Quality and Robustness Improvement for Real World Industrial Systems using a Fuzzy Particle Swarm Optimization

Sai Ho Ling, *Senior Member, IEEE,* Kit Yan Chan, *Member, IEEE,* Frank Hung Fat Leung, *Senior Member, IEEE,* Frank Jiang, *Member, IEEE,* and Hung Nguyen, *Senior member, IEEE*

***Abstract*—This paper presents a novel fuzzy particle swarm optimization with cross-mutated operation (FPSOCM), where a fuzzy logic system developed based on the knowledge of swarm intelligent is proposed to determine the inertia weight for the swarm movement of particle swarm optimization (PSO) and the control parameter of a newly introduced cross-mutated operation. Hence, the inertia weight of the PSO can be adaptive with respect to the search progress. The new cross-mutated operation intends to drive the solution to escape from local optima. A suite of benchmark test functions are employed to evaluate the performance of the proposed FPSOCM. Experimental results show empirically that the FPSOCM performs better than the existing hybrid PSO methods in terms of solution quality, robustness, and convergence rate. The proposed FPSOCM is evaluated by improving the quality and robustness of two real world industrial systems namely economic load dispatch system and self provisioning systems for communication network services. These two systems are employed to evaluate the effectiveness of the proposed FPSOCM as they are the multi-optima and non-convex problems. The performance of FPSOCM is found to be significantly better than that of the existing hybrid PSO methods in a statistical sense. These results demonstrate that the proposed FPSOCM is a good candidate on solving product or service engineering problems which are multi-optima or non-convex natures.**

***Index Terms*—Economic load dispatch, Email communication services, Fuzzy logic system, Particle swarm optimization.**

I. Introduction

Recent research demonstrates that Particle swarm optimization (PSO) is a more effective optimization method in solving hard optimization problems comparing with other commonly used stochastic optimization methods such as evolutionary algorithms, tabu search and simulated annealing, where it has more comparable or even superior search performance with higher and more stable convergence rates in performing the optimizations [1]. As PSO is inspired by the social behaviours of animals or insects for performing optimizations, it has memory [2]; previously visited best positions are remembered, which is different from other evolution algorithms that do not keep the crucial information as the population changes. Hence, best solution quality can be generally obtained by PSO with a shorter computational time on many industrial applications such as product design [3], power systems design [4]–[9], parameter learning of neural networks [10], [11], speech recognition [12], photovoltaic system [13], traffic flow forecasting [14], biomedical system [15], [16], manufacturing process design [17], [18], optimization in dynamic environment [19], parameter estimation and identification [20]–[22], etc. Although reasonable solutions can generally be obtained by the PSO within a reasonable computational time, enhancement of the operations and the mechanisms of the PSO is essentially required in order to obtain better solutions. A commonly used enhancement approach is to integrate other optimization operations into the PSO. Angekine [23] developed a hybrid PSO by integrating with a selection mechanism in order to select elitist particles. Noel and Jannett [24] developed a gradient descent based PSO namely HGPSO by selecting appropriate gradient information in order to achieve faster convergence and aid to obtain global optimum. However, the computational effort is increased by the approach in computing the gradient descents, and also experimental results show that the approach performs poorly in solving multimodal problems which have many sub-optima. Juang [11] proposed a hybrid PSO algorithm namely HGAPSO which use evolutionary operations including crossover, mutation and reproduction to control swarm movement, and also a hybrid PSO namely HPSOM was proposed [4] by integrating the PSO with mutation operation. Both HGAPSO and HPSOM inject random components into particles using mutation, but the mutating space used in both approaches is fixed throughout the search. Although premature convergence is more likely to be avoided, the approach can be further improved by varying the mutating space with respect to the searching progress of the PSO.

Shi [25] developed a PSO of which the inertia weight factor varies linearly with respect to iterations. More recently, a hybrid PSO with wavelet mutation operation (HPSOWM) was proposed in [26], of which the mutating space varied based on the wavelet theory. By solving a few industrial problems, experimental results show that better and more robust solutions can be obtained with smaller computational time. Although the approaches provide a balance between

S.H. Ling and Hung Nguyen are with the Centre for Health Technologies, Faculty of Engineering and Information Technology, University of Technology, Sydney, NSW, Australia e-mail: Steve.Ling@uts.edu.au.

