Swarm and Evolutionary Computation 51 (2019) 100573

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Chaotic particle swarm optimization with sigmoid-based acceleration coefficients for numerical function optimization

Dongping Tian [a,*], Xiaofei Zhao [b], Zhongzhi Shi [b]

[a] Institute of Computer Software, Baoji University of Arts and Sciences, Baoji, Shaanxi, 721007, PR China
[b] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, PR China

ARTICLE INFO

Keywords:
Particle swarm optimization
Acceleration coefficients
Logistic map
Swarm diversity
Inertial weight
Premature convergence

ABSTRACT

Particle swarm optimization (PSO) is a stochastic computation technique motivated by intelligent collective behavior of some animals, which has been widely used to address many hard optimization problems. However, like other evolutionary algorithms, PSO also suffers from premature convergence and entrapment into local optima when dealing with complex multimodal problems. In this paper, we propose a chaotic particle swarm optimization with sigmoid-based acceleration coefficients (abbreviated as CPSOS). On the one hand, the frequently used logistic map is applied to generate well-distributed initial particles. On the other hand, the sigmoid-based acceleration coefficients are formulated to balance the global search ability in the early stage and the global convergence in the latter stage. In particular, two sets of slowly varying function and regular varying function embedded update mechanism in conjunction with the chaos based re-initialization and Gaussian mutation strategies are employed at different evolution stages to update the particles during the whole search process, which can effectively keep the diversity of the swarm and get out of possible local optima to continue exploring the potential search regions of the solution space. To validate the performance of CPSOS, a series of experiments are conducted and the simulation results reveal that the proposed method can achieve better performance compared to several state-of-the-art PSO variants in terms of solution accuracy and effectiveness.

1. Introduction

Particle swarm optimization (PSO) is an efficient population-based stochastic search technique that is based on the metaphors of social interaction and communication (e.g., bird flocking and fish schooling) [19]. Due to the advantages of swarm intelligence, intrinsic parallelism, few parameters, easy implementation and inexpensive computation, PSO has attracted much research attention in the field of evolutionary computation and gained wide applications in many areas in the last two decades [1,7,9,10,16,18,42,43,49]. However, like other nature-inspired evolutionary algorithms [36], PSO also encounters the issues of premature convergence and entrapment into local optimum. For this purpose, researchers have been working hard to develop different PSOs to circumvent these two disadvantages since its advent in 1995 [9,27–29,35,41,42,44,47]. It is known that the performance of PSO is highly related to the parameters (inertia weight, acceleration coefficients and random numbers), especially when attempts are made to avoid premature convergence and to jump out of local optimal solutions. Consequently, a huge amount of inertia weights are formulated to balance the exploration and exploitation and thus result in a better optimal solution [31,42]. Meanwhile, different kinds of acceleration coefficients are developed to affect the movement of particles in the search space [1,6,7,19,34,43,48]. From the literature, it can be clearly observed that the appropriate acceleration coefficients have the capacity to enhance the global search in the early stages of the optimization and to encourage the particles to converge towards the global optima at the end of the search. Besides, the two random numbers in PSO system also have a significant influence on the convergence of a swarm [4,33]. In order to achieve a better trade-off between exploration and exploitation abilities, this paper presents a chaotic particle swarm optimization with sigmoid-based acceleration coefficients (CPSOS). On the one side, the frequently used chaos (logistic map) is applied to generate well-distributed initial particles. On the other side, the sigmoid-based acceleration coefficients are formulated to balance the global search ability in the early stage and the global convergence in the latter stage. Particularly, two sets of slowly varying function and regular varying function embedded update mechanism together with the chaos based re-initialization and Gaussian mutation strategies are exploited at different evolution stages to update the

* Corresponding author.
E-mail addresses: tdp211@163.com, tiandp@ics.ict.ac.cn (D. Tian).

https://doi.org/10.1016/j.swevo.2019.100573
Received 3 March 2019; Received in revised form 3 September 2019; Accepted 4 September 2019
Available online 9 September 2019

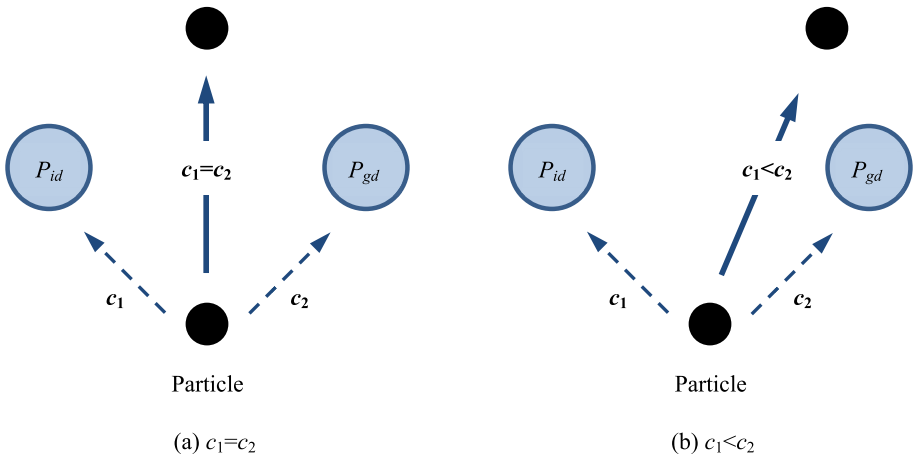


Fig. 1. The effect of acceleration coefficients for particles movement.

particles during the whole search process, which are mostly devoted to keeping the diversity of the swarm and break away from the local optima. Extensive experiments demonstrate that the proposed CPSOS yields better performance in terms of the solution accuracy and effectiveness.

The rest of the paper is structured as follows. Section 2 discusses the related work, especially the acceleration coefficients. In Section 3, the standard PSO is introduced. Section 4 elaborates the proposed CPSOS from three aspects of swarm initialization, sigmoid-based acceleration coefficients and update mechanism, respectively. In Section 5, extensive experiments on two sets of well-known benchmark functions are reported and analyzed. Finally, the concluding remarks and future work are given in Section 6.

2. Related work

Since the introduction of particle swarm optimization in the mid-1990s, a large number of PSO variants have been proposed from different aspects to enhance the performance. For example, the parameter adjustment involving the inertia weight [31,42], acceleration coefficients [6,7] and random numbers [4,33] has a significant influence on the convergence of a swarm; the topology structure of particles mainly involving the circle topology, wheel topology, star topology and random topology [20] aims at enriching the population diversity and undoubtedly influences the flow rate of the best information among particles; the hybridization mechanism involving hybrid PSO with different evolutionary algorithms [9,38], some local search techniques [17,46] and several other auxiliary search strategies [41,44] aims to maintain the balance between exploration (global search) and exploitation (local search) as well as prevent the stagnation and premature convergence of swarm. All of the strategies mentioned above, to some extent, can effectively boost the performance of PSO (e.g., convergence rate and solution quality). More details and a more complete explanation on them can be found in the corresponding literatures. Different from the previous studies of the related work, here, we only focus on a major aspect of PSO algorithm (i.e. acceleration coefficients) and different acceleration coefficients will be reviewed and discussed in the following subsections.

2.1. Equal versus unequal acceleration coefficients

In literature [19], the acceleration coefficients is introduced into the update scheme of PSO. There is no doubt that acceleration coefficients used with the cognitive and the social components will significantly affect the movement of particles. Furthermore, they are very important to the success of PSO in both the social-only model and the cognitive-only model. A large value of the acceleration factor c_1 makes a particle to fly towards its own ever known best position (*pbest*) faster and a large value of the acceleration factor c_2 makes the particle to move towards the best particle (*gbest*) of the whole swarm faster. In general, c_1 and c_2 are determined identical ($c_1 = c_2 = 2$) to give equal weightage to both the cognitive component and the social component simultaneously. It is reported that acceleration coefficients with big values tends to cause the particles to separate and move away from each other, while taking small values causes limitation of the movements of the particles, and not being able to scan the search space adequately [32]. In Ref. [8], the acceleration coefficients is set equal ($c_1 = c_2 = 2.05$) to control the magnitudes of the particle's velocity and guarantee the convergence. Instead, $c_1 = c_2 = 1.49445$ is selected as the constant acceleration coefficients for PSO algorithm in literature [40]. Besides, it should be noted that the unbalanced setting of c_1 and c_2 ($c_1 = 3, c_2 = 1$) is used in Ref. [4] for the worst fit particles to draw the direction of the newly updated particle towards its own *pbest*. That is, the cognitive component is given more weightage in this situation. In other words, the unequal weightage is only applied in the case of stagnation of the particle that helps the newly generated particle to avoid drifting towards the *gbest* and explore a new region in the search space. Hence, it is able to avoid stagnation of the swarm at a suboptimal solution. Besides, the unbalanced settings of acceleration coefficients ($c_1 = 0.5, c_2 = 2.5$) [43] and ($c_1 = 2.8, c_2 = 1.3$) [5] are exploited to accelerate the convergence of PSO with better results for their chosen set of problems. The underlying reason is that if c_1 and c_2 are assigned the same value, the next position of a particle is usually the middle of *pbest* and *gbest*. In this case, it may take more iterations for the particle to move closer to the global best location and thus affect the efficiency of the search (as shown in Fig. 1(a)). In order to accelerate the convergence, a larger value should be assigned to c_2 than c_1. As a result, the new position of a particle will be closer to the global best location *gbest* (as displayed in Fig. 1(b)). More details of them can be gleaned from the corresponding literature.

2.2. Time-varying acceleration coefficients

Apart from the acceleration coefficients with constant values, several strategies have been advanced to dynamically update the acceleration coefficients c_1 and c_2 from iteration to iteration [6,7,16,34,39,50]. In literature [34], a time-varying acceleration coefficients (abbreviated as TVAC) is formulated to enhance the global search in the early stages of the optimization and to encourage the particles to converge towards the global optima at the end of the search, which can be achieved by changing the acceleration coefficients c_1 and c_2 with time in such a manner that the cognitive component is reduced while the social component is increased as the search proceeds. With a large cognitive component and small social component at the beginning, particles are allowed to move around the search space instead of moving toward the

population best during early stages. On the other hand, a small cognitive component and a large social component allow the particles to converge to the global optima in the latter stage of the optimization process. TVAC can be mathematically represented as:

$$c_1 = (c_{1i} - c_{1f}) \times \left(\frac{iter_{\max} - iter}{iter_{\max}}\right) + c_{1f} \tag{1}$$

$$c_2 = (c_{2i} - c_{2f}) \times \left(\frac{iter_{\max} - iter}{iter_{\max}}\right) + c_{2f} \tag{2}$$

where c_{1i} and c_{2i} are the initial values and c_{1f} and c_{2f} are the final values of the acceleration factors c_1 and c_2 respectively. *iter* denotes the current iteration number, $iter_{max}$ is the maximum number of allowable iterations. Alternatively, it should be noted that TVAC defined by Eqs. (1) and (2) can be expressed by another equivalent form as follows:

$$c_1 = (c_{1f} - c_{1i}) \times \frac{iter}{iter_{\max}} + c_{1i} \tag{3}$$

$$c_2 = (c_{2f} - c_{2i}) \times \frac{iter}{iter_{\max}} + c_{2i} \tag{4}$$

Simulations are carried out with various optimal control problems to find out the best ranges of values for c_1 and c_2. From the results it has been observed that best solutions can be determined when changing c_1 from 2.5 to 0.5 and changing c_2 from 0.5 to 2.5, over the full range of search.

2.3. Extended time-varying acceleration coefficients

In literature [16], the evolutionary particle swarm optimization (EPSO) is developed based on the standard PSO with the following velocity update equation. It should be noted that to introduce the time-varying acceleration coefficients into the EPSO, the parameters w_{i0}^*, w_{i1}^*, w_{i2}^* and w_{i3}^* in Eq. (5) are defined as follows:

$$V_i^{k+1} = w_{i0}^* V_i^k + w_{i1}^* \times (pbest_i^k - X_i^k) + w_{i2}^* \times (gbest_i^* - X_i^k) + w_{i3}^* \times (rbest_i^* - X_i^k) \tag{5}$$

$$w_{i0}^* = w + \tau N \tag{6}$$

$$w_{ij}^* = c_j + \tau N, j = 1, 2, 3. \tag{7}$$

The purpose of introducing *rbest* in Eq. (5) is to improve the robustness of EPSO, which is randomly selected from *pbest* of other particles. τ is the learning dispersion factor, N is a random number between 0 and 1, c_1 and c_2 are the cognitive coefficient and social coefficient defined as the standard time-varying acceleration coefficients, c_3 is the TVAC for *rbest* component and calculated as:

$$c_3 = c_1 \times [1 - \exp(-c_2 k)] \tag{8}$$

From c_1 and c_2, it can be seen that the TVAC is able to improve the exploration and exploitation of each particle when c_1 decreases but c_2 increases with the number of iteration and c_1 is not equal to c_2 in each iteration. This is due to the particles will be pulled towards the global solution. When c_1 is large but c_2 is small in the initial iteration, the particles will be pushed to move around the entire solution space. From Eq. (8), c_3 pulls the particle towards *rbest* which can further improve the exploration of the particle towards a better optimum solution.

2.4. Sine cosine acceleration coefficients

In literature [6], the sine cosine acceleration coefficients (termed as SCAC) is formulated as a new parameter adjustment strategy for the cognitive component and social component of PSO. Compared with the

Table 1
Summary of various acceleration coefficients reported in the literature.

