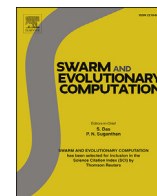




Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

MPSO: Modified particle swarm optimization and its applications

Dongping Tian^{a,*}, 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
Maximal focus distance
Inertial weight
Premature convergence
Local optima
Logistic map
Wavelet mutation

ABSTRACT

Particle swarm optimization (PSO) is a population based meta-heuristic search algorithm that has been widely applied to a variety of problems since its advent. In PSO, 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. However, PSO is easily trapped into the local optima and premature convergence appears when applied to complex multimodal problems. To address these issues, we present a modified particle swarm optimization with chaos-based initialization and robust update mechanisms. On the one side, the Logistic map is utilized to generate uniformly distributed particles to improve the quality of the initial population. On the other side, the sigmoid-like inertia weight is formulated to make the PSO adaptively adopt the inertia weight between linearly decreasing and nonlinearly decreasing strategies in order to achieve better tradeoff between the exploration and exploitation. During this process, a maximal focus distance is formulated to measure the particle's aggregation degree. At the same time, the wavelet mutation is applied for the particles whose fitness value is less than that of the average so as to enhance the swarm diversity. In addition, an auxiliary velocity-position update mechanism is exclusively applied to the global best particle that can effectively guarantee the convergence of MPSO. Extensive experiments on CEC'13/15 test suites and in the task of standard image segmentation validate the effectiveness and efficiency of the MPSO algorithm proposed in this paper.

1. Introduction

Inspired by social behavior observed in nature, such as schools of fish, flocks of birds, swarms of bees, and even human social behavior, particle swarm optimization was first introduced in 1995 for the task of optimization of continuous nonlinear functions [31]. PSO is similar to other population based evolutionary algorithms (EAs) [78] in that it is initialized with a population of random solutions (here refers to the positions of each particle), such as genetic algorithm (GA) [29], ant colony optimization (ACO) [21], firefly algorithm (FA) [79,82,84,86] and cuckoo search (CS) [16,97], etc. It is unlike most of other population based evolutionary algorithms, however, in that PSO is motivated by the simulation of social behavior instead of survival of the fittest, and each candidate solution is associated with a velocity rather than the evolutionary operators like selection, crossover and mutation. Compared with other EAs, it's obvious that PSO has the advantages of fewer control parameters, better convergence and easy implementation. Due to these desirable merits, PSO has attracted much attention worldwide in the field of evolutionary computation since its advent. At present, it becomes one

of the most preferred choices for optimization problems and has been extensively applied to a wide range of application areas such as neural networks [8,36], engineering design [18,65], optimal scheduling [69,76], and so on. However, similar to other nature-inspired evolutionary algorithms, PSO also suffers from the bane of premature convergence and being trapped into local optima when solving complex multimodal problems [27]. Based on this recognition, a huge amount of particle swarm optimization variants have been proposed to deal with these issues. As the representative work, HCLPSO was developed in Ref. [47] which adopted the comprehensive learning particle swarm optimizer [40] to enhance the exploration and utilized the global version of PSO to enhance the exploitation. Subsequently, the ensemble PSO (EPSO) [48] was proposed to solve real-parameter optimization problems by integrating the merits of a pool of particle swarm optimization strategies. Experiments on CEC'05 bore out that the superiority of the EPSO algorithm. From the literature, it can be clearly observed that most of the current existing PSOs can be roughly divided into four categories: swarm initialization, parameter selection, topology structure and hybrid versions respectively.

* Corresponding author.

E-mail addresses: tdp211@163.com, tiandp@ics.ict.ac.cn (D. Tian).<https://doi.org/10.1016/j.swevo.2018.01.011>

Received 30 August 2017; Received in revised form 28 January 2018; Accepted 28 January 2018

Available online xxx

2210-6502/© 2018 Elsevier B.V. All rights reserved.

- **Swarm initialization.** Like other swarm based stochastic optimization algorithm, PSO is initialized with a population of random solutions (position of each particle) in the search space, and subsequently begins to enter a loop in order to continue to search for optimal solutions by updating the particle's velocities and positions until some termination conditions are satisfied. The early notable work [74] used two kinds of chaotic maps to attempt to improve the quality of initial population for PSO with promising results. Subsequent work [26] came up with a similar chaotic opposition-based population initialization instead of the purely random strategy to improve PSO performance. In Ref. [37], two kinds of chaotic maps have been applied to initialize the swarm in which Logistic map was for positions while Cubic map was for velocities of the particles. Note that both of the two PSOs, to some extent, can achieve certain success compared to the PSO with usual random initialization in the same conditions. Particularly, the recent work of Tian et al. [75] introduced chaotic map based initialization and Gaussian mutation as well as a local re-initialization strategies into the standard PSO. Extensive experiments on several well-known benchmark functions demonstrated its effectiveness. In the context of swarm initialization for PSO, there has been very little research on this topic. However, it is reported that PSO tends to the characteristics of low stability due to the non-uniformly distributed initial particles [28]. Moreover, it's obvious that the convergence speed of the particle swarm also depends on the initial population. So how to generate high-quality initial particles is a worthy research direction in the PSO field. This is one of the original intentions of this work, which will be discussed in detail in the following sections.
- **Parameter selection.** The proper selection of control parameters, such as inertia weight and acceleration coefficient, can significantly influence the convergence of PSO. Concerning the acceleration coefficient, it has been studied for many years because of its effect on the self and social cognitions for the convergence of particle swarm optimization [11,12,14,60,70]. As a pioneer work on this topic, Clerc et al. [14] introduced a constriction factor into the standard PSO that was a function of c_1 and c_2 to insure the convergence of particle swarm optimization. Subsequent work [60] put forward a self-organizing hierarchical particle swarm optimizer (HPSO) with time-varying acceleration coefficients (TVAC) to control the local search and convergence to the global optimum solution. Conducted experiments revealed that the performance of HPSO with TVAC was markedly better than that of HPSO with fixed acceleration coefficients ($c_1 = c_2 = 2$). In Ref. [70], a modified particle swarm optimization was brought forward by exploiting the exponential time-varying acceleration coefficients. Besides, various acceleration coefficients [11, 12] have been formulated for PSO to improve its performance in recent years, and more details of them can be gleaned from the corresponding literature. Note that with regard to different inertia weight strategies, they will be comprehensively reviewed in the next section.
- **Topology structure.** To enhance the performance of PSO, different types of topology structures have been studied in the literature. In Ref. [53], a fully informed particle swarm (FIPS) was presented based on an information flow process to update the position of each particle. In FIPS, all members in the neighborhood could fairly offer their search information and the velocity adjustment was not only influenced by the best position in the particle's neighborhood but also by the positions in other neighborhoods. Followed by Liang et al. [43] constructed the DMSPSO by exploiting a dynamic neighborhood strategy rather than a fixed one to improve PSO, which involved a random selection of small swarms with small neighborhood in the early stage to provide better exploration and then dynamically increase the neighborhood by regrouping the swarms to incorporate social interaction and perform better exploitation in the later stage of the search process. In Ref. [51], a PSO with expanding neighborhood topology was developed by combining particle swarm optimization with variable neighborhood search to solve the well-known

constrained shortest path problem. In recent work [49], the fluid neural network was employed to create dynamic neighborhood topologies. As a result, the fluid neural network PSO was introduced with a dynamic neighborhood mechanism. Experiments indicated that this PSO outperformed the standard PSO algorithm and the other PSOs based on partially connected grid topologies. In addition, a hybrid topology scale free Gaussian-dynamic PSO was proposed for real power loss problem involving the fully connected topology and ring topology [80] simultaneously. Besides, a dynamic tournament topology strategy was also exploited to improve particle swarm optimization (DTT-PSO) in Ref. [85] apart from the other PSO variants [5,44]. In sum, a suitable topology has been shown to be able to effectively improve the performance of PSO.

- **Hybrid versions.** It is expected that the performance of PSO can be improved by integrating it with other search techniques, such as chaos search [59], differential evolution (DE) [64], genetic algorithm (GA) [63], simulated annealing (SA) [68], and neighborhood search [81]. The work of [19] combined a quantum-behavior PSO with the simplex algorithm to solve the load flow problem. In literature [38], a hybrid PSO with artificial bee colony (ABC) was proposed (PS-ABC). Simulation results on 13 high-dimensional benchmark functions validated that PS-ABC had the ability to accelerate the convergence and avoid the local optima. Besides, notice that integrating PSO with other evolutionary paradigms like selection [1], crossover [54] and mutation [83] has become a popular research topic in the community of particle swarm optimization in recent years. Since both PSO and evolutionary algorithms (evolution strategy, evolution programming, GA and genetic programming) are based on population, as a result such hybridization can be readily formulated. This is a desirable strategy to achieve better tradeoff between exploration and exploitation as well as to prevent stagnation and premature convergence by harnessing the strengths of each of the components in the corresponding algorithm.

Despite the PSO variants mentioned above belonging to different categories, most of them can achieve encouraging performance and motivate us to better explore PSO methods with the help of their excellent experiences and knowledge. From the literature, it is clearly observed that researchers have paid less attention to the topic on swarm initialization even though it has a direct effect on the performance of PSO. Besides, most existing inertia weights behave either linear or non-linear to attempt to keep the balance between exploration and exploitation. Integrating the characteristics of linear and non-linearly inertia weight together seems to be more efficient for PSO to enhance its search ability and keep fast convergence. Finally, although some PSO variants propose to conduct search process in the context of dynamic environment, they usually lack a certain measure strategy to be followed. To this end, a modified PSO with chaos-based initialization and robust update mechanism is proposed in this paper. On the one hand, the Logistic map is utilized to generate uniformly distributed particles to improve the quality of the initial population. On the other hand, the sigmoid-like inertia weight is formulated to make the PSO adaptively adopt the inertia weight between the linear decreasing and nonlinear decreasing strategies based on the maximal focus distance, which is able to effectively prevent the PSO from plunging into local optima and make the particles proceed with searching in other regions of the solution space. At the same time, the wavelet mutation is applied for the particles whose fitness value is less than that of the average so as to enhance the population diversity. In addition, an auxiliary velocity-position update mechanism is introduced exclusively for the global best particle to ensure the convergence of MPSO. Extensive experiments bear out the effectiveness and efficiency of the proposed PSO algorithm.

The rest of this paper is organized as follows. Section 2 introduces some background and related work, especially various kinds of inertia weight from the aspects of the linear, nonlinear, fuzzy rules, random and other strategies respectively. Section 3 elaborates the MPSO algorithm, including the chaos-based swarm initialization, formulated sigmoid-like

inertia weight, maximum focus distance, exclusive update strategy and the wavelet mutation respectively. Experiments on CEC'13/15 test suites and in the task of standard image segmentation are reported in Section 4. Finally, we end this paper with some important conclusions and future work in Section 5.

2. Background and related work

2.1. Standard PSO

Particle swarm optimization is a population-based stochastic optimization algorithm. It is known that PSO maintains two populations: a population of particle's current positions (i.e. $pbest$) and a population of particle's best positions (i.e. $gbest$) achieved to date. The former is regarded as the candidate solutions in the search space while the latter is used to guide the former's update. In PSO system, each particle is associated with two properties (velocity vector V and position vector X) and it moves in the search space with a velocity that is dynamically adjusted according to the particle's experience and the particles companion's experience simultaneously. Mathematically, the velocity and position of the particles are updated according to the following formula:

$$v_{id}(t+1) = \omega \times v_{id}(t) + c_1 \times r_1 \times [p_{id}(t) - x_{id}(t)] + c_2 \times r_2 \times [p_{gd}(t) - x_{id}(t)] \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

where c_1 and c_2 are acceleration coefficients reflecting the weight of the stochastic acceleration terms that pull each particle toward $pbest$ and $gbest$ positions respectively. r_1 and r_2 denote two random numbers uniformly distributed in the range (0,1). It has characteristics that are reminiscent of the temperature parameter in the simulated annealing (SA). ω is the inertia weight used for balancing the global and local search. In general, a large inertia weight facilitates the global exploration while a small inertia weight tends to facilitate the local exploitation. The position of the i th particle can be represented by a D -dimensional vector $X_i = [x_{i1}, x_{i2}, \dots, x_{ij}, \dots, x_{iD}]$ where $x_{ij} \in [x_{min}, x_{max}]$ denotes the position of the j th dimension of the i th particle, and the corresponding velocity is $V_i = [v_{i1}, v_{i2}, \dots, v_{ij}, \dots, v_{iD}]$ where $v_{ij} \in [v_{min}, v_{max}]$ is used to reduce the likelihood of the particles leaving the search space. The best previous position (the position giving the best fitness value) of the i th particle is recorded $pbest$ and denoted by $P_i = (p_{i1}, p_{i2}, \dots, p_{ij}, \dots, p_{iD})$, while the global best position of the whole swarm achieved so far is recorded $gbest$ and indicated as $P_g = (p_{g1}, p_{g2}, \dots, p_{gj}, \dots, p_{gD})$. The pseudocode of the standard PSO algorithm can be succinctly described as follows:

Algorithm 1: Pseudocode of the standard PSO algorithm

1. **Begin**
 2. Randomly initialize particle swarm
 3. **while** (number of iterations or the stopping criterion is not met)
 4. Evaluate fitness of the particle swarm
 5. **for** $n=1$ to number of particles
 6. Find $pbest$
 7. Find $gbest$
 8. **for** $d=1$ to number of dimensions of particle
 9. Update the velocity of particles by Eq.(1)
 10. Update the position of particles by Eq.(2)
 11. **next** d
 12. **next** n
 13. Update the inertia weight by some corresponding strategy
 14. **next generation until stopping criterion**
 15. **End**
-

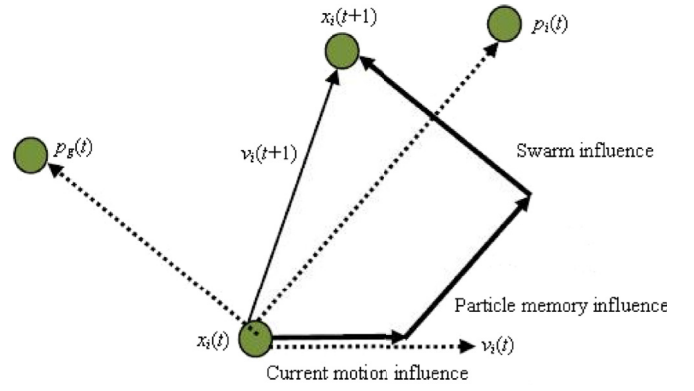


Fig. 1. Graphical representation of the particle evolution.

Fig. 1 illustrates the graphical representation of the particle evolution in PSO system.

2.2. Review of different inertia weights

As is well known, inertia weight plays an important role in controlling the process of exploration (global search) and exploitation (local search) by maintaining a balance in their capabilities. From the perspective of statistical analysis, it is believed that the overall performance of PSO is strongly affected by the inertia weight [58]. In view of this, several kinds of inertia weight will be comprehensively reviewed in this section, including the linear, nonlinear, fuzzy rules, random, and other strategies based inertia weights. The ultimate goal is to gain the comparative analysis and understanding of the merits and demerits of each inertia weight so as to formulate more effective strategies for PSO algorithm. In addition, it should be noted that for the sake of clarity and consistency, the notations t and t_{max} appeared in this section denote the current iteration and the maximum allowed number of iterations respectively. $\omega(t)$ is used to represent the inertia weight of the t th iteration.

2.2.1. Linear strategies to adjust inertia weight

It is known that the tradeoff between global and local search during the evolution is critical to the success of an optimization algorithm. Considering this, the inertia weight was initially set as a constant (such as 0.4) during the search process in the early years of PSO research, but the results illustrated that a constant inertia weight can hardly work due to the failure of balancing exploration and exploitation. In the meanwhile, it was found that a large inertia weight facilitates a global search while a small inertia weight facilitates a local search. As a consequence, a large number of inertia weights have been developed in the area of PSO research. In Ref. [22], a linear decreasing inertia weight was introduced and shown to be effective in improving the fine-tuning characteristic of PSO. In this method, the value of ω is linearly decreased from an initial value ω_{max} to a final value ω_{min} as the number of iterations increases according to the following equation:

$$\omega(t) = \frac{t_{max} - t}{t_{max}} (\omega_{max} - \omega_{min}) + \omega_{min} \quad (3)$$

As observed from the literature, this strategy based inertia weight has been widely applied in the field of particle swarm optimization researches.

In subsequent works [17,92,100], the inertial weight was dynamically adjusted via increasing or decreasing mechanism. In the case of the increasing, initially a small value of inertia weight increased linearly or nonlinearly to be a larger value. On the contrary, a larger value of inertia weight decreased linearly or nonlinearly to be a small one in the decreasing situation. Specifically, Zheng et al. [100] constructed an increasing inertia weight defined by Eq. (4) and confirmed its validity in

terms of the convergence speed and solution precision. Experiments showed that PSO with the increasing inertia weight (increasing from 0.4 to 0.9) outperforms that with the decreasing inertia weight in all benchmark functions used in their tests.

$$\omega(t) = 0.5 \times \frac{t}{t_{\max}} + 0.4 \quad (4)$$

Enlightened by the pros and cons of the increasing and decreasing strategies, an inertia weight that first increased and then decreased has been proposed by Ref. [17], in which the value of inertia weight along the line of linearly increased from 0.4 to 0.9, and then linearly decreased to 0.4 again.

$$\omega(t) = \begin{cases} 1 \times \frac{t}{t_{\max}} + 0.4, & 0 \leq \frac{t}{t_{\max}} \leq 0.5 \\ -1 \times \frac{t}{t_{\max}} + 1.4, & 0.5 < \frac{t}{t_{\max}} \leq 1 \end{cases} \quad (5)$$

In addition, a linear function relationship between inertia weight and the average distance amongst points was established through analyzing the dynamic relationship between the inertia weight and the population diversity [92].

$$\omega(t) = \frac{\omega_{\max} - \omega_{\min}}{D_{\max} - D_{\min}} \times D(t) + \frac{D_{\max}\omega_{\min} - D_{\min}\omega_{\max}}{D_{\max} - D_{\min}} \quad (6)$$

where $D(t)$ denotes the average distance amongst particles and $D_{\min} \leq D(t) \leq D_{\max}$, $\omega_{\min} \leq \omega(t) \leq \omega_{\max}$.

2.2.2. Nonlinear strategies to adjust inertia weight

Inspired by the basic idea of the decreasing inertia weight, Chen et al. [9] proposed two natural exponent inertia weights (as shown in Table 1). Experiments validated that PSO with these two strategies was able to converge faster than that with the linear ones during the early period of

the search process. In the meantime, they came up with another group of three nonlinear decreasing strategies to adjust inertia weight [10], including a parabola opening upwards, a parabola opening downwards and an exponential curve respectively. Simulation results showed that for most continuous optimization problems, the performance of the concave function based decreasing inertia weight surpasses that of the linear strategy, and the linear strategy is superior to the convex function based strategy.

Subsequently, a sigmoid increasing inertia weight was developed by combining the sigmoid function with the linear increasing inertia weight [50] to produce a great improvement in quick convergence and aggressive movement narrowing towards the solution space. Feng et al. [25] made use of a chaotic model (Logistic map) of inertia weight in which a chaotic term was introduced into the linear decreasing inertia weight to improve the performance of PSO. Note that in Ref. [93], a PSO with an adaptive inertia weight was presented for designing IIR digital filter. In this PSO, the modified Versoria function was employed in the new relation of the adaptive inertia weight factor function instead of the commonly used sigmoid function for avoiding the exponential computation and ensuring the small final misadjustment. In addition, a group of nonlinear strategies with multi-stage linear decreasing inertia weight (MLDW) [88] have been put forward for the purpose of easily refining the decreasing process of the inertia weight. A recent work by Tanweer et al. [71] proposed a self regulating particle swarm optimization (SRPSO) that incorporated the self-regulating inertia weight determined by the best particle for better exploration and the self-perception on the global search direction determined by the rest particles for exploitation in the search space. Experiments confirmed that SRPSO has the capability of achieving faster convergence and better solutions in most of the problems. All in all, these nonlinear inertia weights mentioned above were shown to be able to improve the search ability of PSO to some extent, but they still struggled to obtain a good balance between the global convergence and the convergent efficiency. Table 1 summarizes several

Table 1
Summary of nonlinear inertia weights mentioned in this subsection.

Literature	Different inertia weights	Related description
Ref. [9]	$\omega(t) = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \cdot e^{-t/\left(\frac{t_{\max}}{10}\right)}$	–
	$\omega(t) = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \cdot e^{-\left[t/\left(\frac{t_{\max}}{10}\right)\right]^2}$	–
Ref. [10]	$\omega(t) = -(\omega_{\text{start}} - \omega_{\text{end}}) \cdot \left(\frac{t}{t_{\max}}\right)^2 + \omega_{\text{start}}$	convex function based ω
	$\omega(t) = (\omega_{\text{start}} - \omega_{\text{end}}) \cdot \left(\frac{t}{t_{\max}}\right)^2 + (\omega_{\text{end}} - \omega_{\text{start}}) \cdot \left(\frac{2t}{t_{\max}}\right) + \omega_{\text{start}}$	concave function based ω
	$\omega(t) = \omega_{\text{end}} \left(\frac{\omega_{\text{start}}}{\omega_{\text{end}}}\right)^{1/(1+c_3 t/t_{\max})}$	exponential function based ω
Ref. [25]	$\omega(t) = \frac{t_{\max}-t}{t_{\max}} (\omega_{\max} - \omega_{\min}) + \omega_{\min} \cdot z$	z : formula of Logistic map
Ref. [30]	$\omega(t) = \omega_{\text{start}} \times u^t$	$u \in [1.0001, 1.005]$
Ref. [37]	$\omega(t) = \left\{ \frac{(t_{\max}-t)^n}{(t_{\max})^n} \right\} (\omega_{\text{start}} - \omega_{\text{end}}) + \omega_{\text{end}}$	n : nonlinear modulation index
Ref. [39]	$\omega(t) = (\omega_{\text{start}} - \omega_{\text{end}}) \cdot \tan\left\{ 0.875^* \left[1 - \left(\frac{t}{t_{\max}}\right)^{k_1} \right] \right\} + \omega_{\text{end}}$	$k_1 = 0.6$
	$\omega(t) = (\omega_{\text{start}} - \omega_{\text{end}}) \cdot \arctan\left\{ 1.56^* \left[1 - \left(\frac{t}{t_{\max}}\right)^{k_2} \right] \right\} + \omega_{\text{end}}$	$k_2 = 0.4$
Ref. [50]	$\omega(t) = \frac{\omega_{\max} - \omega_{\min}}{1 + e^{-\frac{t}{t_{\max}} \cdot (\omega_{\max} - \omega_{\min})}} + \omega_{\min}$	$u = 10^{(\log t_{\max} - 2)}$
Ref. [71]	$\omega(t) = \omega_i(t) - \Delta\omega$ $\omega(t) = \omega_i(t) + \eta\Delta\omega$	$\Delta\omega = \frac{\omega_{\text{start}} - \omega_{\text{end}}}{N_t}$; η : a constant
Ref. [88]	$\omega(t) = \begin{cases} \frac{(\omega_s - \omega_m)(t_1 - t)}{t_1} + \omega_m & 0 \leq t \leq t_1 \\ \omega_m & t_1 < t \leq t_2 \\ \frac{(\omega_m - \omega_e)(t_{\max} - t)}{t_{\max} - t_2} + \omega_e & t_2 < t \leq t_{\max} \end{cases}$	$\omega_s, \omega_m, \omega_e, t_1$ and t_2 : parameters predefined

nonlinear inertia weights mentioned in this subsection.

2.2.3. Fuzzy rules to adjust inertia weight

As a pioneer work on this topic, Shi et al. [67] first constructed a two-input and one-output fuzzy logic controller (FLC) to improve the performance of PSO. The basic idea behind this fuzzy PSO (FPSO) algorithm is to adjust the inertia weight by applying adaptive FLC to dynamically optimize the inertia weight. To be specific, the two-input variables include the current best performance evaluation (*CBPE*) and the current inertia weight, while the only one-output variable is the change of inertia weight ($\Delta\omega$). The obtained results indicated that the performance of this FPSO can be satisfactory on many issues, but it is difficult to implement due to various factors.

In literature [46], a two-input and two-output FLC was developed. One of the input variables is the normalized *CBPE* whereas the other is the current velocity (CV) of particles. In addition, one of the output variables is ρ , the scaling factor to control the domain of the particle's oscillation, another is V_{ck} used to control the change of the velocity threshold according to the following Eq. (7):

$$V_c = e - [10(1 + V_{ck})] \quad (7)$$

Correspondingly, a new velocity update strategy was formulated as below:

$$V_{ij}(t+1) = \omega \hat{v} + c_1 r_1 (x_{ij}^{\#}(t) - x_{ij}(t)) + c_2 r_2 (x_{ij}^*(t) - x_{ij}(t)) \quad (8)$$

$$\hat{v} = \begin{cases} v_{ij}, & \text{if } |v_{ij}| \geq v_c \\ u(-1, 1)v_{max}/\rho, & \text{if } |v_{ij}| < v_c \end{cases} \quad (9)$$

where $u(-1, 1)$ is the random number uniformly distributed in the interval $[-1, 1]$. v_c is the minimum velocity threshold, a tunable threshold parameter to limit the minimum of the particle's velocity. Through numerical experiment, it validated that the performance of the FPSO does not degrade drastically as the problem dimension increases.

Subsequent work [89] presented a two-input (t and $\Delta v(t)$) and one-output (ω) FLC based particle swarm optimization. In our previous studies [72,73], two fuzzy PSOs were developed based on the two-input and two-output FLC, in which the increment of global optimum (*IGO*) in successive generations and deviation (*Dev*) of particle fitness values as well as the fitness variance (*Delt*) and mean extremal deviation (*Total*) are considered as the input parameters of FLC respectively, while the two-output variables in both methods include the inertia weight and the acceleration coefficients. Simulation results revealed that the increase of the problem dimension cannot significantly deteriorate the performance of these fuzzy PSOs. Besides, in the work of [61], a balanced fuzzy particle swarm optimization (BF-PSO) was put forward to solve the fundamental optimization problem entitled traveling salesman problem.

Table 2
Summary of fuzzy rules based inertia weights mentioned in this subsection.

Literature	Input variables	Output variables	Related description
Ref. [46]	normalized CBPE CV	ρ, V_{ck}	–
Ref. [67]	CBPE current inertia weight	$\Delta\omega$	–
Ref. [72]	$IGO = p_g(t-1) - p_g(t)$ $Dev = \sqrt{\frac{1}{N} \sum_{i=1}^N (f_i - f_{avg})^2}$	ω, c_1, c_2	f_i : fitness value of the i th particle f_{avg} : average fitness value of the swarm $p_g(t)$: global optimum in t th iteration
Ref. [73]	$Delt = \frac{1}{N} \sum_{i=1}^N (f_i - f_{ave})^2$ $Total = \frac{1}{Pop} \sum_{i=1}^{Pop} (pbest - gbest)^2$	ω, c_1, c_2	N : number of particles Pop : swarm size $pbest$: individual extremum $gbest$: global extremum
Ref. [89]	current iteration $\Delta v_{av}(t) = v_{av}(t) - v_{av}(t-1) $	ω	$v_{av}(t) = \frac{1}{mD} \sum_m \sum_D v_{id}$

At the same time, a fuzzy logic based multi-objective PSO was developed to efficiently solve the distributed local area networks topology design problem [33]. Especially in the more recent work [55], to make up for the drawbacks of the trapping into local optima and the premature convergence, a fuzzy adaptive informed particle swarm optimization (FAIPSO) was formulated based on six-input and two-output FLC with ten fuzzy rules. Note that all the FPSO methods mentioned above possess respective advantages and disadvantages, and their common goal is to dynamically adjust the control parameters of PSO during the search process so as to achieve better optimization performance. Table 2 summarizes the fuzzy rules based inertia weights described in this subsection.

2.2.4. Random strategies to adjust inertia weight

Considering the dynamic nature of the most real-world applications, a random inertia weight was proposed for PSO to track the optima in dynamic systems [23]. To be specific, the inertia weight was set to change randomly according to Eq. (10).

$$\omega = 0.5 + rand(\cdot)/2 \quad (10)$$

where $rand(\cdot)$ denotes a random number uniformly distributed within the range $[0, 1]$.

Alternatively, it is difficult to predict whether in a given time the exploration or exploitation would be better in the dynamic environment. So a random value of the inertia weight is selected to address this problem. Note that when the random inertia weight is employed the acceleration coefficients are generally kept constant at 1.494 that coincides well with literature [14]. In view of this, the random strategy was widely applied by the other researches [98,99]. Among these, a strategy developed in literature [98] could tune the expectations of the inertia weight adaptively when they are selected randomly and thus lead to effective balance between the global and local search abilities.

$$\begin{cases} \omega = \alpha_1 + r/2.0, & k \geq 0.05 \\ \omega = \alpha_2 + r/2.0, & k < 0.05 \end{cases} \quad (11)$$

where $k = (f(t) - f(t-10))/f(t-10)$ denotes the change rate of the optimal adaptive value, r is a random number uniformly distributed in the range $[0, 1]$. Let $\alpha_1 > \alpha_2$, if $k \geq 0.05$ then the expectation $E(\omega) = \alpha_1 + 0.25$, otherwise $E(\omega) = \alpha_2 + 0.25$. Note that the expectation value of ω varies adaptively with the change rate of the optimal adaptive value. As a result the balance between the global search and local search can be flexibly adjusted.

In recent work [99], a simplified PSO was developed based on the stochastic inertia weight. It is clearly seen that this variant removes the velocity parameter and obtains the inertia weight by means of the random distribution to enhance the global and local search abilities of PSO algorithm. Meanwhile, the learning coefficients are based on the asynchronous change strategy to improve the search ability of particles.

Table 3
Adaptive inertia weight constructed in literature [24].

The value of ω_i	–	The value of $ F_i / V_i $		
	–	small	middle	large
The directions of \vec{V}_i and \vec{F}_i	same	small	middle	large
	opposite	large	middle	small

$$\omega = m_{\min} + (\mu_{\max} - \mu_{\min}) \times rand() + \sigma \times randn() \quad (12)$$

here μ_{\min} and μ_{\max} are the minimum and maximum of the random inertia weight, $rand()$ denotes a random number uniformly distributed in the interval $[0,1]$ while $randn()$ represents the normally distributed numbers. σ is used to measure the deviation degree between the random variable weight and its average value. To sum up, it has been identified experimentally that the random inertia weight has the advantage of rapid convergence in the early stage of evolution, and it is shown to be able to find fairly good solutions for most of the well-known benchmark functions.

2.2.5. Other strategies to adjust inertia weight

Different from previous proposals to improve PSO by utilizing an adaptive value of the inertia weight in each iteration, an adaptive inertia weight was expressed as the function of the evolution speed and aggregation degree of the swarm in Ref. [95]. Similarly, almost the same strategy was proposed by Ref. [90] in which inertia weight was given by a function of evolution speed and aggregation degree factors, and the value of inertia weight was dynamically adjusted according to the evolution speed and aggregation degree of particles. At the same time, Feng et al. [24] put forward an inertia weight depending on the particle's search states including its location and velocity instead of the iteration times. Table 3 describes the adaptive inertia weight strategy applied in this PSO algorithm. Note that many other PSO variants belonging to this category can be available in Refs. [56,91]. For more details and a more complete explanation on them, please refer to the corresponding literature.

3. Modified particle swarm optimization

In this section, the modified particle swarm optimization is discussed from five aspects of the chaos-based initialization, formulated sigmoid-like inertia weight, maximal focus distance, exclusive update strategy, and position mutation mechanism respectively. More details of them will be elaborated in sequence in the following subsections.

3.1. Chaos-based swarm initialization

As discussed in Section 1, to generate uniformly distributed initial particles in the search space plays a critical role in particle swarm optimization. From the literature, it can be observed that a huge number of chaos-based PSOs have been proposed [2,13,15,26,53,74,77]. Among these PSO variants, most of them can be roughly classified into three categories, that is, chaotic sequence based initialization for PSO [26,74,75], chaotic sequence based parameters update for PSO [2,13,15], and hybrid PSO and chaotic search techniques [20,53,77]. As the representative work of the first category, Tian et al. [74] exploited two kinds of chaos (Tent and Logistic map) to attempt to improve the quality of the initial population for standard PSO in 2010. Gao et al. [26] employed a similar chaotic opposition-based population initialization instead of a pure random initialization for PSO to improve its performance. Conducted experiments verified that these two PSO methods can achieve promising results compared to those with usual random initialization in the same conditions. Besides, it should be noted that both the standard PSO and various improved PSOs, such as HPSO [60], AEPsO [14] and other PSO variants, behave the characteristics of low stability. One of the main reasons, just as proved in literature [28], is that the initial population is non-uniformly distributed. He et al. [28] have just pointed out

the reasons of low stability for PSO, but no specific strategies were given to solve this problem. Based on this recognition, our recent work [75] attempted to deal with the foregoing issue by applying chaotic map based initialization for the standard particle swarm optimization. Extensive experiments demonstrated the merits of the PSO method. In sum, the application of chaotic sequence rather than random sequence in PSO is a powerful strategy to diversify the swarm of particles and improve the performance of PSO by preventing the premature convergence. In addition, as for the latter two kinds of chaos-based PSO algorithms, we will not go into much detail here since it is beyond the scope of our focus in this paper.

To summarize, most of these methods can achieve encouraging performance and motivate us to better explore PSOs with the help of their excellent experiences and knowledge. Without loss of generality, following the core idea of our prior work [75], the chaos-based initialization is exploited here to improve the quality of initial particles. As one of the simplest chaos, Logistic map [52] has been paid much attention by researchers over the last two decades. It can be described as follows:

$$x_{n+1} = f(\mu, x_n) = \mu x_n (1 - x_n), n = 0, 1, 2, \dots \quad (13)$$

where x_n represents the n th chaotic variable, $x_n \in (0, 1)$ under the conditions that the initial $x_0 \in (0, 1)$ except for some periodic fixed points $(0, 0.25, 0.5, 0.75, 1)$. μ is a predetermined constant, also called bifurcation coefficient. When μ increases from zero, the dynamic system generated by Eq. (13) will change from one fixed point to two, and until 2^n . During this process, a large number of multiple periodic components will locate in the 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. On the other hand, when μ is greater than 4, the whole system becomes unstable. Hence the range $[\mu_t, 4]$ is generally considered as the chaotic region of the whole system. Its bifurcation diagram is illustrated in Fig. 2.

Note that the key idea of Logistic map based initialization is to generate the same number of chaotic variables corresponding to the optimization problem. More specifically, when a preset number of chaotic iterations are executed, the chaotic variables will be generated accordingly. Afterwards remapping these variables into the optimization space, it will yield the real initial variables for the original optimization problem. Here, Eq. (13) is chosen as the chaotic signal generator, in which μ is set to be 4. As previously mentioned, the pseudocode of Logistic map can be described as below, which is able to generate uniformly distributed data sequence and avoid plunging into the small periodic cycles effectively.

Algorithm 2: Pseudocode of Logistic map for initialization

```

1. Begin
2. Randomly initialize chaotic variables
3. while (number of maximal iterations is not met)
4.   if chaotic variable plunges into fixed points or the small periodic cycles
5.     Implement a very small positive random perturbation
6.     Map them by Eq.(13)
7.   else
8.     Update the variables by Eq.(13) directly
9.   end
10. next generation until stopping criterion
11. Remap the chaotic variables into the problem space
12. End

```

To further illustrate the distribution performance of chaos-based initialization, Fig. 3 shows the histogram comparison of the Logistic map and random map for 3000 iterations in the range $[0,1]$ respectively. It should be noted that the histogram of Logistic map is depicted under the condition that its initial value and the number of iteration are set to

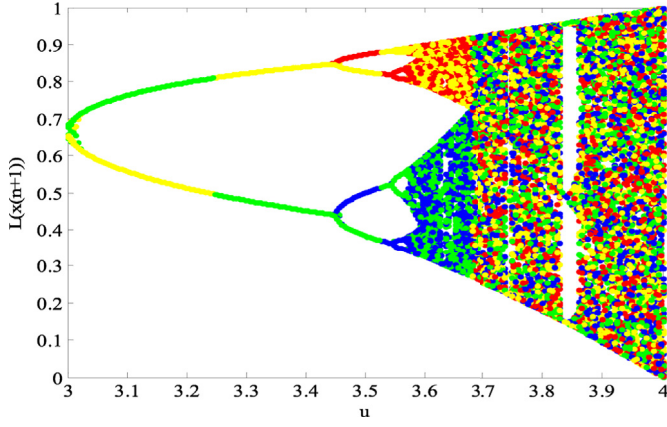
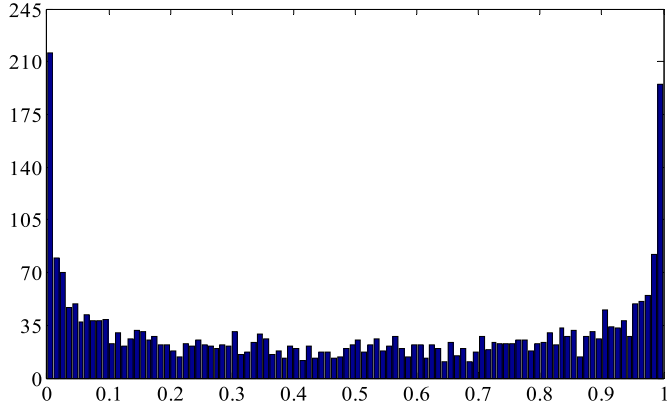
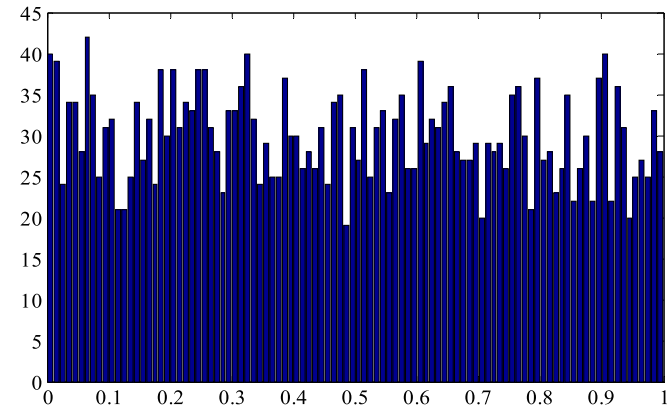


Fig. 2. Bifurcation diagram of Logistic map.

0.4567 and 3000 respectively. By comparing the histograms illustrated below, it can be clearly observed that in Logistic map, the maximal frequency is 216 while the minimal frequency is 11, corresponding to 42 and 19 in random map. In addition, the overall average frequency of Logistic map is about 30 in the range [0.2,0.8] whereas it approximates 27 between [0.3,0.5] for random map, which fully demonstrates that the Logistic map based initialization can yield more uniformly distributed particles in more wider ranges. On the other hand, it is obvious that the histogram trend of Logistic map is intuitively superior to that of random map. This can be easily validated by the empirical cumulative distribution function (ECDF) in case they are stochastically ordered. In sum, the



(a) Logistic map (max.216-min.11)



(b) Random map (max.42-min.19)

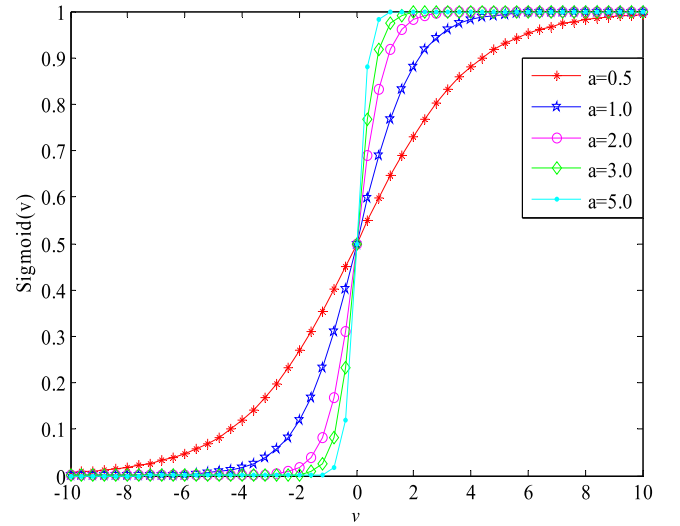
Fig. 3. Histograms of 3000 observations for Logistic and random maps.

Logistic map based initialization is able to generate more uniformly distributed particles in the allowable search space to enhance the stability of PSO algorithm. This is one of the main motivations for this study.

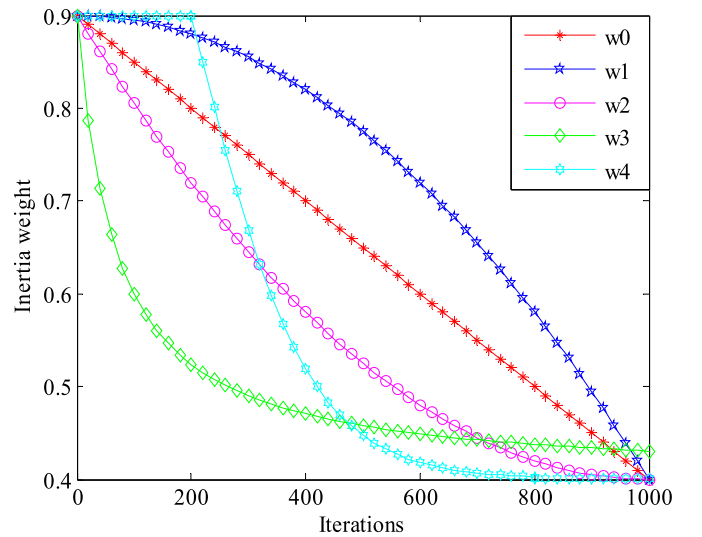
3.2. Formulated sigmoid-like inertia weight

As is known, the most common method for constructing neural activation functions is the sigmoid ($\varphi(v) = 1/(1 + \exp(-av))$) in the neural network, which is able to get excellent balance between linear and nonlinear behavior as displayed in Fig. 4(a). Note that $\varphi(v) = 0$ when $av < -9.903438$. On the contrary, $\varphi(v) = 1$ when $av \geq -9.903438$. Based on this recognition, a novel sigmoid-like inertia weight is formulated as follows. The key idea behind this strategy is to achieve the smooth transition from linear inertia weight to nonlinear inertia weight. It can be defined as below:

$$\omega(t) = \begin{cases} 0.9, & t \leq at_{max} \\ \frac{1}{1 + e^{(10t - 2t_{max})/t_{max}}} + 0.4, & otherwise \end{cases} \quad (14)$$



(a) Curves of sigmoid function



(b) Curves of different inertia weights

Fig. 4. Curves of sigmoid function and different inertia weights.

where t denotes the current iteration, t_{max} is the allowable maximal number of iterations, α is a constant predefined.

It should be noted that in the early stage of evolution $\omega = 0.9$ when $t \leq \alpha t_{max}$ while in the end stage ω approximates to 0.4 when t is equal to t_{max} . Except for these two cases, the values of ω calculated by Eq. (14) at any time during the process of evolution will be ranging from 0.4 to 0.9, which coincide well with the conclusions obtained in Ref. [22] because the performance of PSO can be significantly improved at the time of the inertia weight belonging to the interval [0.4, 0.95]. Fig. 4(b) illustrates several characteristic curves of the inertia weight with $t_{max} = 1000$ and $\alpha = 0.2$ for example, where $\omega_0, \omega_1, \omega_2, \omega_3$ and ω_4 denote the linear decreasing, convex function decreasing, concave function decreasing, exponential function decreasing and the sigmoid-like inertia weight formulated in this paper respectively. Compared with $\omega_0, \omega_1, \omega_2$ and ω_3 , it can be clearly observed that in the early stage of particles evolution, the inertia weight ω_4 keeps a larger value so as to speed up the search process. Quite the reverse, the inertia weight keeps a smaller value in the later stage in order to prevent the PSO from falling into local optima and make the particles proceed with searching in other regions of the solution space. The curve illustrated between these two cases behaves the smooth transition from the early stage to the end stage. Note that with respect to the superiority of the sigmoid-like inertia weight, it will be discussed in detail in the experiment section.

3.3. Maximal focus distance

As one of the inherent drawbacks of PSO, to suffer entrapment in local optima should be given priority to except for the premature convergence. Considering this, how to appropriately measure the swarm diversity or the aggregation degree of swarm plays a crucial role in balancing the exploration and exploitation for PSO. A recent work by Ruan et al. [62] exploited population density to estimate the particle's distribution in the search space by introducing the swarm size, the size of the solution space and a saturated population density respectively. In addition, an aggregation degree (AD) of particles [37] was defined to evaluate the particle's current state of the swarm as $AD = \left| \frac{\min(f_{best}(t), f_{avg}(t))}{\max(f_{best}(t), f_{avg}(t))} \right|$, in which $f_{best}(t)$ and $f_{avg}(t)$ denote the best and average fitness of particles in the t th iteration respectively, and in general, the larger the value is, the higher the aggregation degree. Different from the above-mentioned strategies, in this work, the maximal focus distance (MFD) is formulated to reflect the particle's aggregation degree so as to judge whether PSO algorithm gets stuck in the local optima or not:

$$MFD = \max_{i=1 \dots m} \left(\sqrt{\frac{\sum_{d=1}^D (p_{id} - x_{id})^2}{D}} \right) \quad (15)$$

where m is the number of neighborhood particles, p_{id} is the previous best position, and x_{id} denotes the sub-vector of the d th dimension of the i th particle in the search space.

According to the MFD calculated during the search process, the proposed particle swarm optimization is expected to be able to identify the forthcoming search strategy. That is, either to execute wavelet mutation for the particles whose fitness is less than or equal to the average fitness of the whole swarm, or to reinitialize the same number of particles by the Logistic map based on the average fitness of particles, or to directly update the velocity and position of particles by Eqs. (1)–(2) and (16)–(17) based on the newly formulated inertia weight respectively. By this way, the modified particle swarm optimization can be guaranteed to obtain the global optima without sacrificing too much of its convergence speed, and this is another key motivation for our work.

3.4. Exclusive update strategy

In order to ensure the convergence of PSO, Clerc et al. [14] introduced a constriction factor into the standard particle swarm optimization. Just as discussed in Section 1, this method actually guaranteed the convergence of PSO algorithm via the parameter selection. Similar to our prior work [75], another set of velocity-position update strategy is exploited to keep the global best particle moving until it has reached a local minimum under the assumption of minimization [4], which is able to guarantee the convergence of the MPSO effectively. The corresponding update strategy is described as below:

$$v_{\xi d}(t+1) = -x_{\xi d}(t) + p_{gd}(t) + \omega v_{\xi d}(t) + \rho(t)(1 - 2r_{2d}(t)) \quad (16)$$

$$x_{\xi d}(t+1) = x_{\xi d}(t) + v_{\xi d}(t+1) = p_{gd}(t) + \omega v_{\xi d}(t) + \rho(t)(1 - 2r_{2d}(t)) \quad (17)$$

where ξ denotes the index of the global best particle, $-x_{\xi d}(t)$ resets the particle's position to the global best position $p_{gd}(t)$, $\omega v_{\xi d}(t)$ indicates the current search direction, $\rho(t)(1 - 2r_{2d}(t))$ generates a random sample from a sample space with side lengths $2\rho(t)$. ρ is a scaling factor defined below that determines the size of an area surrounding the global best position to proceed with search.

$$\rho(t+1) = \begin{cases} 2\rho(t), & \text{if } \#successes > s_c \\ 0.5\rho(t), & \text{if } \#failures > f_c \\ \rho(t), & \text{otherwise} \end{cases} \quad (18)$$

where the terms $\#successes$ and $\#failures$ denote the number of consecutive successes and failures respectively. Here, a failure is defined as $f(p_g(t)) = f(p_g(t-1))$ while a success is just the opposite. s_c and f_c denote the preset thresholds. In common cases, a default initial value $\rho(0) = 1.0$ has been found empirically to produce acceptable results.

3.5. Position mutation strategy

From the literature, it can be clearly observed that some evolutionary operators such as selection [1], crossover [54] and mutation [3, 34, 45, 75, 87] are widely exploited to sustain the diversity of the particle swarm. In Ref. [3], an adaptive mutation mechanism was introduced into the conventional particle swarm optimization to enhance the global search ability, in which an extended mutation step size was recommended to facilitate a search phase at new regions when the search population was far away from the optimum point. Alternatively the mutation step size was reduced to initiate local exploration about the isolated search region at the concluding stages of the optimization cycle. In the work of [34], a Gaussian based operator was implemented to induce particle search diversity with probability through mutation. More recent works [45, 75, 87] exploited Cauchy mutation [87], wavelet mutation [45], Gaussian mutation [75] and random strategies [7] to maintain the swarm diversity respectively. Thus, without loss of generality, the wavelet mutation rather than Cauchy, Gaussian, Levy or other mutation mechanism is employed here to retain the swarm diversity in this paper due to its fine-tuning ability in terms of the performance under the same settings compared to other mutation operators.

It is known that the mutation operation is usually used to mutate the position of particles. The details of the wavelet mutation can be described as follows. Each particle has a chance to mutate that is controlled by a probability of mutation $p_m \in [0, 1]$. For the position of each particle, a random number between 0 and 1 will be generated such that if it is less than or equal to p_m , the mutation will take place on that position of the particle. Specifically, let $x_i(t) = [x_{i1}(t), x_{i2}(t), \dots, x_{ij}(t), \dots, x_{iD}(t)]$ be the current selected particle, where $x_{ij}(t)$ denotes the position of the j th dimension of the i th particle in the t th iteration, and it should not exceed the allowable search range for this dimension, viz. $x_{ij}(t) \in [p_{jmin}, p_{jmax}]$.

Hence the resulting particle can be expressed as:

$$\bar{x}_{ij}(t) = \begin{cases} x_{ij}(t) + \sigma \times (p_{jmax} - x_{ij}(t)), & \text{if } \sigma > 0 \\ x_{ij}(t) + \sigma \times (x_{ij}(t) - p_{jmin}), & \text{if } \sigma \leq 0 \end{cases} \quad (19)$$

with

$$\sigma = \frac{1}{\sqrt{a}} e^{-\left(\frac{p}{a}\right)^2 / 2} \cos\left(5 \times \left(\frac{p}{a}\right)\right) \quad (20)$$

where a denotes the dilation parameter, it is usually set to vary with the iteration of particles so as to meet the fine-tuning purpose.

So far, the pseudocode of MPSO algorithm can be succinctly described as follows.

Algorithm 3: Pseudocode of the proposed MPSO algorithm

```

1. Begin
2. Randomly initialize the velocity of the particles and employ Logistic map to initialize the
   position of the particles, let  $f_i$  denote the fitness of each particle,  $MFD^*$  denotes the preset
   threshold of  $MFD$ ,  $f_{avg}$  presents the average fitness of the whole swarm.
3. while (number of maximal iterations is not met)
4.   for  $n=1$  to number of particle
5.     Find  $pbest$ 
6.     Find  $gbest$ 
7.     Calculate  $MFD$  by Eq.(15)
8.     if  $MFD \leq MFD^*$ 
9.       Calculate  $f_i$  and  $f_{avg}$ 
10.      if  $f_i \leq f_{avg}$ 
11.        Execute wavelet mutation for the position of the particles whose fitness is
        less than or equal to the average fitness of the whole swarm
12.      else
13.        Reinitialize the same number of particles using the Logistic map
14.      end
15.    else
16.      Calculate  $\omega$  by Eq.(14)
17.      Update the velocity and position of the global best particle by Eqs.(16)-(17)
18.      Update the velocity and position of the other particles by Eqs.(1)-(2)
19.    end
20.  next  $n$ 
21. next generation until stopping criterion
22. End

```

To better understand the proposed MPSO algorithm described above, we provide a concise yet complete flowchart to illustrate it as shown in Fig. 5.

4. Experimental results and analysis

4.1. Experiments based on basic numerical functions

To validate the effectiveness of the MPSO proposed in this paper, we first conduct experiments on four well-known benchmark functions in this section to determine the related optimal parameters. In particular, the performance of PSO with different initialization methods and different inertia weights is compared and investigated respectively. Note that all the test functions are shown in Table 4, including their expressions, dimensions, search space, allowable search range and global optimum values respectively.

Fig. 6 depicts the graphical shows of the test functions with 2-dimensional decision variables respectively.

To illustrate the effect of the sigmoid-like inertia weight formulated in subsection 3.2, note that PSOs that are randomly initialized but with different inertia weights are compared with each other. For the sake of fair comparison, the parameters are set as follows: the linearly decreasing inertia weight varies from $\omega_{max} = 0.9$ at the beginning of the search to $\omega_{min} = 0.4$ at the end, the number of neighborhood particles $m = 15$, the acceleration coefficients $c_1 = c_2 = 2$, and the swarm size is 40. Besides, the dimension of the test functions is set to 10 except for Schaffer with 2. The success and failure thresholds are $s_c = 15$ and $f_c = 5$ respectively, which implies that the algorithm is quicker to punish a poor ρ setting than it is to reward a successful ρ value to produce acceptable results. In addition, the threshold of MFD is predetermined to be

$MFD^* = 2.80e-006$ by trial and error. It is shown that the performance of MPSO encounters a sharp decline when MFD^* is less than or greater than $2.80e-006$. For each test function, 30 independent runs are performed by each PSO variant, and each run is with 1000 iterations. The PSO algorithm terminates when it reaches the maximum allowed number of iterations. Without loss of generality, the best solution, average solution and standard deviation are employed to measure the performance of different PSOs.

Table 5 presents the optimization results of PSO with different inertia weights. It should be noted that ω_0 -PSO, ω_1 -PSO, ω_2 -PSO, ω_3 -PSO and ω_4 -PSO denote the PSO with linear decreasing inertia weight, convex function decreasing based inertia weight, concave function decreasing based inertia weight, exponential function decreasing based inertia weight and the sigmoid-like inertia weight respectively. As expected, ω_4 -PSO is apparently superior to all the other PSO variants on these four benchmark functions except for the standard deviation of the Griewank,

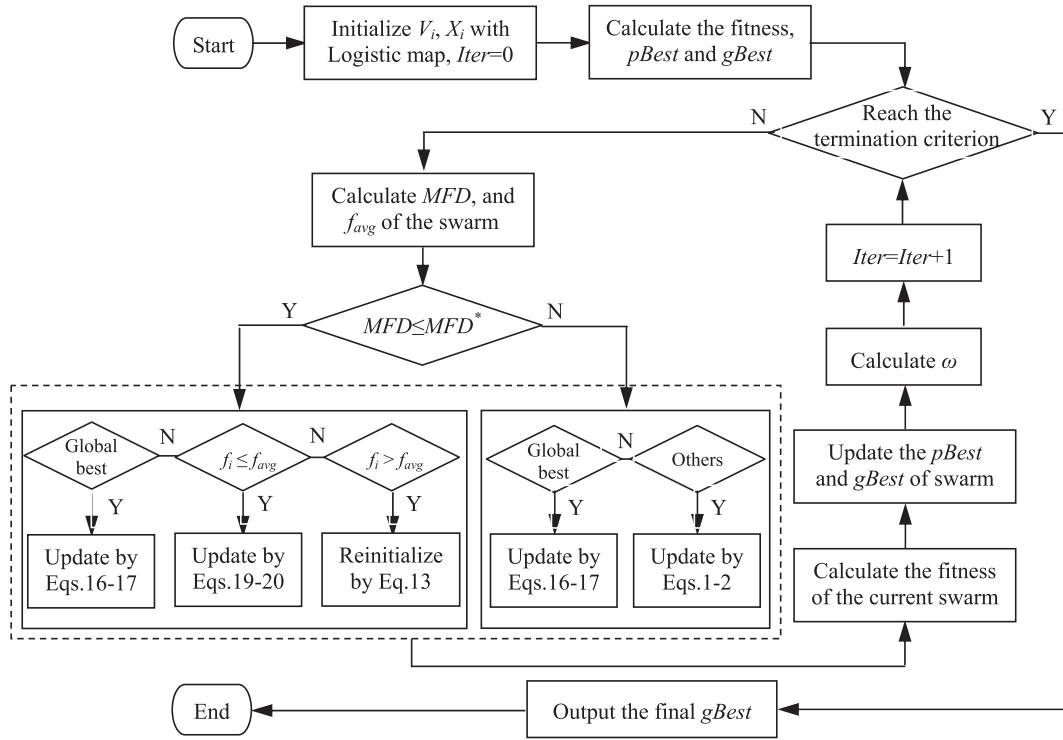


Fig. 5. Flowchart of the MPSO algorithm.

Table 4
Basic benchmark functions employed in this subsection.

Function	Expression	Search space	X_{\max}	V_{\max}	Global optimum
Sphere	$\sum_{i=1}^d x_i^2$	$[-100,100]^n$	100	100	0
Rastrigin	$\sum_{i=1}^d [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-10,10]^n$	10	10	0
Griewank	$\frac{1}{4000} \sum_{i=1}^d (x_i)^2 - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^n$	600	600	0
Schaffer	$0.5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{(1.0 + 0.001(x_1^2 + x_2^2))^2}$	$[-100,100]$	100	100	0

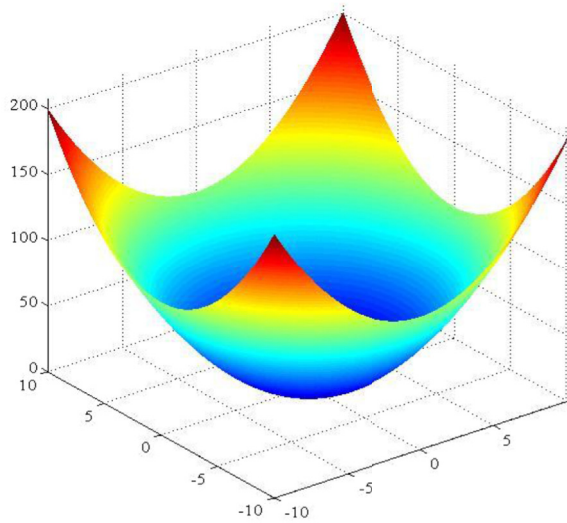
which validates the promising performance of MPSO over the test functions. In other words, this further indicates the importance of the sigmoid-like inertia weight during the search process of particles for the better tradeoff between exploration and exploitation.

Fig. 7 illustrates the convergence curves of PSO with different inertia weights for the four test functions. To show the evolutionary processes clearly, here, the y axes adopt the fitness logarithm values. From the results shown in Fig. 7, one can see that ω_4 -PSO is an effective method with fast convergence speed at the early stage of evolution in almost all cases. Especially in Fig. 7(b), it can be clearly observed that, compared with its counterparts, ω_4 -PSO evolves slightly slower for the first 100 iterations, but after that it quickly converges to the global optimum very significantly. Besides, it should be noted that the former part of Fig. 7(b) is specially scaled up to a certain extent so as to illustrate the variation trends of each curve more clearly. In actual fact, each PSO corresponding to each evolution curve is still run for 1000 iterations. Likewise, it is clear to observe that ω_0 -PSO, ω_2 -PSO and ω_4 -PSO in Fig. 7(d) can achieve the global best solution without being trapped in the local optima. Although the convergence of ω_4 -PSO in the later period is slightly worse than that of ω_0 -PSO, its optimizing speed markedly outperforms ω_0 -PSO in the first 100 iterations. At the same time, both ω_0 -PSO and ω_4 -PSO are apparently superior to the ω_2 -PSO. Another interesting observation comes from the evolution curve of ω_1 -PSO, note that which evolves very slowly before the first 400 iterations, followed by even occurs deterioration in convergence performance. After that ω_1 -PSO improves consistently

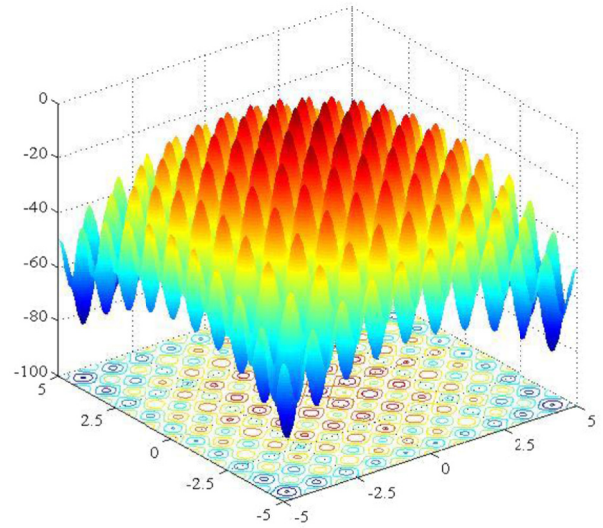
before 750 generations with the increase of iterations, subsequently it almost keeps in a smooth state until the end of the search process. Besides, it can be obviously seen that ω_3 -PSO has the worst performance throughout the iteration. To sum up, PSO with the sigmoid-like inertia weight formulated in this paper is able to get the best performance in most cases by adaptively regulating the balance of exploration and exploitation in the solution space.

To better understand the effectiveness of the chaos-based initialization and the formulated sigmoid-like inertia weight strategies proposed in this paper, different combinations for PSO with an initial population of random map or Logistic map and a constant inertia weight ($\omega = 1$), a linearly decreasing inertia weight or the formulated sigmoid-like inertia weight are exploited respectively. To increase the readability of different PSO paradigms, acronyms PSORC, PSORL, PSORS, PSOLC, PSOLL and PSOLS are specified by Table 6.

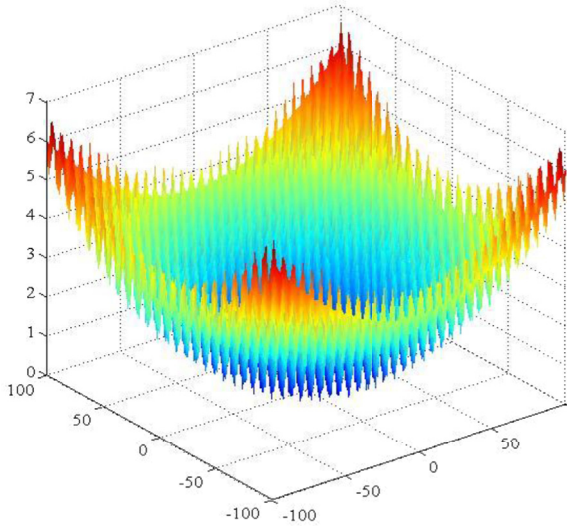
From Table 7, it can be clearly observed that PSOLS outperforms all the others. That is to say, PSO with the Logistic map based initialization and the formulated sigmoid-like inertia weight can get the best optimization performance. To be specific, the performance of PSO with the Logistic map based initialization is far superior to that of the corresponding PSOs with random particles, which means that the distribution of initial particles can be improved by the Logistic map. As can be seen from Table 7, the standard deviations of PSOLS are consistently smaller than that of other PSO variants, which implies that the PSO with an initial population of Logistic map solutions can alleviate its inherent defects of



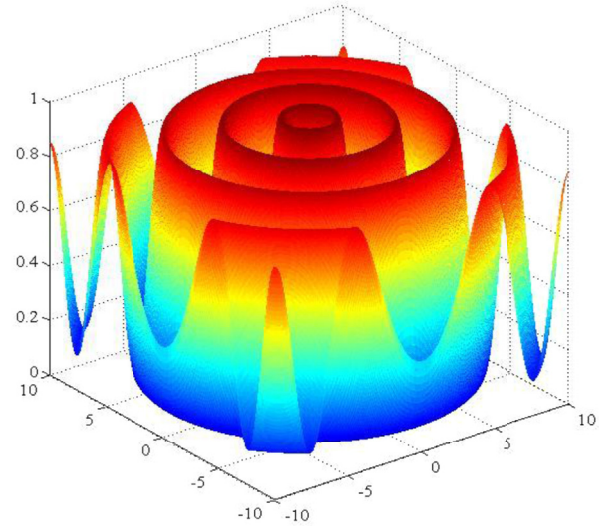
(a) Sphere function



(b) Rastrigin function



(c) Griewank function



(d) Schaffer function

Fig. 6. Graphical shows of the test functions.

low stability. In other words, this further illustrates the importance of the uniformly distributed initial particles to the convergent performance of PSO and the non-linearly sigmoid-like decreasing inertia weight to the tradeoff of exploration and exploitation. At the same time, note that PSO with the formulated sigmoid-like inertia weight also surpasses the corresponding PSO algorithms with a constant inertia weight and a linearly decreasing inertia weight respectively. Meanwhile, the premature convergence is avoided by adopting the wavelet mutation and the local re-initialization strategies based on the maximal focus distance among particles. By this way, the performance of the standard PSO algorithm can be improved to a large extent. To summarize, the Logistic map based initialization, formulated sigmoid-like inertia weight and the newly update mechanism, to some extent, play a good complementary role each other in the particles evolution and should be used together to obtain better optimization performance.

Fig. 8 displays the evolution curves of the *MFD* in each PSO algorithm for the four test functions. As can be seen from Fig. 8, at the points where

the curves of different PSO variants encounter a sharp decline imply that particles tend to trap into the local optima. Subsequently the wavelet mutation as well as the local re-initialization strategy based on the Logistic map is timely leveraged to help PSO escape from the local optima and make the particles proceed with searching in other regions of the solution space. In particular, the introduced velocity-position update mechanism for the global best particle, which is able to keep the search proceeding and effectively guarantee the convergence of MPSO. By comparison, the curves of PSORC and PSORL decrease slowly as the search proceeds. Moreover, both of them behave to be interweaved with each other except for the Sphere function. On the contrary, the curves of PSOLL and PSOLS descend rapidly than that of PSORS and PSOLC for Sphere, Rastrigin and Schaffer functions respectively. In addition, it is noticeable that the evolution curve of the maximal focus distance by PSOLC for Griewank function declines quickly than that of the other PSO algorithms. Particularly, the turning point appears when the iteration approximates 200. On the whole, all the curves of the *MFD* corresponding

Table 5
Optimization results comparison of PSO with different inertia weights.

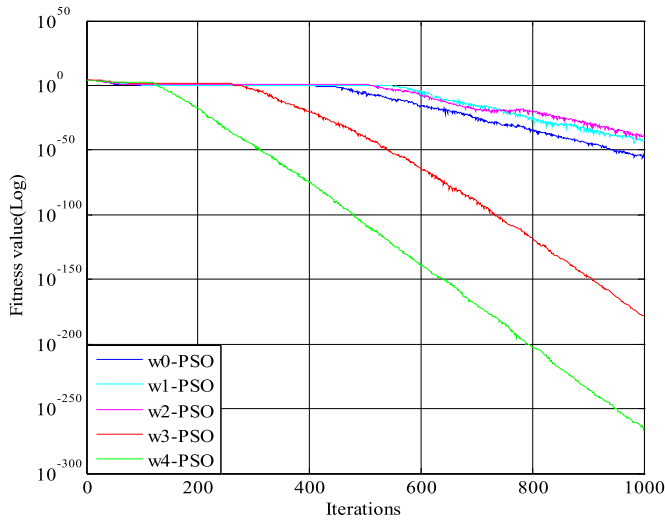
Functions	PSO algorithm	Best solution	Average solution	Standard deviation
Sphere	w_0 -PSO	1.39e-187	1.45e-179	0.00e-000
	w_1 -PSO	1.13e-058	7.62e-017	2.21e-016
	w_2 -PSO	1.40e-058	2.06e-036	5.44e-036
	w_3 -PSO	3.65e-244	4.01e-221	0.00e-000
	w_4 -PSO	1.65e-266	4.13e-261	0.00e-000
Rastrigin	w_0 -PSO	0.00e-000	0.00e-000	0.00e-000
	w_1 -PSO	0.00e-000	6.36e-231	8.31e-230
	w_2 -PSO	0.00e-000	1.63e-271	8.31e-168
	w_3 -PSO	0.00e-000	0.00e-000	0.00e-000
	w_4 -PSO	0.00e-000	0.00e-000	0.00e-000
Griewank	w_0 -PSO	1.61e-001	5.28e-001	5.32e-001
	w_1 -PSO	1.77e-001	1.02e-000	5.46e-001
	w_2 -PSO	1.97e-001	1.15e-000	2.95e-001
	w_3 -PSO	2.86e-001	8.74e-001	4.22e-001
	w_4 -PSO	4.51e-003	9.90e-002	3.16e-001
Schaffer	w_0 -PSO	0.00e-000	0.00e-000	0.00e-000
	w_1 -PSO	3.41e-018	1.11e-017	2.22e-017
	w_2 -PSO	0.00e-000	1.11e-016	4.44e-014
	w_3 -PSO	3.72e-002	1.77e-002	1.09e-014
	w_4 -PSO	0.00e-000	0.00e-000	0.00e-000

to different PSO variants on the test functions remain in a downward trend, which further demonstrates the fact that PSO tends to trap into the local optima especially in the later stage of evolution. In the meanwhile, it also shows the necessity of adopting some effective strategies to help the PSO algorithm escape from the local optima.

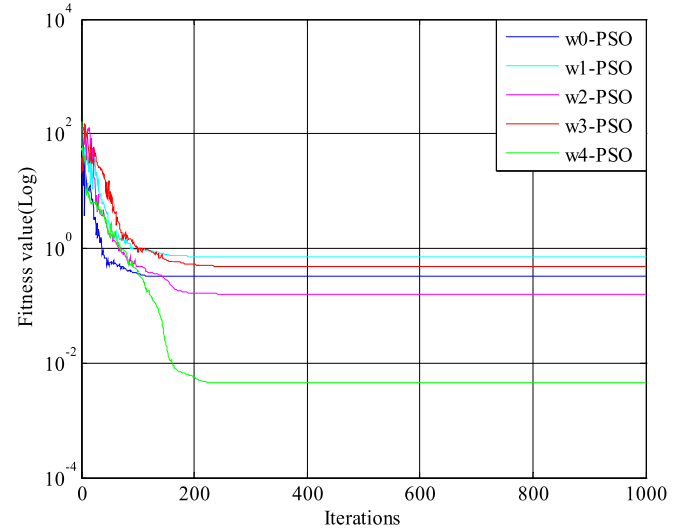
4.2. Experiments based on CEC'13 test suite

As observed from the experimental results shown above, our method is remarkably superior to the others in most cases, which verifies the effectiveness and efficiency of it in the task of the basic function optimization. To investigate its performance in relatively complex multimodal problems, a series of experiments are conducted on CEC'13 test suite [41] in this subsection, which consists of 28 functions, including 5 unimodal functions, 15 multimodal functions and 8 composition functions respectively. Note that nearly half of CEC'13 functions are used to test and analyze here, viz., 4 unimodal functions, 4 multimodal functions and 4 composition functions, resulting in 12 benchmark functions described in Table 8.

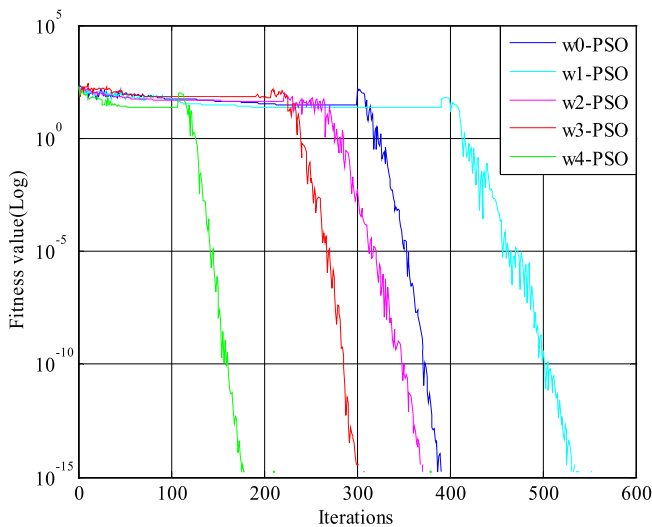
To make a fair comparison with several state-of-the-art PSO variants, including GPSO [66], OLPSO-L [94], DMPPSO [35], SRPSO [71] and



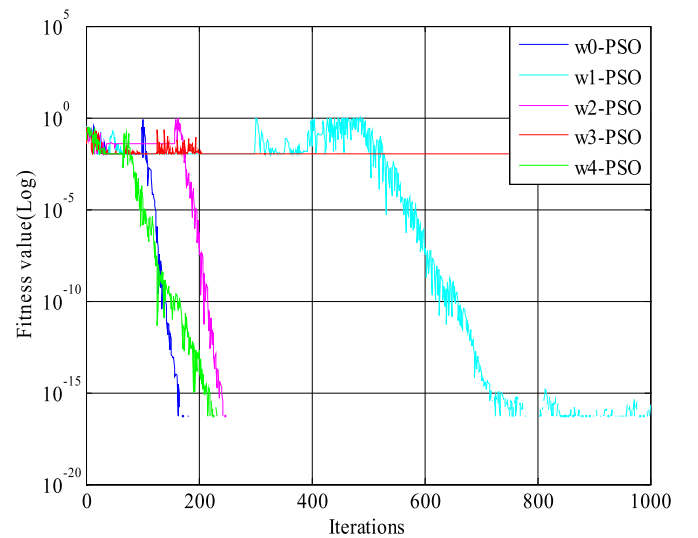
(a) Sphere function



(c) Griewank function



(b) Rastrigin function



(d) Schaffer function

Fig. 7. The convergence curves of PSO with different inertia weights.

Table 6

Each acronym and its corresponding PSO variant.

Initialization method	Inertia weight	Acronym
Random map	constant inertia weight	PSORC
	linearly decreasing inertia weight	PSORL
	sigmoid-like inertia weight	PSORS
Logistic map	constant inertia weight	PSOLC
	linearly decreasing inertia weight	PSOLL
	sigmoid-like inertia weight	PSOLS

Table 7

Optimization results comparison among different PSO variants.

Function	Methods	Best solution	Average solution	Standard deviation
Sphere	PSORC	1.30e−006	1.57e−004	1.42e−004
	PSORL	2.13e−181	2.94e−171	0.00e−000
	PSORS	2.59e−223	2.41e−217	0.00e−000
	PSOLC	5.08e−007	1.02e−004	1.25e−004
	PSOLL	7.86e−192	2.76e−179	0.00e−000
	PSOLS	3.65e−244	4.01e−221	0.00e−000
Rastrigin	PSORC	1.64e+001	3.15e+001	1.33e+001
	PSORL	0.00e+000	0.00e+000	0.00e+000
	PSORS	0.00e+000	0.00e+000	0.00e+000
	PSOLC	1.40e+001	2.56e+001	9.18e+000
	PSOLL	0.00e+000	0.00e+000	0.00e+000
	PSOLS	0.00e+000	0.00e+000	0.00e+000
Griewank	PSORC	4.62e+001	6.21e+001	4.32e−001
	PSORL	2.95e+001	7.36e+001	3.09e−001
	PSORS	1.24e+000	1.72e+000	9.44e−002
	PSOLC	1.53e+000	1.72e+000	1.41e−001
	PSOLL	0.00e+000	5.33e−062	0.00e−000
	PSOLS	0.00e+000	0.00e−000	0.00e−000
Schaffer	PSORC	2.40e−003	1.76e−002	1.32e−002
	PSORL	5.55e−017	1.69e−004	3.38e−004
	PSORS	2.36e−017	1.86e−009	3.24e−009
	PSOLC	5.77e−014	1.96e−012	2.58e−012
	PSOLL	0.00e−000	0.00e−000	0.00e−000
	PSOLS	0.00e−000	0.00e−000	0.00e−000

IDE-PSO [27], the fitness mean (*Mean*) and standard deviation (*Std*) of the fitness errors are utilized to estimate their performance. It should be noted that the fitness error denotes the fitness deviation between the solution yielded by MPSPSO and the ideal solution. Likewise, each PSO is run 30 times on every test function with 1000 iterations for each run, and the stopping criterion is set as reaching the total number of iterations. Besides, it's worth noting that all the experimental parameters are the same as those in subsection 4.1 except for the function dimension with 10 and 30, and the failure threshold $f_c = 3$.

From Tables 9 and 10, we can find that MPSPSO is superior or highly competitive to several state-of-the-art PSO variants. Specifically, there exist the following aspects that can be easily observed. First, the smaller *Mean* value can be obtained by MPSPSO for most functions with $D = 10$ except for f_1 tested by GPSO, OLPSO-L, DMPPSO-L and SRPSO as well as f_4 tested by OLPSO-L and SRPSO respectively. In particular, compared with the most competitive particle swarm optimization variants SRPSO and IDE-PSO, the *Mean* value of our method is consistently smaller than that of IDE-PSO despite with the marginal difference for f_{10} and obviously outperforms that of SRPSO except for f_1 and f_4 . Besides, it should be noted that the *Std* value is markedly superior to that of IDE-PSO, especially for the f_6 and f_9 with one order of magnitude lower, which implies that the PSO with an initial population of Logistic map based solutions can alleviate its inherent shortcomings of low stability. Second, with regard to the experimental results comparison under $D = 30$, MPSPSO is also able to achieve good performance with smaller *Mean* value such as for f_2, f_3, f_5, f_7, f_8 and f_{10} out of the twelve test functions. As for other PSO variants with higher *Mean* value, however, all of them have obtained the better *Std*, such as for f_1, f_4, f_9 and f_{11} with one or more orders of magnitude

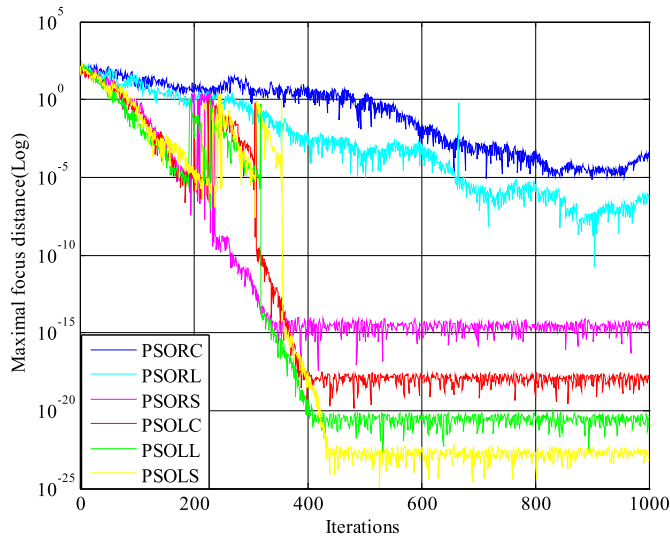
lower than the competitive IDE-PSO and other PSO variants respectively. This further demonstrates the effect of the chaos-based particle swarm initialization. Third, without exception, it is noticeable that the performance of each PSO mentioned above declines as the function's dimension scale increasing from 10 to 30. At the same time, there is no doubt that the computational time increases accordingly. In addition, it is worth noting that like other studies, it seems that the difference of the experimental results among some PSO variants is relatively small, but as far as the intelligent computation itself is concerned, even though a little improvement from the experiments on the surface, they are still of great significance in the field of population based meta-heuristic search algorithm, at least on the experimental results reported in this subsection. To this end, we have validated it from the perspective of statistical analysis by trial and error, and the results indicated that MPSPSO outperforms the other PSO variants such as SRPSO and IDE-PSO with a 95% confidence level.

4.3. Experiments based on CEC'15 test suite

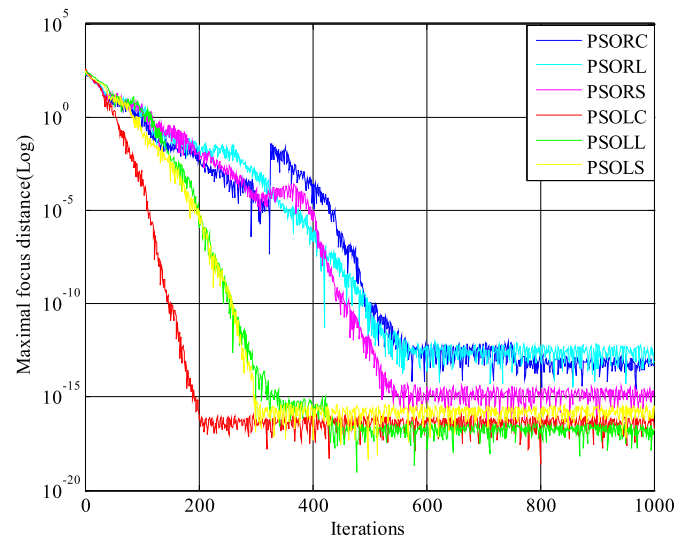
To further investigate the effect of MPSPSO, extensive experiments are also conducted on the latest CEC'15 test suite [42], which consists of 15 learning-based benchmark functions that have several shift vectors and rotation matrices. As a result, it's very difficult to get the optimal solutions with lower minor errors. In Tables 11 and 2 unimodal functions, 3 multimodal functions, 3 hybrid functions and 4 composition functions are selected to be tested in this subsection. Note that all of the problems can be treated as the black-box issues. To ensure a fair comparison with the counterparts such as GPSO [66], LPSO [32], SPSO [6], CLPSO [40], FIPS [53] and DMSPSO [43], the swarm size is set to 40, the maximum number of iterations of MPSPSO is the same as that used in literature [96], that is, 2500 and 7500 for 10- and 30-dimensional problems respectively. Without loss of generality, the performance is also estimated in terms of the fitness mean (*Mean*) and standard deviation (*Std*) of the fitness errors.

The comparison of *Means* and *Stds* between MPSPSO and the other PSO variants is illustrated in Tables 12 and 13. Note that the results of the other six PSO methods to be compared here are directly referenced from literature [96], and the lowest mean and standard deviation values in each line are highlighted in boldface. From the results, one can see that MPSPSO is able to obtain better *Mean* values for most of the test functions with 10 dimensions except for f_2, f_7, f_8, f_{10} and f_{11} . As for *Std*, the proposed method still performs significantly better than the others for almost all cases with the exception of DMSPSO on f_8 , CLPSO on f_{10} and f_{11} as well as SPSO and CLPSO on f_{12} respectively, which indicates that MPSPSO has better solution stability owing to the Logistic map based swarm initialization that yields uniformly distributed initial particles together with the wavelet mutation that enriches the swarm diversity. Similarly, compared with other PSO variants on the selected CEC'15 functions under $D = 30$, even though the *Std* performance of MPSPSO is worse or slightly worse than that of DMSPSO on f_7 , CLPSO and SPSO on f_7 and f_{12} , MPSPSO is still advantageous in its robustness and stability, which is largely ascribed to the initialization strategy and the update mechanism adopted in the PSO algorithm. In sum, compared with other PSO variants mentioned in this subsection, the proposed MPSPSO is quite competitive in terms of the stability, robustness and scalability.

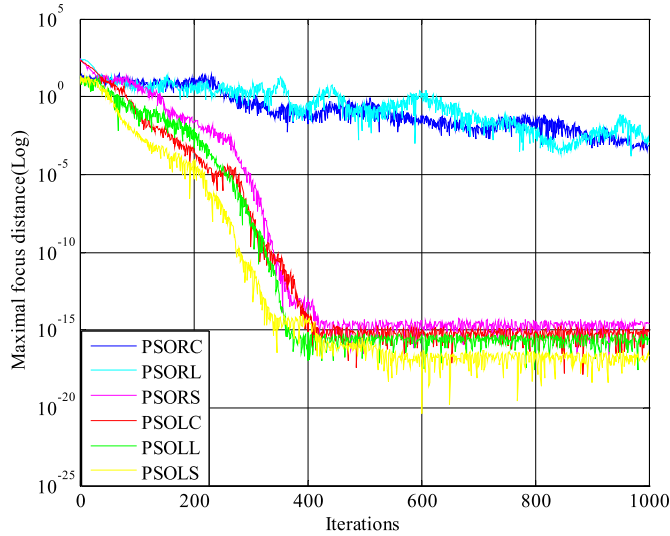
In addition, to thoroughly and fairly discern the experimental results of different PSOs mentioned in this subsection, the nonparametric Wilcoxon rank sum test is conducted with significance level = 0.05 between MPSPSO and its competitors to judge the significance of performance. Table 14 summarizes the experimental results under $D = 10$ and $D = 30$ respectively. Note that the number of benchmark functions (out of the 12 tests) that MPSPSO is significantly better than (*Better*), almost the same as (*Same*) and significantly worse than (*Worse*) its peer algorithm, and the total score (*Merit*) is calculated by subtracting *Worse* from *Better*. From



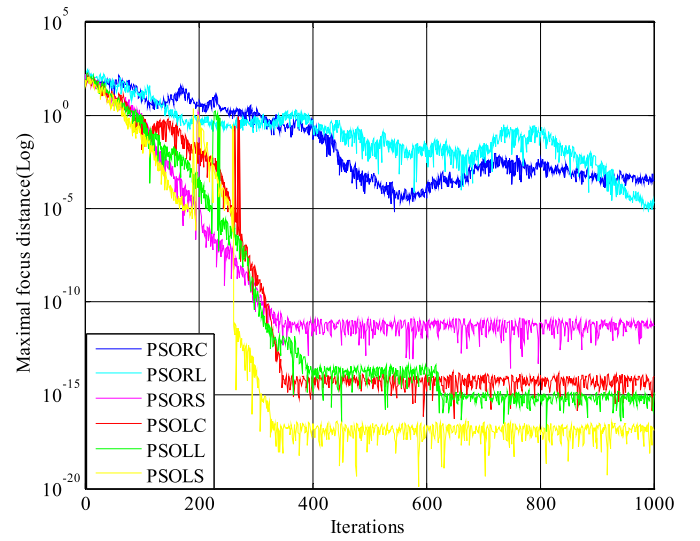
(a) Sphere function



(c) Griewank function



(b) Rastrigin function



(d) Schaffer function

Fig. 8. The evolution curves of MFD with different experimental settings.

Table 8

CEC'13 functions employed in this subsection.

Type	No.	Function name	Search range
Unimodal functions	1	rotated high conditioned elliptic function	[−100,100]
	2	rotated bent cigar function	
	3	rotated discus function	
	4	different powers function	
Multimodal functions	5	rotated Rosenbrock's function	
	6	rotated Weierstrass function	
	7	rotated Schwefel's function	
	8	rotated Katsuura function	
Composition functions	9	composition function 1 (n = 5, rotated)	
	10	composition function 6 (n = 5, rotated)	
	11	composition function 7 (n = 5, rotated)	
	12	composition function 8 (n = 5, rotated)	

Table 14, it can be seen that the number of functions that MPSO outperforms its peers is much larger than the number of functions in which it significantly performs worse than the compared PSO methods. In particular, the *Merit* values shown here apparently demonstrate the significance of MPSO over other selected particle swarm optimization algorithms.

4.4. Experiments based on standard image segmentation

To further illustrate the effect of the proposed MPSO algorithm, we apply it in the task of standard image segmentation. Note that the threshold segmentation is a basic method in the field of image segmentation, and the most commonly used threshold method is the Otsu algorithm [57] whose core idea can be described as follows: let the pixels of a given image be represented in l gray levels $\{0, 1, \dots, l-1\}$, suppose that the pixels are dichotomized into two classes: object and background, denoted by C_0 with gray levels $\{0, 1, \dots, t\}$ and C_1 with gray levels $\{t+1, t+2, \dots, l-1\}$ respectively by a threshold at level t . $p_i = n_i/N$, where

Table 9

Comparison of Means and Stds of MPSO and the other six PSO variants under D = 10.

Func.	Item	GPSO	OLPSO-L	DMPPSO-G	DMPPSO-L	SRPSO	IDE-PSO	MPSO
f_1	Mean	7.30e+05	1.52e+04	2.77e+06	5.82e+05	1.54e+05	3.64e+06	1.82e+06
	Std	1.17e+06	8.62e+03	3.33e+06	6.33e+05	1.72e+05	3.17e+06	2.68e+05
f_2	Mean	3.39e+08	9.12e-07	5.54e+08	8.85e+07	4.65e+07	2.68e+07	2.12e+07
	Std	7.44e+08	2.46e+08	1.10e+09	2.84e+08	1.33e+08	3.31e+07	2.39e+07
f_3	Mean	2.63e+03	1.52e+04	1.46e+04	5.20e+03	3.61e+02	1.34e+02	1.32e+02
	Std	2.76e+03	5.99e+03	5.72e+03	9.57e+03	6.08e+02	1.43e+02	1.29e+02
f_4	Mean	2.02e+01	0.00e+00	6.58e+00	3.68e-02	0.00e+00	6.18e-04	4.37e-04
	Std	2.19e+01	0.00e+00	1.76e+01	2.11e-02	0.00e+00	2.79e-04	1.06e-04
f_5	Mean	3.21e+01	1.11e+01	2.22e+01	6.91e+00	4.34e-01	1.89e-01	1.41e-01
	Std	3.72e+01	1.35e+01	3.23e+01	3.71e+00	1.12e+00	3.23e-02	1.82e-02
f_6	Mean	9.14e+00	7.15e+00	9.16e+00	7.12e+00	8.82e+00	6.11e+00	4.05e+00
	Std	1.42e+00	1.15e+00	1.87e+00	2.38e+00	5.30e-01	1.17e+00	3.33e-01
f_7	Mean	1.29e+03	1.37e+03	1.33e+03	8.63e+02	8.34e+02	7.85e+02	6.96e+02
	Std	3.25e+02	2.40e+02	3.47e+02	4.25e+02	2.83e+02	2.32e+02	1.85e+02
f_8	Mean	8.97e-01	8.99e-01	1.59e+00	1.43e+00	0.00e+00	0.00e+00	0.00e+00
	Std	2.94e-01	2.32e-01	3.36e-01	4.05e-01	0.00e+00	0.00e+00	0.00e+00
f_9	Mean	3.88e+02	3.92e+02	3.74e+02	3.72e+02	3.75e+02	2.39e+02	2.13e+02
	Std	4.45e+01	4.08e+01	6.42e+01	6.80e+01	6.76e+01	1.54e+02	6.68e+01
f_{10}	Mean	2.45e+02	1.67e+02	2.08e+02	1.71e+02	2.11e+02	1.41e+02	1.37e+02
	Std	7.22e+01	2.59e+01	5.10e+01	5.56e+01	5.25e+01	2.04e+01	2.15e+01
f_{11}	Mean	6.79e+02	4.72e+02	6.06e+02	6.01e+02	4.52e+02	3.80e+02	3.39e+02
	Std	1.20e+02	4.24e+01	4.77e+01	4.45e+01	3.93e+01	4.69e+01	2.58e+01
f_{12}	Mean	6.54e+02	4.66e+02	5.27e+02	2.52e+02	3.30e+02	3.23e+02	2.14e+02
	Std	2.60e+02	2.04e+02	2.70e+02	1.42e+02	1.14e+02	1.31e+02	1.17e+02

Bold values indicate the method which achieves the best performance on each function.

Table 10

Comparison of Means and Stds of MPSO and the other six PSO variants under D = 30.

Func.	Item	GPSO	OLPSO-L	DMPPSO-G	DMPPSO-L	SRPSO	IDE-PSO	MPSO
f_1	Mean	5.94e+07	6.37e+05	2.22e+07	7.03e+07	1.75e+09	5.16e+07	6.28e+07
	Std	3.34e+07	1.97e+05	1.99e+07	3.10e+08	7.09e+08	3.47e+07	7.06e+06
f_2	Mean	3.71e+15	3.06e+09	1.50e+10	1.80e+15	7.73e+19	5.46e+07	4.31e+07
	Std	1.82e+16	3.63e+09	7.73e+09	8.76e+15	2.50e+20	7.14e+07	8.01e+06
f_3	Mean	4.21e+04	9.93e+04	6.36e+04	1.48e+04	4.61e+05	8.20e+03	6.85e+03
	Std	1.01e+04	1.72e+04	1.38e+04	3.54e+04	3.12e+05	6.09e+03	5.26e+03
f_4	Mean	8.79e+02	0.00e+00	3.17e+01	1.28e+03	3.71e+04	2.44e+01	3.76e+01
	Std	2.83e+02	0.00e+00	6.70e+01	4.28e+03	1.18e+04	9.23e+01	6.81e+00
f_5	Mean	5.29e+02	7.75e+01	8.49e+01	6.80e+01	1.75e+04	2.45e+01	2.29e+01
	Std	2.57e+02	2.19e+01	4.33e+01	3.49e+01	6.13e+03	2.31e+01	2.18e+01
f_6	Mean	4.13e+01	3.83e+01	4.15e+01	3.97e+01	4.38e+01	3.11e+01	3.64e+01
	Std	3.35e+00	2.29e+00	2.20e+00	6.01e+00	1.22e+00	3.56e+00	1.75e+00
f_7	Mean	5.44e+03	7.16e+03	7.59e+03	4.81e+03	9.42e+03	4.59e+03	4.38e+03
	Std	8.38e+02	5.67e+02	1.35e+03	1.74e+03	5.17e+02	1.03e+02	8.16e+01
f_8	Mean	2.33e+00	2.23e+00	3.14e+00	2.76e+00	0.00e+00	0.00e+00	0.00e+00
	Std	5.77e-01	3.74e-01	5.01e-01	3.91e-01	0.00e+00	0.00e+00	0.00e+00
f_9	Mean	1.34e+03	1.30e+03	3.39e+02	3.08e+02	5.33e+03	2.87e+02	3.17e+02
	Std	5.29e+02	4.63e+02	8.74e+01	9.87e+01	7.52e+02	1.49e+02	9.32e+01
f_{10}	Mean	3.72e+02	2.34e+02	2.82e+02	2.83e+02	4.13e+02	2.03e+02	1.98e+02
	Std	8.81e+01	7.18e+01	1.06e+02	1.03e+02	2.36e+01	3.56e+00	2.81e+00
f_{11}	Mean	1.61e+03	1.25e+03	1.44e+03	1.45e+03	1.47e+03	1.21e+03	1.72e+03
	Std	1.82e+02	7.90e+01	3.81e+01	4.63e+01	4.42e+01	1.03e+02	3.06e+01
f_{12}	Mean	4.80e+03	3.56e+03	2.86e+03	8.23e+02	8.87e+03	3.15e+02	5.39e+02
	Std	9.73e+02	5.47e+02	9.95e+02	1.62e+03	1.42e+03	6.46e+01	3.66e+01

Bold values indicate the method which achieves the best performance on each function.

Table 11

CEC'15 functions employed in this subsection.

Type	No.	Function name	Search range
Unimodal functions	1	rotated high conditioned elliptic function	[-100,100]
	2	rotated cigar function	
Multimodal functions	3	shifted and rotated Ackley's function	
	4	shifted and rotated Rastrigin's function	
Hybrid functions	5	shifted and rotated Schwefel's function	
	6	hybrid function 1 (n = 3)	
Composition functions	7	hybrid function 2 (n = 4)	
	8	hybrid function 3 (n = 5)	
	9	composition function 1 (n = 3)	
	10	composition function 3 (n = 5)	
	11	composition function 6 (n = 7)	
	12	composition function 7 (n = 10)	

Table 12

Comparison of Means and Stds of MPSO and the other six PSO variants under D = 10.

Func.	Item	GPSO	LPSO	SPSO	CLPSO	FIPS	DMSPSO	MPSO
f_1	Mean	5.05e+06	1.17e+05	2.78e+04	5.53e+05	2.51e+05	9.64e+04	2.63e+04
	Std	2.31e+07	7.30e+04	1.01e+04	7.42e+04	3.50e+04	2.46e+05	6.61e+03
f_2	Mean	6.02e+08	2.94e+07	6.15e+03	4.73e+04	6.49e+03	1.38e+04	6.36e+03
	Std	1.05e+09	7.79e+07	2.83e+04	8.12e+04	3.86e+02	5.47e+03	2.12e+02
f_3	Mean	2.03e+01	2.02e+01	2.02e+01	2.02e+01	2.03e+01	2.01e+01	1.87e+01
	Std	1.12e-01	4.07e-02	2.99e-02	2.41e+00	1.51e-01	1.64e-01	1.18e-02
f_4	Mean	1.48e+01	1.35e+01	4.87e+00	1.02e+01	6.75e+00	9.19e+00	3.23e+00
	Std	1.12e+01	9.61e+00	4.98e+00	3.73e-01	1.72e+00	6.16e+00	1.76e-02
f_5	Mean	5.76e+02	2.91e+02	4.14e+02	6.11e+02	5.14e+02	3.34e+02	1.89e+02
	Std	9.36e+01	8.95e+02	6.49e+02	1.46e+02	4.58e+02	1.76e+02	8.85e+01
f_6	Mean	4.69e+03	5.83e+03	1.13e+03	1.72e+03	7.11e+02	1.78e+03	5.20e+02
	Std	6.41e+03	1.04e+04	5.75e+03	3.96e+02	7.15e+02	1.38e+04	3.17e+02
f_7	Mean	5.56e+00	3.60e+00	1.34e+00	1.53e+00	8.87e-01	1.96e+00	8.99e-01
	Std	2.12e+00	4.30e-01	1.99e-01	4.24e-01	2.39e-01	4.07e-01	1.23e-01
f_8	Mean	6.53e+03	1.81e+03	1.27e+03	4.87e+02	6.89e+02	1.79e+02	4.46e+02
	Std	4.98e+04	3.51e+03	2.54e+03	2.88e+02	1.67e+03	6.33e+01	1.28e+02
f_9	Mean	1.07e+02	1.01e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02
	Std	1.79e+01	1.01e+00	7.99e-02	1.39e-02	1.92e-01	5.41e-01	1.06e-02
f_{10}	Mean	3.14e+02	2.59e+02	2.15e+02	1.62e+01	4.64e+01	1.31e+02	1.90e+02
	Std	2.22e+01	1.11e+02	5.62e+02	5.15e+00	1.13e+02	4.49e+02	2.66e+01
f_{11}	Mean	6.25e+03	5.47e+03	6.19e+03	2.45e+03	2.29e+03	3.17e+03	2.71e+03
	Std	4.37e+03	6.32e+03	8.62e+03	4.51e+02	1.70e+03	2.89e+03	7.25e+02
f_{12}	Mean	1.25e+02	1.06e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02
	Std	8.43e+00	4.05e+01	0.00e+00	0.00e+00	1.73e-12	2.63e-13	5.68e-19

Table 13

Comparison of Means and Stds of MPSO and the other six PSO variants under D = 30.

Func.	Item	GPSO	LPSO	SPSO	CLPSO	FIPS	DMSPSO	MPSO
f_1	Mean	2.72e+08	3.52e+07	2.23e+05	5.56e+06	4.00e+06	6.72e+06	1.79e+06
	Std	3.54e+08	5.92e+07	2.42e+05	5.80e+06	2.42e+05	1.15e+07	2.38e+05
f_2	Mean	2.23e+10	1.91e+09	3.70e+03	4.48e+03	6.29e+03	3.37e+03	1.46e+03
	Std	1.96e+09	2.09e+09	2.44e+04	1.70e+03	1.43e+04	2.92e+03	7.28e+01
f_3	Mean	2.08e+01	2.08e+01	2.09e+01	2.09e+01	2.10e+01	2.05e+01	2.03e+01
	Std	7.54e-03	3.58e-01	8.30e-02	1.90e-02	9.90e-02	1.53e-01	2.33e-03
f_4	Mean	1.55e+02	1.03e+02	3.55e+01	9.02e+01	1.54e+02	8.32e+01	1.91e+01
	Std	8.14e+01	1.29e+01	3.38e+00	3.63e+00	4.96e+01	1.54e+01	4.06e-01
f_5	Mean	3.54e+03	3.19e+03	3.97e+03	4.62e+03	6.31e+03	3.79e+03	2.62e+03
	Std	3.22e+02	8.32e+02	1.39e+02	6.03e+02	1.51e+03	5.40e+02	1.24e+02
f_6	Mean	1.00e+07	1.30e+06	1.14e+05	5.53e+05	4.37e+05	1.70e+05	4.41e+04
	Std	7.16e+07	5.56e+05	3.13e+04	1.71e+05	5.05e+05	8.56e+04	7.17e+03
f_7	Mean	4.77e+01	2.46e+01	9.20e+00	9.10e+00	1.24e+01	1.29e+01	9.98e+00
	Std	3.55e+01	3.69e+01	1.25e+00	8.60e-01	4.09e+00	9.84e-02	3.15e+00
f_8	Mean	1.73e+06	2.36e+05	3.22e+04	6.36e+04	4.73e+04	7.95e+04	2.64e+04
	Std	2.05e+07	4.11e+05	2.60e+04	9.24e+04	9.13e+03	1.91e+05	3.36e+03
f_9	Mean	2.15e+02	1.29e+02	1.03e+02	1.04e+02	1.03e+02	1.04e+02	1.03e+02
	Std	8.37e+01	5.08e+00	1.98e-01	1.81e-01	8.50e-02	7.00e-02	8.07e-03
f_{10}	Mean	1.24e+03	1.04e+03	5.91e+02	3.55e+02	4.39e+02	5.98e+02	6.03e+02
	Std	3.88e+02	2.09e+02	7.22e+01	4.81e+01	8.05e+01	7.56e+02	1.88e+01
f_{11}	Mean	4.86e+04	3.85e+04	3.36e+04	2.89e+04	2.73e+04	3.07e+04	2.73e+04
	Std	4.47e+04	4.23e+03	3.82e+03	3.80e+02	2.97e+03	1.27e+03	9.13e+02
f_{12}	Mean	7.58e+02	1.22e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02
	Std	3.01e+03	2.77e+01	0.00e+00	1.88e-13	3.50e-10	1.59e-05	2.12e-10

Table 14

Statistical analysis of Wilcoxon test between MPSO and its competitors.

Dim.	Item	GPSO	LPSO	SPSO	CLPSO	FIPS	DMSPSO
10	Better	12	12	9	8	7	8
	Same	0	0	2	2	2	2
	Worse	0	0	1	2	3	2
	Merit	12	12	8	6	4	6
30	Better	12	12	8	9	8	10
	Same	0	0	2	1	3	1
	Worse	0	0	2	2	1	1
	Merit	12	12	6	7	7	9



(a) original image



(b) segmented by GA (threshold: 116)



(c) segmented by SPSO (threshold: 117)



(d) segmented by MPSO (threshold: 119)

Fig. 9. The original and segmented images of Lena.

n_i represents the number of pixels at level i while N denotes the total number of pixels appeared in the image. As a result, the probabilities of class occurrence and the class mean levels for C_0 can be defined as below:

$$\omega_o(t) = \sum_{i=0}^t p_i, \mu_o(t) = \sum_{i=0}^t i p_i / \omega_o \quad (21)$$

Similarly, the probabilities of class occurrence and the class mean levels for the background can be described as:

$$\omega_1(t) = \sum_{i=t+1}^{l-1} p_i, \mu_1(t) = \sum_{i=t+1}^{l-1} i p_i / \omega_1 \quad (22)$$

Note that the variance formula between these two groups (object and background) can be defined by $d(t) = \omega_o(t)\omega_1(t)(\mu_o(t) - \mu_1(t))^2$. The corresponding gray level value t^* is the best threshold when the variance function achieving the maximum value, i.e., $t^* = \text{Argmax}\{d(t)\}$. So it can be seen that how to determine the threshold value of Otsu method is the key to the task of image segmentation. Here, we exploit the MPSO algorithm proposed in this paper to solve the segmentation threshold. Note that due to the limited space, the standard images Lena and Peppers with 512*512 pixels are employed here to validate the performance of the MPSO. At the same time, we compare it with the standard PSO (SPSO) and genetic algorithm (GA) respectively. The main parameter settings of GA are as follows: elite selection strategy, crossover rate is 0.7, mutation rate is 0.4, migration fraction is 0.2, population size is 50 and

the maximum generation is 100 served as the stopping criteria. Figs. 9 and 10 illustrate the original images and segmented results yielded by GA, SPSO and the proposed MPSO respectively, which further verifies the superiority of our method over other swarm intelligence based approaches in the task of image segmentation.

5. Conclusions and future work

Due to the effect on particle swarm optimization, we have proposed a modified PSO algorithm in this paper. The main contributions of this work can be summarized as follows. First, the chaos-based (Logistic map) solutions are utilized to initialize the uniformly distributed initial particles to enhance the stability of PSO. Second, a novel sigmoid-like inertia weight is formulated to make the PSO adaptively adopt the inertia weight between linear decreasing and nonlinear decreasing strategies based on the maximal focus distance, which is able to keep the balance between exploration and exploitation. Conducted experiments in subsection 4.1 validate its effectiveness and efficiency. Third, the wavelet mutation is applied for the particles whose fitness value is less than that of the average in order to effectively prevent the PSO from plunging into local optima and make the particles proceed with searching in other regions of the solution space. Besides, an auxiliary velocity-position update strategy is introduced exclusively for the global best particle to guarantee the convergence of the MPSO. Extensive experiments on the benchmark test suites and in the task of standard image segmentation validate its effectiveness, robustness and scalability.



(a) original image



(b) segmented by GA (threshold: 119)



(c) segmented by SPSO (threshold: 119)



(d) segmented by MPSO (threshold: 120)

Fig. 10. The original and segmented images of Peppers.

For future work, we intend to compare MPSO with other state-of-the-art PSO variants, such as HCLPSO and EPSO, in the task of solving complex problems. More importantly, we will apply the novel initialization method proposed in this paper to these state-of-the-art PSOs in the future research. In addition, we plan to introduce MPSO into some real-world research fields, such as integrated circuit design, multi-objective optimization, multimedia semantic understanding and engineering optimal scheduling, etc. Besides, we also intend to delve deeper into the parallelization of MPSO for large-scale optimization problems and exploring the use of different inertia weights in different scenarios simultaneously, especially for the adequate parameter tuning in a wide range of problems. Lastly, and arguably most importantly, the qualitative relationship between the chaos-based initialization and the convergence of PSO algorithm, from the viewpoint of mathematics, will be elaborated and proved comprehensively.

Acknowledgements

The authors would like to sincerely thank the editors and anonymous reviewers for their valuable comments and insightful suggestions that have helped us to improve the paper. In addition, this work is partially supported by the National Program on Key Basic Research Project (973 Program) (No. 2013CB329502), National Natural Science Foundation of China (No. 61035003, No. 61202212) and Key Research Project of Baoji University of Arts And Sciences (No. ZK2018061).

References

- [1] P. Agarwalla, S. Mukhopadhyay, Efficient player selection strategy based diversified particle swarm optimization algorithm for global optimization, *Inf. Sci.* 397 (2017) 69–90.
- [2] B. Alatas, E. Akin, A. Ozer, Chaos embedded particle swarm optimization algorithms, *Chaos Solit. Fractals* 40 (4) (2009) 1715–1734.
- [3] A. Alireza, PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems, *Acta Autom. Sin.* 37 (5) (2011) 541–549.
- [4] F. Bergh, A. Engelbrecht, A new locally convergent particle swarm optimizer, in: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'02)*, 2002, pp. 94–99.
- [5] M. Bonyadi, X. Li, Z. Michalewicz, A hybrid particle swarm with a time-adaptive topology for constrained optimization, *Swarm Evol. Comput.* 18 (2014) 22–37.
- [6] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: *Proceedings of the IEEE Swarm Intelligence Symposium (SIS'07)*, 2007, pp. 120–127.
- [7] X. Cai, X. Gao, Y. Xue, Improved bat algorithm with optimal forage strategy and random disturbance strategy, *Int. J. Bio-Inspired Comput.* 8 (4) (2016) 205–214.
- [8] E. Camci, D. Kripalani, L. Ma, et al., An aerial robot for rice farm quality inspection with type-2 fuzzy neural networks tuned by particle swarm optimization-sliding mode control hybrid algorithm, *Swarm Evol. Comput.* (2017), <https://doi.org/10.1016/j.swevo.2017.10.003>.
- [9] G. Chen, X. Huang, J. Jia, et al., Natural exponential inertia weight strategy in particle swarm optimization, in: *Proceedings of the World Congress on Intelligent Control and Automation (WCICA'06)*, 2006, pp. 3672–3675.
- [10] G. Chen, J. Jia, Q. Han, Study on the strategy of decreasing inertia weight in particle swarm optimization algorithm, *J. Xi'an Jiaotong Univ.* 40 (1) (2006) 53–56.
- [11] K. Chen, F. Zhou, L. Yin, et al., A hybrid particle swarm optimizer with sine cosine acceleration coefficients, *Inf. Sci.* 422 (2018) 218–241.