K.Y. Chan is with Department of Electrical and Computer Engineering ,Curtin University, WA, Australia.

F.H.F. Leung is with Centre for Signal Processing, Dept of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hum, Hong Kong.

Frank Jiang is with Faculty of Engineering,UNSW, Australia.

Manuscript received xx xx, 2014; revised xx xx, 201x.

the global exploration and local exploitation, they are not appropriate to assume the searching progresses of the PSO are linear or wavelet characteristics, and it is also impractical and almost impossible to mathematically model the searching progress of the PSO, in order to determine the appropriate inertia weight for searching the optimum.

In this paper, a fuzzy logic based PSO with cross-mutated operation namely FPSOCM is proposed by introducing it with two novel components, namely (i) fuzzy inertia weight and (ii) cross-mutated (CM) operation. (i) the value of the fuzzy inertia weight is determined based on a fuzzy inference system which consists of a set of linguistic rules in representing the searching characteristics of the PSO. By dynamically changing the fuzzy inertia weight, the dynamic of the swarm can be varied with respect to the searching progress of the PSO. Hence, solutions with better qualities are more likely to be searched; ii) the CM injects momentum to the swarm when the progress of the PSO is saturating, where the amount of momentum is controlled based on the fuzzy inference system. It intends to further avoid the PSO in searching the local optima. Hence, the limitation of the classical PSO can be further tackled.

A suite of 17 benchmark test functions with different optimization characteristics were used to test the performance of the FPSOCM. The resulting fuzzy inertia weight and CM operation aid the proposed algorithm to offer better solution quality and solution reliability than the improved PSO (IPSO) with inertia weight [25] and constriction factor [27], and other recently developed hybrid PSO methods [4], [11], [26]. We further evaluate the performance of the FPSOCM by improving the quality and robustness of two real world industrial systems namely economic load dispatch system and self provisioning systems for communication network services. The results are compared with those from other existing hybrid PSO methods on these applications. We can see that the performance of FPSOCM in tackling these applications are improved with statistical significance. Hence, the optimal systems can achieved better results in term of robustness and solution quality, that can offer more stable but cheaper power supply or network service to customers, than the exiting PSO methods.

This paper is organized as follows. Section II presents the details of FPSOCM. Experimental study and analysis will be discussed in Section III, and 17 benchmark test functions with different optimization characteristics will be given to evaluate the performance of the proposed method. Furthermore, the sensitivity of the parameter of the cross-mutated operation will be discussed in this section. In Section IV, the three real-world applications will be discussed. Finally, a conclusion will be drawn in Section V.

II. Fuzzy particle swarm optimization with cross-mutated operation (FPSOCM)

Particle swarm optimization (PSO) models the social behavior of a swarm like bird flocking and fish schooling. The swarm is composed of a number of particles. Every particle traverses a search space for the best fitness value.

1) PSO with constriction and inertia weight factors (IPSO): PSO with inertia weight [25] and PSO with constriction factor [27] were reported to show improved searching ability over the standard PSO [28]. Algorithm II.1 gives the pseudo code of the PSO with constriction and inertia weight factors (IPSO). Under Algorithm II.1, $X(t)$ denotes a swarm at the t-th iteration. Each particle $\mathbf{x}^i(t) \in X(t)$ contains κ elements $x_j^i(t) \in \mathbf{x}^i(t)$, where i = 1, 2,... , γ and j = 1, 2,... , κ; γ denotes the number of particles in the swarm and κ is the dimension of a particle. At the beginning, the swarm particles are initialized and then evaluated by a defined fitness function $f\left(\mathbf{x}^i(t)\right)$. The current generation number t is initialized to 0. The job of PSO is to minimize the fitness value through an iterative process.

