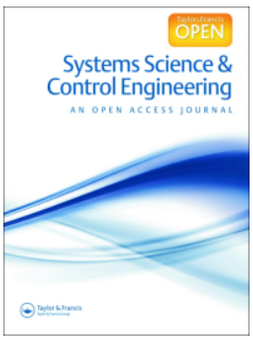

Systems Science & Control Engineering

An Open Access Journal

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tssc20

An improved particle swarm optimization algorithm with adaptive weighted delay velocity

Lin Xu, Baoye Song & Maoyong Cao

To cite this article: Lin Xu, Baoye Song & Maoyong Cao (2021) An improved particle swarm optimization algorithm with adaptive weighted delay velocity, Systems Science & Control Engineering, 9:1, 188-197, DOI: 10.1080/21642583.2021.1891153

To link to this article: https://doi.org/10.1080/21642583.2021.1891153

Published online: 26 Feb 2021.

SYSTEMS SCIENCE & CONTROL ENGINEERING: AN OPEN ACCESS JOURNAL
2021, VOL. 9, NO. 1, 188–197
https://doi.org/10.1080/21642583.2021.1891153

Taylor & Francis
Taylor & Francis Group

An improved particle swarm optimization algorithm with adaptive weighted delay velocity

Lin Xu, Baoye Song and Maoyong Cao

College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao, People's Republic of China

ABSTRACT

An improved particle swarm optimization (PSO) with adaptive weighted delay velocity (PSO-AWDV) is proposed in this paper. A new scheme blending weighted delay velocity is firstly presented for a new PSO with weighted delay velocity (PSO-WDV) algorithm. Then, to adaptively update the velocity inertia weight, an adaptive PSO-AWDV algorithm is developed based on the evolutionary state of the particle swarm evaluated via a new estimation method. The newly proposed adaptive PSO-AWDV algorithm is tested based on some famous benchmark functions, which can confirm that the performance of PSO-AWDV is superior to several well-known PSO variants and intelligent optimization algorithms in literature.

ARTICLE HISTORY
Received 2 January 2021
Accepted 12 February 2021

KEYWORDS
PSO; particle swarm optimization; adaptive weighted delay velocity; intelligent optimization algorithm

1. Introduction

Particle swarm optimization (PSO) is a meta-heuristic inte lligent optimization algorithm developed by Kennedy and Eberhart to mimic the behaviour of the biological swarms, such as bird flock and fish swarm (Kennedy & Eberhart, 1995). Up to now, PSO has been widely used in various optimization problems, e.g. controller design (Sain et al., 2020; Song et al., 2020), system identification (Liu et al., 2018; Sun et al., 2020), and robot path planning (Song et al., 2017, 2019), etc.

In PSO, the position vectors of the particles are regarded as the candidate optimal solutions of the optimization problem. By flying in the search space of the optimization problem, PSO can accomplish the fast exploration of the search space and converge to a global/local optimal solution. The velocity and position vectors of the classical PSO algorithm are updated by the following equations:

$$v_i(k+1) = wv_i(k) + c_1 r_1 (p_{il}(k) - x_i(k)) + c_2 r_2 (p_g((k) - x_i(k)), \quad (1)$$

$$x_i(k+1) = x_i(k) + v_i(k+1), \quad (2)$$

where $v_i(k)$ and $x_i(k)$ are, respectively, the velocity and position vectors of the ith particle at the kth iteration; w represents the velocity inertia weight; c_1 and c_2, respectively, indicate the acceleration factors that are also called cognitive and social parameters; $p_{il}(k)$ and $p_g((k)$ (whose optimal values are, respectively, denoted as *pbest* and *gbest*) stand for, respectively, the optimal positions that are ever experienced by the ith particle and the particle swarm up to the current iteration; r_1 and r_2 indicate the uniformly distributed random numbers between 0 and 1.

Since the birth of the classical PSO algorithm, a great amount of variants have been presented to improve its performance, e.g. PSO-LDIW (Shi & Eberhart, 1998), PSO-TVAC (Ratnaweera et al., 2004), FIPS (Mendes et al., 2004), CLPSO (Liang et al., 2006), PSO-ACO (Shuang et al., 2011), and PSO-SA (Behnamian & Ghomi, 2010), etc. These algorithms have promoted the performance of the classical PSO based on diverse improvement strategies. Particularly, to combat the defects of PSO such as premature convergence and local trapping, adaptive PSO algorithms have been under intensive investigations in recent years. For example, Zhan and his colleagues proposed an APSO (adaptive particle swarm optimization) algorithm (Zhan et al., 2009), which firstly evaluates the evolutionary state of the particle swarm in terms of the calculated evolutionary factor, and then switches the velocity updating function adaptively among a set of equations. Tang and Wang presented an SPSO (switching particle swarm optimization) algorithm (Tang et al., 2011a), which can predict the evolutionary state of the particle swarm based on Markov chain and evolutionary factor, and thereby the velocity updating function is regulated adaptively in accordance with the predicted evolutionary state. Song et al. developed an MDPSO (multimodal delayed particle swarm optimization) algorithm (Song et al., 2017),

CONTACT Baoye Song songbaoye@sdust.edu.cn; Maoyong Cao my-cao@263.net

which records all of the optima of the particles and the swarm in the search history, and some additional terms are adaptively blended into the velocity updating function on account of the evolutionary state. Although the aforementioned adaptive schemes can improve the performance of the classical PSO algorithm, they still have several shortcomings such as high computational complexity and large data storage.

In recent years, fractional calculus has been widely used in theory and engineering. Inspired by the fractional calculus, Pires et al. proposed an FOPSO (particle swarm optimization with fractional-order velocity) algorithm (Pires et al., 2010), where the fractional-order velocity is brought to the velocity updating function. Essentially, in FOPSO, the sum of some weighted historical velocities is blended into the velocity updating function, and the search performance of the algorithm can be regulated via the fractional order. In comparison with the classical PSO algorithm and some of its variants, this scheme can promote the search ability of the optimization algorithm, and meanwhile, there is no obvious growth on the computational burden. After that, some more improved PSOs are presented based on this scheme, which has also been applied to other intelligent optimization algorithms for their performance enhancement (Ates et al., 2017; Couceiro & Sivasundaram, 2016; Yousri et al., 2020; Yousri & Mirjalili, 2020).

