1)$$

$$x_{in}^{t+1} = x_{in}^t + v_{in}^{t+1}. \quad (2)$$

As shown in the above formula, Equations (1) and (2) are the velocity updating formula and the position updating formula, respectively. The particles whose population is m search for the optima in the n -dimensional space. In that process, the i -th particle's position in the n -dimensional space is x_{in} and the current velocity is v_{in} . p_{best} is the individual history best solution and g_{best} is the global one. t is the current iteration numbers. c_1 and c_2 are cognitive and social factors, and r_1 and r_2 are random numbers belonging to $[0,1)$. Figure 1 is an optimization procedure of PSO.

From Figure 1, the area U is the solution space of a function. O is the theoretical optima that needs to be found. x_i^t is the position of the initial particle. The velocity v_i^t is the current particle velocity. v_i^{t+1} is the velocity after being affected by various aspects. Particle memory influence and swarm

influence are parallel to the connecting lines from x_i^t to g_{best} and p_{best} , respectively, indicating the influence from g_{best} and p_{best} . In this generation, the particle i is affected by v_i^t first. After particle memory influence and swarm influence, i arrives at x_i^{t+1} from x_i^t at velocity v_i^{t+1} . From the next iteration, the particle will move from x_i^{t+1} towards the new position. It keeps iterating as the step above and moves to the theoretical optima more and more close.

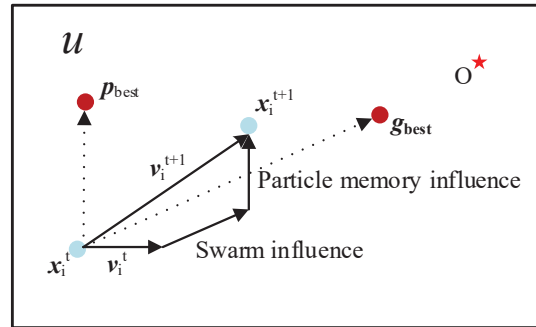


Figure 1. Optimization procedure of bPSO.

The velocity updating formula had changed to Equation (3), when Shi and Eberhart put the inertia weight ω into it, and the position updating formula remained unchanged:

$$v_{in}^{t+1} = \omega v_{in}^t + c_1 r_1 (p_{best}^t - x_{in}^t) + c_2 r_2 (g_{best}^t - x_{in}^t). \quad (3)$$

The introduction of inertia weight effectively keeps a balance between the local and global search capability. The larger the inertia weight, the stronger the global search capability of the algorithm. On the contrary, the local search capability is more prominent. This particle swarm optimization model is the most commonly used nowadays, and many scholars have improved it.

The steps to achieve it are as follows:

- Step 1:** Initialize the population randomly. Set the maximum number of iterations, population size, inertia weight, cognitive factors, social factors, position limits and the maximum velocity limit.
- Step 2:** Calculate the fitness of each particle according to fitness function or models.
- Step 3:** Compare the fitness of each particle with its own history best solution p_{best} . If the fitness is smaller than p_{best} , the smaller value is assigned to p_{best} , otherwise, p_{best} is reserved. Then, the fitness is compared with the global best solution g_{best} , and the method is the same as selecting p_{best} .
- Step 4:** Use Equations (2) and (3) to update the particle position and velocity. In addition, we must make sure that its velocity and position are, respectively, within the maximum velocity limit and position limits.
- Step 5:** Check if the theoretical optimum is reached, output the value and stop the operation; otherwise, return to **Step 2** (Section 2.1) until it reaches the theoretical optima or peaks the maximum number of iterations.

In this paper, a basic particle swarm optimization with decreasing linear inertia weight is used. The weight formula is as Equation (4):

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{T} \times t. \quad (4)$$

In Equation (4), ω_{max} is starting weight. ω_{min} is final weight. t_{max} is the maximum number of iterations. PSO needs to set a still more larger starting weight ω_{max} according to the influence of inertia weight on the search capability of PSO, so as to pay more attention to the global optima. As the number of iterations increases, the weight will be decreased. The search process would be more inclined to explore the local optima, which is more conducive to the final convergence.

2.2. The PSO with a Distribution-Based Position Update Rule

In 2017, a distribution-based update rule for PSO (PSOd) [52] algorithm was proposed by Kiran. This improved strategy changed PSO's iteration formula.

$$x_{in}^{t+1} = \mu + \sigma \times Z. \quad (5)$$

Those three variables in Equation (5) work by Equations (6)–(8):

$$\mu = \frac{x_{in}^t + p_{best}^t + g_{best}^t}{3}, \quad (6)$$

$$\sigma = \sqrt{\frac{(x_{in}^t - \mu)^2 + (p_{best}^t - \mu)^2 + (g_{best}^t - \mu)^2}{3}}, \quad (7)$$

$$Z = (-2 \ln k_1)^{\frac{1}{2}} \times \cos(2\pi k_2). \quad (8)$$

It works as follows:

Step 1: The population is initialized randomly.

Step 2: The fitness is calculated and compared to get the best individual history solution and the best global one.

Step 3: Equation (5) is used to update the particle position that is limited in the upper and lower limits.

Step 4: If the termination condition is met, the best solution is reported.

2.3. A Hybrid PSO with Sine Cosine Acceleration Coefficients

In order to make better use of parameters on PSO algorithm, such as inertia weight, learning factors, etc., Chen et al. proposed a hybrid PSO algorithm with the sine cosine acceleration coefficients (HPSOscac) [67].

Step 1: The population is initialized randomly.

Step 2: The reverse population of the initial population is calculated by Equation (9)

$$x'_{in} = x_{max} + x_{min} - x_{in}. \quad (9)$$

In this equation, x_{in} and x'_{in} are initial population and reverse population, respectively. x_{max} and x_{min} are combined the upper and lower limits of particles position i.e., the solution space boundary.

Step 3: Fitness values of those two populations are sorted, and the best half is used as the initial population. Then, the p_{best} and g_{best} are obtained by comparing.

Step 4: Equations (10) and (11) are used to update the inertia weight and learning factors, respectively:

$$\begin{cases} \omega^{t+1} = \frac{c}{4} \times \sin(\pi\omega^t), \\ \omega^1 = 0.4; c \in (0, 4], \end{cases} \quad (10)$$

$$\begin{cases} c_1 = 2 \times \sin((1 - \frac{t}{T}) \times \frac{\pi}{2}) + 0.5, \\ c_2 = 2 \times \cos((1 - \frac{t}{T}) \times \frac{\pi}{2}) + 0.5. \end{cases} \quad (11)$$

Among them, c is a constant among 0 and 4. c_1 and c_2 are cognitive and social factors, respectively.

Step 5: Updating the particle velocity and position, use Equations (1) and (12). The particle position updating formula is as follows:

$$x_{in}^{t+1} = x_{in}^t \times W_{in}^t + v_{in}^t \times W_{in}^{t'} + \rho \times g_{best}^t \times W_{in}^t. \quad (12)$$

W_{in}^t and $W_{in}^{t'}$ are the dynamic weights that control position and velocity terms. Its formula is like Equation (13). ρ is a random value between 0 and 1:

$$\begin{cases} W_{in}^t = \frac{\exp \frac{f_i}{f_{avg}}}{1 + \exp \frac{-f_i}{f_{avg}}}, \\ W_{in}^{t'} = 1 - W_{in}^t. \end{cases} \quad (13)$$

In this formula, f_i is the particle fitness value, and f_{avg} is the average one.

Step 6: The iteration is ended if end condition is reached. Otherwise, it comes back to **Step 2** (Section 2.3).

2.4. A Two-Swarm Cooperative PSO

A two-swarm cooperative particle swarm optimization (TCPSO) [45] was proposed who uses two particle swarms, the slave swarm and the master swarm with the clear division of their works to overcome the shortcomings such as lack of diversity, slow convergence in the later period, etc. It works like the following:

Step 1: Initialization. Initialize the slave swarm and the master swarm's velocity and position randomly.

Step 2: Calculate the fitness of these two swarms and get the g_{best}^S , p_{best}^S , g_{best} and p_{best}^M . The first two come from the slave swarm and the last two come from the master swarm.

Step 3: Reproduction and updating.

Step 3.1: Update the slave swarm by Equations (14) and (15). Ensure that velocity and position are within the limits:

$$v_{in}^{S,t+1} = c_1^S r_1 (1 - r_2) (x_{kn}^{S,t} - x_{in}^{S,t}) + c_2^S (1 - r_1) r_2 (g_{best} - x_{in}^{S,t}), \quad (14)$$

$$x_{in}^{S,t+1} = x_{in}^{S,t} + v_{in}^{S,t+1}. \quad (15)$$

S in these two formulas means that this variable from the slave swarm, except g_{best} in Equation (14) from the master swarm. Finally, we will get the g_{best}^S . x_k is randomly chosen from the neighborhood of the x_i according to Equation (16) [42]:

$$k \in \begin{cases} [i - \frac{l}{2} + 1, i + \frac{l}{2}], & \text{if } l \text{ is even,} \\ [i - \frac{l-1}{2}, i + \frac{l-1}{2}], & \text{if } l \text{ is odd.} \end{cases} \quad (16)$$

l is the size of neighborhood. Sun and Li found that the size of neighborhood equal to 2 is best in their experiments.

Step 3.2: Update the master swarm by Equations (17) and (18). Ensure that velocity and position are within the limits:

$$v_{in}^{M,t+1} = \omega^M v_{in}^{M,t} + c_1^M r_1 (1 - r_2) (1 - r_3) (p_{best}^M - x_{in}^{M,t}) + c_2^M r_2 (1 - r_1) (1 - r_3) (g_{best}^S - x_{in}^{M,t}) \\ + c_3^M r_3 (1 - r_1) (1 - r_2) (g_{best} - x_{in}^{M,t}) \quad (17)$$

$$x_{in}^{M,t+1} = x_{in}^{M,t} + v_{in}^{M,t+1}. \quad (18)$$

M here means that this variable is from the master swarm. In the end of **Step 3.2** (Section 2.4), g_{best} will be obtained for the next iteration.

Step 4: Get the optima if it meets the termination condition; otherwise, go to **Step 2** (Section 2.4).

3. SPSO, SPSOC, SPSORC

3.1. Simple PSO

Zou et al. proposed a novel harmony search algorithm [68] that used the optimal harmony and worst harmony in the harmony memory to guide the configuration of the harmony vector. It obtained very suitable results. Inspired by its thoughts, we try to round off the velocity formula and cognitive term of PSO and directly use the social term to control the algorithm optimization, so that the formula Equation (21) is the most simplified, namely the Simple Particle Swarm Optimization (SPSO) algorithm. According to the results of the literature [69], the influence of the velocity term on the performance of the particle swarm algorithm can be neglected. Drawing on literature [69], we can simply do the following derivation. Before abandoning the velocity updating formula, SPSO velocity updating formula is shown as follows. Particles' positions are updated according to Equation(2):

$$v_{in}^{t+1} = \omega v_{in}^t + cr(g_{best}^t - x_{in}^t). \quad (19)$$

According to Equations (19) and (2), we make the following assumptions:

Hypothesis 1. *The update of particles per dimension is independent from each other, except that g_{best} is the one that connects the information to the other dimensions.*

Hypothesis 2. *When particle i is updated, the other particles' velocities and positions are not changed.*

Hypothesis 3. *The particles' positions are moving continuously.*

According to the above assumptions, it is only necessary to prove a certain dimension of a certain particle search process that can be universal. Iterating over Equations (19) and (2) yields a second-order differential equation:

$$x^{t+2} + (rc - \omega - 1)x^{t+1} + \omega x^t = rcg_{best}^t. \quad (20)$$

We can observe that there is no velocity updating in Equation (20). This result can be applied to each dimension update of other particles. Now, we get SPSO's updating formula:

$$x_{in}^{t+1} = \omega x_{in}^t + cr(g_{best}^t - x_{in}^t). \quad (21)$$

SPSO only uses this formula to iterate. The experimental results show that this strategy improves the search efficiency and stability of the bPSO.

It works like the following:

- Step 1:** The maximum generation, population number, inertia weight, learning factor are set up. Population is initialized.
- Step 2:** Fitness is calculated according to the function.
- Step 3:** Every particle compares with its history best solution to get the p_{best} and compares with the global best one to get the g_{best} .
- Step 4:** Particle position is updated by Equation (21).
- Step 5:** If the theoretical optimal value is not found, the program returns to **Step 2** (Section 3.1); otherwise, the program stops.

After changing, the particle direction is only affected by the global optima. Graphical display of one of the particle optimization process is shown in Figure 2.

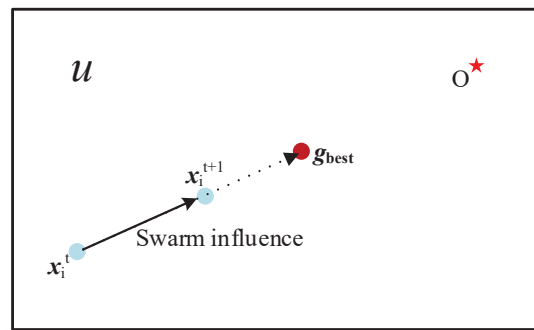


Figure 2. Optimization procedure of SPSO.

As shown in the figure, compared with the bPSO optimization process diagram in Section 2.1, in the optimization of SPSO, the particles are only affected by g_{best} and the direction of the particles always faces g_{best} . This feature also brings some drawbacks. For example, whether the algorithm can or cannot search for the theoretical optima depends entirely on the selection position of the global optima, which makes it likely for particles develop in a certain local optimal direction. It is possible to reach the current g_{best} value directly if the movement is fast enough. This is a search trajectory of one, while when all particles are only optimized in one direction, it obviously reduces the difference between the population. The lack of diversity directly leads to the fact that SPSO are easily trapped in local optimal solutions.

What is gratifying is that SPSO is very fast because of the simplification. This is very suitable for single-peak problems. This advantage can be clearly reflected in the experimental results in Section 4. However, the unconstrained functions, especially single-peak problems, are a minority after all. In order to make this improvement apply into more functions or environment, we propose adding a confidence term so that some part of the particles can determine the distance to advance based on its own level of trust to g_{best} , so as to get rid of the defects that all particles are looking for at one point.

3.2. SPSO with Confidence Term

In order to better solve the multimodal problem and make the improvement universal, we decided to add a confidence item (SPSOC) that rewrites Equation (21) into Equation (22):

$$x_{in}^{t+1} = \omega_1 x_{in}^t + cr_1(g_{best}^t - x_{in}^t) - \omega_2 r_2 g_{best}^t. \quad (22)$$

Compared with SPSO, the algorithm formula adds one item, namely the confidence term. ω_2 is the inertia weight of the confidence term. r_2 is the random value between $[0, 1]$.

Referring to Figure 3, the principle of the item can be understood as: at a certain iteration, the position calculated by the SPSO moves a distance suffered from confidence influence. The effect is equivalent to the particle being optimized from x_i^t to $x_i'^{t+1}$. Then, the particle retreats a distance from the beginning in the opposite direction to the g_{best} direction. Finally, this particle reaches the position of x_i^{t+1} . Using the inertia weight ω_2 and the random number r_2 , the distance of the particles retreating in the opposite direction would be uncertain. It can be imagined that the degree of particles trust at different generation is different, that is, the influence of g_{best} is different. This improvement can effectively slow the convergence of particles, so that the particles are not too dense, thus maintaining particle diversity.

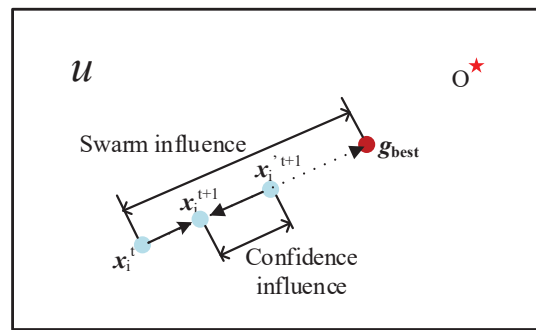


Figure 3. Optimization procedure of SPSOC.

A discussion of the impact of this improved algorithm using a combination of different weights will be explained in the experiment of Section 4.4. In order to minimize the program running time and ensure that the program structure is simple and the effect is optimal, this paper makes $\omega_1 = \omega_2$. SPSOC's iteration process is the same as SPSO.