Algorithm II.1: Pseudo code for IPSO($X(t)$)

```
t ← 0
Initialize X(t)
output (f(X(t)))
while <not termination condition>
  do { t ← t + 1
       Update velocity v(t) based on (3) − (5).
       Ensure the updated velocity is in the region of v_max
       if v(t) > v_max
         then v(t) = v_max
       if v(t) < −v_max
         then v(t) = −v_max
       Generate a new swarm X(t) based on (2).
       Ensure the updated x_j^i(t) is inside the boundary
       if x_j^i(t) > ρ_max_j
         then x_j^i(t) = ρ_max_j
       if x_j^i(t) < ρ_min_j
         then x_j^i(t) = ρ_min_j
       output (f(X(t)))
return (g)
comment: g is the best particle among all particles (solution)
```

The evolution realised by PSO is governed by the velocity (flight speed) of the particles in the search space. The velocity $v_j^i(t)$ and the position $x_j^i(t)$ of the j-th element of the i-th particle at the t-th generation is given by the following formulae:

$$v_j^i(t) = 2 \cdot r_1 \cdot \left(p_j^i - x_j^i(t-1)\right) + 2 \cdot r_2 \cdot \left(g_j - x_j^i(t-1)\right) \quad (1)$$

and

$$x_j^i(t) = x_j^i(t-1) + v_j^i(t) \quad (2)$$

where

$p^i = \left[\begin{array}{cccc} p_1^i & p_2^i & \ldots & p_\kappa^i \end{array}\right]$ is the best position of the particle i, and $g = \left[\begin{array}{cccc} g_1 & g_2 & \ldots & g_\kappa \end{array}\right]$ is the best particle among all the particles; r_1 and r_2 are random numbers in the range of [0,1]. In [29], an improved PSO (IPSO) makes use of a

constriction factor and an inertia weight factor to obtain $v_j^i(t)$:

$$v_j^i(t) = k \cdot \{\omega(t) \cdot v_j^i(t-1) + \varphi_1 \cdot r_1 \cdot (p_j^i - x_j^i(t-1)) \varphi_2 \cdot r_2 \cdot (g_j - x_j^i(t-1))\} \quad (3)$$

where $\omega(t)$ is the inertia weight factor; φ_1 and φ_2 are acceleration constants; k is the constriction factor derived from the stability analysis of (3) for assuring the system to converge but not prematurely [29]. In this paper, k is related to φ_1 and φ_2 as follows:

$$k = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad (4)$$

where $\varphi = \varphi_1 + \varphi_2$ and $\varphi > 4$.

The particle velocity is limited by a maximum value $v_{\max}$ (3). This parameter $v_{\max}$ determines the resolution of the searched regions between the present position and the target position. The value of this limit governs the local exploitation of the problem space. Practically, it emulates the incremental changes of human learning. If the value of $v_{\max}$ is too large, the particles might fly past good solutions. If the value of $v_{\max}$ is too small, the particles may not sufficiently explore beyond the local solutions. Based on our experiments, it is suggested $v_{\max}$ can be assigned with a value of 10% to 20% of the dynamic range of each element. After updating the velocity of all particles, we get a new swarm $X(t)$ based on (2). To ensure every particle element x_j^i in $X(t)$ falls within the range $\left[\ \rho_{min_j},\ \ \rho_{max_j}\ \right]$, we add the following conditions. If $x_j^i(t) > \rho_{\max_j}$, the updated $x_j^i(t)$ should be equal to $\rho_{\max_j}$. Similarly, if $x_j^i(t) < \rho_{\min_j}$, the updated $x_j^i(t)$ should be equal to $\rho_{\min_j}$. Here, $\boldsymbol{\rho_{\min}} = \left[\ \rho_{\min_1}\ \ \rho_{\min_2}\ \ \cdots\ \ \rho_{\min_\kappa}\ \right]$ and $\boldsymbol{\rho_{\max}} = \left[\ \rho_{\max_1}\ \ \rho_{\max_2}\ \ \cdots\ \ \rho_{\max_\kappa}\ \right]$; $\rho_{\min_j}$ and $\rho_{\max_j}$ are the minimum and maximum values of $x_j^i(t)$ respectively, and $j = 1, 2, ..., \kappa$.

IPSO utilizes p^i and g to affect the searching direction so that the particles will move in different manner. Also, the convergence can be made gradual towards p^i and g. A suitable choice of the value of $\omega(t)$ offers a balance between the global exploration and local exploitations. $\omega(t)$, in general, can be given by the equation:

$$\omega(t) = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{T} \times t \quad (5)$$

where T is the total number of iteration, t is the current iteration number, $\omega_{\max}$ and $\omega_{\min}$ are the upper and lower limits of $\omega(t)$, which are normally set to 1.1 and 0.1 respectively [26], [30].