In this paper, a new adaptive PSO with adaptive weighted delay velocity (PSO-AWDV) is developed to deal with the frequently appeared obstacles in the optimization, e.g. premature convergence and local trapping. The main contributions of this paper can be summarized as follows: (1) *A new PSO with weighted delay velocity (PSO-WDV) is proposed by blending the weighted delay velocity, which can provide some power for the particles to jump out of the local trap.* (2) *A new method is presented to evaluate the evolutionary state of the particle swarm.* (3) *Based on the estimated evolutionary state, an adaptive PSO with adaptive weighted delay velocity (PSO-AWDV) is developed by adaptively updating the velocity inertia weight of the PSO-WDV algorithm.* (4) *The superiority of the new PSO-AWDV algorithm is verified by several simulation tests on some famous benchmark functions.*

The remainder of the paper is organized as follows. The related works of PSO variants are described in Section 2. In Section 3, the PSO-WDV algorithm is firstly presented by blending the weighted delay velocity, and then based on the new evaluation method of the evolutionary state, a new adaptive PSO-AWDV algorithm is developed by adaptively updating the inertia weight of the velocity function. In Section 4, some benchmark functions are introduced, and then several simulation experiments are carried out to test the performance of the new adaptive PSO algorithm. Finally, the paper is concluded and future works are also pointed out.

2. Related works of PSO variants

Up to now, a great amount of PSO variants have been proposed to enhance the performance of original PSO algorithm. The main improvement strategy of PSO can be classified as parameter regulation, population size, and hybrid intelligent algorithm, etc.

Parameter regulation is to find the appropriate parameters of PSO algorithm via off-line experiment or update the parameters iteratively in on-line operation (Eiben et al., 1999). For example, HPSO (hierarchical PSO) was proposed in Janson and Middendorf (2005) to arrange the particles in a tree structure with predefined branching factors, where the structure of the tree was updated iteratively according to the *pbest* of swarm particles. In SFIPSO (scale-free fully informed PSO) (Zhang & Yi, 2011), the swarm particles were divided into active and inactive particles. The active particles can update their positions in the search space to find the optimal value, while the states of the inactive particles were not updated before activation. In Yang et al. (2007), the velocity factor and aggregation factor were calculated for the particle swarm, and the inertia weight was adaptively regulated according to the two factors. In Leu and Yeh (2012), GPSO (grey PSO) was presented to dynamically regulate the inertia and acceleration coefficients of the updating function based on the similarity between the swarm particles and the current global optimal particle.

As for the population size, LPSO (ladder PSO) was developed by Chen and his colleagues (Chen & Zhao, 2009), where the diversity of the swarm was evaluated by Manhattan distance and the size of the swarm was dynamically adjusted accordingly. Hsieh et al. proposed an EPUS-PSO (efficient population utilization strategy PSO), where the swarm size was dynamically adjusted via a number of swarm management strategies such as adding/deleting/replacing the swarm particles, thus the learning and sharing of the optimal position can be achieved for the particle swarm (Hsieh et al., 2009). In Zeng et al. (2020), a DNSPSO (dynamic-neighbourhood-based switching PSO) was proposed, where a new velocity updating scheme was designed to regulate the personal best position and the global best position in terms of a distance-based dynamic neighbourhood to make full use of the population evolution information among the entire swarm.

The scheme of hybrid intelligent algorithm includes a mixture of different strategies of PSOs and the hybridization of PSO and other intelligent optimization algorithms. There are a great number of references on this topic,

so we will not repeat them due to page limits. We refer interested readers to Tang et al. (2011b), Behnamian and Ghomi (2010) and Shuang et al. (2011) for more details.

3. Improved PSO algorithm

Inspired by the above-mentioned PSO improvement strategies, a new PSO with weighted delay velocity (PSO-WDV) is firstly proposed in this section. The updating equations of the new PSO are expressed as follows:

$$v_i(k+1) = wv_i(k) + (1-w)v_i(k-1) + c_1r_1(p_{il}(k) - x_i(k)) + c_2r_2(p_g((k) - x_i(k)), \quad (3)$$

$$x_i(k+1) = x_i(k) + v_i(k+1), \quad (4)$$

where w is the inertia weight of the velocity $v_i(k)$ and $w < 1$; $(1-w)$ is the inertia weight of the delayed velocity $v_i(k-1)$; and other parameters can be explained like Equations (1) and (2). It can be found that the updating functions of PSO-WDV are as same as the classical PSO except for one additional term, i.e. the delayed velocity and its inertia weight. Thus, the computational burden of the new PSO is not obviously increased in comparison with the classical PSO because only one weighted delay velocity is added. Besides, it should be mentioned that the PSO-WDV has simpler form and less amount of calculation than FOPSO algorithm for the same reason mentioned above.

Remark 3.1: It is worth pointing out that the PSO-WDV is actually a new scheme with weighted delay velocity, which can provide additional 'power' for the particles of the swarm to more likely fly out of the local trapping and achieve the global optimum. Based on this scheme, new variants of PSO can be developed by specifying the rule of parameter regulation. Motivated by the above considerations, a new adaptive PSO is presented in the remainder of this section.

The new adaptive scheme is developed according to the estimation of the evolutionary state of the particle swarm, thereby a new PSO with adaptive weighted delay velocity (PSO-AWDV) is presented on the basis of the updating functions Equations (3) and (4) of PSO-WDV, where the acceleration factors c_1 and c_2 are compatible with the PSO-TVAC algorithm, that is

$$c_1 = (c_{1i} - c_{1f}) \times \frac{k_{\max} - k}{k_{\max}} + c_{1f}, \quad (5)$$

$$c_2 = (c_{2i} - c_{2f}) \times \frac{k_{\max} - k}{k_{\max}} + c_{2f}, \quad (6)$$

where c_{1i} and c_{1f} indicate, respectively, the initial and final values of the acceleration factor c_1; likewise, the initial and final values of the acceleration factor c_2 are represented by c_{2i} and c_{2f}, respectively; k and $k_{\max}$ denote, respectively, the current and maximal iterations in the optimization. In particular, the inertia weight of velocity is adaptively regulated in accordance with the evolutionary state in the optimization, and it is calculated as follows:

$$w = 1 - \frac{a}{1 + e^{b \cdot E(k)}}, \quad (7)$$

where a and b are two parameters that can be designed to adjust the search performance of PSO; $E(k)$ is the estimation value (EV) of the evolutionary state at the kth iteration and can be computed as follows:

$$E(k) = \frac{|f_{\max}(k)| - |f_{\min}(k)|}{|f_{\max}(k)|}, \quad (8)$$

where $f_{\max}$ and $f_{\min}$ stand for, respectively, the maximal and minimal fitness values of the particles at the kth iteration; $|*|$ denotes the operation of absolute value. In sum, the PSO-AWDV algorithm is described in Algorithm 1.

Algorithm 1 PSO-AWDV algorithm

1: Initialize the parameters of the PSO-AWDV algorithm, including maximum iteration $k_{\max}$, parameters of the inertia weight a and b, parameters of the acceleration factor c_{1i}, c_{1f}, c_{2i}, and c_{2f};
2: **repeat**
3: Evaluate the fitness of every particles of the swarm and estimate the evolutionary state $E(k)$ of the particle swarm at the current iteration according to Equation (8);
4: Compare the fitness value of each particle with the *pbest* of itself and the *gbest* of the swarm, and select the corresponding better particle to replace the original one;
5: Compute the acceleration factors and inertia weight according to Equations (5)- -(7);
6: Update the velocity and position vectors of the particles at the current iteration according to Equations (3) and (4);
7: **until** Maximum iteration

Actually, the particles are scattered in the search space at the initial stage of the optimization. At that moment, the differences between the particles are obvious so that there are large gaps among the fitness values of the particles. In contrast, the particles of the swarm will gather to the global/local optima at the end of the optimization. Consequently, the fitness values of the particles are very similar with each other, and accordingly, the difference between the maximal and minimal fitness values will close to zero. As a result, we can conclude that the

evolutionary state of the particle swarm can be represented by the estimation value of Equation (8). Note that the mean Euclidean distance of the particles is used to estimate the evolutionary state of the particle swarm in some algorithms such as APSO, SPSO, and MDPSO, and then the updating functions are adaptively regulated in terms of the estimated evolutionary state. Nevertheless, the huge computational burden of the estimation makes it impractical in the real engineering. Comparatively, the estimation of the evolutionary state in Equation (8) has not only simpler formulation but also less computational burden.

Remark 3.2: It can be found from Equations (7) and (8) that the particle swarm will converge quickly due to the large velocity inertia weight of the current iteration when the particles are scattered in the swarm (i.e. EV $\rightarrow$ 1). By contrast, when the particles are concentrated (i.e. EV $\rightarrow$ 0), the inertia weight of the velocity at the current iteration is smaller than that of the delayed velocity, thereby this property can postpone the fast convergence of the particles to the current optimal solution. Additionally, owing to the directions of the current and delayed velocities are usually different with each other, the trapped particles can be given more 'power' to jump out of the local trapping so as to explore the search space more thoroughly.

4. Performance evaluation of PSO-AWDV

4.1. Benchmark functions for test

In this section, the performance of the above-mentioned adaptive PSO algorithm is evaluated by several frequently used benchmark functions that are expressed as follows:

$$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2, \tag{9}$$

$$f_2(\vec{x}) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|, \tag{10}$$

$$f_3(\vec{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2, \tag{11}$$

$$f_4(\vec{x}) = \max_i \left\{ |x_i|, 1 \le i \le D \right\}, \tag{12}$$

$$f_5(\vec{x}) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], \tag{13}$$

$$f_6(\vec{x}) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10], \tag{14}$$

$$f_7(\vec{x}) = -20\exp\left[-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2} \right] \tag{15}$$

$$- \exp\left[\frac{1}{n}\sum_{i=1}^{D} \cos(2\pi x_i) \right] + 20 + e, \tag{16}$$

$$f_8(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1, \tag{17}$$

$$f_9(\vec{x}) = \frac{\pi}{n}\left\{ 10\sin(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2 \right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4), \tag{18}$$

$$y_i = 1 + \frac{x_i + 1}{4},$$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

$$f_{10}(\vec{x}) = \frac{1}{10}\left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_D - 1)^2 [1 + \sin^2(2\pi x_d)] \right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4), \tag{19}$$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

where $\vec{x}$ denotes the solution vector of the optimization problem; $x_i (i = 1, 2, \ldots, D)$ is the component of the ith dimension, and D denotes the dimension of the solution vector. In these benchmark functions, $f_1(\vec{x})$–$f_5(\vec{x})$ are unimodal functions that are usually used to test the accuracy and speed of the intelligent optimization algorithms. Specifically, $f_1(\vec{x})$ is called *Sphere function*, whose global optimum value and corresponding solution are respectively 0 and $[0]^D$. $f_2(\vec{x})$–$f_4(\vec{x})$ are three *Schwefel* functions with global optimum value 0 and corresponding solution $[0]^D$. $f_5(\vec{x})$ is the famous *Rosenbrock* function, which can be a unimodal or multimodal function in the cases of different dimensions. Its global optimum value and solution are, respectively, 0 and $[1]^D$. The multimodal functions $f_6(\vec{x})$–$f_{10}(\vec{x})$, all of which have the global optimum value 0 and optimum solution $[0]^D$, are usually used to test

Table 1. Configuration of benchmark function.

Functions	Search space	Optimal solution	Optimum	Threshold
$f_1(\vec{x})$: *Sphere*	$[-100, 100]^D$	$[0]^D$	0	0.01
$f_2(\vec{x})$: *Schwefel 2.22*	$[-10, 10]^D$	$[0]^D$	0	0.01
$f_3(\vec{x})$: *Schwefel 1.2*	$[-100, 100]^D$	$[0]^D$	0	0.01
$f_4(\vec{x})$: *Schwefel 2.21*	$[-100, 100]^D$	$[0]^D$	0	0.01
$f_5(\vec{x})$: *Rosenbrock*	$[-30, 30]^D$	$[1]^D$	0	10
$f_6(\vec{x})$: *Rastrigin*	$[-5.12, 5.12]^D$	$[0]^D$	0	10
$f_7(\vec{x})$: *Ackley*	$[-32, 32]^D$	$[0]^D$	0	0.01
$f_8(\vec{x})$: *Griewank*	$[-600, 600]^D$	$[0]^D$	0	0.1
$f_9(\vec{x})$: *Penalized 1*	$[-50, 50]^D$	$[0]^D$	0	0.01
$f_{10}(\vec{x})$: *Penalized 2*	$[-50, 50]^D$	$[0]^D$	0	0.01

the ability of algorithm to jump out of the local trapping. Particularly, $f_6(\vec{x})$ is the *Rastrigin* function, which has a great amount of local optima that are similar to the global optimum, thereby the diversity of the particles should be enhanced in order to obtain the global optimal solution. $f_7(\vec{x})$ is the *Ackley* function, which has many local optima that are distinctly different from the global optimum. $f_8(\vec{x})$ is the *Griewank* function, which is hard to attain the theoretical optimum in the optimization because the components of each dimension are related with each other. As for the two *Penalized* functions $f_9(\vec{x})$ and $f_{10}(\vec{x})$, they are also quite difficult to obtain the global optimal solution for the misleading local optima that are similar to the global optimum. The configurations of the benchmark functions are illustrated in Table 1, where the column of search space gives the range of each dimension; and the threshold listed in the last column is used to judge whether a successful solution is obtained.

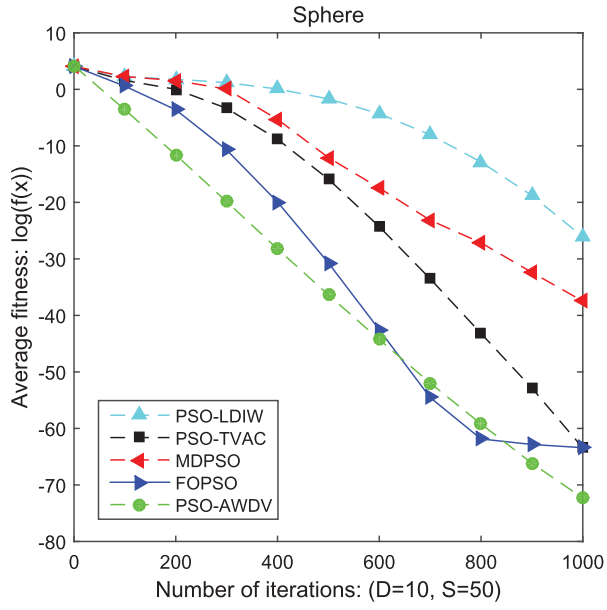

Figure 1. Average fitness of *Sphere* function.

4.2. Result and discussion

In this section, the performance of PSO-AWDV is evaluated using above benchmark functions. The main configurations of the computer for the algorithm test are as follows: Window 7 (64bit) and MATLAB R2014b for the software environment, Intel® Core™ i7-6700 CPU @ 3.40GHz and 16.0GB RAM for the hardware environment.

Apparently, the PSO-AWDV algorithm is originated from the algorithms of PSO-TVAC and FOPSO, and it is closely related to the algorithms of PSO-LDIW and MDPSO. As such, in the following performance evaluation, the PSO-AWDV is firstly compared with these related PSO algorithms, whose performances have been verified in many studies. Without loss of generality, the parameters in the tests are taken as follows: $D = 10$ for dimension, $S = 50$ for swarm size, $T = 1000$ for iteration time, and $R = 100$ for the repetition time of each test. Additionally, the parameters of the PSO-AWDV algorithm are set as follows: $a = 0.9$, $b = 0.5$, $c_{1i} = c_{2f} = 2.5$, $c_{2i} = c_{1f} = 0.5$, and the readers can refer to related references for more details on the parameters of other PSOs (Pires et al., 2010; Ratnaweera et al., 2004; Shi & Eberhart, 1998; Song et al., 2017).

Figures 1–10 show the variation of the average optimal fitness values (logarithm) of the above benchmark functions in each iteration, where the test results of various PSO algorithms are described with different marks and line styles. It is apparent that the PSO-AWDV proposed in this paper outperforms other PSOs. For example, in the tests of five unimodal benchmark functions, PSO-AWDV can converge more quickly and obtain better optimal fitness value than other PSOs. In the tests of five multimodal benchmark functions, the convergence speed and obtainable optimal fitness value of the PSO-AWDV are superior to those of other PSO algorithms except for few values that are similar to those of *Ackley* and *Penalized* functions.

The aforementioned conclusions can also be confirmed by the statistical results illustrated in Table 2, where more information about the test results can be seen, including optimal value (OV), average optimal value (AOV), standard deviation (SD), and success rate (SR).

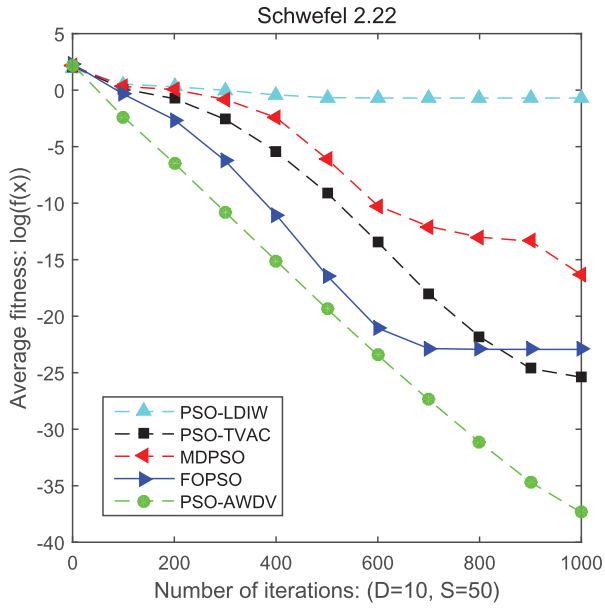

Figure 2. Average fitness of *Schwefel 2.22* function.

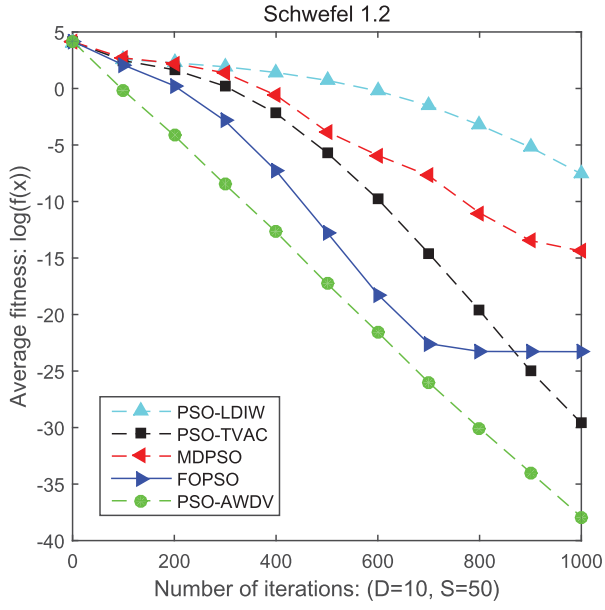

Figure 3. Average fitness of *Schwefel 1.2* function.

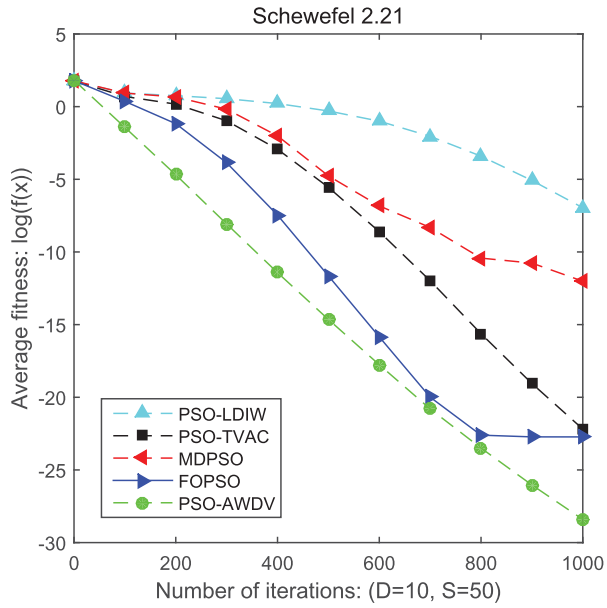

Figure 4. Average fitness of *Schwefel 2.21* function.

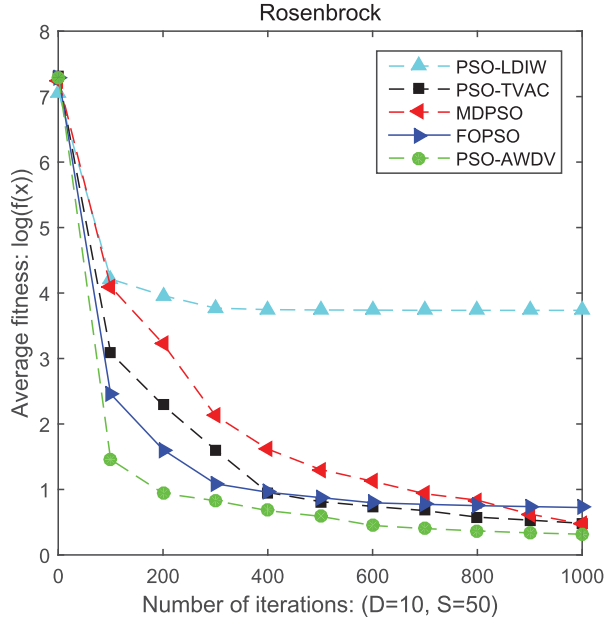

Figure 5. Average fitness of *Rosenbrock* function.

It can be obviously found that PSO-AWDV has smaller AOVs than other PSOs on most benchmark functions, and meanwhile the SDs of PSO-AWDV are also superior to other ones. From these properties, we can conclude that the search performance of the new algorithm is more stable than other PSO variants. Besides, it is worth noting that PSO-AWDV can attain better OVs for most of the benchmark functions, not to mention the superior SRs of PSO-AWDV on some harsh optimization problems (e.g. $f_8(\vec{x})$).

To further confirm the superiority of the newly developed PSO algorithm, more simulation experiments are implemented to compare PSO-AWDV algorithm with some other famous intelligent optimization algorithms proposed in recent years, including ABC (Artificial Bee Colony) algorithm (Karaboga & Basturk, 2007), BBO (Biogeography-based Optimization) algorithm (Simon, 2008), GSA (Gravitational Search Algorithm) (Rashedi et al., 2009), and SCA (Sine Cosine Algorithm) (Mirjalili, 2016). To make the comparison on the same basis, the parameters in the test are compatible with that of above

Table 2. Statistical results of related PSO algorithms.

		PSO-LDIW	PSO-TVAC	MDPSO	FOPSO	PSO-AWDV
$f_1(\vec{x})$	OV	2.07e−31	5.97e−68	1.62e−66	3.37e−96	1.57e−77
	AOV	8.18e−27	3.62e−64	3.33e−38	3.92e−64	4.67e−73
	SD	1.69e−26	8.99e−64	2.52e−37	3.92e−63	2.04e−72
	SR	100%	100%	100%	100%	100%
$f_2(\vec{x})$	OV	6.83e−18	7.31e−36	3.96e−35	4.65e−42	1.86e−40
	AOV	2.00e−1	3.98e−26	4.75e−17	1.15e−23	4.26e−38
	SD	1.40	2.58e−25	4.48e−16	9.20e−23	2.15e−37
	SR	98%	100%	100%	100%	100%
$f_3(\vec{x})$	OV	2.61e−11	1.64e−36	4.67e−29	1.05e−51	1.67e−46
	AOV	2.75e−8	2.48e−30	4.14e−15	5.17e−24	1.25e−28
	SD	5.31e−8	2.03e−29	3.54e−14	4.31e−23	7.58e−38
	SR	100%	100%	100%	100%	100%
$f_4(\vec{x})$	OV	1.66e−9	1.83e−26	1.91e−22	7.42e−37	1.21e−31
	AOV	1.10e−7	6.50e−23	9.70e−13	1.87e−23	3.93e−29
	SD	1.90e−7	3.36e−22	8.25e−12	1.24e−22	8.27e−29
	SR	100%	100%	100%	100%	100%
$f_5(\vec{x})$	OV	6.51e−3	5.83e−9	7.89e−4	2.13e−4	2.29e−3
	AOV	5.43e + 3	3.03	2.96	5.29	2.07
	SD	2.14e + 4	4.72	4.61	2.39	1.24
	SR	72%	96%	95%	94%	97%
$f_6(\vec{x})$	OV	1.48e−2	9.94e−1	0.00	0.00	0.00
	AOV	3.59	4.10	4.69	3.96	2.82
	SD	1.81	1.83	2.33	2.31	1.92
	SR	100%	100%	99%	98%	100%
$f_7(\vec{x})$	OV	2.66e−15	2.66e−15	2.66e−15	2.66e−15	2.66e−15
	AOV	2.86e−14	3.69e−15	3.80e−15	3.41e−15	2.80e−15
	SD	3.45e−14	1.62e−15	1.66e−15	1.45e−15	6.99e−16
	SR	100%	100%	100%	100%	100%
$f_8(\vec{x})$	OV	1.96e−2	1.23e−2	0.00	0.00	0.00
	AOV	7.74e−2	6.39e−2	6.48e−2	5.94e−2	3.46e−2
	SD	3.40e−2	3.07e−2	3.22e−2	3.09e−2	2.27e−2
	SR	81%	87%	87%	90%	99%
$f_9(\vec{x})$	OV	6.26e−32	4.71e−32	4.71e−32	4.71e−32	4.71e−32
	AOV	1.31e−28	4.71e−32	4.73e−32	4.71e−32	4.71e−32
	SD	2.98e−28	4.95e−47	2.32e−33	4.95e−47	4.95e−47
	SR	100%	100%	100%	100%	100%
$f_{10}(\vec{x})$	OV	1.03e−30	1.34e−32	1.34e−32	1.34e−32	1.34e−32
	AOV	6.67e−26	1.34e−32	1.34e−32	1.34e−32	1.34e−32
	SD	5.32e−25	2.47e−47	2.47e−47	2.47e−47	2.47e−47
	SR	100%	100%	100%	100%	100%