Sources	Different acceleration coefficients	Adaptation mechanism	Related parameters
[4]	$c_1 = 3$, $c_2 = 1$	Constant	–
[5]	$c_1 = 2.8$, $c_2 = 1.3$	Constant	–
[8]	$c_1 = c_2 = 2.05$	Constant	–
[19]	$c_1 = c_2 = 2$	Constant	–
[40]	$c_1 = c_2 = 1.49445$	Constant	–
[43]	$c_1 = 0.5$, $c_2 = 2.5$	Constant	–
[45]	$c_1 = c_2 = 1.49618$	Constant	–
[1]	$c_1 = c_1 \times (1 - \lambda)$ $c_2 = c_2 \times (1 + \lambda)$	Adaptive	λ
[3]	$c = 1 + \frac{gbest_i}{pbest_i}$	Adaptive	$pbest_i$, $gbest_i$
[6]	$c_1 = \partial \times \sin\left(\left(1 - \frac{iter}{iter_{\max}}\right) \times \frac{\pi}{2}\right) + \delta$ $c_2 = \partial \times \cos\left(\left(1 - \frac{iter}{iter_{\max}}\right) \times \frac{\pi}{2}\right) + \delta$	Adaptive	$\partial = 2, \delta = 0.5$, *iter*, $iter_{max}$
[7]	$c_1 = -(c_{1f} - c_{1i}) \times \left(\frac{iter}{iter_{\max}}\right)^2 + c_{1f}$ $c_2 = c_{1i} \times \left(1 - \frac{iter}{iter_{\max}}\right)^2 + c_{1f} \times \frac{iter}{iter_{\max}}$	Adaptive	*iter*, $iter_{max}$
[18]	$c_1 = c_1 - \left(\frac{0.5}{iter_{\max}}\right)$ $c_2 = c_2 + \left(\frac{0.5}{iter_{\max}}\right)$	Adaptive	$iter_{max}$
[21]	$c_1 = 0.5 + 0.5 \times \left(\exp\left(-\frac{iter}{500}\right)\right) + 1.4 \times \frac{\sin(iter)}{30}$ $c_2 = 1 + 1.4 \times \left(1 - \exp\left(-\frac{iter}{500}\right)\right) + 1.4 \times \frac{\sin(iter)}{30}$	Adaptive	*iter*
[34]	$c_1 = (c_{1i} - c_{1f}) \times \left(\frac{iter_{\max} - iter}{iter_{\max}}\right) + c_{1f}$ $c_2 = (c_{2i} - c_{2f}) \times \left(\frac{iter_{\max} - iter}{iter_{\max}}\right) + c_{2f}$	Adaptive	*iter*, $iter_{max}$
[48]	$c_{1i}^{k+1} = c_{1i}^k + \alpha_i^k (cbest_1^k - c_{1i}^k)$ $c_{2i}^{k+1} = c_{2i}^k + \alpha_i^k (cbest_2^k - c_{2i}^k)$	Adaptive	α_i^k

TVAC, SCAC can better balance the global search ability in the early stage and the global convergence in the latter stage.

$$c_1 = \partial \times \sin\left(\left(1 - \frac{iter}{iter_{\max}}\right) \times \frac{\pi}{2}\right) + \delta \tag{9}$$

$$c_2 = \partial \times \cos\left(\left(1 - \frac{iter}{iter_{\max}}\right) \times \frac{\pi}{2}\right) + \delta \tag{10}$$

where ∂ and δ are constants with values 2 and 0.5 respectively. *iter* denotes the current iteration number, $iter_{max}$ is the maximum number of allowable iterations. In SCAC, the range of c_1 is between 2.5 and 0.5 and the range of c_2 is between 0.5 and 2.5, which are completely consistent with the conclusions in Ref. [34].

2.5. Nonlinear dynamic acceleration coefficients

In literature [7], the nonlinear dynamic acceleration coefficients (NDAC) is added in the PSO method as a parameter update mechanism that has powerful capability of tuning cognitive component c_1 and social component c_2. NDAC can be expressed as follows:

$$c_1 = -\left(c_{1f} - c_{1i}\right) \times \left(\frac{iter}{iter_{\max}}\right)^2 + c_{1f} \tag{11}$$

$$c_2 = c_{1i} \times \left(1 - \frac{iter}{iter_{\max}}\right)^2 + c_{1f} \times \frac{iter}{iter_{\max}} \tag{12}$$

where c_{1i} and c_{1f} are positive constants ($c_{1i} = 0.5$, $c_{1f} = 2.5$), *iter* and $iter_{max}$ denote the current iteration and the maximum iteration, respectively. According to Eqs.(11)-(12), the cognitive component c_1 and the social component c_2 are changing from 2.5 to 0.5 and from 0.5 to 2.5 with the growth of iteration respectively. This coincides well with the empirical results reported in literature [34].

2.6. Other versions of acceleration coefficients

In the early work [39], a linear decreasing of the acceleration coefficients with time has been first proposed, even if better results can be obtained for different problems when c_1 and c_2 are viewed as constants, but not necessarily set at 2.0. In literature [50], the value of the acceleration coefficients c_1 and c_2 are updated with time based on four different evolutionary states: exploration, exploitation, convergence and jumping-out. To be specific, increasing c_1 and decreasing c_2 in an exploration state, increasing c_1 slightly and decreasing c_2 slightly in an exploitation state, increasing both c_1 and c_2 slightly in a converging state, decreasing c_1 and increasing c_2 in an jumping-out state. The adjustment of the acceleration coefficients according to the evolutionary state are carried out gradually by limiting their maximum increment or decrement between two successive iterations to the interval [0.05,0.1]. The sum of c_1 and c_2 is also restricted to the interval [3.0,4.0]. Finally a jumping-out mechanism based on an elitist learning strategy is proposed to enhance the ability of the swarm to escape from local minima. Such a strategy is applied to the best particle when the swarm is classified in the evolutionary state of convergence. The recent work [21] utilizes the time-varying nature of the acceleration coefficients c_1 and c_2 to categorically increase and decrease the effects of the *pbest* and *lbest* (local best) in different stages of the search process. It is worth noting that all the adaptive strategies mentioned above update the value of acceleration coefficients c_1 and c_2 from iteration to iteration, but at any iteration all the particles share the same values of c_1 and c_2. In contrast, a different mechanism is formulated in Ref. [48] to update the acceleration coefficients with time. Note that in Table 1, where k is the current iteration, $cbest_1$ and $cbest_2$ are the acceleration coefficients c_1 and c_2 of the particle that is able to update *gbest* in a previous iteration, α_i denotes the step size, which can control the diversity of each particle, is selected from two values: 0 or $2/T_{max}$, to remain some diversity at the end of a search. A simple mechanism that the step size α_i takes 0 or the inverse of total iteration T_{max} is adopted as the coordinating of each particle. Unlike previous updating approaches, particles may differ from one another owing to different values of the acceleration coefficients c_1 and c_2. In Ref. [3], the acceleration coefficients is neither set to a constant nor set as a linearly decreasing time varying variable. Instead, it is defined as a function of *pbest* and *gbest* (shown in Table 1) by setting c_1 and c_2 be equivalent, and the velocity update equation is modified as below:

$$v_i(t+1) = w \times v_i(t) + c \times r_1(pbest_i + gbest - 2x_i(t)) \tag{13}$$

Considering the effect of acceleration coefficients on explorative and exploitative capabilities for PSO, the personal acceleration coefficients is proposed to decrease (from 2 to 1.5) during the course of run while the social acceleration coefficients is increased (from 1.5 to 2) [18]. In this way, an appropriate tradeoff between the exploration and exploitation can be established and the premature convergence is significantly mitigated. Besides, the numerical coefficients of the cognitive and social component of the stochastic acceleration are related to the distance of each particle from the best position found so far by the swarm, with different distributions over the swarm [2]. Note that in more recent work [15], it is believed that the balance between social and cognitive coefficients does have a significant effect on the regions of parameter space that leads to optimal performance. At the same time, the optimal parameters are time-dependent and a general recommendation for selecting the acceleration coefficients values for the PSO control parameters is to set $c_1 \in [0.5,1.0]$ and $c_2 \in [2.5,3.0]$.

3. Standard PSO

In the standard PSO (SPSO) algorithm, each particle searches for an optimal solution to the objective function in the search space. In a D-dimension hyperspace, two vectors are related to the ith particle ($i = 1,2, \ldots,N$). One is the velocity vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, and the other is the position vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where N denotes the swarm size. During the search process, each particle dynamically updates its position based on its previous position and new information regarding velocity. Its best location found in the search space so far is called *pbest* and the best location found for all the particles in the swarm is called *gbest*. After the two best values are identified, the particle updates its velocity and position according to the following equations:

$$v_{id}^{t+1} = w \times v_{id}^{t} + c_1 \times rand_1 \times \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 \times rand_2 \times \left(g_{id}^{t} - x_{id}^{t}\right) \tag{14}$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{15}$$

where w is the inertia weight, v_{id}^{t} is the velocity of the ith particle at dth dimension in tth iteration, $x_{id}^{t}$is the current position of the ith particle at dth dimension, p_{id}^{t} and g_{id}^{t} denote *pbest* and *gbest* respectively, $rand_1$ and $rand_2$ are two random numbers uniformly distributed in the range [0,1], c_1 and c_2 are two acceleration coefficients that determine the relative learning weights for *pbest* and *gbest* respectively. In addition, the velocity is clamped to a range $[-v_{max}, v_{max}]$ to reduce the likelihood that the particle might leave the search space and $x_{id} \in [l_d, u_d]$, where l_d and u_d are the lower and upper bounds of the corresponding variables.

So far, the complete computational procedure of the SPSO can be briefly described as follows [37].

Step 1: Initialization.

Initialize the parameters and the swarm with random positions and velocities.

Step 2: Evaluation.

Evaluate the fitness value (the desired objective function) for each particle.

Step 3: Find the *pbest*.

If the fitness value of particle i is better than its best fitness value

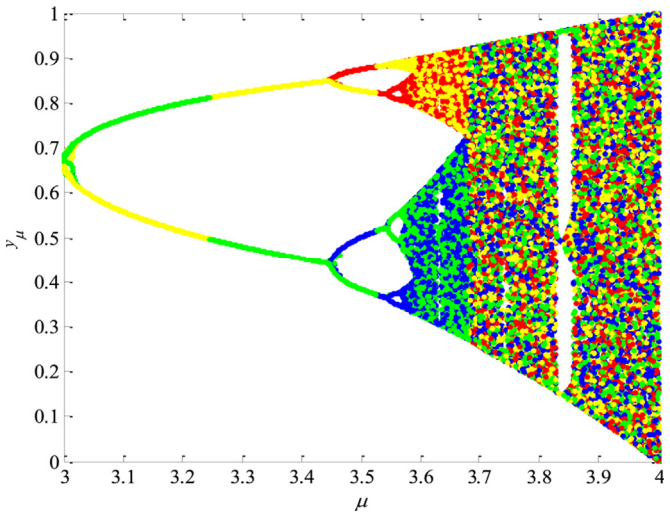


Fig. 2. Bifurcation diagram of logistic map.

(*pbest*) in history, then set the current fitness value as the new *pbest* of particle *i*.

Step 4: Find the *gbest*.

If any updated *pbest* is better than the current *gbest*, then set *pbest* as the current value of the whole swarm.

Step 5: Update the velocity and position.

Update the velocity and move to the next position according to Eqs.(14)-(15).

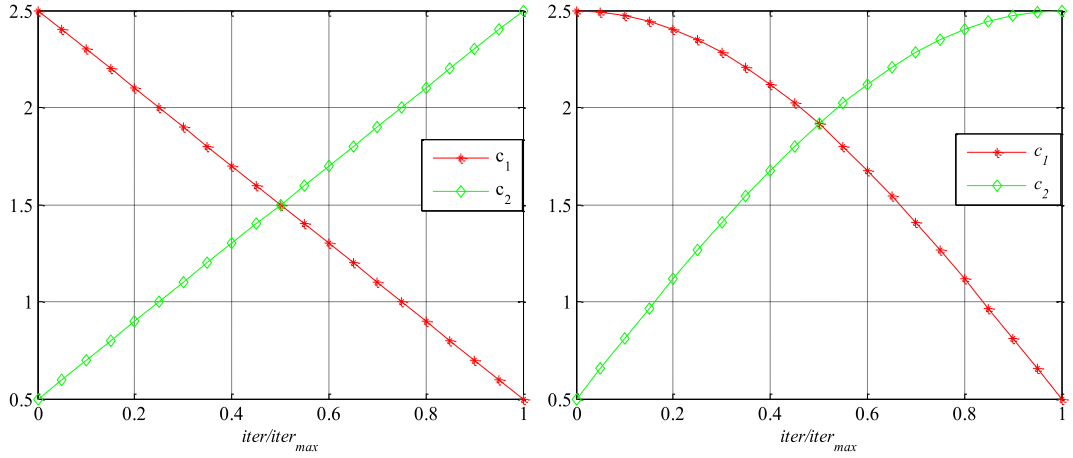
Step 6: Termination.

If the termination condition is reached, then stop; otherwise turn to Step 2.