A. Fuzzy PSO with cross-mutated operation

The pseudo code for FPSOCM is given in Algorithm II.2. A fuzzy inertia weight $\tilde{\omega}(t)$ is first proposed to improve the searching quality. A cross-mutated (CM) operation is also added to tackle the limitation of IPSO [25] [27] being easy to trap in some local minima.

1) fuzzy inertia weight: In IPSO, the inertia weight $\omega(t)$ provides a balance between the global exploration and local exploitation of the swarm. When ω is linearly related to t as in (5), if the value of t/T is smaller, more global exploration is done; if it is larger, more fine-tuning (local exploitation) is realised. However, a linear relation between $\omega(t)$ and t may not be so appropriate because the search progress of the swarm is not a linear movement. We propose a nonlinear inertia weight $\tilde{\omega}(t)$ to enhance the searching performance. The value of $\tilde{\omega}(t)$ is evaluated by a 2-input fuzzy inference system as one of its 2 outputs. (The other output is the control parameter $\beta(t)$ of the CM operation that will be discussed in the later sub-section.) The inputs of the fuzzy inference system are $||\varsigma(t)||$ and t/T. $||\varsigma(t)||$ is the normalized standard deviation of fitness values among all the particles, of which a larger value implies the particles being far apart from one another. The term $||\varsigma(t)||$ is given by:

$$||\varsigma(t)|| = \sqrt{\frac{1}{\gamma}\sum_{i=1}^{\gamma}\left(||f(\mathbf{x}^i(t))|| - ||\bar{f}(\mathbf{x}^i(t))||\right)^2} \quad (6)$$

where

$$||\bar{f}(\mathbf{x}^i(t))|| = \frac{1}{\gamma}\sum_{i=1}^{\gamma}||f(\mathbf{x}^i(t))|| \quad (7)$$

and $\|\cdot\|$ denotes the l_2 vector norm.

Algorithm II.2: PSEUDO CODE FOR FPSOCM($X(t)$)

```
t ← 0
Initialize X(t)
Define the probability of CM operation p_cm
output (f(X(t)))
while <not termination condition>
    do
        t ← t + 1
        output (t/T)
        output (||ς(t)|| based on (6) and (7))
        Find the inertia weight ω̃_k(t) by
         using fuzzy inference system based on (8) − (10).
        Update velocity v(t) based on (11).
        Find the control parameter β(t) by
         using fuzzy inference system based on (14) − (15).
        Generate a random number R_cm
        if R_cm > p_cm
          then Perform cross-mutated operation
         based on (12) − (13).
        Ensure the updated velocity is in the region of v_max
        Generate a new swarm X(t) based on (2).
        Ensure the updated x_j^i(t) is inside the boundary
        output (f(X(t)))
return (g)
comment: g is the best particle among all particles (solution)
```

The following fuzzy rules govern the fuzzy inertia weight $\tilde{\omega}(t)$:

Rule j : IF $||\varsigma(t)||$ is N_1^j, AND t/T is N_2^j, THEN $\tilde{\omega}\left(t\right)=\sigma_j$,

$$j = \quad 1, \quad 2, \quad \ldots, \quad \varepsilon \tag{8}$$

where N_1^j and N_2^j are fuzzy terms of rule j, ε is the number of rules, $\sigma_j \in \left[\begin{array}{cc} \omega_{min} & \omega_{max} \end{array}\right]$ is a singleton to be determined, with ω_{min} and ω_{max} being set at 0.1 and 1.1 respectively [26], [30]. The final value of $\tilde{\omega}\left(t\right)$ is given by:

$$\tilde{\omega}\left(t\right)=\sum_{j=1}^{\varepsilon} m_j\left(t\right)\sigma_j \tag{9}$$

where

$$m_j\left(t\right)=\frac{\mu_{\mathrm{N}_1^j}\left(||\varsigma(t)||\right)\times\mu_{\mathrm{N}_2^j}\left(t/T\right)}{\sum\limits_{j=1}^{\varepsilon}\left(\mu_{\mathrm{N}_1^j}\left(||\varsigma(t)||\right)\times\mu_{\mathrm{N}_2^j}\left(t/T\right)\right)} \tag{10}$$