3.3. SPSOC Based on Random Weight

Adding a confidence item to the SPSO does significantly enhance the search ability of the algorithm, but it does not achieve theoretical optimization when searching for most of the benchmark functions. Compared with many improved PSOs proposed recently, SPSOC has no big advantage except for the short amount of time. Therefore, we think about randomization improvement of inertia weight named SPSOC based on random weight (SPSORC). The improved inertia weight formula is shown in Equation (23):

$$\omega = \begin{cases} \frac{p_{best}^r - f_{best}}{f_{worst} - f_{best}}, & \text{if set minimum as target,} \\ \frac{f_{best} - p_{best}^r}{f_{best} - f_{worst}}, & \text{if set maximum as target.} \end{cases} \quad (23)$$

In this formula, if we set the minimum as the target we want to find, f_{best} is the minimum fitness in the current iteration, f_{worst} is the worst fitness target value in the current iteration, and p_{best}^r is one of the most p_{best} that a random particle has searched for from total swarm.

The use of Equation (23) allows the weights to be generated randomly, which effectively reduces the possibility that the algorithm falls into a local solution and enhance the exploitation capability. This strategy will at least make algorithms better for some multimodel problems. The random weight, however, also increases the risk of finding non-optimal solutions. This will be reflected in the large amount of experimental data in Section 4, but the experimental results show that the overall search ability of SPSOC has been very significantly improved.

It is more concise that the flow of SPSORC is similar to that of the bPSO, which just calculates the random weight. Its procedure is shown in Table 1.

Table 1. The procedure of SPSORC.

Line	Procedure of SPSORC
1	Initialize parameters: dimension N , population size m , iteration number T , weight ω , learning factors c_1, c_2 , etc; % Step 1
2	Initialize and reserve matrix space: $p_{best} = [\text{Inf}_{11} \cdots \text{Inf}_{mN}]$, $g_{best} = [\text{Inf}_1 \cdots \text{Inf}_m]$, x_{min} = lower limits of position, x_{max} = upper limits;
3	For $i = 1:m$
4	For $j = 1:N$
5	Randomly initialize velocity and position: v_{in}, x_{in} ; % Step 2
6	End For
7	End For
8	For $i = 1:m$
9	Calculate the fitness. Compared to get the p_{best}^1 and g_{best}^1
10	End For
11	While the optima is not found or the termination condition is not met
12	Calculate the f_{best} and f_{worst} . Then, get the ω by Equation (23); % Step 3
13	For $i = 1:m$
14	For $j = 1:N$
15	Update the particle position according to Equation (22); % Step 4
16	If $x_{in}^t > x_{max}$
17	$x_{in}^t = x_{max}$;
18	Elseif $x_{in}^t < x_{min}$
19	$x_{in}^t = x_{min}$;
20	End If
21	End For
22	Substitute the current particle into the fitness formula to calculate the fitness value of the current particle;
23	Compare to get the p_{best} and g_{best} ;
24	End For
25	End While
26	Return Results. % Step 5

4. Experimental Study and Results Analysis

4.1. Benchmark Functions

The aim of this improved strategy is to solve the problem of unconstrained optimization better. In order to demonstrate the effectiveness of the algorithm more fully, this experiment will use 22 commonly used benchmark functions to simulate and contrast, including the unimodal benchmark functions represented by Sphere Function, the complex multimodal solution functions such as Rastrigrin Problem, the ill-conditioned quadratic Rosenbrock function, Xin–She Yang 3 with discontinuity and trap near the optimal solution, noise-containing functions like Quartic Function and other functions which is hard to find the best solution. Of course, these 22 functions also contain four test functions (f_7, f_{15}, f_{20} and f_{21}) with negative optima.

These 22 benchmark functions arranged in alphabetical order are shown in Table A1. The following test functions may change slightly in form for consistency or convenience because of the large number of types for test function versions, but the test results will not be affected. The last column, ‘Accuracy (50)’, is the convergence accuracy we want to reach for the test function in the 50-dimensional case, which will be used in Section 4.5.3, ‘Success Rate and Average Iteration Times’.

4.2. Parameters Setting and Simulation Environment

One of the reasons why particle swarm optimization algorithm was proposed late but had a relatively wide range of use is that it needs fewer parameters and is set up simply. When dealing with general problems, its requirements on population numbers, the maximum iteration numbers and other parameters are not high, which also determines that the algorithm has the advantages of small size and fast searching speed when it is implemented. Under normal circumstances, the population set at

40 can get a good solution for most problems. More complex problems can be solved by increasing the population number and the maximum iteration times.

Table 2 is about the specific parameter settings. NR is the number of times each algorithm searches for the benchmark functions. m is population number. T is the maximum iteration times per search. ω_{max} and ω_{min} are the maximum weight and the minimum weight. c_1 and c_2 are the acceleration factors.

Table 2. Parameters for candidates.

	NR	m	T	ω_{max}	ω_{min}	c_1	c_2	c_3
bPSO	30	40	100	0.9	0.4	2	2	-
PSOd	30	40	100	-	-	-	-	-
HPSOscac	30	40	100	Equation (10)	-	Equation (11)	Equation (11)	-
TCP SO	30	80	100	0.9	-	1.6	1.6	1.6
SPSO	30	40	100	0.9	0.4	-	2	-
SPSOC	30	40	100	0.9	0.4	-	2	-
SPSORC	30	40	100	Equation (23)	-	-	2	-

Simulation environment is shown in Table 3.

Table 3. Simulation environment.

Operation System	Windows 7 Professional (× 32)
CPU	Core 2 Duo 2.26 GHz
Memory	4.00 GB
Platform	Matlab R2014a
Network	Gigabit Ethernet

4.3. Discussion on Improvement Necessity For SPSO

The search speed is a great advantage of SPSO because of a simple structure. However, its advantages are its disadvantages. The over-simplified structure makes the SPSO's population lack of diversity, which makes it converge to local optima quickly, so the further improvement of SPSO becomes indispensable. Therefore, in Section 3, we present two improvements to SPSO. In this section, we will let SPSO, SPSOC and SPSORC solve the high dimension benchmark functions. Then, we discuss the necessity of those two improvement steps in Section 3 by analyzing its results.

In this experiment, the function dimension is set to 200 dimensions. The other parameters are consistent with the parameter setting table in Table 2 of Section 4.2. Table 4 shows the optimal results of the experiment. The minimum number in each set of data (min and mean) is represented in bold in the following table.

From the experimental results for the 200-dimensional benchmark functions in Table 4, we can see that SPSORC can search the other 21 functions for the theoretical optimal solution or a better solution than SPSO and SPSOC except for searching Quartic Function with noises. The optimization results of SPSO and SPSOC, however, are in straitened circumstances compared to SPSORC. SPSO gets better solutions four times, while SPSOC gets better solutions six times. Compared with the 30 search average solutions of SPSO and SPSOC, the optima of SPSOC is also smaller. It is indicated that the solution searched by SPSO after adding the confidence item can be kept smaller and its performance is greatly improved and the optimization capability is more enhanced after using random inertia weight. Thus, it can be seen that the two improvements to the SPSO are very necessary. SPSO is more inclined to exploration, which is more conducive to the local search of particles. Confidence term change the trajectory of some particles, which increases particle diversity. Meanwhile, the random inertia weight balances the exploitation capabilities of the algorithm so that it improves the search range and robustness of the algorithm significantly.

Table 4. Discussion on the necessity of improving SPSO.

Instance	SPSO		SPSOC		SPSORC	
	min	mean	min	mean	min	max
f_1	4.44×10^{-15}	4.44×10^{-15}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	1.48×10^{-15}
f_2	1.53×10^{-18}	5.91×10^{-3}	6.94×10^{-56}	6.93×10^{-50}	0	6.65×10^{-268}
f_3	2.16×10^{-35}	2.68×10^{-35}	3.84×10^{-106}	1.33×10^{-97}	0	0
f_4	5.40×10^{-76}	7.42×10^{-76}	2.27×10^{-216}	8.96×10^{-195}	0	0
f_5	0	8.32×10^{-3}	0	0	0	3.37×10^{-16}
f_6	1.61×10^{-30}	2.63×10^{-30}	2.88×10^{-102}	4.12×10^{-93}	0	0
f_7	-1.12×10^1	7.87×10^{-2}	-1.49×10^2	-1.49×10^2	-1.49×10^2	-1.49×10^2
f_8	9.51×10^{-1}	6.61×10^1	0	0	0	1.87×10^1
f_9	2.33×10^{-4}	3.15×10^{-2}	9.12×10^{-4}	2.30×10^{-2}	5.75×10^{-3}	3.63×10^{-1}
f_{10}	0	4.52×10^1	0	0	0	6.51×10^{-16}
f_{11}	1.49×10^2	1.49×10^2	1.49×10^2	1.49×10^2	1.48×10^2	1.49×10^2
f_{12}	7.97×10^{-33}	2.79×10^{-32}	2.95×10^{-82}	8.39×10^{-21}	0	1.77×10^{-04}
f_{13}	6.25×10^{-49}	1.07×10^{-45}	1.89×10^{-75}	1.24×10^{-60}	0	0
f_{14}	1.20×10^{-17}	1.36×10^{-17}	5.46×10^{-55}	3.75×10^{-47}	0	1.48×10^{-270}
f_{15}	-5.26×10^3	-3.37×10^3	-4.31×10^3	-2.61×10^3	-5.99×10^3	-3.80×10^3
f_{16}	1.21×10^{-34}	1.47×10^{-34}	1.91×10^{-106}	1.81×10^{-96}	0	1.90×10^{-321}
f_{17}	0	1.64×10^{-315}	0	0	0	1.36×10^{-256}
f_{18}	1.76×10^{-29}	2.81×10^{-6}	2.34×10^{-60}	1.85×10^{-18}	0	0
f_{19}	7.48×10^{-25}	1.02×10^{-16}	1.75×10^{-21}	1.04×10^{-14}	0	3.96×10^{-20}
f_{20}	3.63×10^{-55}	3.95×10^{-43}	7.93×10^{-46}	1.69×10^{-33}	-1	-1
f_{21}	1.75×10^{-44}	1.78×10^{-42}	1.59×10^{-36}	4.71×10^{-32}	-1	-1
f_{22}	2.61×10^{-36}	3.15×10^{-36}	2.20×10^{-33}	6.20×10^{-14}	0	2.14×10^{-8}

4.4. Discussion on Weight Selection for SPSOC

The proposed SPSOC has two inertia weights. The first inertia weight balances the search ability to global optima and the local one, while the second weight determines the degree to which the particle converges to the global optima in current generation. Obviously, whether these two weights are set properly or not has a significant impact on the performance of the algorithm. Then, the discussion of how these two inertia weights should be selected becomes quite necessary. The experiment comparing the optimal solution and the average solution found by the algorithm with different weights introduces three kinds of inertia weight strategies, which are divided into six kinds of situations. Those three kinds of inertia weights used in the experiment are as follows:

1. Linear decreasing inertia weight, i.e., Equation (4);
2. Classic nonlinear dynamic inertia weight, i.e., Equation (24);

$$\omega = \begin{cases} \omega_{max}, & x_{in} > f_{avg}, \\ \omega_{min} - (\omega_{max} - \omega_{min}) \times \frac{x_{in} - f_{min}}{f_{avg} - f_{min}}, & x_{in} \leq f_{avg}. \end{cases} \quad (24)$$

3. Random inertia weight proposed in this paper, i.e., Equation (23).

Table 5 reports the results of this experiment. Taking $\omega_{2,1}$ for example, the first subscript 2 indicates that ω_1 in Equation (22) uses the second kind of weight formula i.e., Equation (24), and the second subscript 1 indicates that ω_2 uses the first kind of weight formula i.e., Equation (4). The others are similar. Experimental benchmark functions' upper dimensions are set at 100. We represent the minimum value for min and mean in bold in the following table.

Table 5. Discussion on the weights selection of SPSOC.

Instance		Different Weight Matching					
		ω_{21}	ω_{31}	ω_{32}	ω_{11}	ω_{22}	ω_{22}
f_1	min	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}
	mean	1.80×10^1	3.85×10^{-15}	1.27×10^1	8.88×10^{-16}	1.53×10^1	1.36×10^{-15}
f_2	min	1.54×10^{-85}	4.67×10^{-65}	9.25×10^{-83}	8.09×10^{-61}	2.10×10^{-80}	0
	mean	4.34×10^{-16}	1.53×10^{-57}	7.90×10^{-75}	2.73×10^{-53}	4.14×10^{-63}	0
f_3	min	7.53×10^{-163}	3.54×10^{-125}	1.81×10^{-169}	8.92×10^{-122}	2.07×10^{-156}	0
	mean	1.75×10^{-38}	4.85×10^{-109}	4.84×10^{-91}	1.37×10^{-101}	4.97×10^{-50}	0
f_4	min	1.78×10^{-240}	8.32×10^{-258}	0	4.48×10^{-235}	2.22×10^{-304}	0
	mean	1.68×10^{-34}	7.32×10^{-214}	5.03×10^{-106}	1.99×10^{-205}	2.65×10^{-21}	0
f_5	min	0	0	2.17×10^1	0	0	0
	mean	1.20×10^2	0	2.17×10^1	0	2.17×10^{-2}	0
f_6	min	2.20×10^{-161}	2.59×10^{-121}	1.45×10^{-155}	6.26×10^{-114}	1.41×10^{-151}	0
	mean	3.58×10^{-67}	4.98×10^{-104}	7.88×10^{-142}	2.66×10^{-92}	9.16×10^{-72}	0
f_7	min	−9	−9	−9	−9	−9	−9
	mean	−9	−9	−9	−9	−9	−9
f_8	min	1.30	0	9.02×10^{-1}	0	9.02×10^{-1}	0
	mean	1.50	1.54	1.60	7.29×10^{-2}	1.25	3.25×10^{-1}
f_9	min	9.13×10^{-4}	1.37×10^{-3}	1.51×10^{-3}	1.05×10^{-3}	5.92×10^{-4}	2.71×10^{-3}
	mean	2.01×10^{-2}	3.16×10^{-2}	4.61×10^{-2}	1.51×10^{-2}	3.11×10^{-2}	4.44×10^{-2}
f_{10}	min	0	0	0	0	0	0
	mean	2.89×10^1	0	0	0	5.67×10^{-7}	0
f_{11}	min	7.28	8.03	7.69	8.04	7.86	7.97
	mean	8.09	8.83	8.81	8.45	8.22	8.54
f_{12}	min	1.60×10^{-157}	1.76×10^{-116}	8.56×10^{-158}	5.08×10^{-105}	2.32×10^{-148}	0
	mean	3.04×10^{-22}	7.01×10^{-8}	9.69×10^1	6.71×10^{-67}	3.90×10^{-35}	0
f_{13}	min	1.35×10^{-96}	2.36×10^{-73}	1.39×10^{-93}	2.91×10^{-77}	1.92×10^{-92}	0
	mean	9.62×10^{-26}	3.02×10^{-59}	3.63×10^{-43}	7.25×10^{-58}	2.42×10^{-41}	0
f_{14}	min	1.56×10^{-84}	1.14×10^{-64}	2.91×10^{-85}	1.66×10^{-60}	6.95×10^{-84}	0
	mean	4.04×10^{-56}	1.89×10^{-55}	1.94×10^{-67}	3.04×10^{-52}	5.80×10^{-35}	0
f_{15}	min	-1.60×10^3	-1.57×10^3	-1.52×10^3	-1.36×10^3	-1.38×10^3	-1.34×10^3
	mean	-8.99×10^2	-9.81×10^2	-1.05×10^3	-8.06×10^2	-8.09×10^2	-8.84×10^2
f_{16}	min	9.16×10^{-164}	1.56×10^{-121}	3.15×10^{-164}	3.83×10^{-117}	1.96×10^{-150}	0
	mean	2.09×10^{-27}	9.99×10^{-101}	1.05×10^{-144}	1.05×10^{-97}	1.04×10^{-62}	0
f_{17}	min	1.78×10^{-199}	2.74×10^{-154}	1.09×10^{-192}	8.20×10^{-148}	2.80×10^{-158}	0
	mean	6.33×10^{-1}	1.43×10^{-116}	1.63×10^{-33}	2.37×10^{-118}	6.53×10^{-24}	0
f_{18}	min	5.61×10^{-84}	1.49×10^{-52}	1.97×10^{-82}	2.24×10^{-59}	7.48×10^{-80}	0
	mean	3.89×10^{-13}	7.36×10^{-19}	1.94×10^{-19}	4.60×10^{-29}	1.48×10^{-19}	0
f_{19}	min	3.54×10^{-3}	1.01×10^{-2}	9.08×10^{-3}	1.47×10^{-2}	1.09×10^{-2}	0
	mean	6.15×10^{-2}	4.33×10^{-2}	4.10×10^{-2}	9.36×10^{-2}	7.27×10^{-2}	2.24×10^{-2}
f_{20}	min	3.97×10^{-25}	7.05×10^{-17}	3.97×10^{-25}	1.81×10^{-12}	3.97×10^{-25}	−1
	mean	3.68×10^{-14}	2.75×10^{-11}	4.58×10^{-10}	1.01×10^{-7}	3.97×10^{-25}	−1
f_{21}	min	9.28×10^{-4}	5.44×10^{-4}	1.88×10^{-3}	1.61×10^{-3}	2.95	−1
	mean	2.85	2.35×10^{-3}	1.07×10^{-1}	3.14×10^{-3}	2.95	−1
f_{22}	min	7.70×10^{-155}	7.35×10^{-115}	3.27×10^{-159}	1.75×10^{-113}	1.61×10^{-146}	0
	mean	3.08×10^{-30}	2.71×10^{-45}	5.07×10^{-100}	2.32×10^{-88}	1.26×10^{-40}	0

As can be seen from the experimental data in Table 5, when the ω_1 and ω_2 take the random weights proposed in this paper, the obtained optima are satisfactory. It has 19 times to find the best solution, but only dominated by other algorithms when searching for the three functions (f_9 , f_{11} and f_{15})—followed by ω_1 using the second kind of weight improvement strategy and ω_2 using the first strategy with the way. This method has six times to search for smaller results. The conclusion of this discussion is that the optimization of the algorithm is better when ω_1 is equal to ω_2 . If they all use the randomized weights proposed in this paper at the same time, the capability of the SPSOC will be the best and it can easily do this with most of the benchmark functions. Comparing with the 30-times average values, we can find that, when the weight ω_1 is equal to ω_2 , the average value is smaller and the randomization strategy proposed in this paper is the best among them. If they use the same weight equation, the algorithm will be simpler and faster because only one weight needs to be calculated.