4. Proposed CPSOS

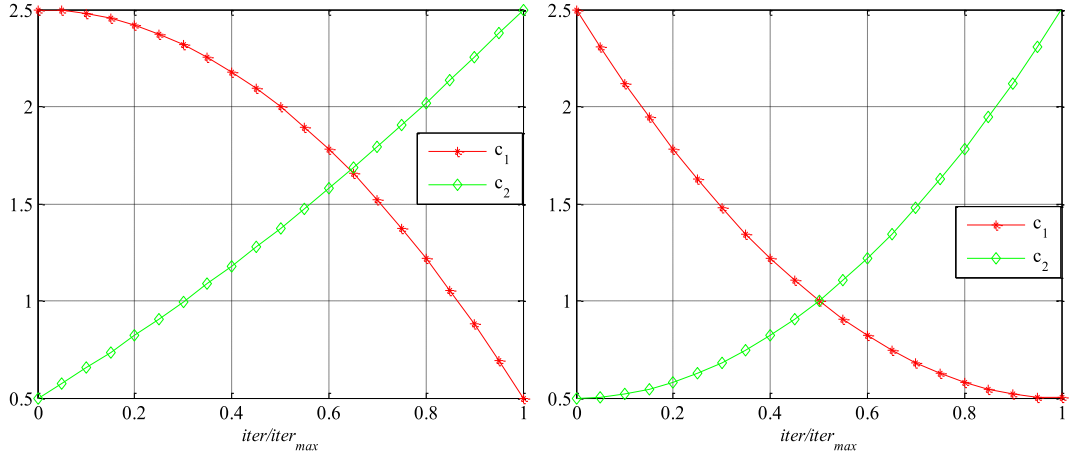
4.1. Swarm initialization

Swarm initialization is the first and a crucial step in any evolutionary algorithms since it affects the convergence speed and quality of the final solution. In general, random initialization is the most commonly used method to generate initial swarm in absence of any information about the solution. From the literature, it can be seen that chaotic sequence instead of random sequence based initialization is a powerful strategy to diversify the particles of swarm and improve the performance of PSO by preventing the premature convergence [10,12,41,42]. Meanwhile, the stability of PSOs and the quality of final solution can also be improved to some extent [41,42]. Based on this recognition, the commonly used chaos logistic map is first employed to generate the initial position



(a) Time-varying acceleration coefficients (TVAC)

(b) Sine cosine acceleration coefficients (SCAC)



(c) Nonlinear dynamic acceleration coefficients (NDAC)

(d) Sigmoid-based acceleration coefficients (SBAC)

Fig. 3. Different kinds of acceleration coefficients (TVAC, SCAC, NDAC and SBAC).

instead of the uniform position, which can be described as follows:

$$y_{i+1} = \mu \times y_i \times (1 - y_i), i = 0, 1, 2, \cdots \quad (16)$$

where y_i is distributed in the interval (0,1), such that the initial $y_0 \in (0,1)$ and $y_0 \notin (0, 0.25, 0.5, 0.75, 1)$. μ is a predetermined constant called bifurcation coefficient. When μ increases from zero, the dynamic system generated by Eq. (16) changes from one fixed-point to two, three, …, and until 2^i. During this process, a large number of multiple periodic components will locate in narrower and narrower intervals of μ as it increases. This phenomenon is obviously free from constraint. But μ has a limit value $\mu_t = 3.569945672$. Note that when μ approaches the μ_t, the period will become infinite or even non-periodic. At this time, the whole system evolves into the chaotic state (behavior). On the other hand, when μ is greater than 4, the whole system becomes unstable. Hence the range $[\mu_t, 4]$ is considered as the chaotic region of the whole system. Its bifurcation diagram is illustrated in Fig. 2.

Thus, the detailed process of the logistic map based initialization can be described as follows. To begin with, the logistic map is utilized to generate the chaotic variable. For the sake of representation, Eq. (16) is rewritten as follows:

$$y_j^{i+1} = \mu \times y_j^i \times \left(1 - y_j^i\right), j = 1, 2, \cdots, D \quad (17)$$

where y_j denotes the jth chaotic variable, and i is the iteration number. Firstly set $i = 0$ and generate D chaotic variables by Eq. (17). Followed by let $i = 1,2, \ldots, N$ in turn and generate the initial chaotic variables. Finally map these yielded variables back to the search space of the problem as below:

$$x_{ij} = x_{\min,j} + y_j^i \times \left(x_{\max,j} - x_{\min,j}\right), j = 1, 2, \cdots, D \quad (18)$$

As a result,

$$x_{ij} = (x_{i1}, x_{i2}, \cdots, x_{iD}), i = 1, 2, \ldots N \quad (19)$$

That is, the logistic map based swarm initialization can be achieved.

As described above, the pseudocode of the logistic map based swarm initialization can be succinctly described as below.

Algorithm 1: Pseudocode of chaos-based swarm initialization

Input: N: swarm size, D: swarm dimension, μ: bifurcation coefficient.
Process:
1. Generate D initial chaotic variables.
2. **while** *number of maximal iterations is not met* **do**
3. | **if** *chaotic variable plunges into the fixed points or the small periodic cycles*
4. | | Implement a very small positive random perturbation.
5. | | Map them by Eq.(17).
6. | **else**
7. | | Update the chaotic variables by Eq.(17) directly.
8. | **end if**
9. **end while**
10. Remap the chaotic variables into the problem space by Eq.(18).

Output: the generated N particles as the initial swarm.

4.2. Sigmoid-based acceleration coefficients

It is generally believed that the performance of PSO heavily depends on the balance between exploration and exploitation. To this end, an appropriate ratio between them should be well established. What's more, a common belief is that PSO should start with exploration and then gradually change into exploitation [47]. Based on this recognition, many time-varying strategies have been proposed to regulate the parameters of PSO. As discussed in Section 2, different acceleration coefficients (c_1 and c_2) pay different attention to the local search and the global search. Motivated by the success of these paradigms, we present a new sigmoid-based acceleration coefficients (SBAC) in this work as follows. The key idea behind this strategy is that the formulated acceleration coefficients has the ability to transform from the linear to nonlinear states smoothly. To be specific, the value of c_1 changes from 2.5 while c_2 starts from 0.5 in the early stages of search through a nonlinear way defined by Eqs. (20) and (21). This allows particles to diversify the search space. Subsequently, the value of c_1 continues to decrease to 0.5 whereas c_2 increase to 2.5 in the latter stages of the search process, which allows the particles to reinforce the search towards the global optima (as illustrated in Fig. 3). This is completely consistent with the empirical results

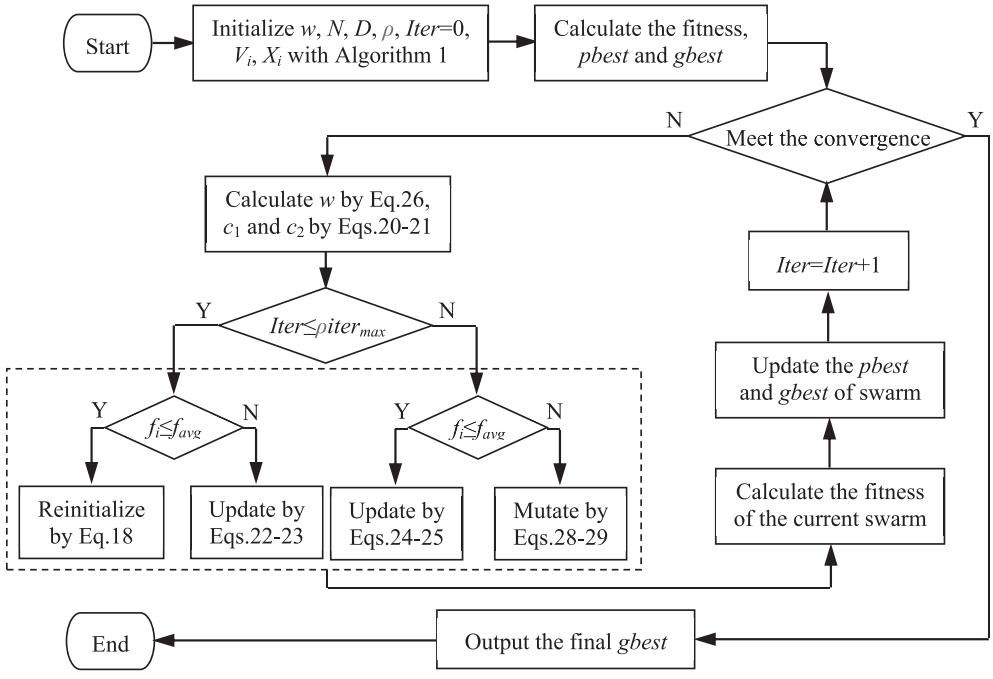


Fig. 4. Flowchart of the CPSOS algorithm.

reported in Ref. [34].

$$c_1(iter) = \frac{1}{1+\exp(-\lambda iter/iter_{\max})} + 2(c_{1f}-c_{1i})\left(\frac{iter}{iter_{\max}}-1\right)^2 \quad (20)$$

$$c_2(iter) = \frac{1}{1+\exp(-\lambda iter/iter_{\max})} + (c_{1f}-c_{1i})\left(\frac{iter}{iter_{\max}}\right)^2 \quad (21)$$

where λ is a control parameter used to adjust the value of the sigmoid-based acceleration coefficients ($\lambda = 0.0001$), c_{1i} and c_{1f} are two positive constants ($c_{1i} = 0.5$, $c_{1f} = 2.5$), *iter* and $iter_{max}$ denote the current iteration and the maximum number of allowable iterations, respectively. Note that c_1 decreases from 2.5 to 0.5 while c_2 increases from 0.5 to 2.5 when $\lambda = 0.0001$, $c_{1i} = 0.5$ and $c_{1f} = 2.5$, which coincides well with the conclusions reported in the literatures [6,7,34] because the performance of PSO can be significantly improved at the time of the acceleration coefficients belonging to these corresponding change intervals.

4.3. Update mechanism

During the search process of PSO algorithm, it is generally believed that the exploration should be enhanced in the early stage so as to intensify the global search in a wider range of the solution space while the exploitation should be strengthened in the later stage in order to improve the local search ability of PSO. Note that two different kinds of functions are defined as follows [22,26]:

- **Slowly varying function (SVF)**: For anyone given function $l(x)$, $\forall\delta > 0$ and $a>0$, there have $l(x) > 0$ in $[a,+\infty)$, if $x^{\delta}l(x)$ is monotone increasing and $x^{-\delta}l(x)$ is monotone decreasing, then $l(x)$ is termed as the slowly varying function.
- **Regular varying function (RVF)**: For anyone given slowly varying function $l(x)$, $\forall\delta > 0$ and $a>0$ in $[a,+\infty)$, $x^{\delta}l(x)$ is termed as regular varying function.

From the definitions and their related properties of the two functions described above [11], it can be clearly observed that SVF is helpful for particles to traverse the solution space as well as to speed up the convergence rate. At the same time, RVF is beneficial to enhance the local search ability with perturbation, which is conducive to improving the accuracy of solution. In view of this, the regular varying function is leveraged to improve the global search ability of PSO with fast convergence in the early stage of the search process.

$$v_{id}^{t+1} = w \times v_{id}^{t} + c_1 \times rand_1 \times \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 \times rand_2 \times \left(g_{id}^{t} - x_{id}^{t}\right) \quad (22)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} + x^{\delta}l(x) \quad (23)$$

where δ denotes the control factor.

In the meanwhile, the slowly varying function is introduced into the update formula of CPSOS algorithm to avoid falling into local optima and to promote the local search ability in the later stage of the search process.

$$v_{id}^{t+1} = w \times v_{id}^{t} + c_1 \times rand_1 \times \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 \times rand_2 \times \left(g_{id}^{t} - x_{id}^{t}\right) \quad (24)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} + l(x) \quad (25)$$

Without loss of generality, the inertia weight w appeared in Eqs. (22) and (24) can be defined as below:

$$w(iter) = (w_{\max} - w_{\min}) \times \frac{iter_{\max} - iter}{iter_{\max}} + w_{\min} \quad (26)$$

where w_{max} and w_{min} denote the initial and final values of the inertia weight, respectively.

It is worth noting that the slowly varying function has proven to be convergence [11], and there is no increase (or decrease) the fastest SVF, nor swing (or shaking) the fastest SVF. Thus a slowly varying function with increasing speed and disturbance relatively faster is selected in our work, which is shown as below:

$$l(x) = (10Inx)^{\alpha} \quad (27)$$

where α is the control parameter.

In addition, it is well known that as an effective strategy to maintain the diversity of a swarm, mutation operation is a widely acceptable mechanism and has been widely applied in the field of bio-inspired computation [42], especially for PSO. Without loss of generality, here, Gaussian mutation is leveraged to mutate the particles when the fitness of particle is greater than the average of the whole swarm in the later stage of the search process because it is believed to be good at local search in the community of evolutionary computation [45].

$$pbest_{ij}^{'} = pbest_{ij} + gaussian_j() \quad (28)$$

$$gbest_{j}^{'} = gbest_{j} + gaussian_j() \quad (29)$$

where $gaussian_j$ () is a random number generated by Gaussian distribution.