$\mu_{\mathrm{N}_1^j}\left(||\varsigma(t)||\right)$ and $\mu_{\mathrm{N}_2^j}\left(t/T\right)$ are the membership function values corresponding to N_1^j and N_2^j respectively. It should be noticed that $\tilde{\omega}\left(t\right)$ will replace $\omega(t)$ in (3) to produce $v_j^i\left(t\right)$:

$$\begin{aligned} v_j^i(t) = k\cdot\{\tilde{\omega}(t)\cdot v_j^i(t-1)+ \\ \varphi_1\cdot r_1\cdot(p_j^i-x_j^i(t-1)) \\ \varphi_2\cdot r_2\cdot(g_j-x_j^i(t-1))\} \end{aligned} \tag{11}$$

As shown in Fig. 1, the fuzzy inference system uses three membership functions to model each input: L (Low), M (Medium), and H (High). The output singletons use five terms, namely VL (Very Low), L (Low), M (Medium), H (High), and VH (Very High). The values for these five terms are set at 0.1, 0.35, 0.6, 0.85, 1.1 respectively. The values are determined based on the values of ω_{min} and ω_{max}. For example, the output term is Very High, the value is equal to ω_{max} (=1.1). If the output term is Medium, then the value is equal to (ω_{min}+ω_{max})/2 (=0.6).With $||\varsigma(t)||$ and t/T as inputs, the 9 linguistic IF-THEN fuzzy rules for determining $\tilde{\omega}(t)$ are given as follows:

Rule 1: IF $||\varsigma(t)||$ is "L" AND t/T is "L", THEN $\tilde{\omega}\left(t\right)$ is "VH" (= 1.1)
Rule 2: IF $||\varsigma(t)||$ is "M" AND t/T is "L", THEN $\tilde{\omega}\left(t\right)$ is "H" (= 0.85)
Rule 3: IF $||\varsigma(t)||$ is "H" AND t/T is "L", THEN $\tilde{\omega}\left(t\right)$ is "VH" (= 1.1)
Rule 4: IF $||\varsigma(t)||$ is "L" AND t/T is "M", THEN $\tilde{\omega}\left(t\right)$ is "M" (= 0.6)
Rule 5: IF $||\varsigma(t)||$ is "M" AND t/T is "M", THEN $\tilde{\omega}\left(t\right)$ is "M" (= 0.6)
Rule 6: IF $||\varsigma(t)||$ is "H" AND t/T is "M", THEN $\tilde{\omega}\left(t\right)$ is "H" (= 0.85)
Rule 7: IF $||\varsigma(t)||$ is "L" AND t/T is "H", THEN $\tilde{\omega}\left(t\right)$ is "VL" (= 0.1)
Rule 8: IF $||\varsigma(t)||$ is "M" AND t/T is "H", THEN $\tilde{\omega}\left(t\right)$ is "VL" (= 0.1)
Rule 9: IF $||\varsigma(t)||$ is "H" AND t/T is "H", THEN $\tilde{\omega}\left(t\right)$ is "L" (= 0.35)