However, this paper uses only six kinds of collocation ways which are combined into three kinds of improvement strategies to carry on the simulation experiment. Whether there is a better weight improvement strategy to make SPSOC have a better performance needs to be further developed and improved.

4.5. Comparison and Analysis with Other PSOs

The most commonly used method which better reflects that the improved algorithm is excellent is bound to be compared with other classical improvement methods. In this section, we have a comparative test between three improved strategies proposed in this paper and bPSO and its three representative improved ones namely, bPSO, PSOd, HPSO-SCAC and TCP SO. The experiment consists of three parts mainly. The first part is to test the seven kinds of particle swarm algorithms separately for 0-dimensional, 50-dimensional and 100-dimensional functions. Each function is searched for 30 times. A *t*-test is used to analyze the large amount of experimental data obtained. Twenty-two distinct evolution curves of fitness from optimizing 100-dimensional functions will be analyzed briefly. Here, all the experiments were conducted on the same conditions as Zhang et al. [70,71]. The second part is to analyze the complexity by the Big O notation [72] and the actual running time for search for the optima in 50-dimensional problems. The third part is to calculate the success rate and the average iteration times of seven algorithms in solving twenty-two 50-dimensional problems, respectively. The stability and effectiveness of the algorithm will be analyzed by these two indices. More details of those three parts will be elaborated in sequence in the following subsections.

4.5.1. Different Dimensional Experiments and *t*-Test Analysis

Students' *t*-test (*t*-test) is a frequently used method of data analysis in statistics to compare whether two sets of data is in one solution space or not, that is, the comparison for data differences. In this paper, a two-independent-samples *t*-test as the following formulas is used to analyze the difference between the 30 optima searched by SPSORC and the 30 ones by others:

$$t = \frac{(\bar{X}_1 - \bar{X}_2) - (\mu_1 - \mu_2)}{S_{\bar{X}_1 - \bar{X}_2}} = \frac{(\bar{X}_1 - \bar{X}_2)}{S_{\bar{X}_1 - \bar{X}_2}}, \quad (25)$$

$$S_{\bar{X}_1 - \bar{X}_2} = \sqrt{\frac{S_c^2}{n_1} + \frac{S_c^2}{n_2}}, \quad (26)$$

where \bar{X}_1 and \bar{X}_2 are, respectively, the average of two sets of data; S_c^2 is the combined variance; The sample size is 30; the two-tailed test level is taken as 0.05. The Matlab R2014a test2 function (MathWorks, Natick, MA, USA) instruction is used to calculate directly so as to avoid unnecessary calculation error in the paper. Table 6 shows the optima of the seven improved algorithms for the 10-dimensional, 50-dimensional and 100-dimensional benchmark functions from Table A1, respectively. In Table 6, each algorithm solves the specified function 30 times separately and minimum value(min), average values(mean) and standard deviation values(std) of them are calculated. The minimum one of this three sets of data are highlighted in boldface. '+', '-' and '=' respectively indicate that the SPSORC results are 'better' than, 'worse' than and 'same' as the improved algorithm. To calculate the SPSORC's net score for convenience, '1', '-1', and '0' corresponding to the three symbols here indicate the score of the SPSORC.

Table 6. Optimization results for the function in 3 kinds of dimension.

		10				50				100			
		min	mean	std	ttest	min	mean	std	ttest	min	mean	std	ttest
f_1	bPSO	3.16×10^{-1}	1.29	6.68×10^{-1}	+(1)	1.26×10^1	1.77×10^1	1.82	+(1)	1.95×10^1	2.02×10^1	3.23×10^{-1}	+(1)
	PSOd	1.16	3.32	1.64	+(1)	1.09×10^1	1.36×10^1	1.16	+(1)	1.39×10^1	1.54×10^1	6.67×10^{-1}	+(1)
	HPSOscac	2.08×10^{-10}	2.20	5.69	+(1)	1.68×10^{-11}	1.13	2.98	+(1)	6.66×10^{-10}	6.94×10^{-1}	2.65	=(0)
	TCPSO	8.92×10^{-01}	2.08	6.35×10^{-1}	+(1)	1.03×10^1	1.39×10^1	2.08	+(1)	1.72×10^1	1.85×10^1	7.74×10^{-1}	+(1)
	SPSO	8.88×10^{-16}	3.38×10^{-15}	1.66×10^{-15}	+(1)	8.88×10^{-16}	3.73×10^{-15}	1.45×10^{-15}	+(1)	8.88×10^{-16}	3.85×10^{-15}	1.35×10^{-15}	+(1)
	SPSOC	8.88×10^{-16}	8.88×10^{-16}	0	=(0)	8.88×10^{-16}	8.88×10^{-16}	0	=(0)	8.88×10^{-16}	8.88×10^{-16}	0	-(−1)
	SPSORC	8.88×10^{-16}	8.88×10^{-16}	0		8.88×10^{-16}	8.88×10^{-16}	0		8.88×10^{-16}	2.19×10^{-15}	4.01×10^{-15}	
f_2	bPSO	3.46×10^{-2}	6.10×10^{-1}	6.44×10^{-1}	+(1)	3.97×10^1	5.60×10^1	8.63	+(1)	1.43×10^2	1.68×10^2	1.25×10^1	+(1)
	PSOd	4.28×10^{-3}	1.29×10^{-1}	1.79×10^{-1}	+(1)	1.53×10^1	2.12×10^1	4.77	+(1)	5.40×10^1	6.94×10^1	1.07×10^1	+(1)
	HPSOscac	0	6.88×10^{-77}	3.77×10^{-76}	=(0)	0	1.27×10^{-68}	6.95×10^{-68}	=(0)	0	8.67×10^{-49}	4.75×10^{-48}	=(0)
	TCPSO	6.42×10^{-2}	1.46	1.69	+(1)	2.24×10^1	4.92×10^1	1.33×10^1	+(1)	1.05×10^2	1.35×10^2	2.11×10^1	+(1)
	SPSO	2.58×10^{-24}	2.81×10^{-2}	1.49×10^{-1}	=(0)	4.28×10^{-20}	2.40×10^{-4}	1.25×10^{-3}	=(0)	3.06×10^{-19}	1.94×10^{-3}	1.06×10^{-2}	=(0)
	SPSOC	7.18×10^{-59}	1.01×10^{-45}	5.54×10^{-45}	=(0)	2.12×10^{-56}	5.22×10^{-41}	2.73×10^{-40}	=(0)	1.07×10^{-53}	5.71×10^{-42}	2.35×10^{-41}	=(0)
	SPSORC	0	0	0		0	5.61×10^{-281}	0		0	0	0	
f_3	bPSO	1.31×10^{-3}	8.95×10^{-1}	4.78	=(0)	7.69×10^2	1.37×10^3	4.27×10^2	+(1)	1.12×10^4	1.37×10^4	1.33×10^3	+(1)
	PSOd	6.53×10^{-4}	2.91×10^{-1}	3.50×10^{-1}	+(1)	2.95×10^2	5.68×10^2	1.66×10^2	+(1)	2.88×10^3	4.32×10^3	1.02×10^3	+(1)
	HPSOscac	0	8.09×10^{-132}	4.43×10^{-131}	=(0)	0	1.62×10^{-155}	8.88×10^{-155}	=(0)	0	3.60×10^{-120}	1.97×10^{-119}	=(0)
	TCPSO	1.66×10^{-2}	7.68×10^{-2}	6.09×10^{-2}	+(1)	1.70×10^2	4.56×10^2	2.85×10^2	+(1)	3.40×10^3	5.24×10^3	1.15×10^3	+(1)
	SPSO	1.80×10^{-52}	2.39×10^{-46}	6.91×10^{-46}	=(0)	2.97×10^{-38}	1.11×10^{-32}	4.16×10^{-32}	=(0)	3.39×10^{-35}	2.39×10^{-29}	1.31×10^{-28}	=(0)
	SPSOC	4.92×10^{-112}	3.50×10^{-90}	1.89×10^{-89}	=(0)	1.91×10^{-108}	9.54×10^{-73}	5.22×10^{-72}	=(0)	7.38×10^{-104}	5.55×10^{-69}	3.04×10^{-68}	=(0)
	SPSORC	0	0	0		0	0	0		0	0	0	
f_4	bPSO	2.10×10^{-8}	5.60×10^{-7}	5.78×10^{-7}	+(1)	7.19	3.05×10^1	1.35×10^1	+(1)	2.32×10^2	4.93×10^2	1.54×10^2	+(1)
	PSOd	3.81×10^{-7}	4.00×10^{-4}	9.29×10^{-4}	+(1)	1.69	4.84	2.59	+(1)	1.94×10^1	6.03×10^1	2.18×10^1	+(1)
	HPSOscac	0	9.26×10^{-285}	0	+(1)	0	6.20×10^{-224}	0	+(1)	0	1.36×10^{-274}	0	+(1)
	TCPSO	2.30×10^{-8}	3.68×10^{-6}	4.79×10^{-6}	+(1)	4.50×10^{-1}	4.35	6.46	+(1)	4.11×10^1	1.07×10^2	5.20×10^1	+(1)
	SPSO	6.17×10^{-108}	4.10×10^{-94}	2.23×10^{-93}	=(0)	9.65×10^{-83}	3.38×10^{-72}	1.77×10^{-71}	=(0)	4.66×10^{-78}	1.25×10^{-69}	4.73×10^{-69}	=(0)
	SPSOC	5.88×10^{-234}	1.19×10^{-183}	0	=(0)	2.87×10^{-218}	1.52×10^{-152}	8.31×10^{-152}	=(0)	1.56×10^{-222}	4.13×10^{-155}	2.26×10^{-154}	=(0)
	SPSORC	0	1.73×10^{-321}	0		0	2.96×10^{-323}	0		0	2.02×10^{-320}	0	
f_5	bPSO	4.22×10^{-1}	9.36×10^{-1}	1.64×10^{-1}	+(1)	6.46×10^1	2.11×10^2	6.99×10^1	+(1)	8.91×10^2	1.15×10^3	1.48×10^2	+(1)
	PSOd	1.37×10^{-1}	8.46×10^{-1}	6.67×10^{-1}	+(1)	4.36×10^1	8.91×10^1	2.69×10^1	+(1)	2.38×10^2	3.18×10^2	4.27×10^1	+(1)
	HPSOscac	3.38×10^{-6}	6.26×10^1	1.14×10^2	+(1)	7.84×10^{-2}	4.02×10^2	5.68×10^2	+(1)	1.57×10^{-5}	1.37×10^3	1.67×10^3	+(1)
	TCPSO	1.04	1.16	9.72×10^{-2}	+(1)	1.90×10^1	6.24×10^1	3.63×10^1	+(1)	2.46×10^2	4.14×10^2	7.97×10^1	+(1)
	SPSO	0	3.87×10^{-1}	3.41×10^{-1}	+(1)	0	5.40×10^{-2}	1.49×10^{-1}	=(0)	0	7.33×10^{-3}	2.37×10^{-2}	=(0)
	SPSOC	0	0	0	-(−1)	0	0	0	=(0)	0	0	0	=(0)
	SPSORC	0	3.70×10^{-18}	2.03×10^{-17}		0	0	0		0	5.18×10^{-17}	1.23×10^{-16}	

Table 6. Cont.