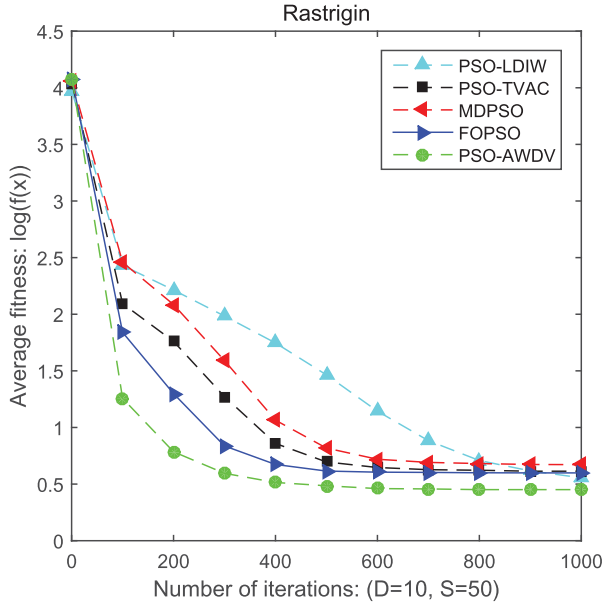

Figure 6. Average fitness of *Rastrigin* function.

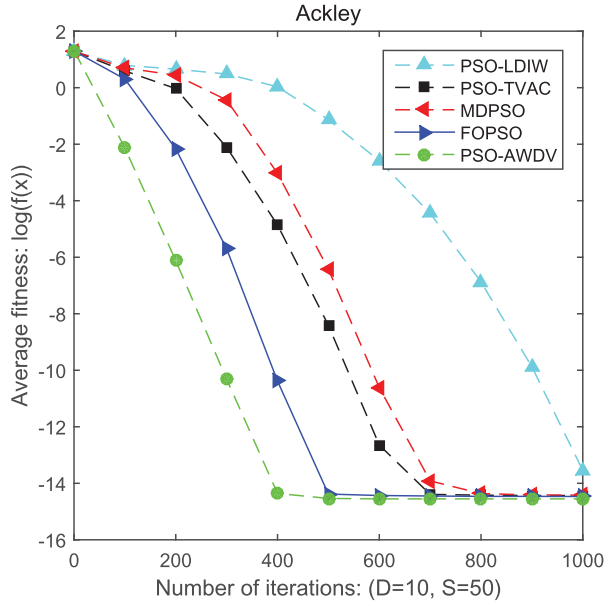

Figure 7. Average fitness of *Ackley* function.

Table 3. Statistical results of different intelligent optimization algorithms.

		ABC	BBO	GSA	SCA	PSO-AWDV
$f_1(\vec{x})$	AOV	1.00e−16	4.18e−2	2.64e−18	2.77e−28	0
	SD	4.04e−17	4.07e−2	1.13e−18	7.74e−28	0
$f_2(\vec{x})$	AOV	3.80e−16	2.61e−2	4.58e−9	4.66e−20	1.93e−37
	SD	8.94e−17	2.05e−2	9.27e−10	1.41e−19	2.15e−37
$f_3(\vec{x})$	AOV	5.37e+1	1.49e+2	6.21e−6	6.75e−10	9.87e−37
	SD	5.09e+1	8.31e+1	2.77e−5	2.98e−9	1.98e−36
$f_4(\vec{x})$	AOV	1.93e−1	6.87e−1	1.11e−9	3.95e−9	1.36e−27
	SD	1.26e−1	1.99e−1	2.16e−10	1.10e−8	3.85e−27
$f_5(\vec{x})$	AOV	4.29e−1	1.16e+2	5.49	7.11	2.61
	SD	4.77e−1	2.51e+2	1.41e−1	2.54e−1	5.02
$f_6(\vec{x})$	AOV	0	1.01e+2	4.62	2.11e−9	4.02
	SD	0	1.06	1.77	9.45e−9	2.53
$f_7(\vec{x})$	AOV	9.41e−15	9.15e−2	2.17e−9	7.63e−15	3.01e−15
	SD	2.41e−15	7.04e−2	4.54e−10	1.35e−14	1.09e−15
$f_8(\vec{x})$	AOV	5.43e−3	1.04e−1	3.64e−2	6.23e−2	3.32e−2
	SD	7.66e−3	4.24e−2	6.26e−2	1.50e−1	2.03e−2
$f_9(\vec{x})$	AOV	1.01e−16	2.20e−3	5.35e−20	7.23e−2	4.71e−32
	SD	3.87e−17	4.80e−3	2.04e−20	3.70e−2	5.61e−48
$f_{10}(\vec{x})$	AOV	1.85e−9	6.61e−3	1.21e−32	2.35e−1	1.34e−32
	SD	3.36e−9	6.98e−3	1.45e−32	6.30e−2	2.80e−48

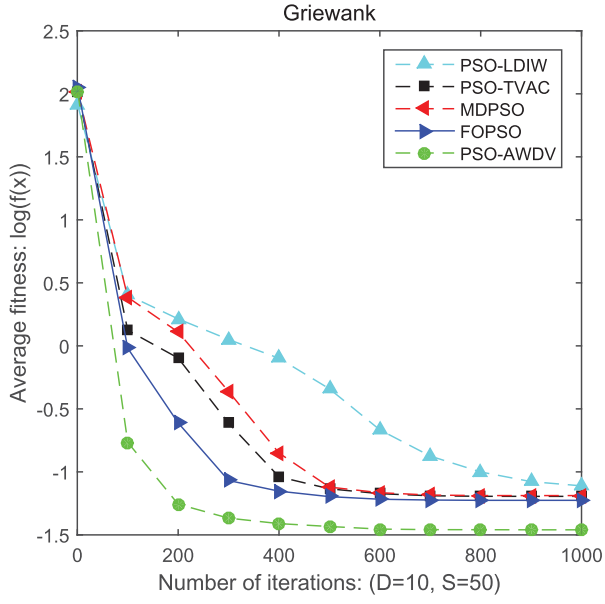

Figure 8. Average fitness of *Griewank* function.