The rationale of the fuzzy rules for determining $\tilde{\omega}\left(t\right)$ is given as follows. The value of t/T represents the evolution stage (a small t/T represents an early stage.) The value of $\tilde{\omega}\left(t\right)$ is set higher when the value of t/T is smaller (in early stage) so that a larger value of the particle velocity is given for global searching. Similarly, a larger value of t/T implies a smaller value of the particle velocity for local searching and fine-tuning. Thus, $\tilde{\omega}\left(t\right)$ of the fuzzy rules 1, 2, and 3 (t/T is "L") has a larger value than that of the rules 4, 5, and 6 (t/T is "M"). As $||\varsigma(t)||$ is the normalized standard deviation of fitness values among all the particles, a large value of $||\varsigma(t)||$ implies that the particle locations are far away from one another. In rules 1 to 3, the searching process is in its early stage (t/T is "L"). When $||\varsigma(t)||$ is "H", the wide-spread particle locations implies a larger value of $\tilde{\omega}\left(t\right)$ should be used for global exploration. When $||\varsigma(t)||$ is "L" in the early stage, the value of $\tilde{\omega}\left(t\right)$ is also set large as the chance of the solution being trapped in a local optimum is high. In rule 2, the value of $||\varsigma(t)||$ is "M", and we set the value of $\tilde{\omega}\left(t\right)$ to be slightly smaller than that in rules 1 and 3 in the early stage (t/T is "L"). In rule 4 to rule 6, the searching process is in its middle stage (t/T is "M"). The rationale for suggesting the value of $\tilde{\omega}\left(t\right)$ is similar to that for rules 1-3. However, when $||\varsigma(t)||$ is "L", the value of $\tilde{\omega}\left(t\right)$ is smaller than that when $||\varsigma(t)||$ is "H". It is because the optimal solution may have been found in the middle stage when a smaller value of $\tilde{\omega}\left(t\right)$ is given. In rule 7 to rule 9, the searching process is in its late stage (t/T is "H"). Hence, the searching process is undergoing a fine-tuning process (local exploitation) to reach the optimal solution. As a result, when the value of $||\varsigma(t)||$ is "L", the locations of particles are close to one another and near the optimal solution and the smallest value of $\tilde{\omega}\left(t\right)$ is used.

2) Cross-mutated operation: The proposed cross-mutated (CM) operation merges the ideas of crossover and mutation operations of the Genetic Algorithm [31] in order to help the particles escaping from some local optima. By injecting random components into particles, the CM operation improves the PSO performance, particularly when it is used to tackle multimodal optimization problems with many local minima.

With the CM operation, the velocity of every particle element will have a chance to undergo CM operation governed by a probability of CM operation, $p_{cm} \in \left[\begin{array}{cc} 0 & 1 \end{array}\right]$, which is defined by the user. A random number R_{cm} between 0 and 1 will be generated for each particle element such that if it is less than or equal to p_{cm}, the CM operation will take place on that element. The value of the p_{cm} affects the solution quality, and its sensitivity analysis with experimental results will be discussed later.

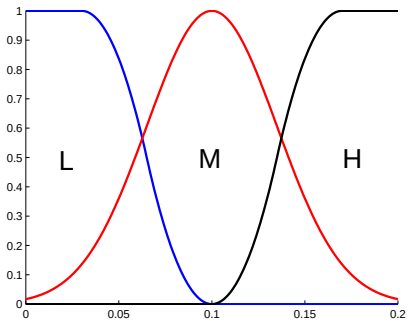
After taking the CM operation, the resulting velocity of a particle element is given by:

$$\bar{v}_j^i(t)=\begin{cases}\left(1-\beta(t)\right)v_j^i\left(t\right)+\beta(t)\tilde{v}_j^i\left(t\right), r_3>0.5\\ \left(1-\beta(t)\right)v_j^i\left(t\right)-\beta(t)\tilde{v}_j^i\left(t\right), r_3\le 0.5\end{cases} \tag{12}$$

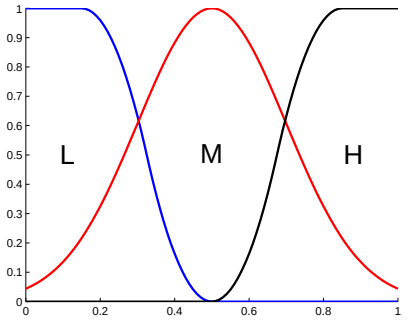
where

$$\tilde{v}_j^i\left(t\right)=0.25\left\{r_4\cdot\left(\rho_{\max_j}-\rho_{\min_j}\right)+\rho_{\min_j}\right\} \tag{13}$$

$r_3, r_4 \in \left[\begin{array}{cc} 0 & 1 \end{array}\right]$ is a random number, $v_j^i\left(t\right)$ is determined by (11), $\tilde{v}_j^i\left(t\right)$ is a random velocity of particle element and its value is bounded within 0.25 of the range of the particle



(a) fuzzy input 1: $||\varsigma(t)||$



(b) fuzzy input 2: t/T

Fig. 1. