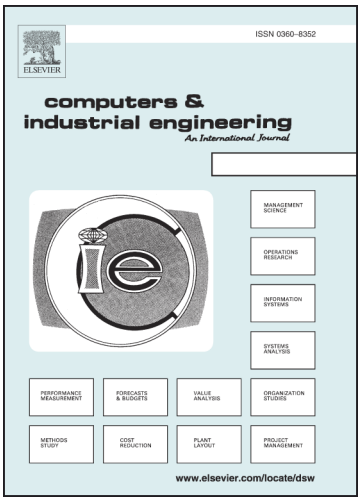


Particle Swarm Optimization with Fitness Adjustment Parameters

Shu-Fen Li, Chen-Yang Cheng

PII: S0360-8352(17)30258-9
DOI: http://dx.doi.org/10.1016/j.cie.2017.06.006
Reference: CAIE 4775

To appear in: *Computers & Industrial Engineering*

Please cite this article as: Li, S-F., Cheng, C-Y., Particle Swarm Optimization with Fitness Adjustment Parameters, *Computers & Industrial Engineering* (2017), doi: http://dx.doi.org/10.1016/j.cie.2017.06.006

Novel particle swarm optimization algorithm with evolutionary parameters based on velocity information

Shu-Fen Li

Department of Industrial Engineering & Enterprise Information

Tunghai University, Taichung, Taiwan

Tel: (+886) 4-2359-4319 ext.113, Email: fennieli@thu.edu.tw

Chen-Yang Cheng†

Department of Industrial Engineering & Management

National Taipei University of Technology, Taipei, Taiwan

Tel: (+886) 2-2771-2171 ext.2341, Email: cycheng@ntut.edu.tw

Particle Swarm Optimization with Fitness Adjustment Parameters

Abstract

Particle swarm optimization (PSO) has been widely applied in solving optimization problems because of its simple execution with fast convergence and high solution quality as known. Previous research has observed the effect of PSO parameters on a particle's movement and the solution performance. Most PSO variants constructed the search strategy of evolution by controlling the movement step. The movement steps of the particles should usually be large in early evolution for exploration and become smaller in the late evolution for exploitation. Therefore, this study proposes a novel PSO algorithm based on the fitness performance (PSOFAP) of particles for rapid convergence to an approximate optimal solution. The experiment is verified through twelve benchmark problems and the results are compared with those of other PSO variants. Furthermore, a well-known nonparametric statistical analysis method, namely the Wilcoxon signed rank test, is applied to demonstrate the performance of the proposed PSOFAP algorithm. The results of the experiment and statistical analysis show that PSOFAP is effective in enhancing the convergence speed, increasing the solution quality, and accurately adapting the parameter value without performing parametric sensitivity analysis.

***Keywords*:** adaptive particle swarm optimization, self-adaptive parameters, fitness performance, free-parameters

1. Introduction

In 1995, Eberhart and Kennedy first proposed the standard particle swarm optimization (PSO). PSO algorithm is a widely applied meta-heuristic algorithm for solving optimization problems, such as resource planning, production scheduling and planning, and image enhancement (Omran, Salman, & Engelbrecht, 2006; Shieh, Kuo, & Chiang, 2011; Sun & Wu, 2011; Zhang, Huang, & Liu, 2007). PSO is a population-based heuristic algorithm that uses a particle swarm, where the position of each particle in the search space directly represents the decision variables of the problem. Each particle updates its velocity and position on the basis of the original movement velocity, the personal best experience, and the global best experience. Considering the balance between the global search and local search strategies in PSO, three parameters are designed for adjusting the influence on the movement velocity of the next iteration: the inertia weight (w), the acceleration of self-cognition (c_1), and the acceleration of social cognition (c_2).

Numerous studies have demonstrated that the inertia weight and two cognition acceleration coefficients are critical in determining the solution quality and convergence speed of PSO (Clerc, 2005; Clerc & Kennedy, 2002; Holland, 1975; Jiang, Luo, & Yang, 2007; Olorunda & Engelbrecht, 2008; Omran *et al*., 2006). Some studies focused on developing methods for selecting the appropriate parameter value. Chatterjee and Siarry (2006); Eiben, Hinterding, & Michalewicz (1999); Shi and Eberhart (1998) denoted two ways of adjusting parameters, parameter tuning and parameter control. Parameter tuning is the commonly practiced approach that finds the appropriate values for parameters before the run of the algorithm, which remain consistent value during evolution. On the other hand, parameter control starts with initial parameter values which are changed during the iterations. The parameter value can be controlled by using adaptively the feedback of solution performance or a simple linear (or nonlinear) function in the evolution. Yasuda and Iwasaki (2004) and Xu (2013) proposed a parameter control based on the actual and ideal velocity information of each particle. Xu (2013) verified that using a nonlinear ideal velocity equation enables obtaining a superior solution to that obtained using a linear equation. These two forms of setting parameter values are both based on the assumption that a better movement of a particle is a larger step in the beginning and getting smaller in the end of the searching.

Most studies on PSO with a self-adaptive parameter value adjust the parameter value through a tuning mechanism that retains the concept of time-varying the parameter value. For example, the inertia weight unit of APSOVI (2013) is used to adjust the velocity to approximate its ideal value, which is a time-based cosine function. All particles apply the same set of parameter values in each iteration of the search process, regardless of whether the individual performance is good or bad. However, particles with higher performance could overshoot and be far from the nearby approximate optimum, which considerably increases the duration of the search process. To improve the efficiency of the search process, this study

focus on developing a method to change the consistent parameter values applied over all particles. The proposed PSOFAP algorithm is constructed on the basis of a fitness-varying ideal velocity function (a transformation cosine function). To increase the efficiency of searching for approximate optimums, PSOFAP obtains the self-adaptive parameter values of particles using the PSO algorithm. The performance of PSOFAP was verified through twelve benchmark problems and the results were compared with those of four other PSO variants by applying the Wilcoxon's test to the proposed PSOFAP algorithm and the APSOVI algorithm. The remaining of this paper is organized as follows. Section 2 reviews SPSO and advanced PSO with parameter selection. Section 3 describes the proposed PSOFAP algorithm in detail. In Section 4, we report and analyze the numerical experimental results on twelve benchmark problems and compare the proposed algorithm results with those of other state-of-the-art PSO variants through non-parametric statistical analysis test. Section 5 concludes the paper.

2. Variants of Particle Swarm Optimization

Eberhart and Kennedy (1995) were inspired to develop the search strategy of PSO by the foraging behavior of flocks of birds and schools of fish. Each particle has its own position and velocity, which represent the values of the decision variables in the current iteration and the movement vector for the next iteration, respectively. The position and velocity of each particle change according to the information shared among all particles in the current iteration. Each particle can record the personal best experience and refresh it with iterations. The global best experience can then be defined by comparing the personal best experiences of all particles. The search strategy of PSO involves identifying the new velocity for calculating the new position at the next iteration according to the original velocity of each particle (v_i), the personal best experience of each particle ($x_{p(i)}$), and the global best experience of all particles (x_g), as shown in Figure 1. PSO can obtain an approximate optimal solution by repeating the procedure over many iterations of evolution.

Figure 2 shows the search procedure of PSO. First, initialize particle swarm, including the particle's position and velocity in the searching space. Then, calculate the fitness of the particle swarm for updating the personal best experience of each particle and the global best experience of all the particles. The most important step in the procedure is to calculate the new velocity and the position of every particle in the next iteration by using equations (1) and (2).

$$v_{id}(t+1) = w * v_{id}(t) + c_1 r_1 \left(x_{p(id)}(t) - x_{id}(t)\right) + c_2 r_2 \left(x_{gd}(t) - x_{id}(t)\right) \qquad (1)$$

where $v_{id}(t)$ represents the velocity value of the d^{th} dimension of the i^{th} particle in the t^{th} iteration. The variable $x_{id}(t)$ represents the position of the d^{th} dimension of the i^{th} particle in the t^{th} iteration. The variable

w represents the inertia weight, c_1 is the self-cognition acceleration coefficient, and c_2 is the social cognition acceleration coefficient.

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

Equation (2) shows that the new position of each particle is updated using the original position and the new velocity from equation (1), where r_1 and r_2 are two separately generated, uniformly distributed random numbers in the range [0,1]. The following section briefly describes these two categories of parameter value selection.

Zhang *et al*. (2015) indicated that PSO has drawbacks such as premature convergence, high computational complexity, slow convergence, and sensitivity to parameters. Because PSO does not apply the crossover operator as in GA or DE, the favorable information can be shared between candidates. Moreover, because PSO does not achieve an appropriate balance between exploitation (local search) and exploration (global search), it can easily and quickly converge to a local minimum. To address these problems, scholars have proposed many solutions, which can be divided into following three types. (i) Major modifications, such as quantum-behaved PSO (Jau *et al.*, 2013), bare-bones PSO (Zhang *et al.*, 2011), and fuzzy PSO (Khan & Engelbrecht, 2012). (ii) Minor modifications, including constriction coefficient (Lin, 2013), velocity clamping (Shahzad *et al.*, 2014), adaptive parameter (Yasuda & Iwasaki, 2004; Xu, 2013), ecological behavior (Lu *et al.*, 2013), neighborhood learning (Zong *et al.*, 2014), and local search (Chen & Ludwig, 2012). (iii) Hybridization of PSO with other algorithms, such as GA (Ghamisi & Benediktsson, 2015), SA (Niknam *et al.*, 2013), and DE (Maione & Punzi, 2013). To maintain the advantages of the simple application of PSO, this study focuses on improving the balance between exploration and exploitation, which belongs to the minor modification category. Therefore, the following section, we review some studies on parameter selection, dynamic tuning of parameters, and self-adaptive parameters.

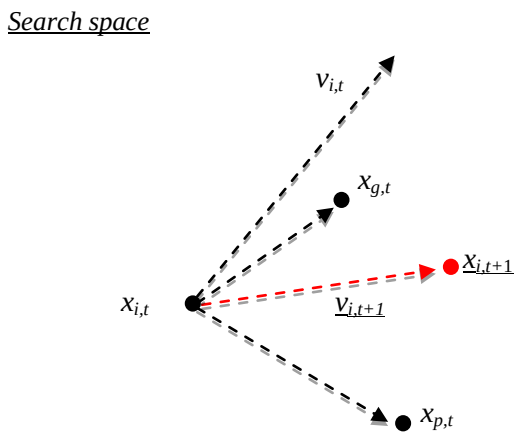


Figure 1: The search strategy of PSO

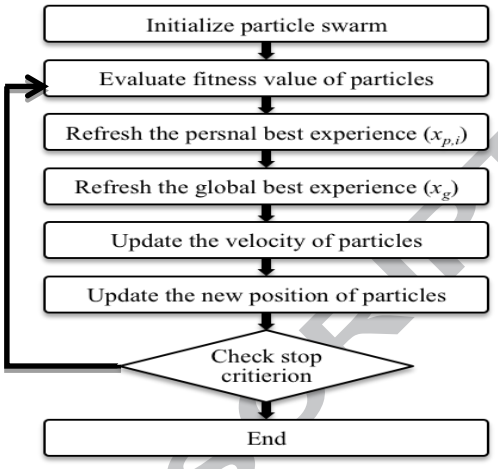


Figure 2: The search procedure of PSO

2.1 Particle Swarm Optimization Algorithm with Time-Varying Parameter Values

In the time-varying parameter value method, the tuning mechanism follows either a linear function or a nonlinear function with the evolution. The following paragraphs provide a brief introduction to the most widely applied PSO variants, including linear decreasing inertia weight PSO (LPSO; Shi & Eberhart, 1998) and nonlinear decreasing inertia weight PSO (NWAPSO; Chatterjee & Siarry, 2006). Both studies tend to explore the search space in early stage of the evolution and exploit in later stage of the evolution.

Shi and Eberhart (1998) reported that the appropriate value scope of the inertia weight for solving Schaffer's function ranges from 0.9 to 1.2. Based on the value scope, LPSO with the iteration number was proposed and is shown in equation (3). The variable w_f represents the final value of the inertia weight that can be set in the final iteration. The variable w_i represents the initial value of the inertia weight that can be set in the first iteration. A larger value in early iteration promotes the global search strategy, and a smaller value in late iteration facilitates local searching. Empirical studies have verified that LPSO can obtain a superior solution to that obtained using SPSO and rapidly converge.

$$w^t = w_f + \frac{(w_i - w_f)((T-t))}{T} \qquad (3)$$

Ratnaweera, Halgamuge, and Watson (2004) proposed a PSO variant with linear time-varying cognition acceleration coefficients (LPSO-TVAC) that is based on LPSO and where c_1 is set from 2.5 to 0.5 and c_2 is set from 0.5 to 2.5. As equation (4) shows, the self-cognition acceleration coefficient decreases gradually with the iteration number, preventing premature convergence to a local optimal value.

Conversely, the effect of population experience increases gradually, as shown in equation (5), improving the solution quality.

$$c_1{}^t = c_{1,i} + (c_{1,f} - c_{1,i})\frac{t}{T} \qquad (4)$$

$$c_2{}^t = c_{2,i} + (c_{2,f} - c_{2,i})\frac{t}{T} \qquad (5)$$

Chatterjee and Siarry (2006) proposed NWAPSO and studied the appropriateness of the nonlinear function. The study set $w^t = w_i + (w_f - w_i)(\mathrm{t}/T)^n$ means that if *n* is not equal to 1, then the slope of the inertia weight is a nonlinear curve. Experimental results indicated that setting *n* to 1.2 enables obtaining a superior solution and rapid convergence. The study investigated the improvement of the solution, and the convergence of PSO with a nonlinear decreasing function is quite obvious, compared with the convergence of PSO with a linear decreasing function.

2.2 Particle Swarm Optimization Algorithm with Self-Adaptive Parameter Values

In a PSO algorithm with self-adaptive parameter values, the values adjust dynamically according to the performance of individual particles or the population in the current iteration (Montalvo, Izquierdo, Pérez-García, and Herrera (2010). Yasuda, Ide, and Iwasaki (2003), Yasuda and Iwasaki (2004), and Xu (2013) have proposed the concept of an ideal velocity that is larger in the early periods and gradually becomes smaller in the end of the search process. Therefore, with the ideal velocity, more exploration is performed initially and more exploitation is performed in the later part of the search process; through the inertia weight tuning the particle velocity that approximates the ideal time varying velocity is reached. By using different tuning functions and rules, Yasuda and Iwasaki (2004) and Xu (2013) have applied the ideal velocity in order to adjust the inertia weight value. The ideal velocity of DAPTPSO(2006) is a time-varying linear decreasing function. Conversely, the ideal velocity of APSOVI (2013) is a time-varying nonlinear decreasing function.

Iwasaki, Yasuda, and Ueno (2006) defined the ideal velocity as the linear decreasing function shown in equation (6) according to the concept of LPSO, where $v_{start} = x_{max} - x_{min}$ and $T_{end} = 0.8 * T_{max}$. They use the information defined as the average absolute value of the velocity of all particles, as shown in equation (7), which can be used as an index to understand the vitality of all particles. In equation (7), *M* represents the number of iterations and *N* represents the number of dimensions of the particle's position.

$$v_{ideal}(t+1) = v_{ideal}(t) - \left(\frac{v_{start}}{T_{end}}\right) \qquad (6)$$

$$v_{ave}(t) = \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left| v_{ij}(t) \right| \qquad (7)$$

The scope of the inertia weight limits the value of the inertia weight and is represented as (w_{min}, w_{max}). A tuning unit for the inertia weight, represented as Δw, adjusts the value of the inertia weight. If the average absolute value of velocity of all of the particles is larger than the value of the ideal velocity, then one tuning unit (Δw) is added to the inertia weight; otherwise, one tuning unit (Δw) is subtracted from the inertia weight. Through the mentioned adaptation mechanism, the result of DPATPSO is improved relative to that of LPSO in solution quality and convergence. On the basis of the concept of the ideal velocity from DPATPSO, Xu (2013) proposed an APSOVI algorithm that employs the cosine function as the ideal velocity function. In equation (8), the ideal velocity transfers from the cosine function to limit the value of the ideal velocity to between 0 and 1, where $T_{end} = 0.95 * T_{max}$ and $v_{start} = (x_{max} - x_{min})/2$. The cosine function is a nonlinear function that causes the velocity of the particle to decrease rapidly in the middle of the search process but gradually decrease in the beginning and the end, as shown in Figure 3.

$$v_{ideal}(t) = v_{start} \cdot \frac{1 + cos(t\pi / T_{end})}{2} \qquad (8)$$

Although the results of DPATPSO (Iwasaki *et al*., 2006) and APSOVI (Xu, 2013) are both superior to those of LPSO and LPSO-TVAC, the inertia weight is still tuned based on the change of iteration number instead of the performance of each particle. Therefore, this study attempts to express the effect of performance-related parameter values adaptation mechanism.

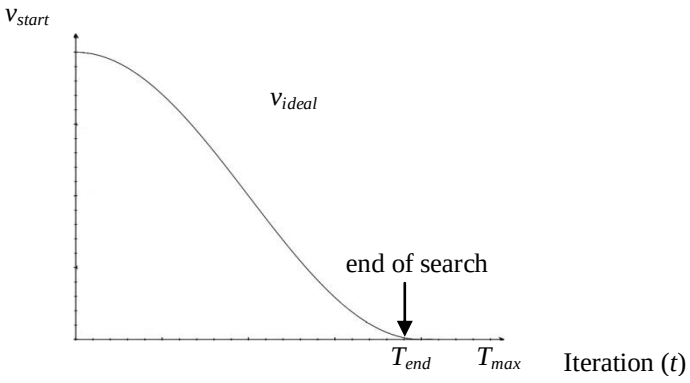
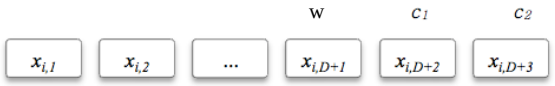


Figure 3: The curve of cosine function as the ideal velocity function of APSOVI

3. Particle Swarm Optimization Algorithm with Parameter Evolution

The PSOFAP algorithm based on individual fitness performance adjusts the absolute velocity of each particle by using Euclidean distance. Each particle has its own inertia weight, self-cognition acceleration coefficient, and social cognition acceleration coefficient. PSOFAP expects that the absolute movement of a particle with strong performance will be small. By contrast, a poorly performing particle must move farther to the following position. Hence, PSOFAP treats these three parameter values as the problem dimensions; the parameter values are self-adjusted by the PSO algorithm. In this study, three parameters are defined as the final three dimensions of problem schema, as shown in Figure 4. The parameters are encoded as the final three dimensions of the PSO code of the problem to be solved, namely the inertia weight (w) and two



cognition acceleration coefficients (c_1 & c_2).