		10				50				100			
		min	mean	std	ttest	min	mean	std	ttest	min	mean	std	ttest
f_6	bPSO	4.16×10^3	6.97×10^5	1.22×10^6	+(1)	1.38×10^8	3.70×10^8	1.82×10^8	+(1)	7.95×10^8	1.90×10^9	6.34×10^8	+(1)
	PSOd	2.60×10^3	5.58×10^4	7.67×10^4	+(1)	1.80×10^7	8.68×10^7	4.80×10^7	+(1)	2.99×10^8	5.38×10^8	1.37×10^8	+(1)
	HPSOscac	0	3.69×10^{-149}	2.02×10^{-148}	=(0)	0	3.77×10^{-157}	2.06×10^{-156}	=(0)	0	3.40×10^{-124}	1.86×10^{-123}	=(0)
	TCP SO	5.51×10^4	8.43×10^5	1.26×10^6	+(1)	4.79×10^7	1.91×10^8	1.33×10^8	+(1)	3.51×10^8	1.04×10^9	6.43×10^8	+(1)
	SPSO	1.44×10^{-45}	5.73×10^{-41}	1.56×10^{-40}	+(1)	1.71×10^{-33}	1.72×10^{-28}	7.68×10^{-28}	=(0)	8.51×10^{-33}	2.42×10^{-27}	5.20×10^{-27}	+(1)
	SPSOC	3.77×10^{-113}	8.59×10^{-84}	4.71×10^{-83}	+(1)	1.15×10^{-101}	1.94×10^{-76}	7.40×10^{-76}	=(0)	1.47×10^{-102}	1.67×10^{-76}	9.16×10^{-76}	+(1)
	SPSORC	0	0	0		0	0	0		0	0	0	
f_7	bPSO	−6.76	−5.32	6.63×10^{-1}	+(1)	-1.20×10^1	−9.16	1.64	+(1)	-1.66×10^1	-1.23×10^1	2.52	+(1)
	PSOd	−7.93	−6.49	6.48×10^{-1}	+(1)	-2.64×10^1	-2.13×10^1	2.01	+(1)	-3.78×10^1	-3.27×10^1	2.57	+(1)
	HPSOscac	−2.46	−2.46	4.73×10^{-1}	+(1)	−3.51	−4.39	4.35	+(1)	-1.48×10^1	−5.40	4.02	+(1)
	TCP SO	−6.83	−4.91	9.48×10^{-1}	+(1)	-1.43×10^1	−8.66	3.01	+(1)	-1.58×10^1	−9.52	3.65	+(1)
	SPSO	−9	−5.47	2.81	+(1)	-4.90×10^1	-1.20×10^1	1.79×10^1	+(1)	-3.31×10^1	−3.96	8.44	+(1)
	SPSOC	−9	−9	0	=(0)	-4.90×10^1	$-4.90 \times 10^{+01}$	0	=(0)	-9.90×10^1	-9.90×10^1	0	−(−1)
	SPSORC	−9	−9	0		-4.90×10^1	-4.90×10^1	0		-9.90×10^1	-9.90×10^1	2.64×10^{-15}	
f_8	bPSO	9.02×10^{-1}	1.71	5.13×10^{-1}	+(1)	1.04×10^1	1.24×10^1	1.36	+(1)	2.39×10^1	2.69×10^1	1.36	+(1)
	PSOd	1.65	2.41	3.26×10^{-1}	+(1)	1.94×10^1	2.11×10^1	6.65×10^{-1}	+(1)	4.29×10^1	4.53×10^1	9.08×10^{-1}	+(1)
	HPSOscac	2.22×10^{-16}	1.82	1.47	+(1)	6.75×10^{-12}	1.43×10^1	9.26	+(1)	9.03×10^{-6}	2.68×10^1	2.09×10^1	+(1)
	TCP SO	7.05×10^{-1}	1.68	7.04×10^{-1}	+(1)	9.91	1.19×10^1	1.05	+(1)	2.33×10^1	2.56×10^1	1.65	+(1)
	SPSO	1.47	2.81	5.56×10^{-1}	+(1)	2.11×10^{-4}	1.87×10^1	6.17	+(1)	6.62×10^{-1}	4.30×10^1	8.94	+(1)
	SPSOC	0	2.87×10^{-2}	1.57×10^{-1}	−(−1)	0	6.58×10^{-1}	3.60	−(−1)	0	1.50	8.22	+(1)
	SPSORC	0	4.43×10^{-1}	1.16		0	1.47	5.59		0	0	0	
f_9	bPSO	1.97×10^{-2}	9.62×10^{-2}	5.02×10^{-2}	+(1)	4.21	2.94×10^1	1.63×10^1	+(1)	2.64×10^2	4.90×10^2	1.22×10^2	+(1)
	PSOd	1.23×10^{-2}	5.86×10^{-2}	3.47×10^{-2}	+(1)	3.09	6.44	3.01	+(1)	3.35×10^1	6.07×10^1	2.06×10^1	+(1)
	HPSOscac	4.17×10^{-1}	1.27×10^1	1.70×10^1	+(1)	6.76	8.72×10^2	6.17×10^2	+(1)	7.34×10^1	3.51×10^3	2.21×10^3	+(1)
	TCP SO	2.68×10^{-2}	7.16×10^{-2}	4.05×10^{-2}	+(1)	2.39	5.04	3.05	+(1)	6.29×10^1	1.07×10^2	4.38×10^1	+(1)
	SPSO	2.16×10^{-4}	2.27×10^{-2}	2.16×10^{-2}	=(0)	2.39×10^{-3}	2.13×10^{-2}	1.70×10^{-2}	−(−1)	2.52×10^{-3}	3.08×10^{-2}	2.90×10^{-2}	−(−1)
	SPSOC	9.79×10^{-4}	2.07×10^{-2}	2.35×10^{-2}	=(0)	1.00×10^{-3}	2.15×10^{-2}	1.65×10^{-2}	+(1)	9.11×10^{-4}	2.15×10^{-2}	1.66×10^{-2}	−(−1)
	SPSORC	7.03×10^{-4}	3.65×10^{-2}	3.08×10^{-2}		4.43×10^{-3}	6.27×10^{-2}	5.32×10^{-2}		5.05×10^{-3}	1.72×10^{-1}	2.82×10^{-1}	
f_{10}	bPSO	8.03	2.76×10^1	1.21×10^1	+(1)	4.55×10^2	5.47×10^2	5.13×10^1	+(1)	1.05×10^3	1.31×10^3	9.25×10^1	+(1)
	PSOd	1.05×10^{-1}	8.86	4.17	+(1)	1.65×10^2	2.06×10^2	2.38×10^1	+(1)	5.15×10^2	6.36×10^2	5.58×10^1	+(1)
	HPSOscac	9.35×10^{-5}	5.13×10^1	5.40×10^1	+(1)	2.50×10^{-1}	3.47×10^2	3.03×10^2	+(1)	3.08×10^{-1}	6.13×10^2	5.71×10^2	+(1)
	TCP SO	1.12×10^1	4.68×10^1	2.10×10^1	+(1)	4.02×10^2	5.42×10^2	7.35×10^1	+(1)	1.03×10^3	1.22×10^3	1.15×10^2	+(1)
	SPSO	0	1.78×10^1	2.31×10^1	+(1)	0	1.09	3.67	=(0)	0	3.24×10^{-1}	1.24	=(0)
	SPSOC	0	0	0	=(0)	0	0	0	=(0)	0	0	0	=(0)
	SPSORC	0	0	0		0	0	0		0	2.96×10^{-16}	9.43×10^{-16}	

Table 6. Cont.

		10				50				100			
		min	mean	std	ttest	min	mean	std	ttest	min	mean	std	ttest
f_{11}	bPSO	1.31×10^1	6.35×10^3	2.28×10^{04}	=(0)	5.06×10^6	1.71×10^7	8.00×10^6	+(1)	1.80×10^8	2.98×10^8	7.48×10^7	+(1)
	PSOd	6.50	8.95×10^2	2.03×10^3	+(1)	1.89×10^6	6.49×10^6	3.13×10^6	+(1)	2.08×10^7	4.22×10^7	1.46×10^7	+(1)
	HPSOscac	9.00×10^3	8.68×10^7	8.12×10^7	+(1)	5.64×10^7	1.01×10^9	6.13×10^8	+(1)	2.42×10^8	2.41×10^9	1.33×10^9	+(1)
	TCPSo	4.24×10^1	8.01×10^2	1.04×10^3	+(1)	5.18×10^5	2.45×10^6	1.01×10^6	+(1)	3.84×10^7	7.88×10^7	3.19×10^7	+(1)
	SPSO	7.74	8.15	1.28×10^{-1}	-(−1)	4.81 × 10 ¹	4.87 × 10 ¹	3.08×10^{-1}	-(−1)	9.81 × 10 ¹	9.88 × 10 ¹	2.02×10^{-1}	=(0)
	SPSOC	8.11	8.32	1.99×10^{-1}	+(1)	4.81×10^1	4.87×10^1	3.05×10^{-1}	+(1)	9.82×10^1	9.89×10^1	1.48 × 10 ^{−1}	=(0)
	SPSORC	8.00	8.64	6.52×10^{-1}		4.86×10^1	4.89×10^1	1.05 × 10 ^{−1}		9.82×10^1	9.89×10^1	1.54×10^{-1}	
f_{12}	bPSO	6.41×10^2	3.89×10^3	1.89×10^3	+(1)	3.09×10^5	1.53×10^6	1.12×10^6	+(1)	5.44×10^6	1.94×10^7	1.18×10^7	+(1)
	PSOd	1.66×10^2	1.73×10^3	7.79×10^2	+(1)	1.23×10^5	3.16×10^5	1.43×10^5	+(1)	1.26×10^6	4.46×10^6	2.47×10^6	+(1)
	HPSOscac	0	7.30×10^{-131}	4.00×10^{-130}	=(0)	0	1.56×10^{-144}	8.56×10^{-144}	=(0)	0	1.10×10^{-120}	6.01×10^{-120}	=(0)
	TCPSo	4.44×10^2	3.89×10^3	1.64×10^3	+(1)	5.48×10^5	2.50×10^6	1.72×10^6	+(1)	1.05×10^7	3.86×10^7	3.14×10^7	+(1)
	SPSO	2.39×10^{-45}	9.73×10^{-42}	3.27×10^{-41}	=(0)	1.84×10^{-38}	1.73×10^{-36}	3.39×10^{-36}	+(1)	1.09×10^{-36}	5.55×10^{-35}	8.02×10^{-35}	+(1)
	SPSOC	3.41×10^{-117}	3.44×10^{-52}	1.56×10^{-51}	=(0)	8.83×10^{-74}	2.38×10^{-24}	9.28×10^{-24}	+(1)	8.79×10^{-66}	2.06	1.13×10^1	+(1)
	SPSORC	0	0	0		0	0	0		0	0	0	
f_{13}	bPSO	5.56×10^{-9}	1.23×10^{-6}	2.22×10^{-6}	+(1)	7.47×10^{-10}	1.01×10^{-6}	1.96×10^{-6}	+(1)	1.25×10^{-8}	9.08×10^{-7}	1.18×10^{-6}	+(1)
	PSOd	1.42×10^{-34}	1.88×10^{-30}	4.49×10^{-30}	+(1)	7.15×10^{-36}	6.07×10^{-31}	1.23×10^{-30}	+(1)	6.10×10^{-35}	4.29×10^{-30}	1.65×10^{-29}	=(0)
	HPSOscac	0	1.06×10^{-83}	5.79×10^{-83}	=(0)	0	1.13×10^{-73}	6.15×10^{-73}	=(0)	3.62×10^{-321}	3.14×10^{-86}	1.25×10^{-85}	=(0)
	TCPSo	3.23×10^{-4}	2.43×10^{-2}	2.73×10^{-2}	+(1)	4.45×10^{-04}	1.88×10^{-2}	1.83×10^{-2}	+(1)	2.17×10^{-4}	2.45×10^{-2}	2.75×10^{-2}	+(1)
	SPSO	3.11×10^{-51}	5.41×10^{-47}	1.16×10^{-46}	+(1)	3.93×10^{-53}	6.62×10^{-47}	1.57×10^{-46}	+(1)	8.32×10^{-52}	1.34×10^{-45}	5.05×10^{-45}	=(0)
	SPSOC	7.15×10^{-77}	4.92×10^{-60}	2.68×10^{-59}	+(1)	2.13×10^{-77}	7.72×10^{-60}	4.23×10^{-59}	+(1)	3.79×10^{-79}	1.10×10^{-60}	6.01×10^{-60}	=(0)
	SPSORC	0	0	0		0	0	0		0	0	0	
f_{14}	bPSO	6.81×10^{-2}	2.34×10^{-1}	1.12×10^{-1}	+(1)	8.67×10^1	3.08×10^2	9.07×10^2	=(0)	2.94×10^2	3.55×10^2	2.47×10^1	+(1)
	PSOd	2.74×10^{-2}	6.15×10^{-1}	5.46×10^{-1}	+(1)	3.43×10^1	5.81×10^1	1.55×10^1	+(1)	1.19×10^2	1.62×10^2	2.39×10^1	+(1)
	HPSOscac	0	1.04×10^{-59}	5.71×10^{-59}	=(0)	1.79×10^{-238}	3.77×10^{-61}	1.45×10^{-60}	=(0)	0	5.56×10^{-63}	2.57×10^{-62}	=(0)
	TCPSo	2.30×10^{-1}	4.88×10^{-1}	1.66×10^{-1}	+(1)	9.95×10^1	8.94×10^{13}	4.83×10^{14}	=(0)	3.25×10^2	1.60×10^{37}	7.61×10^{37}	=(0)
	SPSO	1.38×10^{-25}	1.28×10^{-23}	2.21×10^{-23}	+(1)	6.30×10^{-20}	3.34×10^{-17}	6.71×10^{-17}	+(1)	1.70×10^{-19}	2.24×10^{-15}	5.95×10^{-15}	+(1)
	SPSOC	8.49×10^{-58}	9.72×10^{-48}	3.86×10^{-47}	+(1)	7.86×10^{-55}	1.25×10^{-34}	6.82×10^{-34}	+(1)	2.92×10^{-56}	3.42×10^{-37}	1.85×10^{-36}	+(1)
	SPSORC	0	0	0		0	0	0		0	0	0	
f_{15}	bPSO	-3.83×10^3	-3.41×10^3	3.02×10^2	+(1)	-1.31×10^4	−1.12 × 10 ⁴	1.03×10^3	-(−1)	-1.91×10^4	-1.57×10^4	1.40×10^3	-(−1)
	PSOd	-3.83×10^3	-3.18×10^3	3.54×10^2	+(1)	-1.10×10^4	-9.40×10^3	7.22×10^2	+(1)	-1.77×10^4	-1.46×10^4	1.31×10^3	-(−1)
	HPSOscac	-1.91×10^3	-1.28×10^3	3.96×10^2	+(1)	-5.62×10^3	-3.34×10^3	1.02×10^3	+(1)	-7.58×10^3	-4.87×10^3	1.20×10^3	+(1)
	TCPSo	−4.06 × 10 ³	−3.41 × 10 ³	3.01×10^2	-(−1)	−1.34 × 10 ⁴	-1.11×10^4	9.24×10^2	+(1)	−2.04 × 10 ⁴	−1.72 × 10 ⁴	1.78×10^3	-(−1)
	SPSO	-1.51×10^3	-9.53×10^2	2.22×10^2	=(0)	-3.05×10^3	-1.91×10^3	5.22×10^2	=(0)	-4.12×10^3	-2.69×10^3	6.78×10^2	=(0)
	SPSOC	-1.19×10^3	-7.02×10^2	1.89 × 10 ²	=(0)	-2.61×10^3	-1.54×10^3	4.49×10^2	=(0)	-3.62×10^3	-2.09×10^3	6.72×10^2	=(0)
	SPSORC	-1.35×10^3	-8.69×10^2	2.55×10^2		-2.62×10^3	-1.87×10^3	4.16 × 10 ²		-4.26×10^3	-2.82×10^3	6.11 × 10 ²	

Table 6. Cont.

		10				50				100			
		min	mean	std	ttest	min	mean	std	ttest	min	mean	std	ttest
f_{16}	bPSO	2.15×10^{-1}	2.48	2.76	+(1)	7.73×10^3	1.87×10^4	6.62×10^3	+(1)	9.11×10^4	1.15×10^5	1.47×10^4	+(1)
	PSOd	9.32×10^{-1}	1.02×10^1	1.21×10^1	+(1)	4.26×10^3	9.67×10^3	2.81×10^3	+(1)	2.34×10^4	3.40×10^4	6.67×10^3	+(1)
	HPSOscac	0	6.95×10^{-126}	3.81×10^{-125}	=(0)	0	5.30×10^{-144}	2.90×10^{-143}	=(0)	0	1.15×10^{-92}	6.31×10^{-92}	=(0)
	TCPSO	1.41	7.52	5.62	+(1)	1.74×10^3	6.23×10^3	4.27×10^3	+(1)	3.12×10^4	5.11×10^4	1.05×10^4	+(1)
	SPSO	1.38×10^{-50}	3.20×10^{-44}	1.11×10^{-43}	=(0)	2.69×10^{-36}	5.43×10^{-32}	1.61×10^{-31}	=(0)	3.40×10^{-35}	8.65×10^{-30}	2.53×10^{-29}	=(0)
	SPSOC	2.11×10^{-114}	3.46×10^{-89}	1.66×10^{-88}	=(0)	8.39×10^{-107}	5.65×10^{-70}	3.10×10^{-69}	=(0)	2.21×10^{-103}	5.27×10^{-79}	2.85×10^{-78}	=(0)
	SPSORC	0	0	0		0	0	0		0	0	0	
f_{17}	bPSO	2.52×10^{-12}	1.09×10^{-8}	2.37×10^{-8}	+(1)	7.44×10^{-55}	2.75×10^{-3}	2.89×10^{-3}	+(1)	1.48×10^{-2}	2.36×10^{-1}	2.58×10^{-1}	+(1)
	PSOd	2.07×10^{-11}	3.03×10^{-6}	9.79×10^{-6}	=(0)	5.86×10^{-8}	4.53×10^{-5}	7.70×10^{-5}	+(1)	7.17×10^{-7}	1.85×10^{-4}	3.29×10^{-4}	+(1)
	HPSOscac	0	5.38×10^{-135}	2.95×10^{-134}	=(0)	0	2.62×10^{-148}	1.34×10^{-147}	=(0)	0	2.87×10^{-154}	1.57×10^{-153}	=(0)
	TCPSO	1.51×10^{-8}	6.10×10^{-7}	6.72×10^{-7}	+(1)	6.45×10^{-6}	5.69×10^{-4}	1.10×10^{-3}	+(1)	1.48×10^{-3}	1.27×10^{-1}	3.65×10^{-1}	=(0)
	SPSO	1.16×10^{-94}	4.96×10^{-86}	1.50×10^{-85}	=(0)	6.17×10^{-94}	1.10×10^{-84}	5.95×10^{-84}	=(0)	1.78×10^{-92}	8.69×10^{-87}	1.71×10^{-86}	+(1)
	SPSOC	8.84×10^{-141}	1.97×10^{-115}	1.08×10^{-114}	=(1)	6.16×10^{-158}	1.01×10^{-121}	5.52×10^{-121}	=(0)	2.52×10^{-146}	1.22×10^{-118}	6.19×10^{-118}	+(1)
	SPSORC	0	0	0		0	0	0		0	4.94×10^{-324}	0	
f_{18}	bPSO	5.10×10^{-4}	6.20×10^{-1}	2.15	=(0)	1.82×10^{13}	2.27×10^{19}	1.11×10^{20}	=(0)	2.83×10^{34}	6.97×10^{46}	3.76×10^{47}	=(0)
	PSOd	5.79×10^{-4}	3.73×10^{-1}	6.15×10^{-1}	+(1)	2.48×10^4	8.59×10^{10}	4.26×10^{11}	=(0)	8.86×10^{20}	5.66×10^{32}	2.80×10^{33}	=(0)
	HPSOscac	5.26×10^{-319}	1.20×10^3	3.72×10^3	+(1)	4.63×10^{-237}	3.53×10^{28}	1.66×10^{29}	=(0)	0	2.88×10^{54}	1.34×10^{55}	=(0)
	TCPSO	2.75×10^{-3}	3.15×10^{-1}	4.89×10^{-1}	+(1)	1.10×10^8	4.18×10^{16}	2.20×10^{17}	=(0)	4.98×10^{30}	1.22×10^{42}	6.28×10^{42}	=(0)
	SPSO	2.25×10^{-31}	1.23×10^{-4}	6.69×10^{-4}	=(0)	1.65×10^{-33}	3.98×10^{-6}	2.12×10^{-5}	=(0)	1.97×10^{-34}	1.67×10^{-5}	6.35×10^{-5}	=(0)
	SPSOC	4.34×10^{-70}	2.51×10^{-21}	1.37×10^{-20}	=(0)	5.00×10^{-64}	2.23×10^{-8}	1.22×10^{-7}	=(0)	7.22×10^{-63}	3.73×10^{-20}	2.04×10^{-19}	=(0)
	SPSORC	0	0	0		0	0	0		0	0	0	
f_{19}	bPSO	9.08×10^{-4}	2.66×10^{-3}	4.25×10^{-4}	=(0)	1.59×10^{-19}	1.68×10^{-19}	3.35×10^{-21}	+(1)	1.88×10^{-40}	1.94×10^{-40}	3.45×10^{-42}	−(−1)
	PSOd	5.66×10^{-4}	9.64×10^{-4}	3.49×10^{-4}	−(−1)	8.35×10^{-18}	1.41×10^{-16}	2.64×10^{-16}	+(1)	3.03×10^{-31}	1.19×10^{-28}	3.03×10^{-28}	−(−1)
	HPSOscac	1.59×10^{-3}	3.43×10^{-3}	3.74×10^{-4}	=(0)	1.79×10^{-19}	1.79×10^{-19}	4.90×10^{-35}	+(1)	2.04×10^{-40}	2.04×10^{-40}	4.15×10^{-56}	−(−1)
	TCPSO	2.12×10^{-3}	3.50×10^{-3}	2.34×10^{-3}	=(0)	1.45×10^{-19}	9.59×10^{-16}	5.21×10^{-15}	+(1)	1.69×10^{-40}	4.24×10^{-34}	2.32×10^{-33}	−(−1)
	SPSO	7.91×10^{-3}	5.49×10^{-2}	3.83×10^{-2}	+(1)	1.32×10^{-10}	4.32×10^{-7}	8.07×10^{-7}	=(0)	1.24×10^{-16}	2.36×10^{-12}	1.22×10^{-11}	=(0)
	SPSOC	2.34×10^{-2}	9.80×10^{-2}	5.49×10^{-2}	+(1)	1.79×10^{-8}	3.53×10^{-5}	4.99×10^{-5}	=(0)	2.94×10^{-13}	1.58×10^{-9}	4.91×10^{-9}	=(0)
	SPSORC	0	9.35	2.22×10^{-2}		0	3.81×10^{-7}	6.51×10^{-7}		3.57×10^{-18}	1.13×10^{-13}	2.35×10^{-13}	
f_{20}	bPSO	3.97×10^{-25}	3.97×10^{-25}	1.87×10^{-40}	+(1)	9.83×10^{-123}	9.83×10^{-123}	0	+(1)	9.66×10^{-245}	9.66×10^{-245}	0	+(1)
	PSOd	4.66×10^{-25}	2.90×10^{-19}	1.51×10^{-18}	+(1)	4.66×10^{-63}	1.10×10^{-51}	5.13×10^{-51}	+(1)	7.68×10^{-90}	1.59×10^{-73}	8.73×10^{-73}	+(1)
	HPSOscac	3.97×10^{-25}	3.97×10^{-25}	1.87×10^{-40}	+(1)	9.83×10^{-123}	9.83×10^{-123}	0	+(1)	9.66×10^{-245}	9.66×10^{-245}	0	+(1)
	TCPSO	3.97×10^{-25}	3.97×10^{-25}	1.87×10^{-40}	+(1)	9.83×10^{-123}	9.83×10^{-123}	0	+(1)	9.66×10^{-245}	9.66×10^{-245}	0	+(1)
	SPSO	9.48×10^{-14}	2.11×10^{-8}	9.45×10^{-8}	+(1)	8.01×10^{-31}	1.32×10^{-21}	4.61×10^{-21}	+(1)	1.95×10^{-49}	3.65×10^{-34}	1.91×10^{-33}	+(1)
	SPSOC	1.15×10^{-12}	4.77×10^{-7}	1.37×10^{-6}	+(1)	1.19×10^{-22}	2.22×10^{-16}	4.89×10^{-16}	+(1)	1.69×10^{-41}	5.40×10^{-22}	2.89×10^{-21}	+(1)
	SPSORC	−1.00	−1.00	0		−1.00	$−9.33 \times 10^{-1}$	2.54×10^{-1}		−1.00	$−9.00 \times 10^{-1}$	3.05×10^{-1}	

Table 6. Cont.

		10				50				100			
		min	mean	std	ttest	min	mean	std	ttest	min	mean	std	ttest
f_{21}	bPSO	3.73×10^{-7}	3.57×10^{-6}	3.53×10^{-6}	+(1)	5.81×10^{-19}	9.30×10^{-17}	1.28×10^{-16}	+(1)	1.04×10^{-32}	4.59×10^{-29}	9.83×10^{-29}	+(1)
	PSOd	2.54×10^{-8}	8.53×10^{-6}	1.06×10^{-5}	+(1)	1.35×10^{-20}	6.03×10^{-20}	5.19×10^{-20}	+(1)	3.97×10^{-39}	7.51×10^{-38}	9.24×10^{-38}	+(1)
	HPSOscac	−1.00	-5.27×10^{-1}	5.13×10^{-1}	+(1)	−1.00	-3.33×10^{-2}	1.83×10^{-1}	+(1)	3.14×10^{-31}	1.87×10^{-14}	5.55×10^{-14}	+(1)
	TCPSO	8.71×10^{-7}	1.33×10^{-5}	1.67×10^{-5}	+(1)	6.93×10^{-20}	2.42×10^{-18}	3.80×10^{-18}	+(1)	1.30×10^{-38}	1.17×10^{-32}	5.38×10^{-32}	+(1)
	SPSO	4.39×10^{-4}	1.04×10^{-3}	3.19×10^{-4}	+(1)	4.00×10^{-16}	2.15×10^{-14}	1.99×10^{-14}	+(1)	2.51×10^{-29}	6.77×10^{-27}	1.10×10^{-26}	+(1)
	SPSOC	6.88×10^{-4}	3.24×10^{-3}	1.69×10^{-3}	+(1)	9.35×10^{-14}	7.28×10^{-12}	1.12×10^{-11}	+(1)	1.13×10^{-25}	3.10×10^{-22}	6.75×10^{-22}	+(1)
	SPSORC	−1.00	−1.00	0		−1.00	−1.00	2.92×10^{-17}		−1.00	−1.00	2.92×10^{-17}	
f_{22}	bPSO	2.26	2.62×10^1	2.76×10^1	+(1)	1.16×10^3	2.76×10^3	4.53×10^3	+(1)	3.39×10^3	1.70×10^5	5.90×10^5	=(0)
	PSOd	5.39×10^{-1}	4.98	2.55	+(1)	1.80×10^2	3.28×10^2	9.49×10^1	+(1)	7.23×10^2	9.14×10^2	1.20×10^2	+(1)
	HPSOscac	0	1.93×10^{-174}	0	+(1)	0	1.17×10^{-167}	0	+(1)	0	1.89×10^{-143}	1.04×10^{-142}	=(0)
	TCPSO	5.77×10^{-1}	1.15×10^1	1.52×10^1	+(1)	1.03×10^3	1.60×10^4	5.36×10^4	=(0)	2.51×10^3	6.23×10^5	2.09×10^6	=(0)
	SPSO	1.63×10^{-48}	1.59×10^{-43}	6.83×10^{-43}	=(0)	1.69×10^{-40}	3.13×10^{-38}	6.42×10^{-38}	+(1)	1.89×10^{-40}	3.88×10^{-38}	8.35×10^{-38}	=(0)
	SPSOC	1.29×10^{-113}	1.69×10^{-81}	8.46×10^{-81}	=(0)	3.35×10^{-61}	2.71×10^{-29}	1.08×10^{-28}	+(1)	8.00×10^{-61}	3.24×10^{-20}	1.76×10^{-19}	=(0)
	SPSORC	0	0	0		0	0	0		0	1.64×10^{-4}	8.99×10^{-4}	

The search results of the heuristic algorithm are random, so the average value (mean) of the results after multiple searches is the most valuable data. Observing the mean values in Table 6, when searching for 10-dimensional functions, we can see that SPSORC outperforms the other six PSO on 16 functions ($f_1, f_2, f_3, f_4, f_6, f_7, f_{10}, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{18}, f_{20}, f_{21}$ and f_{22}) in terms of the criteria ‘mean’, and 15 out of 16 were theoretical optimal solutions. Secondly, the number of that SPSOC find the minimum mean solutions for 10-dimensional functions is 6 times ($f_1, f_5, f_7, f_8, f_9, f_{10}$). PSOd, TCPSo and SPSO only find the minimum mean once. bPSO and HPSOscac are unable to search for the minimum mean at one time. Regarding the three functions (f_1, f_7, f_{10}), SPSOC and SPSORC can obtain the same mean.

On the other hand, both SPSORC and HPSOscac can find the same values in many functions including $f_2, f_3, f_4, f_6, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{21}$ and f_{22} . In addition, SPSORC has achieved the best results for ninth, indicating that SPSORC and HPSOscac have the opportunity to yield a better solution than the average one, but the results are volatility, especially for HPSOscac.

The use of standard deviation (std) can observe the volatility of the algorithm results. The standard deviation is a measure of the degree to which a set of data averages is dispersed. A larger standard deviation represents a larger difference between most of the values and their average values; a smaller standard deviation means that these values are closer to the average. It is clear that the standard deviation of SPSORC is almost the smallest of all algorithms.

The comparison of ‘min’, ‘mean’ and ‘std’ from 50-dimensional and 100-dimensional dimensional functions between SPSORC and other six PSO variants is also illustrated in Table 6. It is clearly seen that, for both 50-dimensional and 100-dimensional functions except for $f_9, f_{11}, f_{15}, f_{19}$, SPSORC is able to obtain better ‘mean’ than the most of the other improved strategies. In 50-dimensional optima values, SPSORC outperforms the other six PSOs on 17 functions ($f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_{10}, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{18}, f_{20}, f_{21}$ and f_{22}), in which, 14 out of 17 were searched for theoretical optima. SPSOC has searched for the best solution five times ($f_1, f_5, f_7, f_8, f_{10}$). Then, PSOd and SPSO has two times (f_{15}, f_{19}) and once (f_9), respectively. Regrettably, other algorithms have no chance. Almost the same situation also appears in 100-dimensional results. In addition, as for ‘std’ of SPSORC, it should be noted that the ‘std’ without any fluctuations many times is markedly superior to that of the others. Second, with regard to the experimental results comparison in 100-dimensional results of Table 6, SPSOC is also able to achieve good performance with smaller mean value such as for f_1, f_5, f_7, f_9 and f_{10} out of the twenty-two 100-dimensional test functions.

To sum up, from Table 6, it has been identified experimentally that SPSORC is superior or highly competitive with several improved PSO variants, and this improving strategy is shown to be able to find fairly good solutions for most of the well-known benchmark functions.

Table 7 is a summary of the scores based on the *t*-test analysis of search results in three kinds of dimensions. *Score* is the net score of SPSORC, which is better than the score obtained by the comparison function minus the number of comparison functions. Take the comparison result of SPSOC and RSMPSoC in Table 7 as an example, that is, the A6 algorithm in 100-dimensional in Table 7. The SPSORC result is seven times better than SPSOC and three times worse than SPSOC. Therefore, the net score of SPSORC is: $Score = 7 - 3 = 4$, and the calculation process of other net scores is the same.

Observed from Table 7, SPSORC has a stable net score for these six algorithms, all greater than 0. Careful observation can show us that the performance is slightly higher in the 50-dimensional scores compared to 10-and-100 dimensions. It is again proven that the capability of SPSORC is better than bPSO, PSOd, HPSOscac, TCPSo, SPSO and SPSOC, especially when solving the 50-dimensional problem. The convergence curves of seven improved PSO algorithms on twenty-two benchmark functions with 100 dimensions are plotted in Section 4.5.1 Figure 4, respectively.

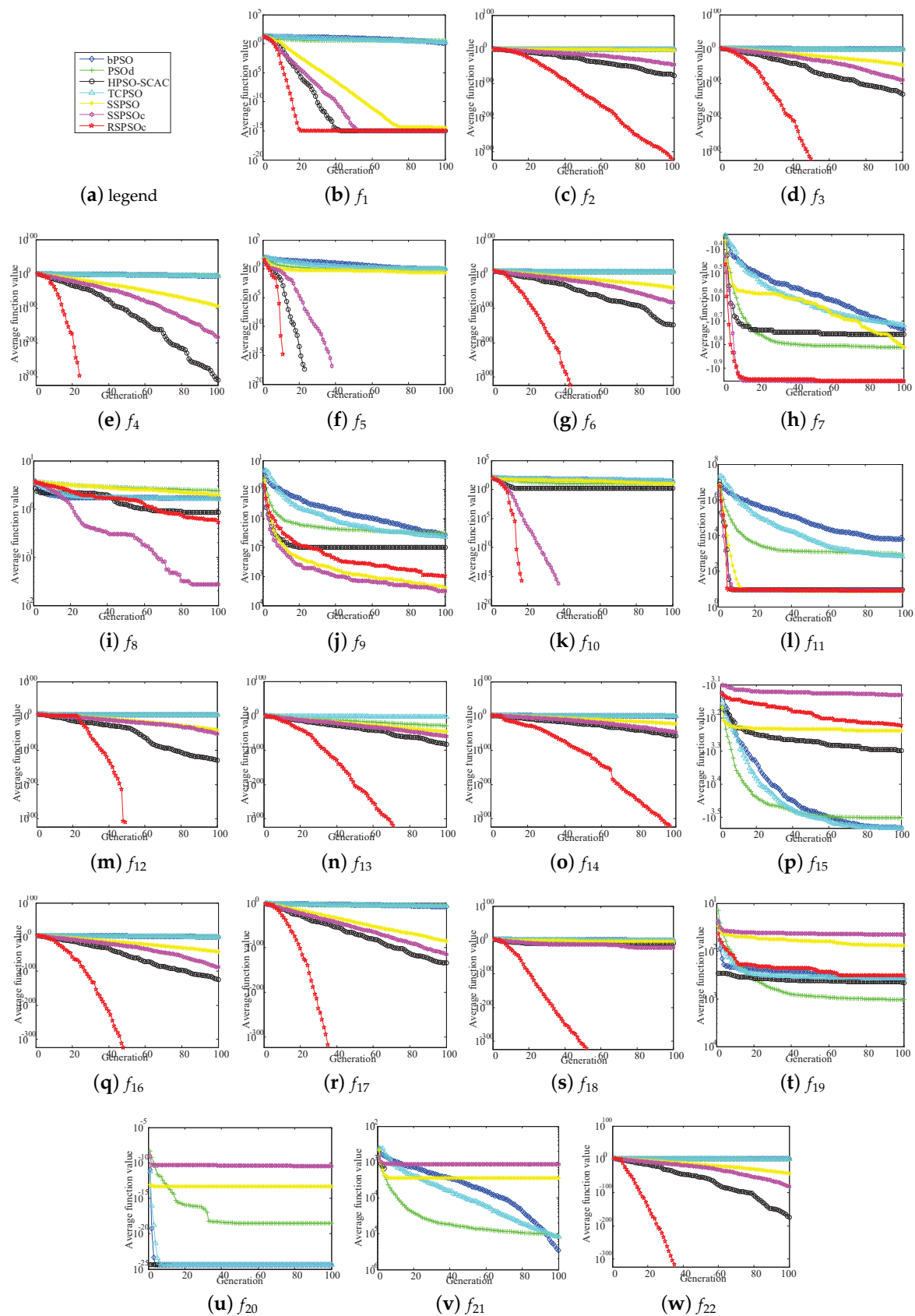


Figure 4. Average convergence curves of seven improved PSO algorithms for twenty-two functions in 10 dimensions.

Table 7. Simulation environment.

<i>N</i>	Results	bPSO	PSOd	HPSOscac	SPSO	SPSOC	SPSORC
10	+	18	20	12	20	11	6
	=	4	1	10	1	10	14
	−	0	1	0	1	1	2
	Score	18	19	12	19	10	4
50	+	19	21	13	19	9	8
	=	2	1	9	3	11	13
	−	1	0	0	0	2	1
	Score	18	21	13	19	7	7
100	+	18	18	10	16	9	7
	=	2	2	11	4	12	12
	−	2	2	1	2	1	3
	Score	16	16	9	14	8	4

Figure 4a is the legend for the other twenty-two convergence curves. Figure 4b indicates that, on f_1 , PPSOC converges the fastest in the early stage among the seven improvements. HPSOscac converges relatively slowly compared to SPSORC. The order of performance on f_1 is SPSORC, HPSOscac, SPSOC, SPSO, bPSO, PSOd, TCP SO. Almost the same situation occurs simultaneously on the other 11 function convergence curves. This should be the effect of algorithm simplifying so that the algorithm can converge very quickly in the early stage. On f_3, f_4, f_5 , et al., RPSO has found the best solution within the maximum generation. Figure 4 l,m, on f_{11} and f_{12} , show that SPSORC converges relatively slowly compared to HPSOscac in the beginning, but it surpasses HPSOscac in about 20th generation.

Next, it is further analyzed by the box diagram in Figure 5. Box diagram is mainly used to reflect the characteristics of the original data distribution, and can also compare the distribution characteristics of multiple sets of data. On the same number of axes, the box plots of several sets of data are arranged in parallel. Shape information such as median, tail length, outliers, and distribution intervals of several batches of data can be seen at a glance. + indicates an abnormal point.

From Figure 5a, the order of these boxes from high to low is bPSO, TCP SO, PSOd, HPSOscac, SPSO, SPSORC and SPSOC, respectively. The upper quartile and median values of bPSO and TCP SO are closer to the upper edge, which indicates that the data of the two algorithms are more biased toward larger values. In comparison, the box of PSOd is more symmetrical and the data distribution is relatively uniform. Unfortunately, the distribution of the boxes of these three algorithms is too high, and the search results are not good. The box of HPSOscac is at the bottom of the coordinate system. However, we can clearly see that there are many outliers in its data. Some even have exceeded the median of PSOd. Its skewed nature tends to be smaller, but the distribution of data is more scattered. Compared with the above four algorithms, the distribution of the boxes of the three algorithms proposed in this paper is obviously more optimistic. The optimization results of the three algorithms are almost neat, concentrated and smaller. The situation of other box diagrams is not much different from that of Figure 5a. Throughout the 22 box diagrams in Figure 5, the bPSO, PSOd, HPSOscac and TCP SO seem to have more difficulty locating the solution than the SPSORC for from the box diagrams. The boxes of PSOd are mostly too top, followed by TCP SO. HPSOscac has a lot of outliers. Its maximum and minimum span is large, and distribution is extremely non-uniform and decentral. It is observed that the results of HPSOscac are highly volatile and the improvement of the algorithm is unstable. This may be related to its weight mixed with the trigonometric function. For the above reasons, the results of SPSORC and SPSOC are not obvious in the box diagram, almost all posted at the bottom. Combining the results of Table 6, we can roughly know that SPSORC has better performance, and the more oblate box can show that the 30 search results have little differences and the performance is very stable.

To sum up, the test results indicate that: Both confidence term and random weight can enhance diversity. The former can yield a significant improvement in performance, while the latter can preserve much more diversity. The aforementioned two methods are compatible. Combining both of them with SPSO can preserve the highest diversity and achieve the best overall performance among the six improved strategies.

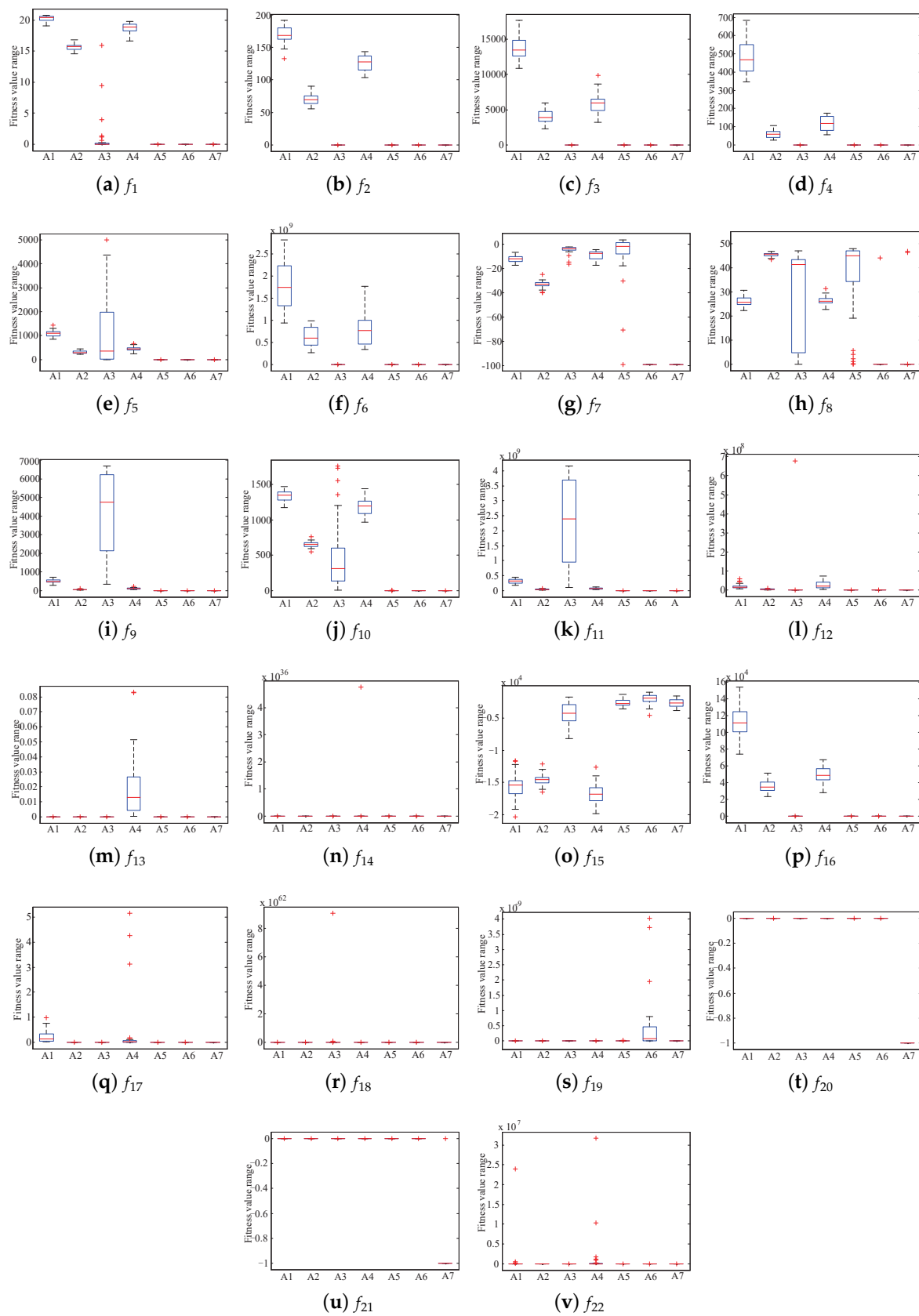


Figure 5. Box diagram of thirty results, each function with 100 dimensions.

4.5.2. Algorithm Complexity Analysis

Comparing the steps of the bPSO algorithm, the time complexity of SPSO's two improvements mainly depends on two aspects: (1) random initialization, and (2) particle velocity and position updating. These two parts can all be expressed as $O(m \times N)$ by the Big O notation [72], in that, m is the population and N is the problem dimensions. In this paper, we haven't changed the algorithm's initialization method, so we only compare the time complexity from particle velocity and position updating. The SPSO, which does not consider the inertia weight updating and confidence term calculating, has a reduced computational complexity compared to the basic particle swarm optimization algorithm, but the Big O notation can also be represented by $O(m \times N)$. Compared with the bPSO and the SPSO, the most complex algorithm we proposed is named SPSORC, which has increased weight, and the confidence term has surely increased in computational complexity, but we can see from Table 1 that its loop body has not changed. According to the Big O notation, its time complexity is still $O(m \times N)$. Overall, the complexity of SPSO and its two improved ones are not increased by orders of magnitude.

Then, we analyze the real computational time from Table 8. In Table 8, we show the computational time for three kinds of dimension functions.

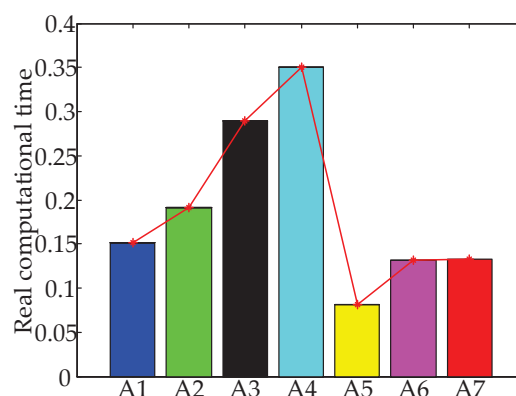
Table 8. Real computational time.

N	Alg	bPSO	PSOd	HPSOscac	TCP SO	SPSO	SPSOC	SPSORC
10	f_1	0.0346	0.0408	0.0681	0.0755	0.0193	0.0293	0.0298
	f_2	0.0324	0.0396	0.0651	0.0731	0.0188	0.0295	0.0296
	f_3	0.0309	0.0374	0.0606	0.0673	0.0178	0.0280	0.0274
	f_4	0.0378	0.0465	0.0700	0.0857	0.0269	0.0374	0.0367
	f_5	0.0334	0.0421	0.0655	0.0783	0.0196	0.0298	0.0304
	f_6	0.0358	0.0444	0.0676	0.0825	0.0245	0.0349	0.0348
	f_7	0.0380	0.0449	0.1343	0.0856	0.0244	0.0328	0.0332
	f_8	0.0542	0.0626	0.0867	0.1214	0.0423	0.0517	0.0561
	f_9	0.0484	0.0559	0.0815	0.1066	0.0362	0.0470	0.0475
	f_{10}	0.0329	0.0410	0.0654	0.0758	0.0199	0.0294	0.0311
	f_{11}	0.0303	0.0388	0.0623	0.0704	0.0174	0.0278	0.0286
	f_{12}	0.0428	0.0517	0.0760	0.0967	0.0313	0.0421	0.0423
	f_{13}	0.0288	0.0378	0.0622	0.0676	0.0180	0.0285	0.0289
	f_{14}	0.0290	0.0381	0.0608	0.0690	0.0181	0.0286	0.0280
	f_{15}	0.0355	0.0435	0.0736	0.0803	0.0215	0.0325	0.0324
	f_{16}	0.0288	0.0379	0.0624	0.0681	0.0181	0.0289	0.0280
	f_{17}	0.0365	0.0462	0.0681	0.0852	0.0265	0.0368	0.0363
	f_{18}	0.0471	0.0555	0.0800	0.1043	0.0347	0.0466	0.0466
	f_{19}	0.0358	0.0428	0.0676	0.0812	0.0220	0.0319	0.0319
	f_{20}	0.0447	0.0543	0.0769	0.1006	0.0339	0.0439	0.0407
	f_{21}	0.0406	0.0488	0.0745	0.0914	0.0269	0.0369	0.0347
	f_{22}	0.0294	0.0381	0.0617	0.0695	0.0181	0.0292	0.0287
50	f_1	0.1515	0.1912	0.2889	0.3502	0.0813	0.1311	0.1325
	f_2	0.1515	0.1874	0.2897	0.3465	0.0815	0.1329	0.1329
	f_3	0.1323	0.1718	0.2726	0.3145	0.0732	0.1247	0.1241
	f_4	0.1798	0.2172	0.3179	0.4042	0.1174	0.1713	0.1704
	f_5	0.1589	0.1976	0.2964	0.3633	0.0868	0.1378	0.1383
	f_6	0.1761	0.2157	0.3130	0.3969	0.1163	0.1687	0.1678
	f_7	0.1820	0.2190	0.5396	0.4080	0.1150	0.1543	0.1552
	f_8	0.2847	0.3195	0.4206	0.5978	0.2137	0.2620	0.2682
	f_9	0.2330	0.2727	0.3809	0.5160	0.1716	0.2246	0.2250
	f_{10}	0.1554	0.1933	0.2921	0.3546	0.0851	0.1343	0.1356
	f_{11}	0.1380	0.1775	0.2819	0.3236	0.0765	0.1289	0.1295
	f_{12}	0.2696	0.3012	0.4074	0.5677	0.1996	0.2533	0.2527
	f_{13}	0.1306	0.1709	0.2713	0.3072	0.0702	0.1236	0.1246
	f_{14}	0.1324	0.1733	0.2629	0.3136	0.0744	0.1287	0.1276
	f_{15}	0.1569	0.1965	0.3368	0.3581	0.0913	0.1459	0.1466
	f_{16}	0.1348	0.1727	0.2703	0.3113	0.0727	0.1262	0.1244
	f_{17}	0.1738	0.2125	0.3016	0.3952	0.1126	0.1623	0.1649
	f_{18}	0.2277	0.2688	0.3719	0.5035	0.1682	0.2176	0.2197
	f_{19}	0.1644	0.2027	0.2985	0.3724	0.0985	0.1469	0.1495
	f_{20}	0.2089	0.2507	0.3380	0.4681	0.1501	0.1996	0.1881
	f_{21}	0.1892	0.2241	0.3240	0.4226	0.1209	0.1707	0.1541
	f_{22}	0.1308	0.1680	0.2694	0.3077	0.0700	0.1220	0.1222

Table 8. Cont.

N	Alg	bPSO	PSOd	HPSOscac	TCPSO	SPSO	SPSOC	SPSORC
100	f_1	0.2977	0.3719	0.5642	0.6830	0.1561	0.2571	0.2612
	f_2	0.3023	0.3793	0.5775	0.6944	0.1595	0.2655	0.2660
	f_3	0.2665	0.3431	0.5417	0.6247	0.1416	0.2485	0.2491
	f_4	0.3588	0.4369	0.6381	0.8083	0.2379	0.3400	0.3387
	f_5	0.3158	0.3929	0.5894	0.7232	0.1716	0.2729	0.2756
	f_6	0.3555	0.4288	0.6287	0.7992	0.2284	0.3352	0.3343
	f_7	0.3655	0.4419	1.3028	0.8107	0.2323	0.3049	0.3061
	f_8	0.5580	0.6271	0.8395	1.2018	0.4311	0.5230	0.5311
	f_9	0.4651	0.5443	0.7505	1.0211	0.3380	0.4447	0.4465
	f_{10}	0.3098	0.3841	0.5771	0.7101	0.1667	0.2646	0.2687
	f_{11}	0.2744	0.3521	0.5583	0.6413	0.1477	0.2529	0.2522
	f_{12}	0.6752	0.7524	0.9544	1.4332	0.5468	0.6518	0.6535
	f_{13}	0.2651	0.3426	0.5400	0.6193	0.1385	0.2442	0.2468
	f_{14}	0.3217	0.3646	0.5121	0.6165	0.1431	0.2478	0.2477
	f_{15}	0.3154	0.3918	0.6178	0.7155	0.1852	0.2899	0.2911
	f_{16}	0.2632	0.3413	0.5416	0.6519	0.1466	0.2518	0.2498
	f_{17}	0.3599	0.4450	0.6326	0.8335	0.2255	0.3279	0.3377
	f_{18}	0.4758	0.5581	0.7631	1.0589	0.3491	0.4483	0.4592
	f_{19}	0.3406	0.4093	0.6189	0.7589	0.1980	0.2975	0.3253
	f_{20}	0.4381	0.5238	0.6735	0.9385	0.2970	0.4000	0.3777
	f_{21}	0.3812	0.4604	0.6413	0.8513	0.2433	0.3474	0.3103
	f_{22}	0.2658	0.3424	0.5421	0.6204	0.1367	0.2440	0.2443

The time in Table 8 is the average time required to run 30 times independently. The average length of time varies slightly depending on the problem. Observing the running time of the seven algorithms, it is certain that the running time of SPSORC is similar to the computational time of other algorithms. It is clear that the lowest running time is obtained by SPSO, since it greatly simplifies bPSO. SPSOC has increased slightly over time due to confidence term. HPSOscac's trigonometric function improvement strategy makes the algorithm better applicable to multimodal problems. However, because the regularity distribution of the trigonometric function increases the particle diversity, the particle is difficult to converge at the later stage, and the actual calculation time is longer. TCPSO uses the dual population to optimize problems through information exchange. Thus, SPSO and its improved strategies do not simply consume runtime to improve algorithm performance. The real computational time is basically distributed as Figure 6. A1–A7 are namely bPSO, PSOd, HPSOscac, TCPSO, SPSO, SPSOC and SPSORC.

Figure 6. Real computational time of f_{11} in 50 dimensions.

4.5.3. Success Rate and Average Iteration Times

The success rate (SR) is the percentage between the times that each algorithm can successfully achieve convergence accuracy for function optimization and the total number of times. The average iterations times (AIT) is the average iteration numbers required by the algorithm to find the convergence accuracy. The former can examine the stability and accuracy of the algorithm, while the

latter mainly examines the efficiency of the algorithm. The convergence accuracy used for the success rate and the average iteration times in this paper is the accuracy of the 50-dimensional test functions we want to meet. The specific values can refer to the last column of Table A1. The other parameters are set according to Table 2. Figure 7 shows the Radar charts with average iteration times of eight functions. We can surely, in Figure 7, find that the point of SPSORC is always close to the center origin.

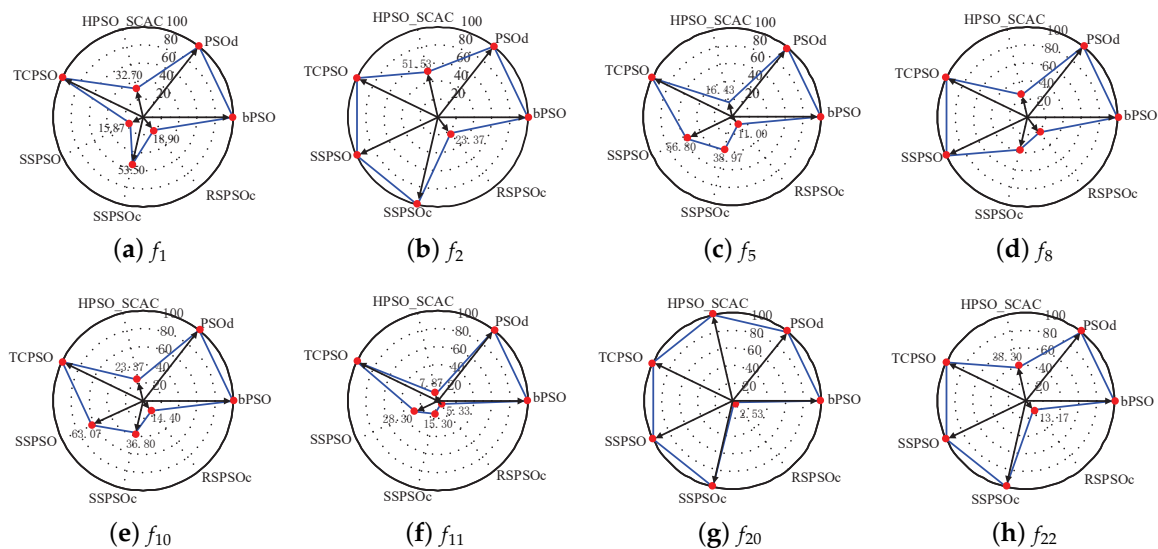


Figure 7. Radar charts of average iteration times.

Specific data for the average iteration times and the success rate of the seven algorithms in solving 50-dimensional problems are referred to in Table ‘AIT’ is the average iteration times. ‘SR’ represents the success rate. In order to facilitate the use of differentials, ‘SR’ is expressed in percentage form. The results round off to retain two digits after the decimal point. ‘—’ means that the algorithm fails to search this function when convergence accuracy is reached within the maximum generations. For the convenience of observation, we will show the minimum AIT and the maximum SR for each function in the bold format.

The results can be analyzed from Table 9. bPSO, PSOd, TCPSO and SPSO have higher success rates three times. SPSOC and HPSOscac gets six and seven times, respectively. SPSORC, surprisingly, has 15 times, and seven of them (f_{12} , f_{13} , f_{14} , f_{17} , f_{18} , f_{20} , f_{21}) have the best success rate that none of the other six algorithms have achieved. One can see that the SPSORC strategy has a wider range of types, high precision and stability. Comparing the average iteration times, it is clearly shown that TCPSO, SPSO and SPSOC do not get the lowest average iteration times.

The above two data comparisons reveal that SPSORC has a large advantage compared with the other six algorithms. Not only is it more stable, but the search efficiency is also faster. When faced with a unimodal function, SPSORC can converge to effective precision more quickly. For other multi-peak complex problems, it is not to be outdone, except for the extremely difficult functions such as Rosenbrock Function and Schwefel’s Problem 2.26, which show weak stability. It can be stated that the improved method can be adapted to a variety of test environments, and the results are quite excellent.

Table 9. Success rate and average iteration times.

	bPSO		PSOd		HPSO _{scac}		TCPSO		SPSO		SPSOC		SPSORC	
	AIT	SR	AIT	SR	AIT	SR	AIT	SR	AIT	SR	AIT	SR	AIT	SR
f_1	-	0.00%	-	0.00%	32.70	0.00%	-	0.00%	15.87	3.33%	53.50	100.00%	18.90	93.33%
f_2	-	0.00%	-	0.00%	51.53	100.00%	-	0.00%	-	0.00%	-	0.00%	23.37	100.00%
f_3	-	0.00%	-	0.00%	35.33	100.00%	-	0.00%	-	100.00%	77.33	100.00%	17.50	100.00%
f_4	-	0.00%	-	0.00%	58.60	100.00%	-	0.00%	-	0.00%	-	0.00%	16.47	100.00%
f_5	-	0.00%	-	0.00%	16.43	0.00%	-	0.00%	56.80	76.67%	38.97	100.00%	11.00	100.00%
f_6	-	0.00%	-	0.00%	54.73	100.00%	-	0.00%	-	0.00%	-	0.00%	20.70	100.00%
f_7	-	100.00%	-	100.00%	-	36.67%	-	100.00%	-	100.00%	34.50	100.00%	13.77	100.00%
f_8	-	0.00%	-	0.00%	25.00	0.00%	-	0.00%	-	0.00%	37.03	96.67%	21.80	96.67%
f_9	-	0.00%	-	0.00%	6.27	0.00%	-	0.00%	12.30	96.67%	7.40	100.00%	11.80	66.67%
f_{10}	-	0.00%	-	0.00%	23.37	0.00%	-	0.00%	63.07	53.33%	36.80	100.00%	14.40	100.00%
f_{11}	-	0.00%	-	0.00%	7.87	0.00%	-	0.00%	28.30	100.00%	15.30	100.00%	5.33	100.00%
f_{12}	-	0.00%	-	0.00%	56.93	100.00%	-	0.00%	-	0.00%	-	0.00%	20.97	100.00%
f_{13}	2.83	0.00%	-	0.00%	62.93	93.33%	4.67	0.00%	-	0.00%	-	0.00%	19.57	100.00%
f_{14}	-	0.00%	-	0.00%	66.70	96.67%	-	0.00%	-	0.00%	-	0.00%	16.50	100.00%
f_{15}	1.20	100.00%	1.10	100.00%	1.20	60.00%	1.37	100.00%	1.23	16.67%	4.27	0.00%	2.30	10.00%
f_{16}	-	0.00%	-	0.00%	57.73	100.00%	-	0.00%	-	0.00%	-	0.00%	24.50	100.00%
f_{17}	-	0.00%	-	0.00%	26.63	43.33%	-	0.00%	-	0.00%	-	0.00%	20.07	100.00%
f_{18}	-	0.00%	-	0.00%	14.63	36.67%	-	0.00%	-	0.00%	-	0.00%	8.20	100.00%
f_{19}	1.67	100.00%	6.30	100.00%	1.00	100.00%	1.60	100.00%	9.20	3.33%	4.07	0.00%	1.20	16.67%
f_{20}	-	0.00%	-	0.00%	-	0.00%	-	0.00%	-	0.00%	-	0.00%	2.53	90.00%
f_{21}	-	0.00%	-	0.00%	0.63	0.00%	-	0.00%	-	0.00%	-	0.00%	16.73	96.67%
f_{22}	-	0.00%	-	0.00%	38.30	100.00%	-	0.00%	-	0.00%	-	0.00%	13.17	100.00%

5. Conclusions

Due to the effect on particle swarm optimization, in this paper, a Simple Particle Swarm Optimization based on Confidence term and Random inertia weight namely SPSORC has been proposed. SPSORC adopts three different improving strategies—first, particle updating formulas only use positional items and social items to enhance the exploration capability; second, the confidence term is introduced to increase particle diversity and avoid excessive particle convergence. Finally, a random inertia weight is formulated to keep the balance between exploration and exploitation. Extensive experiments in Section 4 on twenty-two benchmark functions validate and discuss SPSO and its further improvements' effectiveness, efficiency, robustness and scalability. It has been demonstrated that, in most cases, SPSORC performs a better capability of exploitation and exploration than, or at least highly competitively with, basic PSO and its state-of-the-art improved ones introduced in this paper.

In our future work, we intend to incorporate different initialization strategies, multi-swarm and hybrid algorithms into SPSORC. This may result in very competitive algorithms. Obviously, many adaptive methods for PSO have been proposed. In order to improve the performance of the proposed approach and its application, the research on particle swarm optimization algorithm and its improvements is a promising research direction. Furthermore, we will apply the proposed approach to solve some other practical existing engineering optimization problems, e.g., machine-tool spindle design, logistics distribution region partitioning problem, economic load dispatch problem, etc. With these evolutionary algorithms, it is unnecessary to know the computing environment and to calculate the gradient and other information. Thus, it is helpful to save on the cost of computing. Even better, we can calculate the problem with more dimensions and goals at once, including some discontinuous problems.

Author Contributions: X.Z. suggested the improving strategy and wrote the original draft preparation. D.Z. was responsible for checking this paper. X.S. provided a provincial project.

Funding: The National Natural Science Foundation of China (No. 61403174) and the Postgraduate Research and Practice Innovation Program of Jiangsu Province (No. KYCX17_1575, No. KYCX17_1576).

Acknowledgments: This work was supported by the National Natural Science Foundation of China (No. 61403174) and the Postgraduate Research and Practice Innovation Program of Jiangsu Province (No. KYCX17_1575, No. KYCX17_1576).

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

Abbreviations

The following abbreviations are used in this manuscript:

PSO	Particle Swarm Optimization
bPSO	The basic PSO [21,22]
PSOd	A distribution-based update rule for PSO [52]
HPSOscac	A hybrid PSO with sine cosine acceleration coefficients [67]
TCPSO	A two-swarm cooperative PSO [45]
SPSO	Simple PSO
SPSOC	Simple PSO with Confidence Term
SPSORC	Simple PSO based on Random weight and Confidence term
v	Particle velocity
x	Particle position
p_{best}	Personal historical best solution
g_{best}	Global best solution
ω	Inertia weight
ω_{max}	The maximum weight
ω_{min}	The minimum weight
c_1	Self-cognitive factor
c_2	Social communication factor
U (in Figures 1–3)	Solution space of a function
O (in Figures 1–3)	The theoretical optima of a function
i	The current particle
n	The current dimension

N	The maximum dimension
t	The current generation
T	The upper limit of generation
NR	The number of times that algorithm search for problem
m	Population size
min	The minimum values from the optima in which algorithms search for the problem 30 times
mean	The average values for the optima in which algorithms search for the problem 30 times
std	The average values for the optima in which algorithms search for the problem 30 times
ttest (in Table 6)	t -test results
AIT (in Section 4.5.2)	Average iteration times
SR (in Section 4.5.2)	Success rate
A1–A7 (in Figures 5 and 6)	bPSO, PSOd, HPSOscac, TCPSO, SPSO, SPSOC and SPSORC, respectively

Appendix A. Benchmark Function Appendix

Table A1. Benchmark functions.

Instance	Expression	Domain	Analytical Solution	Accuracy (50)
Ackley's Path Function	$f_1(x) = -20e^{-0.2\sqrt{\frac{1}{30}\sum_{i=1}^N x_i^2}} - e^{\frac{1}{30}\sum_{i=1}^N \cos 2\pi x_i} + 20 + e$	$[-32, 32]$	$f_1(0, \dots, 0) = 8.88 \times 10^{-16}$	1×10^{-15}
Alpine Function	$f_2(x) = \sum_{i=1}^N x_i \sin(x_i) + 0.1x_i $	$[-10, 10]$	$f_2(0, \dots, 0) = 0$	1×10^{-60}
Axis Parallel Hyperellipsoid	$f_3(x) = \sum_{i=1}^N ix_i^2$	$[-5.12, 5.12]$	$f_3(0, \dots, 0) = 0$	1×10^{-15}
De Jong's Function 4 (no noise)	$f_4(x) = \sum_{i=1}^N ix_i^4$	$[-1.28, 1.28]$	$f_4(0, \dots, 0) = 0$	1×10^{-240}
Girewank Problem	$f_5(x) = \frac{1}{4000} \sum_{i=1}^N x_i^2 - \prod_{i=1}^N (\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	$f_5(0, \dots, 0) = 0$	1×10^{-15}
High Conditioned Elliptic Function	$f_6(x) = \sum_{i=1}^N (10^6)^{\frac{i-1}{n-1}} x_i^2$	$[-100, 100]$	$f_6(0, \dots, 0) = 0$	1×10^{-110}
Inverted Cosine	$f_7(x) = -\sum_{i=1}^{N-1} (e^{\frac{-x_i^2 - x_{i+1}^2 - 0.5x_i x_{i+1}}{8}})$	$[-5, 5]$	$f_7(0, \dots, 0) = -N + 1$	-4.9×10^{-1}
Wave Function	$\times \cos(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1}})$			
Pathological Function	$f_8(x) = \sum_{i=1}^{N-1} (0.5 + \frac{\sin^2 \sqrt{100x_i^2 + x_{i+1}^2 - 0.5}}{1 + \frac{1}{1000}(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2)})$	$[-100, 100]$	$f_8(0, \dots, 0) = 0$	1×10^{-5}
Quartic Function, i.e, noise	$f_9(x) = \sum_{i=1}^N ix_i^4 + \text{rand}[0, 1)$	$[-10, 10]$	$f_9(0, \dots, 0) = 0$	1×10^{-1}
Rastrigin Problem	$f_{10}(x) = \sum_{i=1}^N [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	$f_{10}(0, \dots, 0) = 0$	1×10^{-20}
Rosenbrock Problem	$f_{11}(x) = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]$	$f_{11}(0, \dots, 0) = 0$	5×10^1
Schwefel's Problem 1.2	$f_{12}(x) = \sum_{i=1}^N (\sum_{j=1}^m x_j)^2$	$[-100, 100]$	$f_{12}(0, \dots, 0) = 0$	1×10^{-100}
Schwefel's Problem 2.21	$f_{13}(x) = \max_i x_i , 1 \leq i \leq 30$	$[-100, 100]$	$f_{13}(0, \dots, 0) = 0$	1×10^{-80}
Schwefel's Problem 2.22	$f_{14}(x) = \sum_{i=1}^N x_i + \prod_{i=1}^N x_i $	$[-10, 10]$	$f_{14}(0, \dots, 0) = 0$	1×10^{-60}
Schwefel's Problem 2.26	$f_{15}(x) = \sum_{i=1}^N x_i \sin(x_i) + 0.1x_i $	$[-500, 500]$	$f_{15}(s, \dots, s) = -419 \text{ N}$ $s \approx 420.97$	-2.5×10^3
Sphere Function	$f_{16}(x) = \sum_{i=1}^N x_i^2$	$[-100, 100]$	$f_{16}(0, \dots, 0) = 0$	1×10^{-120}
Sum of Different Power Function	$f_{17}(x) = \sum_{i=1}^N x_i ^{i+1}$	$[-1, 1]$	$f_{17}(0, \dots, 0) = 0$	1×10^{-300}
Xin-She Yang 1	$f_{18}(x) = \sum_{i=1}^N \text{rand}[0, 1) \times x_i ^i$	$[-5, 5]$	$f_{18}(0, \dots, 0) = 0$	1×10^{-60}
Xin-She Yang 2	$f_{19}(x) = \frac{\sum_{i=1}^N x_i }{e^{\sum_{i=1}^N \sin x_i^2}}$	$[-2\pi, 2\pi]$	$f_{19}(0, \dots, 0) = 0$	1×10^{-8}

Table A1. Cont.

Instance	Expression	Domain	Analytical Solution	Accuracy (50)
Xin-She Yang 3	$f_{20}(x) = e^{-\sum_{i=1}^N (\frac{x_i}{\beta})^{2\alpha}} - 2e^{-\sum_{i=1}^N x_i^2} \prod_{i=1}^N \cos^2 x_i$ $\beta = 15, \alpha = 3$	$[-20, 20]$	$f_{20}(0, \dots, 0) = -1$	-1
Xin-She Yang 4	$f_{21}(x) = [\sum_{i=1}^N \sin^2 x_i - e^{-\sum_{i=1}^N x_i^2}] \sum_{i=1}^N \sin^2 \sqrt{ x_i }$	$[-10, 10]$	$f_{21}(0, \dots, 0) = -1$	-1
Zakharov Function	$f_{22}(x) = \sum_{i=1}^N x_i^2 + (\sum_{i=1}^N 0.5ix_i^2)^2 + (\sum_{i=1}^N 0.5ix_i^2)^4$	$[-5, 10]$	$f_{22}(0, \dots, 0) = 0$	1×10^{-80}

References

- Denysiuk, R.; Gaspar-Cunha, A. Multiobjective Evolutionary Algorithm Based on Vector Angle Neighborhood. *Swarm Evol. Comput.* **2017**, *37*, 663–670. [\[CrossRef\]](#)
- Wang, G.G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memet. Comput.* **2016**, *10*, 1–14. [\[CrossRef\]](#)
- Feng, Y.H.; Wang, G.G. Binary moth search algorithm for discounted 0–1 knapsack problem. *IEEE Access* **2018**, *6*, 10708–10719. [\[CrossRef\]](#)
- Grefenstette, J.J. Genetic algorithms and machine learning. *Mach. Learn.* **1988**, *3*, 95–99. [\[CrossRef\]](#)
- Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [\[CrossRef\]](#)
- Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [\[CrossRef\]](#)
- Kirkpatrick, S.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [\[CrossRef\]](#) [\[PubMed\]](#)
- Wang, G.G.; Gandomi, A.H.; Alavi, A.H.; Deb, S. A multi-stage krill herd algorithm for global numerical optimization. *Int. J. Artif. Intell. Tools* **2016**, *25*, 1550030. [\[CrossRef\]](#)
- Wang, G.G.; Deb, S.; Gandomi, A.H.; Alavi, A.H. Opposition-based krill herd algorithm with Cauchy mutation and position clamping. *Neurocomputing* **2016**, *177*, 147–157. [\[CrossRef\]](#)
- Wang, G.G.; Gandomi, A.H.; Alavi, A.H. An effective krill herd algorithm with migration operator in biogeography-based optimization. *Appl. Math. Model.* **2014**, *38*, 2454–2462. [\[CrossRef\]](#)
- Wang, G.G.; Guo, L.H.; Wang, H.Q.; Duan, H.; Liu, L.; Li, J. Incorporating mutation scheme into krill herd algorithm for global numerical optimization. *Neural Comput. Appl.* **2014**, *24*, 853–871. [\[CrossRef\]](#)
- Wang, G.G.; Gandomi, A.H.; Hao, G.S. Hybrid krill herd algorithm with differential evolution for global numerical optimization. *Neural Comput. Appl.* **2014**, *25*, 297–308. [\[CrossRef\]](#)
- Ding, X.; Guo, H.; Guo, S. Efficiency Enhancement of Traction System Based on Loss Models and Golden Section Search in Electric Vehicle. *Energy Procedia* **2017**, *105*, 2923–2928. [\[CrossRef\]](#)
- Ramos, H.; Monteiro, M.T.T. A new approach based on the Newton's method to solve systems of nonlinear equations. *J. Comput. Appl. Math.* **2016**, *318*, 3–13. [\[CrossRef\]](#)
- Fazio, A.R.D.; Russo, M.; Valeri, S.; Santis, M.D. Linear method for steady-state analysis of radial distribution systems. *Int. J. Electr. Power Energy Syst.* **2018**, *99*, 744–755. [\[CrossRef\]](#)
- Du, X.; Zhang, P.; Ma, W. Some modified conjugate gradient methods for unconstrained optimization. *J. Comput. Appl. Math.* **2016**, *305*, 92–114. [\[CrossRef\]](#)
- Pooranian, Z.; Shojafar, M.; Abawajy, J.H.; Abraham, A. An efficient meta-heuristic algorithm for grid computing. *J. Comb. Optim.* **2015**, *30*, 413–434. [\[CrossRef\]](#)
- Shojafar, M.; Chiaraviglio, L.; Blefari-Melazzi, N.; Salsano, S. P5G: A Bio-Inspired Algorithm for the Superfluid Management of 5G Networks. In Proceedings of the GLOBECOM 2017: 2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–7. [\[CrossRef\]](#)
- Shojafar, M.; Cordeschi, N.; Abawajy, J.H.; Baccarelli, E. Adaptive Energy-Efficient QoS-Aware Scheduling Algorithm for TCP/IP Mobile Cloud. In Proceedings of the IEEE Globecom Workshops, San Diego, CA, USA, 6–10 December 2015; pp. 1–6. [\[CrossRef\]](#)
- Zou, D.X.; Deb, S.; Wang, G.G. Solving IIR system identification by a variant of particle swarm optimization. *Neural Comput. Appl.* **2018**, *30*, 685–698. [\[CrossRef\]](#)

21. Kennedy, J. Particle Swarm Optimization. In Proceedings of the 1995 International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [\[CrossRef\]](#)
22. Shi, Y.H.; Eberhart, R.C. Empirical study of particle swarm optimization. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99(Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; pp. 1945–1950. [\[CrossRef\]](#)
23. Chen, Y.; Li, L.; Xiao, J.; Yang, Y.; Liang, J.; Li, T. Particle swarm optimizer with crossover operation. *Eng. Appl. Artif. Intell.* **2018**, *70*, 159–169. [\[CrossRef\]](#)
24. Zou, D.; Gao, L.; Li, S.; Wu, J.; Wang, X. A novel global harmony search algorithm for task assignment problem. *J. Syst. Softw.* **2010**, *83*, 1678–1688. [\[CrossRef\]](#)
25. Niu, W.J.; Feng, Z.K.; Cheng, C.T.; Wu, X.Y. A parallel multi-objective particle swarm optimization for cascade hydropower reservoir operation in southwest China. *Appl. Soft Comput.* **2018**, *70*, 562–575. [\[CrossRef\]](#)
26. Feng, Y.; Wang, G.G.; Deb, S.; Lu, M.; Zhao, X.J. Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. *Neural Comput. Appl.* **2017**, *28*, 1619–1634. [\[CrossRef\]](#)
27. Wang, G.G.; Deb, S.; Coelho, L.D.S. Elephant Herding Optimization. In Proceedings of the International Symposium on Computational and Business Intelligence, Bali, Indonesia, 7–9 December 2015; pp. 1–5. [\[CrossRef\]](#)
28. Wang, G.; Guo, L.; Gandomi, A.H.; Cao, L.; Alavi, A.H.; Duan, H.; Li, J. Levy-flight krill herd algorithm. *Math. Probl. Eng.* **2013**, *2013*, 682073. [\[CrossRef\]](#)
29. Wang, G.G.; Gandomi, A.H.; Alavi, A.H. Stud krill herd algorithm. *Neurocomputing* **2014**, *128*, 363–370. [\[CrossRef\]](#)
30. Wang, G.G.; Deb, S.; Coelho, L. Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Int. J. Bio-Inspired Comput.* **2018**, *12*, 1–22. [\[CrossRef\]](#)
31. Wang, G.G.; Chu, H.; Mirjalili, S. Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerosp. Sci. Technol.* **2016**, *49*, 231–238. [\[CrossRef\]](#)
32. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK, USA, 4–9 May 1998; pp. 69–73. [\[CrossRef\]](#)
33. Parsopoulos, K.E.; Vrahatis, M.N. Particle swarm optimizer in noisy and continuously changing environments. In *Artificial Intelligence and Soft Computing*; Hamza, M.H., Ed.; IASTED/ACTA Press: Anaheim, CA, USA, 2001; pp. 289–294.
34. Kennedy, J.; Mendes, R. Population structure and particle swarm performance. In Proceedings of the IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1671–1676. [\[CrossRef\]](#)
35. Clerc, M.; Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [\[CrossRef\]](#)
36. Xu, G.; Yu, G. Reprint of: On convergence analysis of particle swarm optimization algorithm. *J. Shanxi Norm. Univ.* **2018**, *4*, 25–32. [\[CrossRef\]](#)
37. Sun, J.; Wu, X.; Palade, V.; Fang, W.; Lai, C.H.; Xu, W.B. Convergence analysis and improvements of quantum-behaved particle swarm optimization. *Inf. Sci.* **2012**, *193*, 81–103. [\[CrossRef\]](#)
38. Li, S.F.; Cheng, C.Y. Particle Swarm Optimization with Fitness Adjustment Parameters. *Comput. Ind. Eng.* **2017**, *113*, 831–841. [\[CrossRef\]](#)
39. Li, N.J.; Wang, W.; Hsu, C.C.J. Hybrid particle swarm optimization incorporating fuzzy reasoning and weighted particle. *Neurocomputing* **2015**, *167*, 488–501. [\[CrossRef\]](#)
40. Chih, M.; Lin, C.J.; Chern, M.S.; Ou, T.Y. Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem. *J. Chin. Inst. Ind. Eng.* **2014**, *33*, 77–102. [\[CrossRef\]](#)
41. Ratnaweera, A.; Halgamuge, S.K.; Watson, H.C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans. Evol. Comput.* **2004**, *8*, 240–255. [\[CrossRef\]](#)
42. Kennedy, J.; Mendes, R. Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *IEEE Trans. Syst. Man Cybern. Part C* **2006**, *36*, 515–519. [\[CrossRef\]](#)
43. Zhao, S.Z.; Suganthan, P.N.; Pan, Q.K.; Tasgetiren, M.F. Dynamic multi-swarm particle swarm optimizer with harmony search. *Expert Syst. Appl.* **2011**, *38*, 3735–3742. [\[CrossRef\]](#)

44. Majercik, S.M. Using Fluid Neural Networks to Create Dynamic Neighborhood Topologies in Particle Swarm Optimization. In Proceedings of the International Conference on Swarm Intelligence, Brussels, Belgium, 10–12 September 2014; Springer: Cham, Switzerland; New York, NY, USA, 2014; Volume 8667, pp. 270–277. [\[CrossRef\]](#)
45. Sun, S.; Li, J. A two-swarm cooperative particle swarms optimization. *Swarm Evol. Comput.* **2014**, *15*, 1–18. [\[CrossRef\]](#)
46. Netjinda, N.; Achalakul, T.; Sirinaovakul, B. Particle Swarm Optimization inspired by starling flock behavior. *Appl. Soft Comput.* **2015**, *35*, 411–422. [\[CrossRef\]](#)
47. Beheshti, Z.; Shamsuddin, S.M. Non-parametric particle swarm optimization for global optimization. *Appl. Soft Comput.* **2015**, *28*, 345–359. [\[CrossRef\]](#)
48. Mendes, R.; Kennedy, J.; Neves, J. The fully informed particle swarm: Simpler, maybe better. *IEEE Trans. Evol. Comput.* **2004**, *8*, 204–210. [\[CrossRef\]](#)
49. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Subramanian, B. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [\[CrossRef\]](#)
50. Gong, Y.J.; Li, J.J.; Zhou, Y.; Li, Y.; Chung, H.S.H.; Shi, Y.H.; Zhang, J. Genetic Learning Particle Swarm Optimization. *IEEE Trans. Cybern.* **2017**, *46*, 2277–2290. [\[CrossRef\]](#)
51. Zou, D.; Li, S.; Li, Z.; Kong, X. A new global particle swarm optimization for the economic emission dispatch with or without transmission losses. *Energy Convers. Manag.* **2017**, *139*, 45–70. [\[CrossRef\]](#)
52. Kiran, M.S. Particle Swarm Optimization with a New Update Mechanism. *Appl. Soft Comput.* **2017**, *60*, 607–680. [\[CrossRef\]](#)
53. Wang, G.G.; Gandomi, A.H.; Alavi, A.H. A chaotic particle-swarm krill herd algorithm for global numerical optimization. *Kybernetes* **2013**, *42*, 962–978. [\[CrossRef\]](#)
54. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [\[CrossRef\]](#)
55. Yang, X.S. *Nature-Inspired Metaheuristic Algorithm*; Luniver Press: Beckington, UK, 2008; ISBN 1905986106, 9781905986101.
56. Wang, G.G.; Gandomi, A.H.; Yang, X.S.; Alavi, A.H. A novel improved accelerated particle swarm optimization algorithm for global numerical optimization. *Eng. Comput.* **2014**, *31*, 1198–1220. [\[CrossRef\]](#)
57. Liu, Y.; Niu, B.; Luo, Y. Hybrid learning particle swarm optimizer with genetic disturbance. *Neurocomputing* **2015**, *151*, 1237–1247. [\[CrossRef\]](#)
58. Chen, Y.; Li, L.; Peng, H.; Xiao, J.; Yang, Y.; Shi, Y. Particle Swarm Optimizer with two differential mutation. *Appl. Soft Comput.* **2017**, *61*, 314–330. [\[CrossRef\]](#)
59. Liu, Z.G.; Ji, X.H.; Liu, Y.X. Hybrid Non-parametric Particle Swarm Optimization and its Stability Analysis. *Expert Syst. Appl.* **2017**, *92*, 256–275. [\[CrossRef\]](#)
60. Bewoor, L.A.; Prakash, V.C.; Sapkal, S.U. Production scheduling optimization in foundry using hybrid Particle Swarm Optimization algorithm. *Procedia Manuf.* **2018**, *22*, 57–64. [\[CrossRef\]](#)
61. Xu, X.; Rong, H.; Pereira, E.; Trovati, M.W. Predatory Search-based Chaos Turbo Particle Swarm Optimization (PS-CTPSO): A new particle swarm optimisation algorithm for Web service combination problems. *Future Gener. Comput. Syst.* **2018**, *89*, 375–386. [\[CrossRef\]](#)
62. Guan, G.; Yang, Q.; Gu, W.W.; Jiang, W.; Lin, Y. Ship inner shell optimization based on the improved particle swarm optimization algorithm. *Adv. Eng. Softw.* **2018**, *123*, 104–116. [\[CrossRef\]](#)
63. Qin, Z.; Liang, Y.G. Sensor Management of LEO Constellation Using Modified Binary Particle Swarm Optimization. *Optik* **2018**, *172*, 879–891. [\[CrossRef\]](#)
64. Peng, Z.; Manier, H.; Manier, M.A. Particle Swarm Optimization for Capacitated Location-Routing Problem. *IFAC PapersOnLine* **2017**, *50*, 14668–14673. [\[CrossRef\]](#)
65. Copot, C.; Thi, T.M.; Ionescu, C. PID based Particle Swarm Optimization in Offices Light Control. *IFAC PapersOnLine* **2018**, *51*, 382–387. [\[CrossRef\]](#)
66. Chen, S.Y.; Wu, C.H.; Hung, Y.H.; Chung, C.T. Optimal Strategies of Energy Management Integrated with Transmission Control for a Hybrid Electric Vehicle using Dynamic Particle Swarm Optimization. *Energy* **2018**, *160*, 154–170. [\[CrossRef\]](#)
67. Chen, K.; Zhou, F.; Yin, L.; Wang, S.; Wang, Y.; Wan, F. A Hybrid Particle Swarm Optimizer with Sine Cosine Acceleration Coefficients. *Inf. Sci.* **2017**, *422*, 218–241. [\[CrossRef\]](#)

68. Zou, D.; Gao, L.; Wu, J.; Li, S. Novel global harmony search algorithm for unconstrained problems. *Neurocomputing* **2010**, *73*, 3308–3318. [[CrossRef](#)]
69. Hu, W.; Li, Z.S. A Simpler and More Effective Particle Swarm Optimization Algorithm. *J. Softw.* **2007**, *18*, 861–868. [[CrossRef](#)]
70. Zhang, X.; Zou, D.X.; Kong, Z.; Shen, X. A Hybrid Gravitational Search Algorithm for Unconstrained Problems. In Proceedings of the 30th Chinese Control and Decision Conference, Shenyang, China, 9–11 June 2018; pp. 5277–5284. [[CrossRef](#)]
71. Zhang, X.; Zou, D.X.; Shen, X. A Simplified and Efficient Gravitational Search Algorithm for Unconstrained Optimization Problems. In Proceedings of the 2017 International Conference on Vision, Image and Signal Processing, Osaka, Japan, 22–24 September 2017; pp. 11–17. [[CrossRef](#)]
72. Müller, P. Analytische Zahlentheorie. In *Funktionentheorie 1*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 386–456. [[CrossRef](#)]



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).