So far, the complete procedure of the CPSOS algorithm proposed in this paper can be succinctly described as follows (the flowchart is illustrated in Fig. 4):

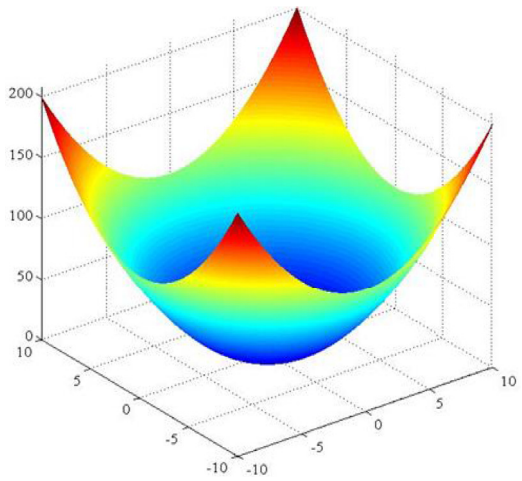
Algorithm 2: Pseudocode of the proposed CPSOS algorithm

Input: w_0: inertia weight, c_1, c_2: acceleration coefficients, N: swarm size, D: swarm dimension, ρ: control factor, f_{avg}: average fitness.

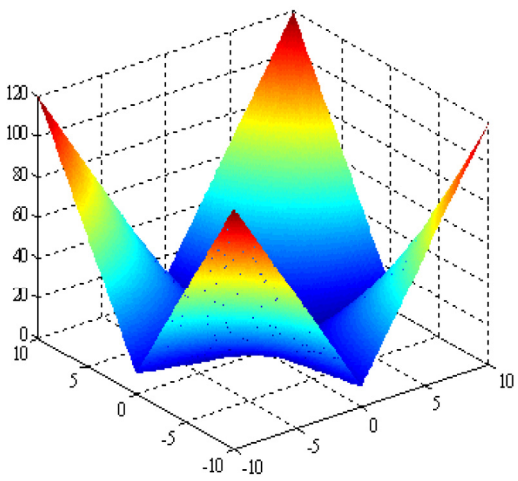
Process:

```
Initialize the particles of swarm by Algorithm 1.
while maximum number of iterations (iter≤iter_max) is not met do
    Calculate w by Eq.(26).
    Calculate c1 and c2 by Eqs.(20)-(21).
    for n=1 to N do
        if iter≤ρiter_max
            if f_i ≤ f_avg
                Reinitialize the same number of particles by Eq.(18).
            else
                Update velocity and position of particles by Eqs.(22)-(23).
            end if
        else
            if f_i ≤ f_avg
                Update velocity and position of particles by Eqs.(24)-(25).
            else
                Execute Gaussian mutation on the positions of particles.
            end if
        end if
        Calculate the fitness values of the new particle X_i.
        if X_i is better than pbest_i
            Set X_i to be pbest_i.
        end if
        if X_i is better than gbest
            Set X_i to be gbest.
        end if
    end for
end while
```

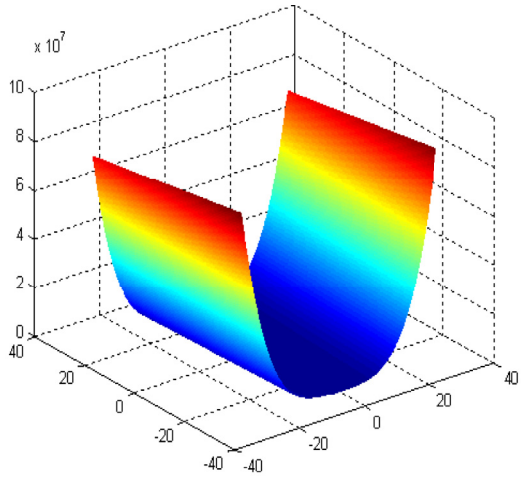
Output: *gbest* particle as the final optimal solution.



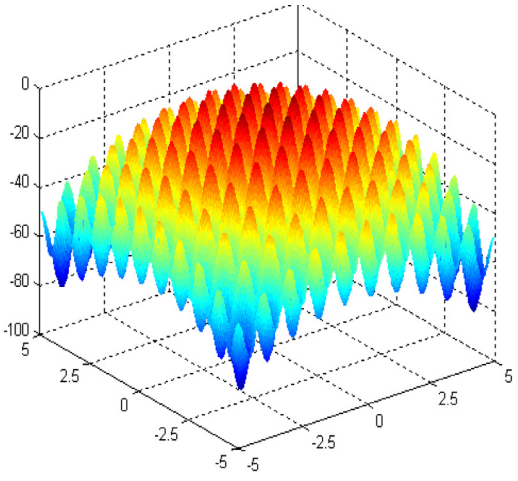
(a) Sphere function (f_1)



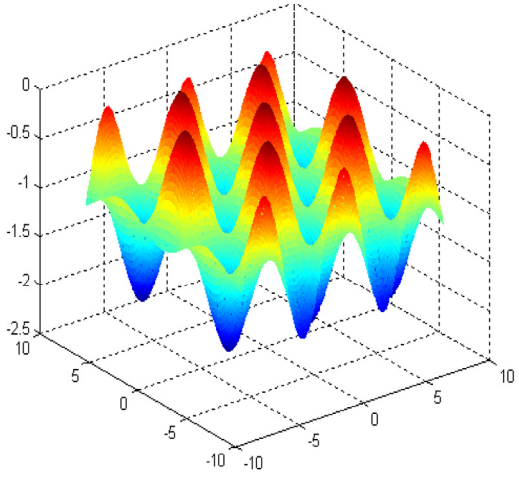
(b) Schwefel function (f_2)



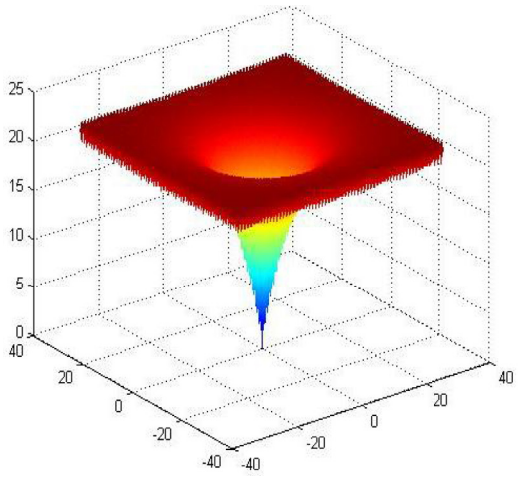
(c) Rosenbrock function (f_3)



(d) Rastrigin function (f_4)



(e) Griewank function (f_5)



(f) Ackley function (f_6)

Fig. 5. Graphical shows of the test functions.

5. Experimental results and analysis

5.1. Experiments based on basic benchmark functions

To ascertain the performance of the proposed CPSOS, extensive experiments are first conducted on a set of well-known benchmark functions consisting of six global optimization problems, which can be divided into two groups in terms of their properties.

Group A: Unimodal problems.

(1) Sphere function

$$\min f_1(x) = \sum_{i=1}^{n} x_i^2$$

where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-10 \le x_i \le 10$.

(2) Schwefel function

$$\min f_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$$

where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-100 \le x_i \le 100$.

(3) Rosenbrock function

$$\min f_3(x) = \sum_{i=1}^{n-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2 \right]$$

where the global optimum x^*=(1,1, …,1) and $f(x^*) = 0$ for $-30 \le x_i \le 30$.

Group B: Multimodal problems.

(4) Rastrigin function

$$\min f_4(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$$

where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-5.12 \le x_i \le 5.12$.

(5) Griewank function

$$\min f_5(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i)^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-600 \le x_i \le 600$.

(6) Ackley function

$$\min f_6\left(x\right) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-32 \le x_i \le 32$.

Fig. 5 depicts the 2-dimensional graphical shows of the six test functions.

Note that all the test functions are to be minimized, they are numbered f_1-f_6 listed in Table 2, including the expression, dimension, allowable search space, global optimum and property, respectively. Since the superiority of chaos-based swarm initialization has already been proved in our previous studies [41,42], that is, enhancing the stability of the optimized results and improving the convergence speed and quality of final solution for the problems to be solved, thus we won't go into much detail here. Instead, to illustrate the effect of different acceleration coefficients for PSO will be highlighted and investigated in this section. It should be noted that for the sake of readability and comparison, we use CPSO-SBAC and CPSOS interchangeably from now on in this work. From this point of view, the implementation of the other three methods CPSO-TVAC, CPSO-SCAC and CPSO-NDAC is just the same as CPSOS except for the use of different acceleration coefficients. Specifically, the experimental settings can be described as follows: the swarm size $N = 40$, the decaying inertia weight starts at 0.9 and ends at 0.4, viz. $w_{max} = 0.9$, $w_{min} = 0.4$, the control factor ρ is set as 0.5 by trial and error by considering the tradeoff between computational accuracy and convergence speed. For each test function, 30 independent runs are performed by each PSO, and each run is with 1000 iterations. All the PSOs terminate when reaching the maximum number of allowed iterations. Without loss of generality, the most commonly used performance metrics including best solution, average solution and standard deviation are employed to evaluate the performance of CPSOS.

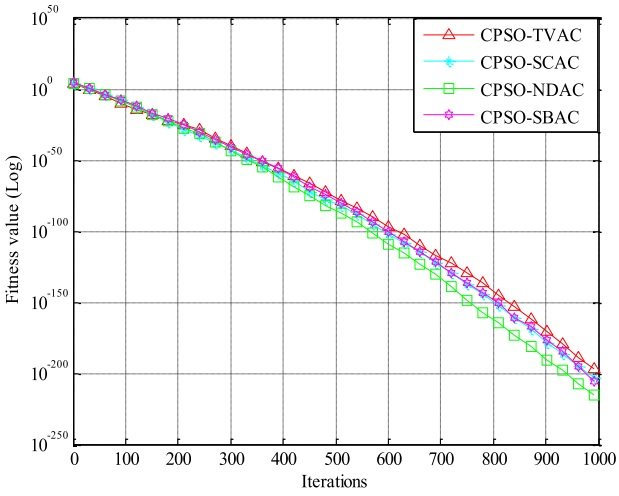
Table 2
Dimensions, search ranges, global optimum and properties of the test functions.

Test functions	Dimensions	Search range	Global optimum	Properties
Sphere (f_1)	10/30	$[-100,100]^D$	0	Unimodal
Schwefel (f_2)	10/30	$[-10,10]^D$	0	Unimodal
Rosenbrock (f_3)	10/30	$[-30,30]^D$	0	Unimodal
Rastrigin (f_4)	10/30	$[-5.12,5.12]^D$	0	Multimodal
Griewank (f_5)	10/30	$[-600,600]^D$	0	Multimodal
Ackley (f_6)	10/30	$[-32,32]^D$	0	Multimodal

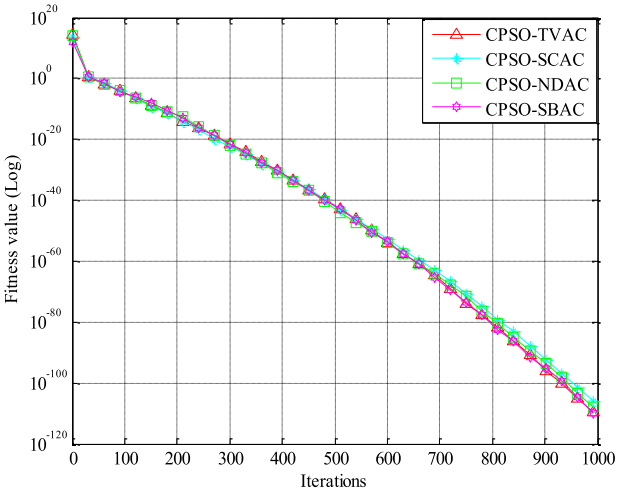
From the results shown in Table 3, it can be clearly observed that PSO with different acceleration coefficients can achieve promising results. To be specific, CPSO-NDAC can get the best solution on all the test functions except on f_1 as well as on f_3 and f_4 with the same optimization results. In particular, note that although CPSO-NDAC seems to give a slightly better best solution on f_5 and f_6, both the average solution and standard deviation of CPSO-SBAC are smaller than that of CPSO-NDAC, which further demonstrates the advantage of the chaos-based swarm initialization for PSO algorithm. Another interesting observation comes from Table 3 is that the optimized results on functions f_3 and f_4. Note that f_3 is also referred to as the valley or banana function. It is a popular test problem for gradient-based optimization algorithms (shown in Fig. 5(c)). This function is unimodal, and the global minimum lies in a narrow, parabolic

Table 3
Results of CPSO with different acceleration coefficients under $D = 10$.