Figure 4: the PSOFAP code with three parameter dimensions

However, the problem's objective function cannot directly reflect the fitness of the parameter values. Hence, this study employs the transformed cosine as the objective function for parameter evolution, which represents the absolute velocity corresponding fitness performance grades. In this study, the fitness performance grades are initially defined, and then each particle is assigned a reference point based on the fitness performance to obtain the corresponding ideal velocity. Next, the velocity bias is measured as the difference between the ideal velocity and current absolute velocity; a smaller velocity bias reflects a more optimal state. Then, we compare the velocity biases of particles assigned to the same reference point. The parameter values of particle with the smallest velocity bias are assigned as the global optimum for the reference point.

The proposed PSOFAP algorithm is implemented on the basis of the SPSO procedure and summarized in the following search steps; Steps 3–6 are added for adjusting the parameter values to efficiently obtain a more optimal solution. The proposed PSOFAP algorithm is summarized with the following notation:

Notation of PSOFAP:

d	dimension index	w	inertia weight
D	numbers of dimensions for the search space	c_1	cognitive learning factor
i	particle index	c_2	social learning factor
I	swarm size	r	reference points index

t	iteration index	R	*numbers of* reference points
T	numbers of Iterations	A	performance of a particle
v	vector of particle's velocity, v = [v1,v2, . . . ,vD+3]	F^p	the objective function of parameters evolution
V_{max}	vector of particle's current maximum velocity		
V^{ideal}	vector of particle's ideal velocity		
x	vector of particle's position, x = [x1,x2, . . . ,xD+3]		
x_w, x_g	vector of particle's position of current worst and best fitness		

Step 1 Initialize the values of the parameters and the velocity and position of each particle.

Step 2 Calculate the fitness value of each particle, and update the individual optimal position, the global optimal position, and the global worst position.

Step 3 Calculate A and $|v|$ for each particle. Randomly generate parameter values and go to Step 7 if the three parameter values are all equal to 0.

Since each particle dynamically tunes its own parameter values according to its current fitness value performance, the index (A_i) represents the performance of each particle's current fitness value compared to the best and worst fitness value in the evolution, as shown in equation (9). In this equation, the difference between the current best fitness value and the current poorest fitness value is the basis for the index and used to measure the performance of each particle. Hence, the global best and global worst position in the evolution would be recorded.

$$A_i(\mathrm{t}) = \frac{\mathrm{f}(\mathrm{x_w}) - \mathrm{f}(\mathrm{x_i}(\mathrm{t}))}{\mathrm{f}(\mathrm{x_w}) - \mathrm{f}(\mathrm{x_g})} \tag{9}$$

Since tuning parameter values of each particle is based on the comparison of the best experience of fitness performance. Hence, we divided the fitness value scope from the experience in the evolution into several sections as subsets. Once a particle moves to a new position then it is classified to one specific subset in accordance with its A_i value, and then the best experience and worst experience of this compared subset would be updated. Consequently, we defined the new parameter R as the number of subsets for parameter evolution, and $r = 1,\ldots, R$ represents the r^{th} fitness performance subsets.

To prevent a particle from stopping moving before it satisfies the stop criterion, the following mechanism is implemented: if the values of all of the last three dimensions of a particle equal "0" then it randomly generates the values of parameter dimensions within certain boundaries.

Step 4 (Only for parameter dimensions) Calculate F^p for each particle and obtain the corresponding reference point for the ideal velocity function of each particle.

This study proposes the parameter values tuning mechanism is the transformation of the cosine function as the ideal velocity function with the individual fitness performance, as equation (10) shown, referred to the ideal velocity concept mentioned by Xu (2013).

$$v_i^{ideal}(t) = v_{max} \cdot \frac{1+\cos(\pi(r-1)/(R-1))}{2} \quad (10)$$

Therefore, particles with similar fitness values should have the same distance of movement ideally, and the ideal velocity (v_i^{ideal}) in equation (10) represents the ideal distance of movement of each particle, and also the (v_{max}) represents the largest absolute distance of all of the particles ever obtained. Consequently, several reference points of the ideal velocity curve are labeled, that correspond to the fitness performance compared subsets by A_i (from 0 to 1), as Figure 5 shown. Figure 5 shows the concept of the reference point of the ideal velocity function. If we set R = 20, then the value of the first reference point is 0 and the value of the second reference point is 1/19. To avoid getting stuck in local optima, we set up the rule for pointing the corresponding reference point of the ideal velocity value. If A_i value is less or equal to the value of the r^{th} reference point, then particle i is classified to the $(r\text{-}1)^{th}$ reference point, as the following description.

$$\text{if } \mathrm{A}_i(t) \leq \frac{\mathrm{r}-1}{\mathrm{R}-1} \text{ , then } x_i \in \ the\ (r-1)th\ particle\ sub-swarm.$$

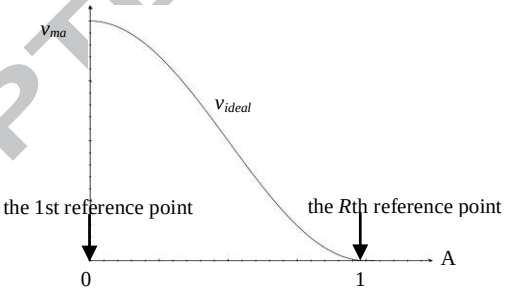


Figure 5: the concept of the reference point of the ideal velocity function.

To achieve parameter evolution, this study designs the velocity bias as the objective of parameters (the final three dimensions), as shown in equation (11). Here, we defined the absolute velocity of a particle as the absolute distance from the current position to the next position obtained by the Euclidean distance, as shown in equation (12), where i represents the particle number, d represents the d^{th} dimension of a particle, and D represents the dimension size. The updated velocity formula of the parameter dimensions of the particle is expressed in equation (13), where $d = D + 1$, $D + 2$ and $D + 3$. Furthermore, the personal best

experience of parameter dimensions would be meaningless in the evolution because of the difference of the reference points. Therefore, the updated formula only considers the original velocity and social best experience as references for the next step.

$$\min \ F_i^p(t) = \left| v_{ideal}(t) - |v_i(t)| \right| \tag{11}$$

$$|v_i(t)| = \sqrt{\sum_{d=1}^{D} v_{id}(t)^2} \tag{12}$$

$$v_{id}^p(t+1) = 0.01 * w_i v_{id}^p(t) + 0.01 * c_{2_i} r_2 \left(x_{gr}^p(t) - x_{id}(t) \right) \tag{13}$$

Step 5 (Only for parameter dimensions) Update the global position of the parameter dimensions from the set of corresponding particle reference points.

Step 6 (Only for parameter dimensions) Update the velocity and position of the parameter dimensions for the next iteration.

Step 7 (Only for dimensions 1 to *D*) Update the velocity and position of each particle according to the new values of parameters from Step 6.

Step 8 Calculate the fitness value of each particle and update the individual optimal position, global optimal position, and global worst position.

Step 9 Stop if *t* is equal to the maximum iteration number *T*; otherwise, proceed to Step 3.

4. Experiments and Discussion

4.1 Benchmark Test

For evaluating the performance of PSOFAP, a series of experiments were conducted for each test function with three different dimension sizes: 10, 30, and 100. In addition, the standard PSO and three other PSO variants namely NWAPSO, DPATPSO and APSOVI were used for evaluating the contribution of PSOFAP. All values of the parameters of each algorithm are shown in Table 1.

To assess the performance of each algorithm compared with that of PSOFAP, twelve benchmark functions were adopted, as listed in Table 2 (Ali, Khompatraporn, & Zabinsky, 2005; Liang *et al*., 2013). Table 2 also lists the corresponding bounds of each dimension (search space), the optimal value of each dimension (x^*), and the minimal fitness value of each function ($f(x^*)$). All test functions were minimization problems, and the compositions of test functions in this study include four unimodal functions (f_1~f_3 and f_7) and eight multimodal functions (f_4~f_6 and f_8~f_{12}). The unimodal function is used to confirm the solution quality and convergence of an algorithm, while the multimodal functions, usually with many local optima, are used to determine how effectively an algorithm can avoid local solutions. This study applies six test functions (f_1–f_6) to compare the proposed algorithm with five PSO variants and another six test functions (f_7–f_{12}) to compare the best two PSO variants. Furthermore, we apply Wilcoxon signed rank nonparametric

test to determine statistically significant differences between APSOVI and PSOFAP. The Wilcoxon signed rank nonparametric test is widely used as an alternative to the paired test when the results cannot be assumed to obey the normal distribution (Martins & Anderemi, 2014).

Each algorithm was executed in more than 30 independent runs with 5,000 iterations and a swarm size of 40. This study used a 64-bit computer with an i7-3770 CPU operating at 3.40 GHz and with 8 GB of RAM.

Table 1: PSO algorithms for comparison with values of parameters

Algorithm	Parameters	Reference
SPSO	w=0.729, c_1= 1.49, c_2= 1.49	(Eberhart & Kennedy, 1995)
NWAPSO	$w_{initial}$=0.2, w_{final}= -0.3, m= -2.5*10^{-4}, and n=1.2	(Chatterjee & Siarry, 2006)
DPATPSO	$w_{\max}$: 0.9, $w_{\min}$: 0.4, Δw = 0.1, c_1= 1.3, c_2= 1.3	(Iwasaki *et al*., 2006)
APSOVI	$w_{\max}$: 0.9, $w_{\min}$: 0.3, Δw= 0.1, c_1= 1.49, c_2= 1.49	Xu (2013)
PSOFAP	w=[-1, 1], c_1= [-2, 2], c_2 =[-2, 2], R=15	This study

Table 2: Twelve testing benchmark problems in this study

Function	Test Function	Searching space	x^*	$f(x^*)$
Rosenbrock	$f1(x) = \sum_{i=1}^{D-1}[100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2]$	[-10, 10]	[1,…,1]	0
Exponential	$f2(x) = -\exp(-0.5\sum_{i=1}^{D} {x_i}^2)$	[-1, 1]	[0,…,0]	-1
Sphere	$f3(x) = \sum_{i=1}^{D} {x_i}^2$	[-10, 10]	[0,…,0]	0
Rastrigin*	$f4(x) = \sum_{i=1}^{D}[{x_i}^2 - 10\cos(2\pi x_i) + 10]$	[-5.12, 5.12]	[0,…,0]	0
Griewank*	$f5(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} {x_i}^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	[-50, 50]	[0,…,0]	0
Ackley*	$f6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	[-30, 30]	[0,…,0]	0
Shifted functions:				
Shifted Sphere	$f7(x) = \sum_{i=1}^{D} {z_i}^2 - 1400;\ z = x - o$	[-10, 10]	O	-1400
Shifted Rastrigin*	$f8(x) = \sum_{i=1}^{D}[{z_i}^2 - 10\cos(2\pi z_i) + 10] - 400;$ $z = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(\frac{5.12(x-o)}{100}))$	[-10, 10]	O	-400
Rotated functions:				
Rotated Rosenbrock*	$f9(x) = \sum_{i=1}^{D-1}[100(z_{i+1} - {z_i}^2)^2 + (z_i - 1)^2] - 900;$ $z = x - o$	[-10, 10]	O	-900
Rotated Rastrigin*	$f10(x) = \sum_{i=1}^{D}[{z_i}^2 - 10\cos(2\pi z_i) + 10] - 300;$ $z = M_1\Lambda^{10}M_2 T_{asy}^{0.2}(T_{osz}(M_1\frac{5.12(x-o)}{100}))$	[-5.12, 5.12]	O	-300
Rotated Griewank*	$f11(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} {z_i}^2 - \prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) - 500;$ $z = \Lambda^{100} M_1(\frac{600(x-o)}{100})$	[-50, 50]	O	-500
Rotated Ackley*	$f12(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {z_i}^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e - 700;$ $z = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(x-o))$	[-30, 30]	O	-700

*: Multimodal functions

4.2 Results and Analysis

The experimental results of five PSO variants regarding the average final best value and the standard deviation (SD) are summarized in Table 3, where the optimal result of each test function is marked bold. PSOFAP obtains superior performance in the benchmark functions among the five PSOs, as shown in Table 3. PSOFAP yields a superior solution compared with the other PSOs according to both the average and the standard deviations of the approximate optimal solution over 30 runs. This result indicates that the solution quality and stability of PSOFAP is considerably superior to that of the other four algorithms. Table 4 shows the comparison of computation time for five PSO variants. Through a unimodal function experiment, we discovered that the difference between PSOFAP and other algorithms increases with the dimension size. Furthermore, as illustrated in Table 3, PSOFAP yields the optimal solution for two multimodal functions with many dimensions, while the other PSO variants yield poorer results.

Table 3: Comparison between different PSO algorithms for benchmark test functions

Algorithms		**SPSO**	**NWAPSO**	**DPATPSO**	**APSOVI**	**PSOFAP**
Function	Dim	Average (Std. Dev)	Average (Std. Dev)	Average (Std. Dev)	Average (Std. Dev)	Average (Std. Dev)
$f1$	10	**0.588783504 (1.369534435)**	8.24875 (16.47729)	10.4373 (55.55572)	16.341147 (62.15159)	7.2031587 (3.4081896)
	30	**23.93017747 (23.155703618)**	806.0424 (1852.007)	378.34265 (1823.4347)	361.79414 (1825.7846)	28.911825 (0.063277383)
	100	975.1584951 (1866.1601913)	188875.2 (70388.31)	63413.165 (240648.88)	63593.371 (240288.02)	**98.934681 (0.039566045**)
$f2$	10	**-1 (0)**	**-1 (9.45E-17)**	**-1 (1.26466E-08)**	**-1 (0)**	**-1 (0)**
	30	**-1 (0)**	-0.99565 (0.004665)	-0.9716677 (0.081779921)	-0.95258047 (0.14876889)	-0.99867953 (0.007232527)
	100	-0.97988003 (0.066718721)	-0.34609 (0.093354)	-0.42813197 (0.31004994)	-0.69245043 (0.31191503)	**-0.99671536 (0.012555968)**
$f3$	10	**1.0072E-100 (4.42788E-100)**	**1.93E-34 (1.06E-33)**	**0 (0)**	**0 (0)**	**0 (0)**
	30	**5.61866E-37 (1.671534E-36)**	1.488839 (1.997052)	0.01568633 (0.054581401)	0.010445162 (0.01853444)	0.057429458 (0.3145541)
	100	5.216357574 (18.597916455)	231.1243 (53.22372)	42.243603 (68.341805)	20 (48.42342)	**0 (0)**
$f4$	10	7.462188058 (3.6181649695)	14.16157 (6.716091)	4.3146184 (7.9319166)	1.6583473 (3.2249247)	**0.14452662 (0.79160492)**
	30	87.93269999 (34.406753541)	90.99567 (19.45191)	58.377695 (28.193002)	53.144474 (36.471547)	**1.2073107 (6.6127132)**
	100	537.5281696 (72.760312266)	563.8737 (51.4619)	309.8266 (107.69485)	358.94247 (182.66932)	**14.674083 (80.373263)**
$f5$	10	0.072391806 (0.019852078)	0.100476 (0.059396)	0.037320181 (0.071672461)	**0 (0)**	**0 (0)**
	30	0.010664288 (0.0118467963)	0.534163 (0.296616)	0.087720183 (0.26471057)	**0 (0)**	**0 (0)**
	100	0.202669283 (0.2368051614)	2.416541 (0.300419)	1.1429422 (0.631638)	0.60486728 (0.73287259)	**0 (0)**
$f6$	10	**3.67113E-15 (6.486338E-16)**	1.512089 (1.161333)	**0 (0)**	**0 (0)**	**0 (0)**
	30	1.853287791 (1.2602973235)	9.936236 (2.124229)	4.2871713 (6.5613841)	0.44345605 (2.4289088)	**0 (0)**
	100	13.25560461 (2.2871116414)	16.12496 (0.764844)	12.220856 (6.144316)	7.5296408 (4.8172143)	**0 (0)**

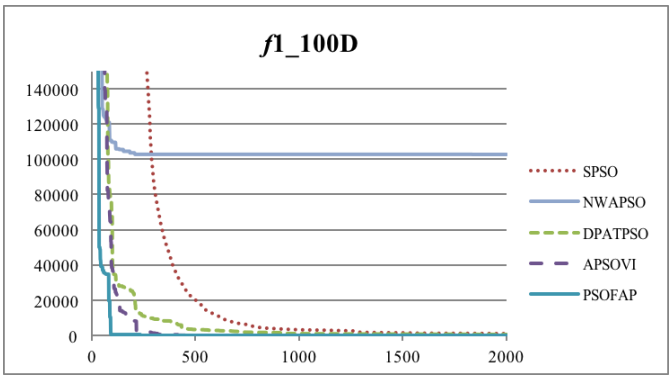
We observe that the best two PSOs are APSOVI and the proposed PSOFAP (Table 4). To test APSOVI and PSOFAP, this study also applies the functions f_7 to f_{12} with 30-dimensional size and more than 30 independent runs; the experiment results are shown in Table 5. The convergence curves for both the algorithms are illustrated in Figure 7. Table 5 indicates that the computational time of PSOFAP is longer than that of APSOVI. Hence, a more detailed analysis of the convergence efficiencies of PSOFAP and APSOVI is provided from the perspective of computational time. This analysis reveals that PSOFAP converges during earlier iterations than do other PSO algorithms. Therefore, four PSO algorithms were compared with six fixed computational times and functions f_7 to f_{12} with 100-dimension size; the results are shown in Table 6. The four PSO algorithms converge to approximate solutions after approximately 2 s in all functions except for f_8, which converged in approximately 1 s. Furthermore, we find that PSOFAP could obtain better results than other PSO algorithms for functions f_7 to f_{12}. We observe that PSOFAP could converge to an approximate result after fewer iterations and with shorted computational time than could other PSO algorithms. Because the approximate results and computational time obtained by APSOVI and PSOFAP are similar, a nonparametric statistical test for APSOVI and PSOFAP is conducted in Section 4.3.

Table 4: Comparison of the computational time of five PSO algorithms for benchmark test functions

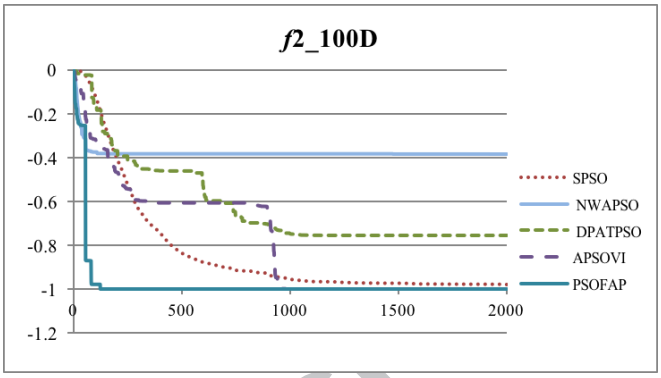
PSO variants	**SPSO**		**NWAPSO**		**DPATPSO**		**APSOVI**		**PSOFAP**	
(Dimension sizes)	30	100	30	100	30	100	30	100	30	100
f_1	0	0	0	0	0	0	0	0	0	0
f_2	100	0	0	0	13.3	0	40	70	96.7	86.7
f_3	100	0	0	0	83.3	16.7	43.3	83.3	96.7	100
f_4	0	0	0	0	0	0	10	0	96.7	96.7
f_5	0	0	0	0	43.3	0	100	43.3	100	100
f_6	0	0	0	0	50	6.7	96.7	23.3	100	100
AVE	33.3	0	0	0	31.65	3.9	48.3	36.7	**81.7***	**80.6***

Table 5: Comparison between PSOFAP and APSOVI for problems with dimension of 30, 40 particles, 5,000 iterations and 30 replications

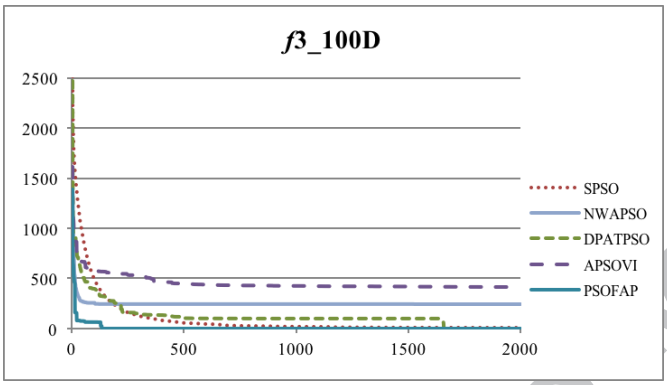
Functions	**APSOVI**				**PSOFAP**			
	Min	Average	Std. Dev.	C/T(sec)	Min	Average	Std. Dev.	C/T(sec)
f_7	-1399.8221	**-1346.5898**	55.559086	5.6408	**-1399.9982**	-1344.9438	**35.519289**	9.6834
f_8	**-400**	-394.76656	7.6796755	5.9438	**-400**	**-396.97597**	**4.0969465**	6.2768
f_9	**-899.99599**	**-896.49563**	4.7293505	10.1163	-899.86202	-889.66318	5.9625695	11.4454
f_{10}	**-298.12433**	-249.73527	28.960414	9.9392	-293.04285	**-257.34868**	**22.299315**	13.8995
f_{11}	-487.91767	-267.30478	190.42892	10.4146	**-499.70217**	**-302.37514**	**152.81647**	11.1136
f_{12}	**-683.30471**	**-679.76787**	0.91926651	9.4564	-679.38524	-679.1428	**0.12955915**	11.3122



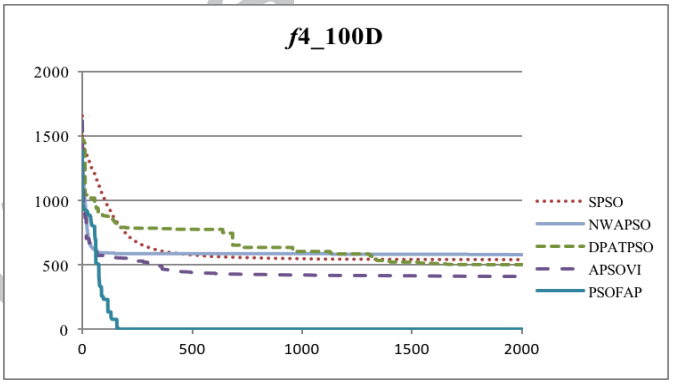
(a) f_1



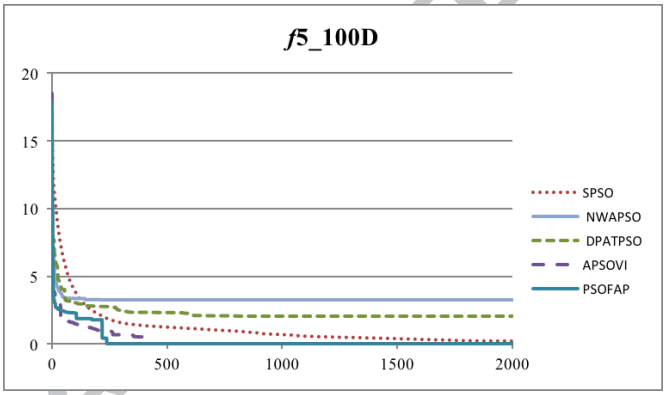
(b) f_2



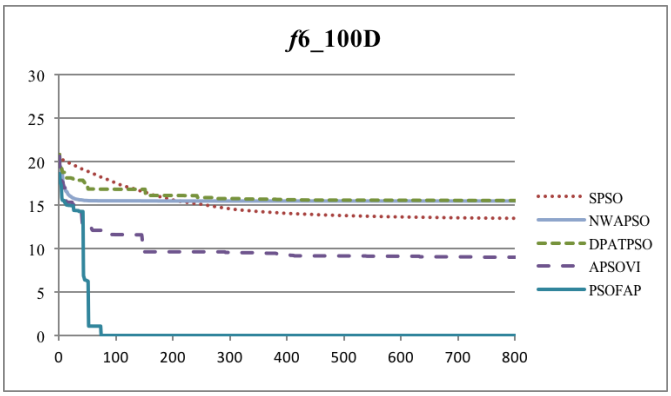
(c) f_3



(d) f_4

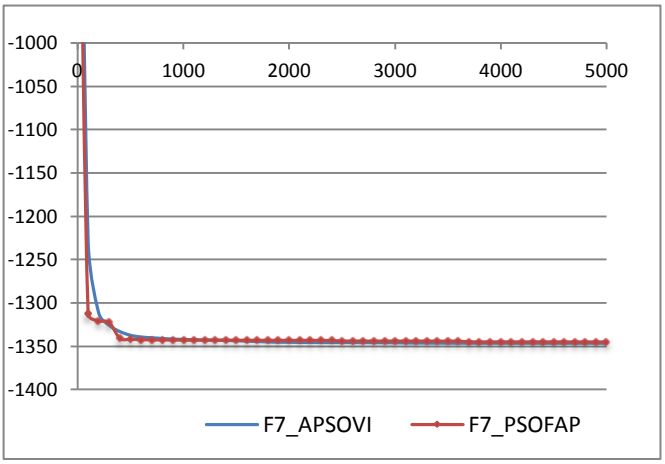


(e) f_5

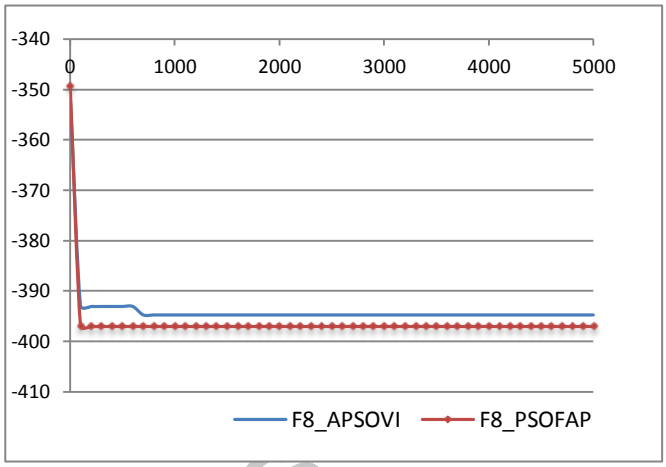


(f) f_6

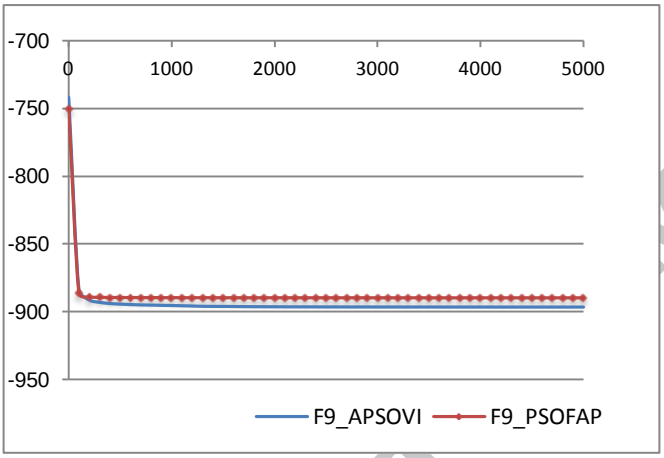
Figure 6: Convergence curves of 100-D test functions (a) f_1, (b) f_2, (c) f_3, (d) f_4, (e) f_5, and (f) f_6



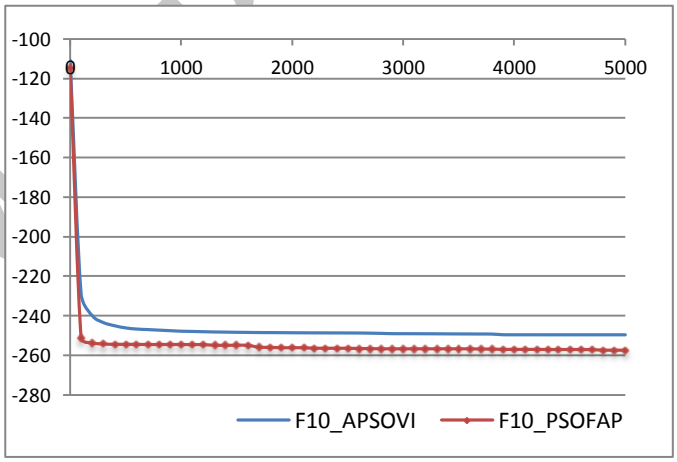
(a) f_7



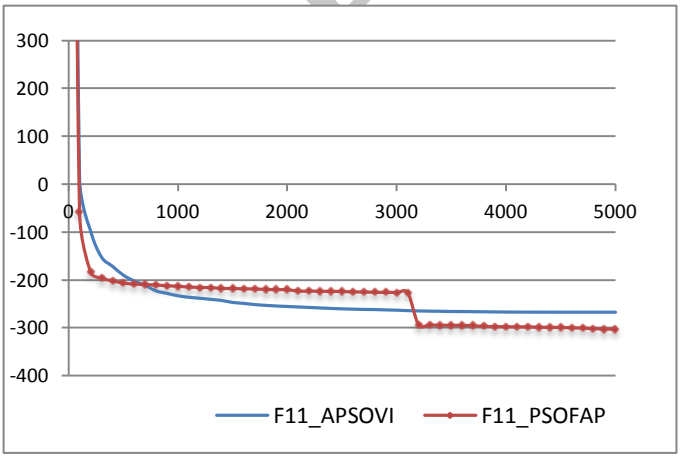
(b) f_8



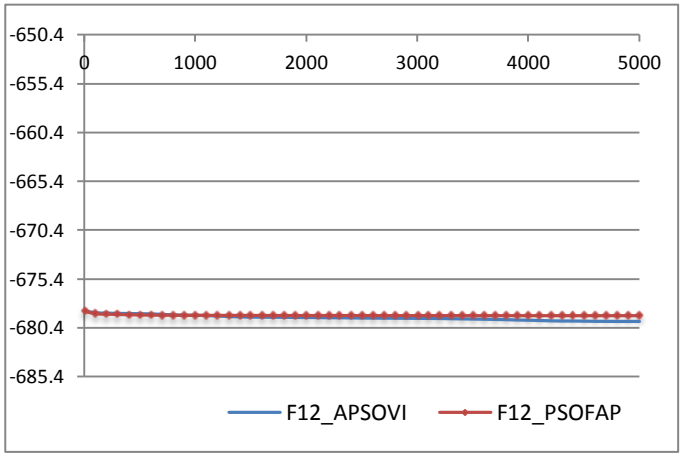
(c) f_9



(d) f_{10}



(e) f_{11}



(f) f_{12}

Figure 7: Convergence curves of 30-D test functions obtained by PSOFAP and APSOVI for test functions 7^{th} to 12^{th}. (a) f_7, (b) f_8, (c) f_9, (d) f_{10}, (e) f_{11}, and (f) f_{12}.

Table 6: Comparison between four PSO variants for functions f_7 to f_{12} at six time sections with dimension of 100, 40 particles, and 10 replications.

Functions	C/T (sec.)	NWAPSO (Min, Average)		DPATPSO (Min, Average)		APSOVI (Min, Average)		PSOFAP (Min, Average)	
*f*7	2s	-988.332	-897.752	-991.788	-764.256	-135.437	90.690	**-1336.377**	**-1179.485**
	4s	-1000.351	-904.396	-1174.759	-1056.606	-687.997	-268.595	**-1353.476**	**-1212.600**
	6s	-1000.351	-904.396	-1188.251	-1065.007	-828.901	-471.511	**-1368.020**	**-1220.980**
	8s	-1000.351	-904.396	-1188.251	-1065.007	-1023.607	-564.866	**-1374.591**	**-1234.807**
	10s	-1000.351	-904.396	-1188.251	-1065.007	-1215.587	-646.393	**-1374.591**	**-1239.431**
	12s	-1000.351	-904.396	-1188.251	-1065.007	-1252.338	-694.767	**-1374.591**	**-1240.205**
*f*8	0.5s	**-395.312**	**-375.535**	-360.031	-282.534	-319.804	-216.509	-387.977	-372.228
	1s	-395.916	-375.929	-372.130	-305.875	-325.786	-219.298	**-396.882**	**-387.373**
	2s	-395.916	-375.929	-391.031	-353.467	-337.654	-297.295	**-396.882**	**-387.373**
	3s	-395.916	-375.929	-394.021	-354.273	-341.044	-305.436	**-396.882**	**-387.373**
	4s	-395.916	-375.929	-395.275	-354.528	-341.359	-308.625	**-396.882**	**-387.373**
	5s	-395.916	-375.929	-395.275	-354.528	-342.117	-311.040	**-396.882**	**-387.373**
*f*9	2s	-788.768	-658.206	-760.059	-735.632	-709.067	-657.869	**-877.265**	**-844.109**
	4s	-788.882	-658.394	-787.407	-773.808	-751.391	-715.232	**-880.563**	**-854.441**
	6s	-788.882	-658.394	-800.800	-800.005	-784.327	-742.400	**-881.464**	**-856.291**
	8s	-788.882	-658.394	-806.423	-806.371	-797.341	-768.014	**-881.930**	**-856.892**
	10s	-788.882	-658.394	-806.423	-806.371	-816.569	-786.819	**-881.964**	**-857.479**
	12s	-788.882	-658.394	-806.423	-806.371	-836.141	-798.837	**-882.010**	**-857.697**
*f*10	2s	-63.714	66.633	-40.487	26.637	108.507	150.142	**-181.737**	**-101.034**
	4s	-63.786	65.938	-87.647	-25.495	33.083	70.805	**-236.783**	**-156.377**
	6s	-63.786	65.938	-107.269	-49.880	-8.408	37.850	**-250.675**	**-169.076**
	8s	-63.786	65.938	-117.157	-59.165	-31.764	10.849	**-262.093**	**-176.494**
	10s	-63.786	65.938	-117.157	-59.166	-51.416	-14.268	**-272.096**	**-182.029**
	12s	-63.786	65.938	-117.157	-59.166	-56.989	-24.834	**-273.158**	**-186.076**
*f*11	2s	1210.527	1480.514	2004.676	2583.066	3368.945	4138.110	**6.170**	**207.285**
	4s	1209.850	1478.436	1560.585	2019.069	2287.076	2999.891	**-217.104**	**-47.428**
	6s	1209.850	1478.436	1365.900	1772.805	1689.058	2288.890	**-283.475**	**-145.303**
	8s	1209.850	1478.436	1344.330	1700.209	1038.643	1706.279	**-338.375**	**-210.502**
	10s	1209.850	1478.436	1303.013	1664.962	805.340	1478.975	**-357.656**	**-244.768**
	12s	1209.850	1478.436	1197.628	1626.680	756.627	1368.601	**-371.089**	**-271.426**
*f*12	2s	-678.689	-678.617	-678.609	-678.592	-678.625	-678.595	**-678.835**	**-678.752**
	4s	-678.689	-678.628	-678.634	-678.600	-678.632	-678.614	**-678.920**	**-678.838**
	6s	-678.699	-678.640	-678.824	-678.701	-678.655	-678.620	**-678.967**	**-678.859**
	8s	-678.749	-678.660	-678.894	-678.792	-678.655	-678.626	**-678.967**	**-678.887**
	10s	-678.749	-678.660	-678.926	-678.848	-678.655	-678.632	**-678.967**	**-678.893**
	12s	-678.749	-678.665	-678.936	-678.880	-678.655	-678.634	**-678.967**	**-678.895**

4.3 Statistical Analysis Test

In this section, this study compares APSOVI and the proposed PSOFAP by a non-parametric statistical analysis method as the Wilcoxon signed rank test to proof the indication of the proposed PSOFAP (Wilcoxon, 1945). The Wilcoxon signed rank test compares two related samples or matched samples without assumptions about the distribution of the data. The approach has been widely used to test the significant of many enhanced evolutionary optimization algorithms (Biswas & Biswas, 2017; Ding *et al.*, 2017; Güvenc & Katırcıoğlu, 2017; Jiang *et al.*, 2017; Martins & Anderemi, 2014; Liu *et al.*, 2017).

We first divide the search process of these two PSO algorithms with 21 cut points per 100 iterations. The average of the global optimal solution for each of the cut points with 10 replications are selected as the measurement for statistical analysis (Derrac *et al.*, 2014). Wilcoxon's test is conducted with a 5% significance level by using the statistical software package, Statistical Product and Service Solutions (SPSS). There are three assessment values for these two algorithms: the R^+ (the sum of ranks, in which the PSOFAP outperformed the APSOVI on all cut points), R^- (the sum of ranks for the opposite case), and *P* values. The *P* value indicates the level of significance of the hypothesis test. If the *P* value is lower than α (the significance level), the hypothesis is statistically significance with $100 \times (1 - \alpha)\%$ confidence level. Table 7 and Table 8 present the nonparametric statistical analysis of the PSOFAP and APSOVI algorithms on twelve test functions. Table 7 displays the sum of ranks and the mean and sum of scaled R^+ and R^- values for the average fitness values of the 21 cut points with 10 replications obtained by the two PSO algorithms. The larger mean of scaled R^+ indicates that the PSOFAP is superior to APSOVI. Because R^+ has 21 cut points in six test functions, PSOFAP is superior to APSOVI for all cut points in 6 of the 12 test functions.

The nonparametric statistical analysis performed using the Wilcoxon signed rank test is presented in Table 8; bold values represent the significance of the null hypothesis. Because the *P* values of f_1–f_6, f_8, and f_{10} are all lower than 0.05, the null hypothesis (H_0: PSOFAP > APSOVI) is rejected. Therefore, PSOFAP outperforms APSOVI in f_1–f_6, f_8, and f_{10}. Overall, we observe that the fitness values obtained using PSOFAP are lower than those obtained using APSOVI on eight test functions (f_1–f_6, f_8, and f_{10}), greater than those obtained using APSOVI on two test functions (f_7 and f_9), and approximately equal to those obtained using APSOVI on two test functions (f_{11} and f_{12}). Although the *P* values of f_7 and f_{11} are both greater than 0.05 (Table 8), the means of scaled R^+ are both greater than mean of scaled R^- (Table 7). Therefore, the performance of PSOFAP outperforms that of APSOVI on test functions f_7 and f_{11}.

Table 7: The signed rank score on mean of the fitness value of each cut points obtained by PSOFAP and APSOVI on test functions.

Measurement	Sum of ranks	Mean of scaled	Sum of scaled	Sum of ranks	Mean of scaled	Sum of scaled	Sum of ranks	Mean of scaled	Sum of scaled
Function	f_1:			f_2:			f_3:		
R^+	21	11	231	21	11	231	16	12.06	193
R^-	0	0	0	0	0	0	5	7.60	38
Sum	21			21			21		
Function	f_4:			f_5:			f_6:		
R^+	21	11	231	21	11	231	21	11	231
R^-	0	0	0	0	0	0	0	0	0
Sum	21			21			21		
Function	f_7:			f_8:			f_9:		
R^+	11	13.27	146	20	11.50	230	2	11.50	23
R^-	10	8.50	85	1	1	1	19	10.95	208
Sum	21			21			21		
Function	f_{10}:			f_{11}:			f_{12}:		
R^+	21	11	231	7	13.86	97	9	7.11	64
R^-	0	0	0	14	9.57	134	12	13.92	167
Sum	21			21			21		

Table 8: Wilcoxon signed rank test on mean fitness obtained by PSOFAP and APSOVI for test functions (α =0.05).

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
z	-4.015[a]	-4.015[a]	-2.964[a]	-4.015[a]	-4.015[a]	-4.015[a]	-1.060[a]	-4.125[a]	-3.215[b]	-4.015[a]	-.643[b]	-1.790[b]
***P* value**	**0**	**0**	**0.007**	**0**	**0**	**0**	.289	**0**	**0.001**	**0**	.520	.073

a. H_0: PSOFAP fitness value> APSOVI fitness value;
b. H_0: PSOFAP fitness value< APSOVI fitness value.

4.4 Solving Open Vehicle Routing Problem with Time Windows

In this study, we employ the open vehicle routing problem with time windows (OVRPTW) as an industrial experiment problem. In a typical OVRPTW problem, most studies discuss time windows, capacity, and routing constraints for vehicle destination. The OVRPTW seeks to obtain a set of nondepot returning vehicle routes for a fleet of capacitated vehicles in order to satisfy customers' requirements within fixed time intervals representing the earliest and latest times during the day that customers' can be serviced (Repoussis *et al.*, 2007). To evaluate the proposed algorithm, we use the well-known Solomon benchmark data sets (Solomon, 1987) as the experimental data. The Solomon benchmark problems consist of three classes according to the geographical distribution of 100 customers: randomly distributed (R), clustered (C), and a mix of randomly distributed and clustered (RC); each class is divided into two sets. These data sets have been widely used to evaluate the delivery subproblem in the OVRPTW problem with hard time windows and stochastic service times (VRPTW-ST; Errico *et al.*, 2016), with time windows and multiple

delivery men (VRPTWMD; Alvarez & Munari, 2017), and with time windows and driver-specific times (VRPTWDST; Schneider, 2016).

In this study, we employ two data sets, namely C1 and C2, for experiments each consisting of 100 customers and 25 vehicles. Data set C1 is characterized by vehicles with smaller capacity and customers with shorter time windows than those in data set C2. The objective of this experiment is to minimize the total travelling distance. The parameter settings for SPSO, APSOVI, and PSOFAP include a population size of 100 and five fixed computational time of 10, 20, 30, 40 and 50 s.

The experimental results reveal travelling distances obtained by the three PSO algorithms (Table 9; computational time (s) is denoted by C/T. Comparing SPSO and APSOVI, the average travelling distance obtained by PSOFAP in data sets C1 and C2 are both shorter for all fixed computational times. Overall, the proposed PSOFAP algorithm is more effective at finding the shortest path for visiting all customers.

Table 9: The experiment results on sets of Solomon's problems C1 and C2 obtained by SPSO, APSOVI and PSOFAP with 100 particles and 10 times replications.

Problems	C/T (sec)	**SPSO**			**APSOVI**			**PSOFAP**		
		Min	Average	Std. Dev.	Min	Average	Std. Dev.	Min	Average	Std. Dev.
C1	10	2281.433	3049.881	614.309	2312.699	2516.458	162.702	**2169.891**	**2333.801**	145.513
	20	2276.312	2670.067	342.323	2167.300	2396.416	169.814	**2085.171**	**2248.715**	121.535
	30	**2009.089**	2499.071	351.012	2068.113	2287.673	180.443	2067.807	**2222.943**	115.365
	40	**1979.636**	2337.580	282.384	2021.119	2245.350	178.279	2051.202	**2203.487**	118.453
	50	**1667.832**	2297.063	308.204	2003.357	2198.565	156.202	2028.266	**2158.237**	123.713
C2	10	2934.218	4138.840	933.239	2748.158	3040.130	192.110	**2616.630**	**2857.977**	172.799
	20	2814.004	3370.114	506.949	2661.096	2936.554	229.496	**2515.112**	**2740.137**	153.261
	30	2626.644	3132.655	316.673	2661.096	2894.075	234.651	**2515.112**	**2702.340**	142.436
	40	2516.945	2996.409	305.142	2626.002	2860.590	238.032	**2472.529**	**2678.565**	152.585
	50	**2354.621**	2980.859	368.070	2624.841	2849.076	237.002	2472.529	**2649.830**	120.966

5. Conclusions

This study proposes the PSOFAP algorithm, which focuses on avoiding premature convergence and enhancing efficiency to obtain more accurate approximate solutions. The proposed PSOFAP algorithm uses fitness performance to adjust three parameters of individual particles according to their individual velocities. In PSOFAP, each particle is assigned an ideal velocity based on its fitness performance over the swarm, and the deviation between its real velocity and ideal velocity is determined. The real velocity of a

particle indicates the movement range of each particle at the next iteration. Hence, the proposed PSOFAP searches for more suitable parameter values for each particle on the basis of its fitness performance, and thus obtains improved candidate solutions at each iteration. Experimental results and statistical analysis indicated that the proposed PSOFAP algorithm obtained an acceptable approximate solution after fewer iterations and in shorter computational time, especially when solving multimodal and large-sized problems. Even for the industrial problem, the OVRPTW problem, PSOFAP obtained a more optimal solution than did SPSO and APSOVI.