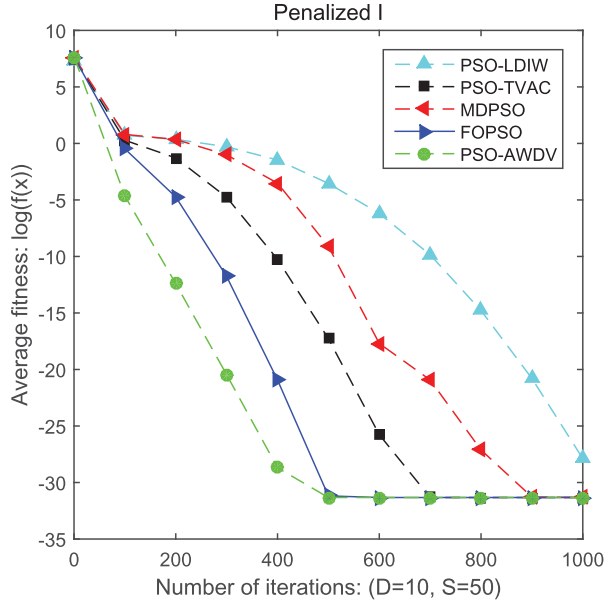

Figure 9. Average fitness of *Penalized I* function.

references as follows: $D = 10$ for dimension, $S = 40$ for swarm size, $T = 1000$ for iteration time, and $R = 20$ for the repetition time of each test (Chen et al., 2018).

The statistical results are illustrated in Table 3 for different intelligent optimization algorithms, including AOV and SD of the test on above benchmark functions, where the value less than 10^{-50} is replaced by 0 for the sake of simplicity. In the test of five unimodal benchmark functions, PSO-AWDV outperforms other algorithms on AOV and SD for the benchmark functions $f_1(\vec{x})$–$f_4(\vec{x})$, and ABC algorithm is superior to others for $f_5(\vec{x})$ function. In the test of five multimodal benchmark functions, ABC and SCA algorithms can achieve better results for $f_6(\vec{x})$ function, especially the results of ABC algorithm are much better than those of others. For $f_7(\vec{x})$ and $f_8(\vec{x})$ functions, the performances of PSO-AWDV and ABC can be ranked at the top two of the algorithms. For $f_9(\vec{x})$ function, the superiority of PSO-AWDV is very obvious in contrast to other algorithms. Finally, PSO-AWDV and GSA have similar performances on AOV of $f_{10}(\vec{x})$ function, but the former can obtain better SD than latter. To sum up, PSO-AWDV developed in this paper can outperform above-mentioned intelligent optimization algorithms.

5. Conclusions

In this paper, a PSO-WDV algorithm is firstly presented based on several successful improved PSO algorithms, where the weighted delay velocity can provide additional 'power' for the particles of the swarm to jump out in

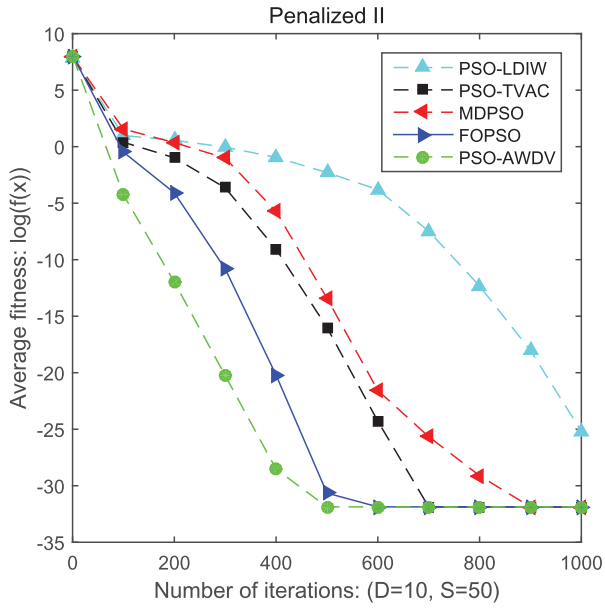

Figure 10. Average fitness of *Penalized II* function.

the case of local trapping. Then, to adaptively regulate the velocity inertia weight, a new adaptive scheme is proposed based on the estimation of the evolutionary state of the particle swarm. Consequently, a PSO-AWDV algorithm is developed in terms of the new adaptive scheme, which has not only simpler formulation but also less computation than other methods. Finally, the performances of the new adaptive PSO algorithm are evaluated by several simulation experiments on some famous benchmark functions, and the test results can confirm the superiority of the PSO-AWDV algorithm to other well-known PSO variants and intelligent optimization algorithms.

In future work, we will pay more attentions to the applications of the newly proposed PSO algorithm (Zeng et al., 2019), and the scheme of adaptive weighted delay velocity on other intelligent optimization algorithms (Zeng et al., 2021).

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported in part by the National Natural Science Foundation of China [grant number 61703242].

References

Ates, A., Alagoz, B. B., Kavuran, G., & Yeroglu, C. (2017). Implementation of fractional order filters discretized by modified fractional order Darwinian particle swarm optimization. *Measurement*, *107*, 153–164. https://doi.org/10.1016/j.measurement.2017.05.017

Behnamian, J., & Ghomi, S. M. T. F. (2010). Development of a PSO-SA hybrid metaheuristic for a new comprehensive regression model to time-series forecasting. *Expert Systems with Applications*, *37*(2), 974–984. https://doi.org/10.1016/j.eswa.2009.05.079

Chen, D. B., & Zhao, C. X. (2009). Particle swarm optimization with adaptive population size and its application. *Applied Soft Computing*, *9*(1), 39–48. https://doi.org/10.1016/j.asoc.2008.03.001

Chen, K., Zhou, F., Yin, L., Wang, S., Wang, Y., & Wan, F. (2018). A hybrid particle swarm optimizer with sine cosine acceleration coefficients. *Information Sciences*, *422*, 218–241. https://doi.org/10.1016/j.ins.2017.09.015

Couceiro, M., & Sivasundaram, S. (2016). Novel fractional order particle swarm optimization. *Applied Mathematics and Computation*, *283*, 36–54. https://doi.org/10.1016/j.amc.2016.02.007

Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *3*(2), 124–141. https://doi.org/10.1109/4235.771166

Hsieh, S. T., Sun, T. Y., Liu, C. C., & Tsai, S. J. (2009). Efficient population utilization strategy for particle swarm optimizer. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *39*(2), 444–456. https://doi.org/10.1109/TSMCB.2008.2006628

Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *35*(6), 1272–1282. https://doi.org/10.1109/TSMCB.2005.850530

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–470. https://doi.org/10.1007/s10898-007-9149-x

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Network* (pp. 1942–1948). IEEE Press.

Leu, M.-S., & Yeh, M.-F. (2012). Grey particle swarm optimization. *Applied Soft Computing*, *12*(9), 2985–2996. https://doi.org/10.1016/j.asoc.2012.04.030

Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal function. *IEEE Transactions on Evolutionary Computation*, *10*(3), 281–295. https://doi.org/10.1109/TEVC.2005.857610

Liu, L., Shan, L., Jiang, C., Dai, Y., Liu, C., & Qi, Z. (2018). Parameter identification of the fractional-order systems based on a modified PSO algorithm. *Journal of Southeast University (English Edition)*, *34*(1), 6–14.

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, *8*(3), 204–210. https://doi.org/10.1109/TEVC.2004.826074

Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, *96*, 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

Pires, E. J. S., Machado, J. A. T., Oliveira, P. B. M., Cunha, J. B., & Mendes, L. (2010). Particle swarm optimization with fractional-order velocity. *Nonlinear Dynamics*, *61*, 295–301. https://doi.org/10.1007/s11071-009-9649-y

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information Sciences*, *179*(13), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

Ratnaweera, A., Halgamure, S. K., & Watson, H. C. (2004). Self-organizing herarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, *8*(3), 240–255. https://doi.org/10.1109/TEVC.2004.826071

Sain, C., Banerjee, A., Biswas, P. K., Babu, T. S., & Dragicevic, T. (2020). Updated PSO optimised fuzzy-PI controlled buck type multi-phase inverter-based PMSM drive with an over-current protection. *IET Electric Power Applications*, *14*(12), 2331–2339. https://doi.org/10.1049/elp2.v14.12

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Proceedings of IEEE International Conference on Evolutionary Computation* (pp. 69–73). IEEE Press.

Shuang, B., Chen, J., & Li, Z. (2011). Study on hybrid PS-ACO algorithm. *Applied Intelligence*, *34*(1), 64–73. https://doi.org/10.1007/s10489-009-0179-6

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, *12*(6), 702–713. https://doi.org/10.1109/TEVC.2008.919004

Song, B., Wang, Z., & Zou, L. (2017). On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm. *Cognitive Computation*, *9*, 5–17. https://doi.org/10.1007/s12559-016-9442-4

Song, B., Wang, Z., Zou, L., Xu, L., & Alsaadi, F. E. (2019). A new approach to smooth global path planning of mobile robots with kinematic constraints. *International Journal of Machine Learning and Cybernetics*, *10*, 107–119. https://doi.org/10.1007/s13042-017-0703-7

Song, B., Xiao, Y., & Xu, L. (2020). Design of PI controller for brushless DC motor based on PSO-GSA algorithm. *Systems Science and Control Engineering*, *8*(1), 67–77. https://doi.org/10.1080/21642583.2020.1723144

Sun, Z., Jiao, X., & Xue, J. (2020). Prescribed performance control based on PSO identification and disturbance observer for automotive electronic throttle system with actuator constraint. *Control Engineering and Applied Informatics*, *22*(1), 24–31.

Tang, Y., Wang, Z., & Fang, J. (2011a). Parameters identification of unknown delayed genetic regulatory networks by a switching particle swarm optimization algorithm. *Expert Systems with Applications*, *38*, 2523–2535. https://doi.org/10.1016/j.eswa.2010.08.041

Tang, Y., Wang, Z., & Fang, J. (2011b). Feedback learning particle swarm optimization. *Applied Soft Computing*, *11*(8), 4713–4725. https://doi.org/10.1016/j.asoc.2011.07.012

Yang, X., Yuan, J., & Mao, H. (2007). A modified particle swarm optimizer with dynamic adaptation. *Applied Mathematics and Computation*, *189*, 1205–1213. https://doi.org/10.1016/j.amc.2006.12.045

Yousri, D., Elaziz, M. A., & Mirjalili, S. (2020). Fractional-order calculus-based flower pollination algorithm with local search for global optimization and image segmentation. *Knowledge-Based Systems*, *197*, 105889. https://doi.org/10.1016/j.knosys.2020.105889

Yousri, D., & Mirjalili, S. (2020). Fractional-order cuckoo search algorithm for parameter identification of the fractional-order chaotic, chaotic with noise and hyper-chaotic financial systems. *Engineering Applications of Artificial Intelligence*, *92*, 103662. https://doi.org/10.1016/j.engappai.2020.103662

Zeng, N., Song, D., Li, H., You, Y., Liu, Y., & Alsaadi, F. E. (2021). A competitive mechanism integrated multi-objective whale optimization algorithm with differential evolution. *Neurocomputing*, *432*, 170–182. https://doi.org/10.1016/j.neucom.2020.12.065

Zeng, N., Wang, Z., Liu, W., Zhang, H., Hone, K., & Liu, X. (2020). A dynamic neighborhood-based switching particle swarm optimization algorithm. *IEEE Transactions on Cybernetics*. doi:10.1109/TCYB.2020.3029748.

Zeng, N., Wang, Z., Zhang, H., Kim, K. E., & Liu, X. (2019). An improved particle filter with a novel hybrid proposal distribution for quantitative analysis of gold immunochromatographic strips. *IEEE Transactions on Nanotechnology*, *18*(1), 819–829. https://doi.org/10.1109/TNANO.7729

Zhan, Z., Zhang, J., Li, Y., & Chung, H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, *39*(6), 1362–1381. https://doi.org/10.1109/TSMCB.2009.2015956

Zhang, C., & Yi, Z. (2011). Scale-free fully informed particle swarm optimization algorithm. *Information Sciences*, *181*(20), 4550–4568. https://doi.org/10.1016/j.ins.2011.02.026