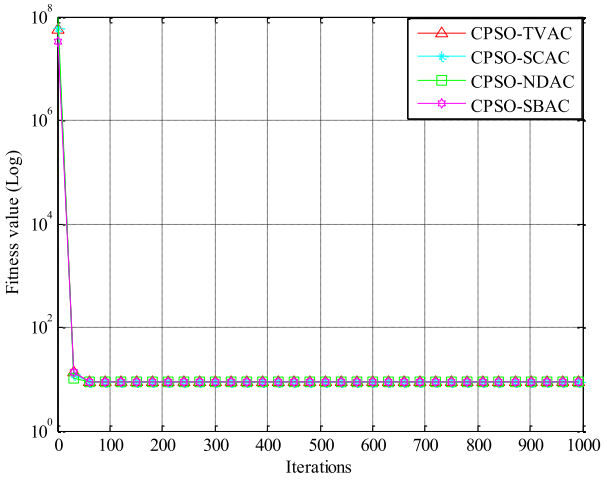
Functions	PSO algorithm	Best solution	Average solution	Standard deviation
f_1	CPSO-TVAC	4.9379e−216	1.2402e−206	0.0000e−000
	CPSO-SCAC	5.7789e−220	2.4951e−201	0.0000e−000
	CPSO-NDAC	1.2435e−219	1.8765e−216	0.0000e−000
	CPSO-SBAC	3.3406e−218	2.3519e−210	0.0000e−000
f_2	CPSO-TVAC	3.9041e−109	5.0448e−106	2.2329e−105
	CPSO-SCAC	1.1572e−111	4.5358e−104	1.6833e−103
	CPSO-NDAC	1.5361e−112	6.9727e−110	3.6898e−109
	CPSO-SBAC	2.5330e−111	9.2311e−108	6.4313e−109
f_3	CPSO-TVAC	9.0000e−000	9.0000e−000	0.0000e−000
	CPSO-SCAC	9.0000e−000	9.0000e−000	0.0000e−000
	CPSO-NDAC	9.0000e−000	9.0000e−000	0.0000e−000
	CPSO-SBAC	9.0000e−000	9.0000e−000	0.0000e−000
f_4	CPSO-TVAC	0.0000e−000	0.0000e−000	0.0000e−000
	CPSO-SCAC	0.0000e−000	0.0000e−000	0.0000e−000
	CPSO-NDAC	0.0000e−000	0.0000e−000	0.0000e−000
	CPSO-SBAC	0.0000e−000	0.0000e−000	0.0000e−000
f_5	CPSO-TVAC	1.3832e−000	4.1863e−000	5.3916e−000
	CPSO-SCAC	2.1622e−001	1.0215e−000	5.1392e−000
	CPSO-NDAC	6.3572e−001	9.1385e−001	4.9603e−001
	CPSO-SBAC	6.6216e−001	8.2182e−001	4.2830e−001
f_6	CPSO-TVAC	3.6535e−016	4.7287e−011	8.6239e−006
	CPSO-SCAC	7.1524e−017	7.7632e−012	5.2132e−007
	CPSO-NDAC	4.3612e−017	8.8136e−012	5.1696e−007
	CPSO-SBAC	4.3621e−017	7.3288e−012	4.8206e−007



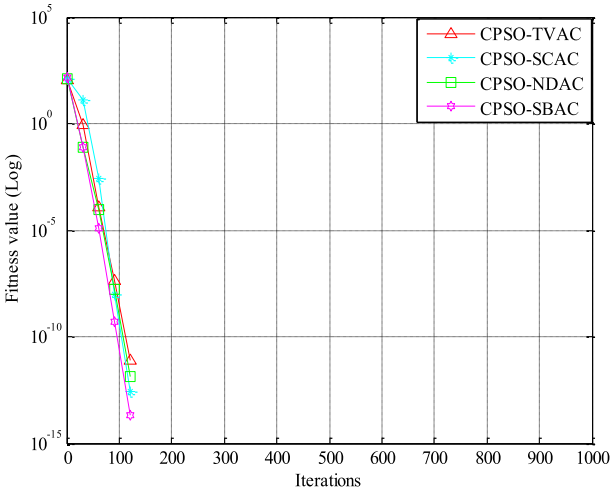
(a) Sphere function (f_1)



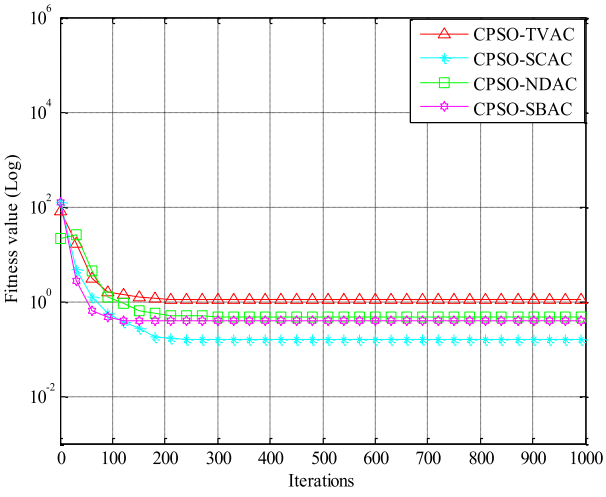
(b) Schwefel function (f_2)



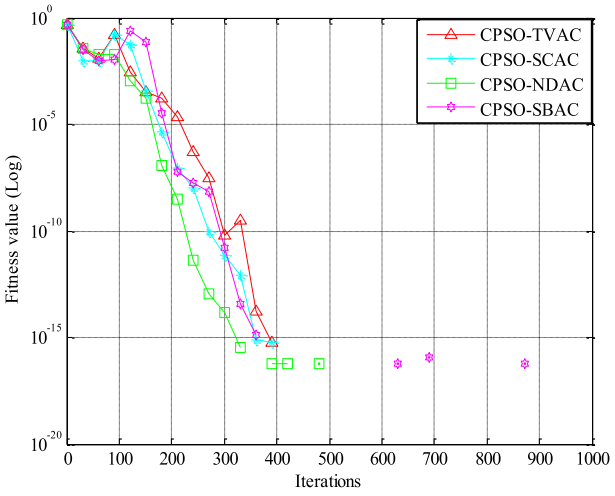
(c) Rosenbrock function (f_3)



(d) Rastrigin function (f_4)



(e) Griewank function (f_5)



(f) Ackley function (f_6)

Fig. 6. Convergence curves of CPSO with different acceleration coefficients.

Table 4
Comparison results of CPSOS with other PSO variants under $D = 30$.

Func.	Item	CLPSO	HPSO-TVAC	FIPSO	LPSO	DMSPSO	LFPSO	CPSOS
f_1	Avg.	1.58e−12	2.83e−33	2.42e−13	3.34e−14	2.65e−31	4.69e−31	**3.28e−99**
	Std.	7.70e−13	3.19e−33	1.73e−13	5.39e−14	6.25e−31	2.50e−30	**4.21e−99**
	Rank	7	2	6	5	3	4	1
f_2	Avg.	2.51e−08	9.03e−20	2.76e−08	1.70e−10	1.57e−18	2.64e−17	**3.20e−60**
	Std.	5.84e−09	9.58e−20	9.04e−09	1.39e−10	3.79e−18	6.92e−17	**5.62e−60**
	Rank	6	2	7	5	3	4	1
f_3	Avg.	**1.14e + 01**	2.39e+01	2.51e+01	2.81e+01	4.16e+01	2.38e+01	2.33e+01
	Std.	9.85e+00	2.65e+01	5.10e−01	2.18e+01	3.03e+01	3.17e−01	**3.61e−01**
	Rank	1	4	5	6	7	3	2
f_4	Avg.	9.09e−05	9.43e+00	6.51e+01	3.51e+01	2.72e+01	4.54e+00	**6.92e−05**
	Std.	1.25e−04	3.48e+00	1.34e+01	6.89e+00	6.02e+00	1.03e+01	**8.05e−05**
	Rank	2	4	7	6	5	3	1
f_5	Avg.	9.02e−09	9.75e−03	9.01e−12	1.53e−03	6.21e−03	**8.14e−17**	7.26e−01
	Std.	8.57e−09	8.33e−03	1.84e−11	4.32e−03	8.14e−03	**4.46e−16**	4.68e−02
	Rank	3	6	2	4	5	1	7
f_6	Avg.	3.66e−07	7.29e−14	2.33e−07	8.20e−08	1.84e−14	**1.68e−14**	1.72e−14
	Std.	7.57e−08	3.00e−14	7.19e−08	6.73e−08	4.35e−15	4.84e−15	**3.69e−15**
	Rank	7	4	6	5	3	1	2
Avg. rank		**4.33**	**3.67**	**5.50**	**5.17**	**4.33**	**2.67**	**2.33**
Final rank		4	3	6	5	4	2	1

valley. However, even though this valley is easy to find, convergence to the minimum is very difficult. In view of this, the four different PSOs, i.e., CPSO-TVAC, CPSO-SCAC, CPSO-NDAC and CPSO-SBAC, converge to the same global minimum with $d = 10$. In comparison, f_4 is multimodal, which is the most widely used test function in the research of PSO. Generally speaking, it is easy for an optimization algorithm to be trapped in a local minimum on its way to the global minimum. However, by virtue of striking a better balance between the local search ability and global search ability, it can be clearly seen from Table 3 that the PSO paradigm proposed in this work has achieved the global optima on each dimension on f_4. This is largely ascribed to the following aspects: first of all, the chaos-based swarm initialization instead of the random initialization is adopted to start the PSO algorithm so as to diversify the swarm particles and enhance the performance of PSO by preventing the premature convergence. In the meanwhile, the stability of PSO and the quality of the final solution can also be improved to a certain extent. Second, the sigmoid-based acceleration coefficients is formulated to dynamically regulate the factors c_1 and c_2 throughout the search, which makes full use of the ability of sigmoid function to transform from the linear to nonlinear states smoothly. Last but not the least, two sets of slowly varying function and regular varying function embedded update mechanism in conjunction with chaos based re-initialization and Gaussian mutation strategies are employed at different evolution stages to update the particles during the whole search process, which can effectively keep the diversity of the swarm and get out of possible local optima to continue exploring the optimal search regions in the whole spaces. Note that to improve the convergence speed of the PSO, the control factor ρ is specifically utilized to divide the evolution process into two stages so as to further refine the search process, involving the exploration (RVF and chaos-based re-initialization) and exploitation (SVF and Gaussian mutation). To summarize, the chaos-based swarm initialization, sigmoid-based acceleration coefficients, nonlinear decreasing inertia weight and the SVF and RVF embedded update mechanism, to some extent, play a good complementary role each other and the combination makes them benefit from each other in the evolution process.

To illustrate the search process of different PSOs, Fig. 6 displays the convergence curves of PSO with different acceleration coefficients for all the test functions. Taking Fig. 6(a) for example, the convergence curves of the four PSOs decrease at almost the same rate in the first 300 iterations. But after that, CPSO-NDAC consistently declines with fast convergence speed compared to the others. It is interesting to note that the evolutionary curves visualized in Fig. 6(b)–(c) converge in almost the same way on all the test functions, especially in Fig. 6(c), one can see that all the PSOs evolve very fast at the early 100 iterations, after that their convergent performance exhibit very slowly and flatly. In other words, the acceleration coefficients TVAC, SCAC, NDAC and SBAC formulated in this work have a marginal effect on the performance of functions f_2 and f_3. As for f_4, it is clear to observe that the four different PSOs are able to achieve the global best solution without being trapped in local optima. Specifically, CPSO-SBAC yields the fastest convergence rate to reach the global optimum on this multimodal function. In contrast, the convergence rate of CPSO-SCAC and CPSO-NDAC is not as fast as that of CPSO-SBAC, but both of them evolve rapidly than CPSO-TVAC as the search proceeds. Concerning the convergence curves illustrated in Fig. 6(e), it should be noted that CPSO-SBAC converges very fast than the other three methods in the former 100 iterations, but CPSO-SCAC proceeds exploring the search space until 200 iterations. On the whole, the performance is improved very slightly for all the PSOs after 200 iterations until the end of the search process. As regards the convergence curves of different PSOs on function f_6 shown in Fig. 6(f), it is clearly observed that all the PSOs evolve better in the first 400 iterations. In sum, under the same experimental conditions, PSO with different acceleration coefficients can get better performance in most cases.

To further validate the merits of our proposal, CPSOS is compared with the other six PSO variants in Table 4, including LPSO [20], FIPSO [30], HPSO-TVAC [34], DMSPSO [25], CLPSO [23] and LFPSO [14]. Here, note that the dimension of all the test functions is set as 30. Likewise, the average best solution (Avg.) and standard deviation (Std.) are used to measure the performance, and the best result in a comparison is emphasized as bold font. From the results, we can find that CPSOS is ranked the first three times, the second on functions f_3 and f_6 two times, even though the Avg. values of CPSOS are slightly worse than that of CLPSO and LFPSO on f_3 and f_6 respectively, it still gets the better Std. values. This better stability results from the good balance between the exploration and the exploitation through using the chaos-based swarm initialization, sigmoid-based acceleration coefficients, the slowly varying function and regular varying function embedded update mechanism as well as the chaos based re-initialization and Gaussian mutation strategies respectively. Besides, according to the final rank values listed at the bottom of Table 4, it can be also seen that CPSOS achieves better performance than the others in terms of the average best solution and standard deviation, which further bears out the effectiveness and efficiency of the proposed PSO paradigm in this work.

5.2. Experiments based on CEC'13 functions

As observed from the experimental results shown above, the proposed

Table 5
Benchmark functions of CEC′13.

Type	No.	Function name	Search range
Unimodal functions	1	sphere function	$[-100,100]^D$
	2	rotated high conditioned elliptic function	
	3	rotated bent cigar function	
	4	rotated discus function	
	5	different powers function	
Multimodal functions	6	rotated rosenbrock's function	
	7	rotated schaffers f7 function	
	8	rotated ackley's function	
	9	rotated weierstrass function	
	10	rotated griewank's function	
	11	rastrigon's function	
	12	rotated rastrigin's function	
	13	non-continuous rotated rastrigin's function	
	14	schwefel's function	
	15	rotated schwefel's function	
	16	rotated katsuura function	
	17	lunacek bi-rastrigin function	
	18	rotated lunacek bi-rastrigin function	
	19	expanded griewank's plus rosenbrock's function	
	20	expanded scaffer's f6 function	
Composition functions	21	composition function 1 ($n = 5$, rotated)	
	22	composition function 2 ($n = 3$, unrotated)	
	23	composition function 3 ($n = 3$, rotated)	
	24	composition function 4 ($n = 3$, rotated)	
	25	composition function 5 ($n = 3$, rotated)	
	26	composition function 6 ($n = 5$, rotated)	
	27	composition function 7 ($n = 5$, rotated)	
	28	composition function 8 ($n = 5$, rotated)	

Table 6
Parameter settings for different PSO variants.

Algorithm	Swarm size	Inertia weight	Acceleration coefficients	Others	Ref.
GPSO	40	0.9–0.4	$c_1 = 2, c_2 = 2$	-	[37]
OLPSO	40	0.9–0.4	$c = 2$	$G = 5$	[51]
DMPPSO	40	0.9–0.4	$c_1 = 0.5 + 0.5\exp(-t/500) + 1.4\sin(t)/30$, $c_2 = 1 + 1.4\,(1\text{-}exp\,(-t/500)) + 1.4\sin(t)/30$	$\lambda = 10$	[21]
SRPSO	40	1.05–0.5	$c_1 = 1.49445, c_2 = 1.49445$	$\eta = 1.0, \lambda = 0.5$	[40]
HCLPSO	40	0.99–0.2	$c_1 = 2.5–0.5, c_2 = 0.5–1.5, c = 3–1.5$	-	[27]
EPSO	40	0.9–0.4	$c_1 = 2.5–0.5, c_2 = 0.5–2.5$	-	[28]
IDE-PSO	40	0.9–0.4	$c_1 = 0.9–0.4, c_2 = 0.4–0.9$	$m_1 = 0–0.4$, $m_2 = 0.9$, $Stag = 10$	[13]

CPSOS is markedly superior to the others in most cases, which verifies the effectiveness of it for basic function optimization problems. To further illustrate the effect of the CPSOS algorithm for relatively complex problems, a series of experiments are conducted on CEC′13 test suite [24] that has 28 functions (shown in Table 5), including 5 unimodal functions, 15 multimodal functions and 8 composition functions. At the same time, we compare it with several well-known PSO variants listed as follows:

- Global PSO (GPSO)
- Orthogonal learning PSO (OLPSO)
- Difference mean based perturbation in PSO (DMPPSO)
- Self-regulating PSO (SRPSO)
- Heterogeneous comprehensive learning PSO (HCLPSO)
- Ensemble particle swarm optimizer (EPSO)
- Individual difference evolution based PSO (IDE-PSO)

Note that the parameter settings of these compared PSOs are consistent with the original literature (listed in Table 6), and they terminate when reaching the maximum number of function evaluations $2000 \times D$ (D is the dimension). As for the proposed CPSOS, it is likewise run 30 times on each benchmark function with 1000 iterations for each run, and the stopping criterion is set as reaching the maximum number of allowable iterations. It is worth noting that all the experimental parameters are the same as those in subsection 5.1, and all the experiments run on the same test machine with an Intel Core i7-8550U CPU at 2.0 GHz, 8 GB of RAM and Window 10 operating system. In addition, the other two often used criteria are applied here to investigate the performance of different PSOs:

- Mean: the mean of the fitness error (defined as $f(x)$-$f(x^*)$, where x is the solution yielded by the algorithm and x^* is the real global minimum of the corresponding function).
- Std: the standard deviation of the fitness error.

According to the experimental setup described in Table 6, we have reproduced all the PSO variants as completely as possible according to the original literature and carried out a large number of simulation experiments. But considering the fairness of comparison, the results for some PSOs, such as GPSO, OLPSO-L, DMPPSO-G and IDE-PSO, are directly taken from literature [13] and the performance of SRPSO, HCLPSO[1)], EPSO[1)] and CPSOS has been evaluated using the experimental guidelines defined in CEC′13 test suite [24]. From Table 7, it can be clearly observed that CPSOS yields better result than the other methods on f_4, f_7, f_9, f_{14}, f_{20}, f_{21}, f_{23}, f_{24}, f_{26}, f_{27} and equals with OLPSO-L on f_1, SRPSO and HCLPSO on f_1 and f_{16}, EPSO on f_1, f_8, f_{16}, f_{26}, IDE-PSO on f_1, f_{14} and f_{16} in terms of the mean error. Compared to the best result achieved by HCLPSO on f_{15} and f_{28} as well as EPSO on f_{12}, f_{13} and f_{25}, CPSOS is only slightly worse than these two PSO variants by a small margin, but the standard deviation of our proposal is sufficiently lower than the other

[1] Source codes can be available from http://www.ntu.edu.sg/home/epnsugan/.

Table 7
Comparison results of CPSOS with other PSO variants under $D = 10$.

Func.	Item	GPSO	OLPSO-L	DMPPSO-G	SRPSO	HCLPSO	EPSO	IDE-PSO	CPSOS
f_1	Mean	3.79e−01	**0.00e + 00**	1.25e+00	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**
	Std	9.88e−01	**0.00e + 00**	2.12e+00	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**
f_2	Mean	7.30e+05	**1.52e + 04**	2.77e+06	1.54e+05	1.58e+05	1.64e+05	3.64e+06	2.79e+06
	Std	1.17e+06	**8.62e + 03**	3.33e+06	1.72e+05	1.07e+05	2.11e+05	3.17e+06	3.28e+05
f_3	Mean	3.39e+08	9.12e+07	5.54e+08	4.65e+07	3.14e+07	2.81e+07	**2.68e + 07**	3.01e+07
	Std	7.44e+08	2.46e+08	1.10e+09	1.33e+08	4.26e+07	3.95e+07	**3.31e + 07**	3.40e+07
f_4	Mean	2.63e+03	1.52e+04	1.46e+04	3.61e+02	1.61e+02	1.52e+02	1.34e+02	**1.31e + 02**
	Std	2.76e+03	5.99e+03	5.72e+03	6.08e+02	1.55e+02	1.79e+02	1.43e+02	**1.38e + 02**
f_5	Mean	2.02e+01	**0.00e + 00**	6.58e+00	**0.00e + 00**	2.59e−05	2.63e−05	6.18e−04	4.11e−06
	Std	2.19e+01	**0.00e + 00**	1.76e+01	**0.00e + 00**	3.04e−05	8.08e−05	2.79e−04	6.32e−06
f_6	Mean	3.21e+01	1.11e+01	2.22e+01	4.34e−01	3.66e+00	4.37e+00	**1.89e−01**	3.12e+01
	Std	3.72e+01	1.35e+01	3.23e+01	1.12e+00	5.12e+00	2.91e+00	**3.23e−02**	3.27e+01
f_7	Mean	1.30e+02	4.43e+01	7.22e+01	2.90e+01	2.83e+01	2.76e+01	3.80e+01	**2.75e + 01**
	Std	5.69e+01	1.64e+01	3.24e+01	2.86e+01	3.97e+01	3.85e+01	1.93e+01	**6.21e + 00**
f_8	Mean	2.03e+01	2.04e+01	2.04e+01	2.03e+01	2.02e+01	**1.98e + 01**	2.02e+01	**1.98e + 01**
	Std	7.65e−02	8.10e−02	8.60e−02	7.25e−02	4.36e−02	3.77e−02	4.52e−02	**3.46e−02**
f_9	Mean	9.14e+00	7.15e+00	9.16e+00	8.82e+00	6.13e+00	5.89e+00	6.11e+00	**5.75e + 00**
	Std	1.42e+00	1.15e+00	1.87e+00	5.30e−01	8.14e−01	6.06e+00	1.17e+00	**4.46e−01**
f_{10}	Mean	1.28e+01	1.02e+00	7.38e+00	6.04e−01	**5.01e−01**	5.49e−01	1.52e+00	2.16e+00
	Std	8.06e+00	9.91e−01	5.58e+00	3.90e−01	2.36e−01	**2.18e−01**	1.62e+00	2.27e−01
f_{11}	Mean	6.79e+01	**4.15e−01**	3.89e+01	2.61e+00	1.65e+00	1.61e+00	1.37e+00	1.46e+00
	Std	2.92e+01	7.14e−01	1.47e+01	1.78e+00	2.87e+00	3.36e+00	1.09e+00	**3.36e−01**
f_{12}	Mean	6.77e+01	3.10e+01	4.28e+01	2.01e+01	1.43e+01	**1.39e + 01**	1.44e+01	1.44e+01
	Std	3.84e+01	8.24e+00	1.83e+01	9.76e+00	5.05e+00	4.92e+00	5.90e+00	**4.62e + 00**
f_{13}	Mean	7.93e+01	4.66e+01	5.30e+01	2.28e+01	1.90e+01	**1.86e + 01**	1.91e+01	1.88e+01
	Std	2.36e+01	1.92e+01	1.31e+01	1.01e+01	6.36e+00	7.05e+00	5.89e+00	**4.77e + 00**
f_{14}	Mean	1.24e+03	8.12e+02	1.48e+03	9.01e+01	2.61e+01	3.53e+01	**3.58e + 00**	**3.58e + 00**
	Std	2.93e+02	2.99e+02	3.84e+02	8.78e+01	4.90e+01	5.18e+01	**3.83e + 00**	4.18e+00
f_{15}	Mean	1.29e+03	1.37e+03	1.33e+03	8.34e+02	**6.88e + 02**	7.43e+02	7.85e+02	6.91e+02
	Std	3.25e+02	2.40e+02	3.47e+02	2.83e+02	4.36e+02	3.79e+02	2.32e+02	**2.26e + 02**
f_{16}	Mean	8.97e−01	8.99e−01	1.59e+00	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**
	Std	2.94e−01	2.32e−01	3.36e−01	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**
f_{17}	Mean	7.47e+01	**1.07e + 01**	6.60e+01	1.26e+01	1.18e+01	1.12e+01	1.15e+01	1.29e+01
	Std	2.09e+01	6.27e−01	1.80e+01	3.76e+00	3.55e+00	3.64e+00	1.95e+00	**4.92e−01**
f_{18}	Mean	6.45e+01	3.86e+01	7.18e+01	**2.45e + 01**	3.05e+01	2.82e+01	2.91e+01	3.17e+01
	Std	2.40e+01	4.83e+00	1.51e+01	6.95e+00	1.21e+00	6.36e+00	1.21e+01	**1.18e−01**
f_{19}	Mean	5.70e+00	2.16e+00	4.34e+00	5.01e−01	**4.41e−01**	6.16e−01	5.96e−01	4.75e−01
	Std	2.94e+00	1.05e+00	2.44e+00	1.87e−01	5.17e−01	5.06e−01	2.54e−01	**1.86e−01**
f_{20}	Mean	4.22e+00	3.44e+00	3.87e+00	3.23e+00	3.42e+00	3.13e+00	3.07e+00	**3.03e + 00**
	Std	3.21e−01	4.96e−01	3.07e−01	3.85e−01	3.69e−01	4.71e−01	4.47e−01	**2.68e−01**
f_{21}	Mean	3.88e+02	3.92e+02	3.74e+02	3.75e+02	2.61e+02	2.46e+02	2.39e+02	**2.38e + 02**
	Std	4.45e+01	4.08e+01	6.42e+01	6.76e+01	4.13e+01	5.26e+01	1.54e+02	**3.98e + 01**
f_{22}	Mean	1.70e+03	9.38e+02	1.71e+03	1.91e+02	1.12e+02	1.24e+02	**9.46e + 01**	9.73e+01
	Std	3.70e+02	2.99e+02	4.40e+02	1.10e+02	2.83e+02	2.66e+02	7.91e+01	**5.38e + 01**
f_{23}	Mean	1.78e+03	1.60e+03	1.62e+03	9.13e+02	1.01e+03	9.93e+02	1.03e+03	**6.62e + 02**
	Std	4.97e+02	3.16e+02	5.54e+02	3.72e+02	2.94e+02	4.26e+02	2.74e+02	**2.39e + 02**
f_{24}	Mean	2.35e+02	2.06e+02	2.30e+02	2.14e+02	1.82e+02	1.60e+02	1.64e+02	**1.58e + 02**
	Std	6.81e+00	2.12e+01	**2.68e + 00**	3.83e+00	3.17e+01	3.32e+01	2.58e+01	4.93e+00
f_{25}	Mean	2.34e+02	2.18e+02	2.27e+02	2.18e+02	1.84e+02	**1.73e + 02**	1.75e+02	1.81e+02
	Std	7.73e+00	4.60e+00	3.35e+00	2.98e+00	2.21e+01	2.06e+01	**1.80e + 01**	1.93e+00
f_{26}	Mean	2.45e+02	1.67e+02	2.08e+02	2.11e+02	1.46e+02	**1.34e + 02**	1.41e+02	**1.34e + 02**
	Std	7.22e+01	2.59e+01	5.10e+01	5.25e+01	2.58e+01	2.62e+01	**2.04e + 01**	2.35e+01
f_{27}	Mean	6.79e+02	4.72e+02	6.06e+02	4.52e+02	3.85e+02	3.75e+02	3.80e+02	**3.67e + 02**
	Std	1.20e+02	4.24e+01	4.77e+01	3.93e+01	3.66e+01	3.73e+01	4.69e+01	**3.41e + 01**
f_{28}	Mean	6.54e+02	4.66e+02	5.27e+02	3.30e+02	**2.57e + 02**	2.87e+02	3.23e+02	2.60e+02
	Std	2.60e+02	2.04e+02	2.70e+02	1.14e+02	2.03e+02	1.52e+02	1.31e+02	**1.13e + 02**

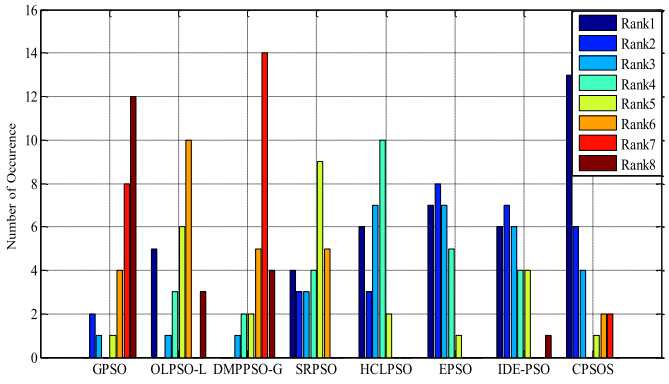
two PSO variants on these functions for 10-dimensional problems except for f_{25}, especially for functions f_{12} and f_{13}, CPSOS outperforms the EPSO algorithm with an order of magnitude difference of the standard deviation. Likewise, despite the mean errors of our method is larger than that of OLPSO-L on f_{11} and f_{17}, SRPSO on f_{18}, HCLPSO on f_{19} and IDE-PSO on f_{22}, it still can be seen that CPSOS is apparently superior to them according to the *Mean* values. This indicates that PSO with the chaos-based swarm initialization can alleviate the inherent drawbacks of the low stability. Thus future work will focus on investigating advanced strategies to make CPSOS more stable. In comparison, the performance of our method is inferior to that of OLPSO-L on f_2 and f_5, SRPSO on f_5, HCLPSO on f_{10} as well as IDE-PSO on f_3 and f_6. On the whole, the proposed CPSOS is superior or highly competitive to the other peer PSO variants in most cases. Just as the "no free lunch theorem" described, one algorithm maintains an average performance on every aspect close to any other when considering a large number of test functions. In other words, an algorithm cannot offer better performance on all kind of problems.

Table 8 reports the experimental results for 30-dimensional problems. The bold font indicates the PSO method which achieves the best performance on each function. As for the results for the unimodal functions f_1~f_5, one can observe that CPSOS surpasses the other PSO variants except for OLPSO-L on f_1, f_2 and f_5, and it is only marginally worse than

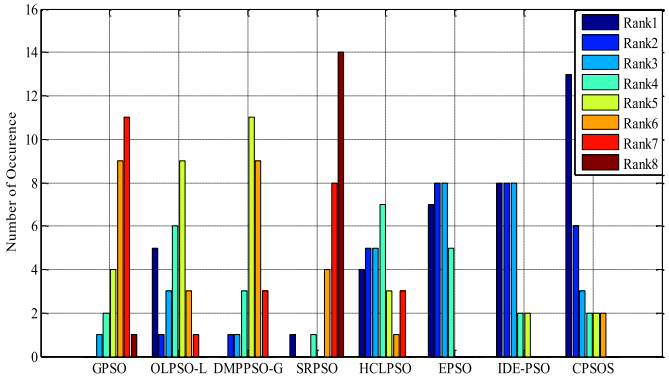
Table 8
Comparison results of CPSOS with other PSO variants under $D = 30$.

Func.	Item	GPSO	OLPSO-L	DMPPSO-G	SRPSO	HCLPSO	EPSO	IDE-PSO	CPSOS
f_1	Mean	5.08e+03	**0.00e + 00**	2.15e+01	7.99e+04	1.03e−14	9.82e−15	6.21e−15	5.41e−15
	Std	2.01e+03	**0.00e + 00**	2.88e+01	1.13e+04	9.48e−14	8.80e−15	3.04e−14	7.03e−15
f_2	Mean	5.94e+07	**6.37e + 05**	2.22e+07	1.75e+09	6.83e+07	4.73e+07	5.16e+07	4.62e+07
	Std	3.34e+07	**1.97e + 05**	1.99e+07	7.09e+08	2.37e+07	2.91e+07	3.47e+07	1.71e+07
f_3	Mean	3.71e+15	3.06e+09	1.50e+10	7.73e+19	5.66e+07	5.25e+07	5.46e+07	**4.60e + 07**
	Std	1.82e+16	3.63e+09	7.73e+09	2.50e+20	5.82e+07	6.13e+07	7.14e+07	**5.29e + 07**
f_4	Mean	4.21e+04	9.93e+04	6.36e+04	4.61e+05	8.52e+03	8.31e+03	8.20e+03	**7.11e + 03**
	Std	1.01e+04	1.72e+04	1.38e+04	3.12e+05	6.86e+03	7.15e+03	6.09e+03	**6.06e + 03**
f_5	Mean	8.79e+02	**0.00e + 00**	3.17e+01	3.71e+04	6.39e+01	2.76e+01	2.44e+01	2.93e+01
	Std	2.83e+02	**0.00e + 00**	6.70e+01	1.18e+04	3.61e+01	3.88e+01	9.23e+01	3.52e+01
f_6	Mean	5.29e+02	7.75e+01	8.49e+01	1.75e+04	5.64e+01	5.83e+01	**2.45e + 01**	5.64e+01
	Std	2.57e+02	**2.19e + 01**	4.33e+01	6.13e+03	7.16e+01	6.89e+01	2.31e+01	6.18e+01
f_7	Mean	6.88e+04	2.95e+02	8.18e+02	3.03e+06	1.10e+02	**1.03e + 02**	1.12e+02	**1.03e + 02**
	Std	9.33e+04	2.35e+02	2.22e+03	7.46e+06	2.65e+01	3.14e+01	2.94e+01	**1.85e + 01**
f_8	Mean	2.09e+01	2.10e+01	2.10e+01	2.12e+01	2.21e+01	**2.06e + 01**	2.08e+01	2.07e+01
	Std	5.32e−02	3.97e−02	5.30e−02	4.53e−02	4.88e−02	5.28e−02	5.92e−02	**3.73e−02**
f_9	Mean	4.13e+01	3.83e+01	4.15e+01	4.38e+01	3.29e+01	3.65e+01	**3.11e + 01**	3.67e+01
	Std	3.35e+00	2.29e+00	2.20e+00	1.22e+00	3.74e+00	3.51e+00	3.56e+00	**1.21e + 00**
f_{10}	Mean	8.52e+02	1.03e+01	8.82e+00	1.17e+04	**6.39e + 00**	7.78e+00	3.42e+01	3.42e+01
	Std	3.32e+02	**1.74e + 01**	2.21e+01	2.10e+03	9.05e+01	8.13e+01	1.10e+02	8.06e+01
f_{11}	Mean	5.90e+02	**7.55e + 00**	4.41e+02	1.21e+03	2.84e+02	1.74e+02	1.70e+02	1.73e+02
	Std	1.35e+02	**5.23e + 00**	1.02e+02	2.31e+02	2.27e+02	2.36e+02	3.11e+01	3.07e+01
f_{12}	Mean	6.07e+02	3.45e+02	4.61e+02	1.27e+03	1.33e+02	**1.21e + 02**	1.25e+02	**1.21e + 02**
	Std	1.27e+02	7.86e+01	8.09e+01	1.29e+02	3.79e+01	4.28e+01	6.08e+01	**3.63e + 01**
f_{13}	Mean	6.06e+02	4.24e+02	3.93e+02	1.26e+03	1.47e+02	**1.38e + 02**	1.52e+02	1.41e+02
	Std	1.21e+02	8.17e+01	7.45e+01	1.85e+02	3.32e+01	4.11e+01	3.60e+01	**3.26e + 01**
f_{14}	Mean	5.13e+03	4.74e+03	7.56e+03	9.26e+03	2.62e+03	3.29e+03	**1.54e + 02**	**1.54e + 02**
	Std	9.89e+02	6.44e+02	9.92e+02	4.73e+02	9.86e+02	1.06e+03	9.07e+01	**8.99e + 01**
f_{15}	Mean	5.44e+03	7.16e+03	7.59e+03	9.42e+03	**4.50e + 03**	4.64e+03	4.59e+03	**4.50e + 03**
	Std	8.38e+02	5.67e+02	1.35e+03	5.17e+02	2.66e+02	2.52e+02	1.03e+02	**1.01e + 02**
f_{16}	Mean	2.33e+00	2.23e+00	3.14e+00	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**
	Std	5.77e−01	3.74e−01	5.01e−01	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**
f_{17}	Mean	6.55e+02	**3.96e + 01**	4.80e+02	2.37e+03	3.50e+02	3.14e+02	3.32e+02	3.59e+02
	Std	1.27e+02	**6.80e + 00**	6.99e+01	3.11e+02	3.17e+01	3.21e+01	6.78e+01	2.75e+01
f_{18}	Mean	6.82e+02	2.26e+02	4.57e+02	2.18e+03	2.79e+02	2.12e+02	**1.83e + 02**	2.09e+02
	Std	1.24e+02	2.60e+01	8.61e+01	2.79e+02	5.43e+01	6.68e+01	8.62e+01	**2.37e + 01**
f_{19}	Mean	2.85e+03	2.08e+01	4.46e+01	8.52e+06	3.01e+00	2.85e+00	3.06e+00	**2.83e + 00**
	Std	1.64e+03	1.63e+01	1.71e+01	4.55e+06	2.88e+00	3.62e+00	9.09e−01	**5.71e−01**
f_{20}	Mean	1.45e+01	1.45e+01	1.45e+01	1.50e+01	1.31e+01	1.24e+01	1.26e+01	**1.21e + 01**
	Std	1.39e−01	8.48e−01	3.80e−01	5.29e-08	3.87e+00	4.35e+00	1.05e+00	**1.22e−01**
f_{21}	Mean	1.34e+03	1.30e+03	3.39e+02	5.33e+03	2.99e+02	2.91e+02	**2.87e + 02**	2.99e+02
	Std	5.29e+02	4.63e+02	8.74e+01	7.52e+02	4.83e+01	5.18e+01	1.49e+02	**4.32e + 01**
f_{22}	Mean	7.00e+03	5.18e+03	7.38e+03	9.73e+03	8.89e+02	9.24e+02	**5.47e + 02**	6.24e+02
	Std	1.00e+03	6.44e+02	1.75e+03	4.33e+02	3.34e+02	1.78e+02	2.73e+02	**9.57e + 01**
f_{23}	Mean	6.23e+03	7.84e+03	8.29e+03	9.92e+03	6.12e+03	5.31e+03	5.16e+03	**5.08e + 03**
	Std	1.03e+03	6.81e+02	1.41e+03	3.21e+02	1.64e+03	2.01e+03	1.20e+03	**2.97e + 02**
f_{24}	Mean	4.70e+02	2.94e+02	3.19e+02	3.18e+02	3.36e+02	**2.85e + 02**	2.89e+02	3.21e+02
	Std	7.25e+01	6.00e+00	**4.18e + 00**	4.29e+00	1.83e+01	2.32e+01	1.12e+01	1.10e+01
f_{25}	Mean	4.27e+02	3.12e+02	3.05e+02	3.14e+02	**2.88e + 02**	2.91e+02	2.93e+02	**2.88e + 02**
	Std	3.16e+01	7.42e+00	3.83e+00	**3.81e + 00**	4.48e+00	4.93e+00	5.76e+00	4.64e+00
f_{26}	Mean	3.72e+02	2.34e+02	2.82e+02	4.13e+02	2.51e+02	2.02e+02	2.03e+02	**2.02e + 02**
	Std	8.81e+01	7.18e+01	1.06e+02	2.36e+01	2.56e+01	3.14e+01	3.56e+00	**2.88e + 00**
f_{27}	Mean	1.61e+03	1.25e+03	1.44e+03	1.47e+03	1.33e+03	1.24e+03	**1.21e + 03**	1.47e+03
	Std	1.82e+02	7.90e+01	3.81e+01	4.42e+01	3.92e+02	4.75e+01	1.03e+02	**3.61e + 01**
f_{28}	Mean	4.80e+03	3.56e+03	2.86e+03	8.87e+03	3.12e+02	3.16e+02	3.15e+02	**3.08e + 02**
	Std	9.73e+02	5.47e+02	9.95e+02	1.42e+03	4.23e+01	5.01e+01	6.46e+01	**4.87e + 01**

EPSO on f_5 according to the *Mean* value. In contrast, OLPSO-L is able to converge to the optimal solution on functions f_1 and f_5 due to its good optimization performance to the unimodal function. With regard to the experimental results for the multimodal functions, viz., f_6~f_{20}, CPSOS achieves better mean error and standard deviation values and it exceeds GPSO and DMPPSO-G by a significant margin. Likewise, with the exception of f_{16}, our proposal markedly outperforms SRPSO on all the basic multimodal functions, some even with three or more orders of magnitude difference in terms of the *Mean* value. Besides, it is worth noting that CPSOS is superior to OLPSO-L on this set of functions apart from f_{10}, f_{11} and f_{17} even though the latter has good performance on the first five unimodal functions. In other words, our method has the capability to deal with the relatively complex multimodal problems with high dimensions. Compared to the rest three PSO variants, CPSOS obtains better result than HCLPSO on f_7, f_8, f_{11}~f_{14}, f_{18}, f_{19}, f_{20}, EPSO on f_6, f_{11}, f_{14}, f_{15}, f_{18}, f_{19}, f_{20}, IDE-PSO on f_7, f_8, f_{12}, f_{13}, f_{15}, f_{19}, f_{20} and equals with HCLPSO on f_6, f_{15}, f_{16}, EPSO on f_7, f_{12}, f_{16} and IDE-PSO on f_{10}, f_{14}, f_{16}. In sum, the proposed CPSOS exhibits good performance for this set of multimodal functions. Concerning the results for the eight composition functions f_{21}~f_{28}, our method can also get encouraging performance on the majority of the test functions, especially with the smallest *Std* values in most cases (6 out of 8). In addition, note that an interesting



(a) For 10-dimensional case



(b) For 30-dimensional case

Fig. 7. Histogram of mean error ranks under $D = 10/30$.

observation comes from $f_3, f_{12}, f_{15}, f_{25}$ and f_{28} is that CPSOS is not the best method on these functions for 10-dimensional cases, but it is the best one (same with HCLPSO on f_{15}) on the same functions in solving the corresponding 30-dimensional problems, which implies that CPSOS has the ability to address complex problems with good scalability. Furthermore, the standard deviation of CPSOS is significantly smaller than that of the other PSO variants in most cases. Alternatively, it is worth noting that the average performance of each PSO mentioned in Table 8 decreases when the function's dimension scale increasing from 10 to 30. At the same time, there is no doubt that the computational time increases accordingly. To summarize, from the extensive experiments conducted above, it can be clearly seen that PSO with the chaos-based swarm initialization is able to yield high-quality initial particles and alleviate the shortcomings of the low stability, the sigmoid-based acceleration coefficients are capable to strike a better balance between the exploration and exploitation, two sets of slowly varying function and regular varying function embedded update mechanism as well as the chaos based re-initialization and Gaussian mutation strategies have the ability to maintain the potential diversity of the whole swarm. All of these result in the better performance of the proposed CPSOS algorithm.