Finally, future studies on variants of the PSO algorithm with self-adaptive parameters should consider the following. First, studies could focus on investigating why unimodal functions with small dimensions obtain poor results, which is relevant to the robustness of the PSOFAP algorithm. Second, studies should also expand the parameter self-adjusting mechanism to other evolutionary algorithms, such as the genetic algorithm, ant colony optimization algorithm, and differential evolution algorithm. Third, the proposed PSOFAP algorithm could be applied to industrial problems.

Reference

Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization, 31*(4), 635-672.

Alvarez, A., & Munari, P. (2017). An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen. *Computers & Operations Research*, 83, 1-12.

Biswas, A. & Biswas, B., (2017). Analyzing evolutionary optimization and community detection algorithms using regression line dominance. *Informance Science*, 396, 185-201.

Chatterjee, A., & Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & Operations Research, 33*(3), 859-871.

Chen, C.-A., & Chiang, T.-C. (2015). *Adaptive differential evolution: A visual comparison.* Paper presented at the Evolutionary Computation (CEC), IEEE Congress on.

Chen, M., & Ludwig, S. A. (2012). Discrete particle swarm optimization with local search strategy for rule classification. In Nature and Biologically Inspired Computing (NaBIC), 2012 *Fourth World Congress on* (pp. 162-167). *IEEE*.

Clerc, M. (2005). *L'optimisation par essaims particulaires*: Hermes Science Publications.

Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on, 6*(1), 58-73.

Derrac, J., García, S., Hui, S., Suganthan, P. N., & Herrera, F. (2014). Analyzing convergence performance of evolutionary algorithms: a statistical approach. *Information Sciences*, 289, 41-58.

Ding, J., Lü, Z., Zhou, T., & Xu, L. (2017), A quality and distance guided hybrid algorithm for the vertex separator problem. *Computers and Operations Research*, 78, 255-266.

Eberhart, R. C., & Kennedy, J. (1995). *A new optimizer using particle swarm theory.* Paper presented at the In Proceedings of the sixth international symposium on micro machine and human science

Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, *3*(2), 124-141.

Errico, F., Desaulniers, G., Gendreau, M., Rei, W., & Rousseau, L. M. (2016). A priori optimization with recourse for the vehicle routing problem with hard time windows and stochastic service times. *European Journal of Operational Research*, *249*(1), 55-66.

Güvenc, U., Katırcıoğlu, F. (2017). Escape velocity: a new operator for gravitational search algorithm. *Neural Comput & Applic*, 1-16.

Ghamisi, P., & Benediktsson, J. A. (2015). Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *IEEE Geoscience and Remote Sensing Letters*, 12(2), 309-313.

Holland, J. H. (1975). Adaptation in natural and artificial system: an introduction with application to biology, control and artificial intelligence. *Ann Arbor, University of Michigan Press*.

Iwasaki, N., Yasuda, K., & Ueno, G. (2006). Dynamic parameter tuning of particle swarm optimization. *IEEJ Transactions on Electrical and Electronic Engineering*, *1*(4), 353-363.

Jau, Y. M., Su, K. L., Wu, C. J., & Jeng, J. T. (2013). Modified quantum-behaved particle swarm optimization for parameters estimation of generalized nonlinear multi-regressions model based on Choquet integral with outliers. *Applied Mathematics and Computation*, 221, 282-295.

Jiang, M., Luo, Y., & Yang, S. (2007). Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, *102*(1), 8-16.

Jiang, S. & Yang, S. (2017), A Steady-State and Generational Evolutionary Algorithm for Dynamic Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, *21*(1), 65-82.

Khan, S. A., & Engelbrecht, A. P. (2012). A fuzzy particle swarm optimization algorithm for computer communication network topology design. *Applied Intelligence*, 36(1), 161-177.

Liang, J. J., Qu, B. Y., Suganthan, P. N., & Hernández-Díaz, A. G. (2013). Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, 201212.

Liu, Y., Bi, J.W., Fan, Z.P. (2017). A method for multi-class sentiment classification based on an improved one-vs-one (OVO) strategy and the support vector machine (SVM) algorithm. *Information Sciences*, 394, 38-52.

Lu, Y. C., Jan, J. C., Hung, S. L., & Hung, G. H. (2013). Enhancing particle swarm optimization algorithm using two new strategies for optimizing design of truss structures. *Engineering Optimization*, 45(10), 1251-1271.

Maione, G., & Punzi, A. (2013). Combining differential evolution and particle swarm optimization to tune and realize fractional-order controllers. *Mathematical and Computer Modelling of Dynamical Systems*, 19(3), 277-299.

Martins, A. A. , & Aderemi O. A. (2014), "Improved Particle Swarm Optimization with a Collective Local Unimodal Search for Continuous Optimization Problems," *The Scientific World Journal*, vol. 2014, 23.

Montalvo, I., Izquierdo, J., Pérez-García, R., & Herrera, M. (2010). Improved performance of PSO with self-adaptive parameters for computing the optimal design of water supply systems. *Engineering Applications of Artificial Intelligence*, *23*(5), 727-735.

Niknam, T., Narimani, M. R., & Jabbari, M. (2013). Dynamic optimal power flow using hybrid particle swarm optimization and simulated annealing. *International Transactions on Electrical Energy Systems*, 23(7), 975-1001.

Olorunda, O., & Engelbrecht, A. P. (2008). *Measuring exploration/exploitation in particle swarms using swarm diversity.* Paper presented at the Evolutionary Computation, 2008. CEC 2008.(IEEE World

Congress on Computational Intelligence). IEEE Congress on.

Omran, M. G., Salman, A., & Engelbrecht, A. P. (2006). Dynamic clustering using particle swarm optimization with application in image segmentation. *Pattern Analysis and Applications, 8*(4), 332-344.

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *Evolutionary Computation, IEEE Transactions on,* 8(3), 240-255.

Repoussis, P. P., Tarantilis, C. D., & Ioannou, G. (2007). The open vehicle routing problem with time windows. *Journal of the Operational Research Society*, 58(3), 355-367.

Schneider, M. (2016). The vehicle-routing problem with time windows and driver-specific times. *European Journal of Operational Research*, *250*(1), 101-119.

Shahzad, F., Masood, S., & Khan, N. K. (2014). Probabilistic opposition-based particle swarm optimization with velocity clamping. *Knowledge and information systems*, 39(3), 703-737.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 69-73.

Shieh, H.-L., Kuo, C.-C., & Chiang, C.-M. (2011). Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification. *Applied Mathematics and Computation*, 218(8), 4365-4383.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, *35*(2), 254-265.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL report*, 2005005, 2005.

Sun, J., & Wu, S. (2011). Route planning of cruise missile based on improved particle swarm algorithm. *Journal of Beijing University of Aeronautics and Astronautics, 37*(10), 1228-1232.

Wilcoxon , F. (1945). Individual comparisons by ranking methods. *Biometrics Bull*, *1*(6), 80-83.

Xu, G. (2013). An adaptive parameter tuning of particle swarm optimization algorithm. *Applied Mathematics and Computation*, 219(9), 4560-4569.

Yasuda, K., Ide, A., & Iwasaki, N. (2003). Adaptive particle swarm optimization. Paper presented at t*he Systems, Man and Cybernetics, 2003. IEEE International Conference on*.

Yasuda, K., & Iwasaki, N. (2004). Adaptive particle swarm optimization using velocity information of swarm. Paper presented at *the Systems, Man and Cybernetics, 2004 IEEE International Conference on*.

Zhang, H., Kennedy, D. D., Rangaiah, 2013 G. P., & Bonilla-Petriciolet, A. (2011). Novel bare-bones particle swarm optimization and its performance for modeling vapor–liquid equilibrium data. *Fluid Phase Equilibria*, 301(1), 33-45.

Zhang, J., Huang, D.-S., & Liu, K.-H. (2007). Multi-sub-swarm particle swarm optimization algorithm for multimodal function optimization. Paper presented at the *Evolutionary Computation, CEC 2007*. IEEE Congress on.

Zhang, Y., Wang, S., & Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015.

Zong, X., Xiong, S., & Fang, Z. (2014). A conflict–congestion model for pedestrian–vehicle mixed evacuation based on discrete particle swarm optimization algorithm. *Computers & Operations Research*, 44, 1-12.

Research Highlights

1. A novel PSO for rapid convergence to an approximate optimal solution is proposed.
2. The algorithm is based on fitness performance to adjust three parameters of individual particle.
3. In comparing with twelve benchmark problems, the results show the fast convergence.