- [12] M. Chih, C. Lin, M. Chern, Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem, *Appl. Math. Model.* 38 (4) (2014) 1338–1350.
- [13] L. Chuang, C. Hsiao, C. Yang, Chaotic particle swarm optimization for data clustering, *Expert Syst. Appl.* 38 (12) (2011) 14555–14563.
- [14] 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.
- [15] L. Coelho, A quantum particle swarm optimizer with chaotic mutation operator, *Chaos Solit. Fractals* 37 (5) (2008) 1409–1418.
- [16] Z. Cui, B. Sun, G. Wang, et al., A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems, *J. Parallel Distr. Comput.* 103 (2017) 42–52.
- [17] H. Cui, Q. Zhu, Convergence analysis and parameter selection in particle swarm optimization, *Comput. Eng. Appl.* 43 (23) (2007) 89–91.
- [18] P. Das, H. Behera, B. Panigrahi, A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning, *Swarm Evol. Comput.* 28 (2016) 14–28.
- [19] E. Davoodi, M. Hagh, S. Zadeh, A hybrid improved quantum-behaved particle swarm optimization-simplex method (IQPSOS) to solve power system load flow problems, *Appl. Soft Comput.* 21 (2014) 171–179.
- [20] S. Dey, S. Bhattacharyya, U. Maulik, Quantum inspired genetic algorithm and particle swarm optimization using chaotic map model based interference for gray level image thresholding, *Swarm Evol. Comput.* 15 (2014) 38–57.
- [21] M. Dorigo, L. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Trans. Evol. Comput.* 1 (1) (1997) 53–66.
- [22] R. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'00)*, 2000, pp. 84–88.
- [23] R. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'01)*, 2001, pp. 94–100.
- [24] C. Feng, S. Cong, X. Feng, A new adaptive inertia weight strategy in particle swarm optimization, in: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'07)*, 2007, pp. 4186–4190.
- [25] Y. Feng, Y. Yao, A. Wang, Comparing with chaotic inertia weights in particle swarm optimization, in: *Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC'07)*, 2007, pp. 329–333.
- [26] 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.
- [27] 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.
- [28] R. He, Y. Wang, Q. Wang, et al., An improved particle swarm optimization based on self-adaptive escape velocity, *J. Softw.* 16 (12) (2005) 2036–2044.
- [29] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.
- [30] B. Jiao, Z. Lian, X. Gu, A dynamic inertia weight particle swarm optimization algorithm, *Chaos Solit. Fractals* 37 (2008) 698–705.
- [31] J. Kennedy, R. Eberhart, Particle swarm optimization, in: *Proceedings of the IEEE International Conference on Neural Networks (ICNN'95)*, 1995, pp. 1942–1948.
- [32] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02)*, 2002, pp. 1671–1676.
- [33] S. Khan, A. Engelbrecht, A fuzzy particle swarm optimization algorithm for computer communication network topology design, *Appl. Intell.* 36 (1) (2012) 161–177.
- [34] M. Khurana, K. Massey, Swarm algorithm with adaptive mutation for airfoil aerodynamic design, *Swarm Evol. Comput.* 20 (2015) 1–13.
- [35] R. Kundu, S. Das, R. Mukherjee, et al., An improved particle swarm optimizer with difference mean based perturbation, *Neurocomputing* 129 (2014) 315–333.
- [36] S. Leung, Y. Tang, W. Wong, A hybrid particle swarm optimization and its application in neural networks, *Expert Syst. Appl.* 39 (1) (2012) 395–405.
- [37] J. Li, Y. Cheng, K. Chen, Chaotic particle swarm optimization algorithm based on adaptive inertia weight, in: *Proceedings of the Chinese Control and Decision Conference (CCDC'14)*, 2014, pp. 1310–1315.
- [38] Z. Li, W. Wang, Y. Yan, et al., PS-ABC: a hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems, *Expert Syst. Appl.* 42 (22) (2015) 8881–8895.
- [39] L. Li, B. Xue, B. Niu, et al., The novel non-linear strategy of inertia weight in particle swarm optimization, in: *Proceedings of the IEEE International Conference on Bio-Inspired Computing: Theories and Applications (BICTA'09)*, 2009, pp. 183–187.
- [40] 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.
- [41] 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.
- [42] J. Liang, B. Qu, P. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-parameter Single Objective Optimization, Technical Report, Nanyang Technological University (Singapore) and Zhengzhou University (China), 2014.
- [43] J. Liang, P. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: *Proceedings of the IEEE Swarm Intelligence Symposium (SIS'05)*, 2005, pp. 124–129.
- [44] W. Lim, N. Isa, Particle swarm optimization with increasing topology connectivity, *Eng. Appl. Artif. Intell.* 27 (2014) 80–102.
- [45] S. Ling, H. Lu, K. Chan, et al., Hybrid particle swarm optimization with wavelet mutation and its industrial applications, *IEEE Trans. Syst. Man Cybernet. Part B Cybernet.* 38 (3) (2008) 743–763.
- [46] H. Liu, A. Abraham, Fuzzy adaptive turbulent particle swarm optimization, in: *Proceedings of the International Conference on Hybrid Intelligent Systems (HIS'05)*, 2005, pp. 445–450.
- [47] N. Lynn, P. Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, *Swarm Evol. Comput.* 24 (2015) 11–24.
- [48] N. Lynn, P. Suganthan, Ensemble particle swarm optimizer, *Appl. Soft Comput.* 55 (2017) 533–548.
- [49] S. Majercik, Using fluid neural networks to create dynamic neighborhood topologies in particle swarm optimization, in: *Proceedings of the International Conference on Swarm Intelligence (ICSI'14)*, 2014, pp. 270–277.
- [50] R. Malik, T. Rahman, S. Hashim, et al., New particle swarm optimizer with sigmoid increasing inertia weight, *Int. J. Comput. Sci. Secur.* 1 (2) (2007) 35–44.
- [51] Y. Marinakis, A. Migdalas, A. Sifaleras, A hybrid particle swarm optimization – variable neighborhood search algorithm for constrained shortest path problems, *Eur. J. Oper. Res.* 261 (3) (2017) 819–834.
- [52] May R, Simple mathematical models with very complicated dynamics, *Nature* 261 (1976) 459–467.
- [53] R. Mendes, J. Kennedy, The fully informed particle swarm: simpler, maybe better, *IEEE Trans. Evol. Comput.* 8 (3) (2004) 204–210.
- [54] A. Meng, Z. Li, H. Yin, et al., Accelerating particle swarm optimization using crisscross search, *Inf. Sci.* 329 (2016) 52–72.
- [55] M. Neshat, FAIPSO: fuzzy adaptive informed particle swarm optimization, *Neural Comput. Appl.* 23 (1) (2013) 95–116.
- [56] A. Nickabadi, M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, *Appl. Soft Comput.* 11 (4) (2011) 3658–3670.
- [57] N. Otsu, A threshold selection method from gray-level histograms, *IEEE Trans. Syst. Man Cybernet.* 9 (1) (1979) 62–66.
- [58] Y. Peng, X. Peng, Z. Liu, Statistic analysis on parameter efficiency of particle swarm optimization, *Acta Electron. Sin.* 32 (2) (2004) 209–213.
- [59] M. Pluhacek, R. Senkerik, D. Davendra, Chaos particle swarm optimization with ensemble of chaotic systems, *Swarm Evol. Comput.* 25 (2015) 29–35.
- [60] 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.
- [61] A. Robati, G. Barani, H. Pour, et al., Balanced fuzzy particle swarm optimization, *Appl. Math. Model.* 36 (5) (2012) 2169–2177.
- [62] Z. Ruan, Y. Yuan, Q. Chen, et al., A new multi-function global particle swarm optimization, *Appl. Soft Comput.* 49 (2016) 279–291.
- [63] L. Sahoo, A. Banerjee, A. Bhunia, et al., An efficient GA-PSO approach for solving mixed-integer nonlinear programming problem in reliability optimization, *Swarm Evol. Comput.* 19 (2014) 43–51.
- [64] A. Sedki, D. Ouazar, Hybrid particle swarm optimization and differential evolution for optimal design of water distribution systems, *Adv. Eng. Inf.* 26 (3) (2012) 582–591.
- [65] F. Sheikholeslami, N. Navimipour, Service allocation in the cloud environments using multi-objective particle swarm optimization algorithm based on crowding distance, *Swarm Evol. Comput.* 35 (2017) 53–64.
- [66] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'98)*, 1998, pp. 69–73.
- [67] Y. Shi, R. Eberhart, Fuzzy adaptive particle swarm optimization, in: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'01)*, 2001, pp. 101–106.
- [68] H. Shieh, C. Kuo, C. Chiang, Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification, *Appl. Math. Comput.* 218 (2011) 4365–4383.
- [69] K. Suresh, N. Kumarappan, Hybrid improved binary particle swarm optimization approach for generation maintenance scheduling problem, *Swarm Evol. Comput.* 9 (2013) 69–89.
- [70] Z. Tang, D. Zhang, A modified particle swarm optimization with adaptive acceleration coefficients, in: *Proceedings of the Asia-Pacific Conference on Information Processing (APCIP'09)*, 2009, pp. 330–332.
- [71] M. Tanweer, S. Suresh, N. Sundararajan, Self regulating particle swarm optimization algorithm, *Inf. Sci.* 294 (2015) 182–202.
- [72] D. Tian, N. Li, Fuzzy particle swarm optimization algorithm, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'09)*, 2009, pp. 263–267.
- [73] D. Tian, T. Zhao, Particle swarm optimization algorithm based on fuzzy controller, *Comput. Eng. Des.* 31 (24) (2010) 5335–5338.
- [74] D. Tian, T. Zhao, Particle swarm optimization based on Tent map and Logistic map, *J. Shaanxi Univ. Sci. Technol.* 28 (2) (2010) 17–23.
- [75] D. Tian, Particle swarm optimization with chaos-based initialization for numerical optimization, *Intell. Autom. Soft Comput.* (2017) 1293881, <https://doi.org/10.1080/10798587.2017>.
- [76] A. Verma, S. Kaushal, A hybrid multi-objective particle swarm optimization for scientific workflow scheduling, *Parallel Comput.* 62 (2017) 1–19.

- [77] 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 Evol. Comput.* 8 (2013) 26–32.
- [78] G. Wang, X. Cai, Z. Cui, et al., High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm, *IEEE Trans. Emerg. Top. Comput.* (2017), <https://doi.org/10.1109/TETC.2017.2703784>.
- [79] H. Wang, Z. Cui, H. Sun, et al., Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism, *Soft Comput.* 21 (2017) 5325–5339.
- [80] C. Wang, Y. Liu, Y. Zhao, et al., A hybrid topology scale-free Gaussian-dynamic particle swarm optimization algorithm applied to real power loss minimization, *Eng. Appl. Artif. Intell.* 32 (2014) 63–75.
- [81] H. Wang, H. Sun, C. Li, et al., Diversity enhanced particle swarm optimization with neighborhood search, *Inf. Sci.* 223 (2013) 119–135.
- [82] H. Wang, W. Wang, H. Sun, et al., Firefly algorithm with random attraction, *Int. J. Bio-Inspired Comput.* 8 (1) (2016) 33–41.
- [83] H. Wang, W. Wang, Z. Wu, Particle swarm optimization with adaptive mutation for multimodal optimization, *Appl. Math. Comput.* 221 (2013) 296–305.
- [84] H. Wang, W. Wang, X. Zhou, et al., Firefly algorithm with neighborhood attraction, *Inf. Sci.* 382 (2017) 374–387.
- [85] L. Wang, B. Yang, J. Orchard, Particle swarm optimization using dynamic tournament topology, *Appl. Soft Comput.* 48 (2016) 584–596.
- [86] H. Wang, X. Zhou, H. Sun, et al., Firefly algorithm with adaptive control parameters, *Soft Comput.* 21 (17) (2017) 5091–5102.
- [87] Q. Wu, Cauchy mutation for decision-making variable of Gaussian particle swarm optimization applied to parameters selection of SVM, *Expert Syst. Appl.* 38 (5) (2011) 4929–4934.
- [88] J. Xin, G. Chen, Y. Hai, A particle swarm optimizer with multi-stage linearly-decreasing inertia weight, in: *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO'09)*, 2009, pp. 505–508.
- [89] P. Yadmellat, S. Salehizadeh, M. Menhaj, A new fuzzy inertia weight particle swarm optimization, in: *Proceedings of the International Conference on Computational Intelligence and Natural Computing (CINCO'09)*, 2009, pp. 507–510.
- [90] X. Yang, J. Yuan, J. Yuan, et al., A modified particle swarm optimizer with dynamic adaptation, *Appl. Math. Comput.* 189 (2007) 1205–1213.
- [91] Z. You, W. Chen, G. He, et al., Adaptive weight particle swarm optimization algorithm with constriction factor, in: *Proceedings of the International Conference on Information Science and Management Engineering (ISME'10)*, 2010, pp. 245–248.
- [92] H. Yu, L. Zhang, D. Chen, et al., Adaptive particle swarm optimization algorithm based on feedback mechanism, *J. Zhejiang Univ. Eng. Sci.* 39 (9) (2005) 1286–1291.
- [93] X. Yu, J. Liu, H. Li, An adaptive inertia weight particle swarm optimization algorithm for IIR digital filter, in: *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI'09)*, 2009, pp. 114–118.
- [94] Z. Zhan, J. Zhang, Y. Li, et al., Orthogonal learning particle swarm optimization, *IEEE Trans. Evol. Comput.* 15 (6) (2011) 832–847.
- [95] X. Zhang, Y. Du, G. Qin, et al., Adaptive particle swarm algorithm with dynamically changing inertia weight, *J. Xi'an Jiaotong Univ.* 39 (10) (2005) 1039–1042.
- [96] Q. Zhang, W. Liu, X. Meng, et al., Vector coevolving particle swarm optimization algorithm, *Inf. Sci.* 394 (2017) 273–298.
- [97] M. Zhang, H. Wang, Z. Cui, et al., Hybrid multi-objective cuckoo search with dynamical local search, *Memet. Comput.* (2017), <https://doi.org/10.1007/s12293-017-0237-2>.
- [98] L. Zhang, H. Yu, D. Chen, et al., Analysis and improvement of particle swarm optimization algorithm, *Inf. Control* 33 (5) (2004) 513–517.
- [99] Z. Zhao, S. Huang, W. Wang, Simplified particle swarm optimization algorithm based on stochastic inertia weight, *Appl. Res. Comput.* 31 (2) (2014) 361–364.
- [100] Y. Zheng, L. Ma, L. Zhang, et al., On the convergence analysis and parameter selection in particle swarm optimization, in: *Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC'03)*, 2003, pp. 1802–1807.