To further illustrate the superiority of CPSOS, Fig. 7 depicts the histogram of mean error ranks for both 10 and 30 dimensional problems, which is mainly used to indicate the number of times each PSO algorithm has acquired the ranks in the range of 1–8. As shown in Fig. 7, CPSOS achieves the top rank nearly twice as compared to EPSO which is the second best algorithm in the 10-dimensional case. Likewise, our method is also the top ranked algorithm that is apparently superior to the second best method IDE-PSO and the third best approach EPSO in the 30-dimensional problem. This fully validates the effectiveness of CPSOS for numerical function optimization, at least in CEC'13 test suite.

In addition, to thoroughly compare CPSOS with the other seven PSO variants, we have validated it from the perspective of statistical analysis by trial and error. To be specific, the statistical *t*-tests are performed and the results are reported in Table 9. Note that the number of benchmark functions showing that CPSOS is significantly better than, almost the same as, and significantly worse than the compared algorithms is illustrated. The level of significance is 0.05, and the "Merit" score is calculated by subtracting the "worse" score from the "better" score. Compared to GPSO, OLPSO-L, DMPPSO-G, SRPSO, HCLPSO, EPSO and IDE-PSO, CPSOS performs significantly better on 26, 15, 24, 14, 8, 6 and 8 functions for 10-dimensional problems as well as 28, 12, 20, 24, 15, 6 and 5 functions for 30-dimensional cases, respectively. Interestingly, some merit values such as GPSO, SRPSO and HCLPSO show the performance gain when dimension increases from 10 to 30, which further bears out that the proposed CPSOS algorithm has good scalability.

6. Conclusions and future work

In this paper, we propose a chaotic particle swarm optimization with sigmoid-based acceleration coefficients. On one side, the frequently used logistic map is applied to generate well-distributed initial particles. On the other side, the sigmoid-based acceleration coefficients is formulated to balance the global search ability in the early stage and the global convergence in the latter stage. In particular, two sets of slowly varying function and regular varying function embedded update mechanism as well as the chaos based re-initialization and Gaussian mutation strategies are used at different evolution stages to update the particles, which can effectively keep the diversity of the swarm and get away from possible local optima to continue exploring the potential search regions of the solution space. Conducted experiments reveal that the proposed CPSOS outperforms several competitive PSO variants regarding their convergence rate and solution accuracy in function optimization tasks.

As for future work, CPSOS will be compared with more PSO algorithms in the task of complex multi-optima and multi-objective problems, especially in some real-world engineering applications. More importantly, we plan to apply the chaos-based swarm initialization and the two sets of SVF and RVF embedded update mechanism to the state-of-the-art PSO variants in the future research. In the meanwhile, we also intend to delve deeper into parallelization of CPSOS for large-scale optimization problems and exploring the use of different acceleration coefficients in different scenarios simultaneously, especially for the adequate parameter tuning in a wide range of problems. Last but not the least, the qualitative relationships between the chaos-based initialization and the PSO's stability as well as the acceleration coefficients and the convergence rate,

Table 9
Statistical analysis of Wilcoxon test between CPSOS and its competitors.

Dim.	Item	GPSO	OLPSO-L	DMPPSO-G	SRPSO	HCLPSO	EPSO	IDE-PSO
10	Better	27	21	26	20	17	15	16
	Same	0	1	0	2	2	4	4
	Worse	1	6	2	6	9	9	8
	Merit	26	15	24	14	8	6	8
30	Better	28	20	24	25	19	15	15
	Same	0	0	0	2	5	4	3
	Worse	0	8	4	1	4	9	10
	Merit	28	12	20	24	15	6	5

from the viewpoint of mathematics, will be elaborated and proved comprehensively.

Acknowledgements

The authors would like to sincerely thank the editor and anonymous reviewers for their valuable comments and insightful suggestions that have helped us to improve the paper. In addition, this work is fully supported by the National Program on Key Basic Research Project (973 Program) (No.2013CB329502), the National Natural Science Foundation of China (No.61971005), the Tianchenghuizhi Fund for Innovation and Promotion of Education (No.2018A03036) and the Key R&D Program of the Shaanxi Province of China (No.2018GY-037).

References

[1] J. Agrawal, S. Agrawal, Acceleration based particle swarm optimization for graph coloring problem, Procedia Computer Science 60 (2015) 714–721.
[2] G. Ardizzon, G. Cavazzini, G. Pavesi, Adaptive acceleration coefficients for a new search diversification strategy in particle swarm optimization algorithms, Inf. Sci. 299 (2015) 337–378.
[3] M. Arumugam, A. Chandramohan, M. Rao, Competitive approaches to PSO algorithms via new acceleration coefficient variant with mutation operators, in: Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, 2005, pp. 225–230.
[4] K. Bharti, P. Singh, Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering, Appl. Soft Comput. 43 (2016) 20–34.
[5] A. Carlisle, G. Dozier, An off-the-shelf PSO, in: Proceedings of the Workshop on Particle Swarm Optimization, 2001, pp. 1–6.
[6] K. Chen, F. Zhou, L. Yin, et al., A hybrid particle swarm optimizer with sine cosine acceleration coefficients, Inf. Sci. 422 (2018) 218–241.
[7] K. Chen, F. Zhou, Y. Wang, et al., An ameliorated particle swarm optimizer for solving numerical optimization problems, Appl. Soft Comput. 73 (2018) 482–496.
[8] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[9] P. Das, H. Behera, B. Panigrahi, A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning, Swarm and Evolutionary Computation 28 (2016) 14–28.
[10] N. Dong, C. Wu, W. Ip, et al., An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection, Comput. Math. Appl. 64 (2012) 1886–1902.
[11] J. Galambos, E. Seneta, Regularly varying sequences, in: Proceedings of the American Mathematical Society, 1973, pp. 110–116.
[12] W. Gao, S. Liu, L. Huang, Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique, Commun. Nonlinear Sci. Numer. Simul. 17 (11) (2012) 4316–4327.
[13] J. Gou, Y. Lei, W. Guo, et al., A novel improved particle swarm optimization algorithm based on individual difference evolution, Appl. Soft Comput. 57 (2017) 468–481.
[14] H. Haklı, H. Uğuz, A novel particle swarm optimization algorithm with Levy flight, Appl. Soft Comput. 23 (2014) 333–345.
[15] K. Harrison, A. Engelbrecht, B. Ombuki-Berman, Optimal parameter regions and the time-dependence of control parameter values for the particle swarm optimization algorithm, Swarm and Evolutionary Computation 41 (2018) 20–35.
[16] H. Illias, X. Chai, A. Bakar, Hybrid modified evolutionary particle swarm optimisation-time varying acceleration coefficient-artificial neural network for power transformer fault diagnosis, Measurement 90 (2016) 94–102.
[17] F. Javidrad, M. Nazari, A new hybrid particle swarm and simulated annealing stochastic optimization method, Appl. Soft Comput. 60 (2017) 634–654.
[18] A. Jordehi, Time varying acceleration coefficients particle swarm optimisation (TVACPSO): a new optimisation algorithm for estimating parameters of PV cells and modules, Energy Convers. Manag. 129 (2016) 262–274.
[19] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the International Conference on Neural Networks, 1995, pp. 1942–1948.
[20] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the Congress on Evolutionary Computation, 2002, pp. 1671–1676.
[21] R. Kundu, S. Das, R. Mukherjee, et al., An improved particle swarm optimizer with difference mean based perturbation, Neurocomputing 129 (2014) 315–333.
[22] H. Liang, F. Kang, Adaptive mutation particle swarm algorithm with dynamic nonlinear changed inertia weight, Optik 127 (19) (2016) 8036–8042.
[23] J. Liang, A. Qin, P. Suganthan, et al., Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[24] J. Liang, B. Qu, P. Suganthan, et al., Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, 2013.
[25] J. Liang, P. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: IEEE Symposium on Swarm Intelligence, 2005, pp. 124–129.
[26] F. Lu, L. Gao, Novel optimization mechanism based on improved particle swarm optimization, J. Northeast. Univ. (Nat. Sci.) 32 (9) (2011) 1221–1224.
[27] N. Lynn, P. Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm and Evolutionary Computation 24 (2015) 11–24.
[28] N. Lynn, P. Suganthan, Ensemble particle swarm optimizer, Appl. Soft Comput. 55 (2017) 533–548.
[29] N. Lynn, M. Ali, P. Suganthan, Population topologies for particle swarm optimization and differential evolution, Swarm and Evolutionary Computation 39 (2018) 24–35.
[30] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (3) (2004) 204–210.
[31] A. Nickabadi, M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, Appl. Soft Comput. 11 (2011) 3658–3670.
[32] Y. Ortakci, Comparison of Particle Swarm Optimization Methods in Applications, Graduate School of Natural and Applied Sciences, Karabuk University, 2011.
[33] M. Pluhacek, R. Senkerik, D. Davendra, Chaos particle swarm optimization with ensemble of chaotic systems, Swarm and Evolutionary Computation 25 (2015) 29–35.
[34] A. Ratnaweera, S. Halgamuge, H. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.
[35] F. Salajegheh, E. Salajegheh, PSOG: enhanced particle swarm optimization by a unit vector of first and second order gradient directions, Swarm and Evolutionary Computation 46 (2019) 28–51.
[36] J. Ser, E. Osaba, D. Molina, et al., Bio-inspired computation: where we stand and what's next, Swarm and Evolutionary Computation 48 (2019) 220–250.
[37] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of the Congress on Evolutionary Computation, 1998, pp. 69–73.
[38] H. Soleimani, G. Kannan, A hybrid particle swarm optimization and genetic algorithm for closed-loop supply chain network design in large-scale networks, Appl. Math. Model. 39 (14) (2015) 3990–4012.
[39] P. Suganthan, Particle swarm optimiser with neighbourhood operator, in: Proceedings of the Congress on Evolutionary Computation, 1999, pp. 1958–1962.
[40] M. Tanweer, S. Suresh, N. Sundararajan, Self-regulating particle swarm optimization algorithm, Inf. Sci. 294 (2015) 182–202.
[41] D. Tian, Particle swarm optimization with chaos-based initialization for numerical optimization, Intelligent Automation and Soft Computing 24 (2) (2018) 331–342.
[42] D. Tian, Z. Shi, MPSO: modified particle swarm optimization and its applications, Swarm and Evolutionary Computation 41 (2018) 49–68.
[43] C. Tsai, I. Kao, Particle swarm optimization with selective particle regeneration for data clustering, Expert Syst. Appl. 38 (6) (2011) 6565–6576.
[44] Z. Wan, G. Wang, B. Sun, A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems, Swarm and Evolutionary Computation 8 (2013) 26–32.
[45] H. Wang, W. Wang, Z. Wu, Particle swarm optimization with adaptive mutation for multimodal optimization, Appl. Math. Comput. 221 (2013) 296–305.
[46] H. Wang, H. Sun, C. Li, et al., Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. 223 (2013) 119–135.
[47] X. Xia, L. Gui, Z. Zhan, A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting, Appl. Soft Comput. 67 (2018) 126–140.
[48] T. Yamaguchi, K. Yasuda, Adaptive particle swarm optimization: self-coordinating mechanism with updating information, in: Proceedings of the International Conference on Systems, Man and Cybernetics, 2006, pp. 2303–2308.
[49] D. Yousri, D. Allam, M. Eteiba, et al., Static and dynamic photovoltaic models' parameters identification using chaotic heterogeneous comprehensive learning particle swarm optimizer variants, Energy Convers. Manag. 182 (2019) 546–563.
[50] Z. Zhan, J. Zhang, Y. Li, et al., Adaptive particle swarm optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (6) (2009) 1362–1381.
[51] Z. Zhan, J. Zhang, Y. Li, et al., Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Comput. 15 (6) (2011) 832–847.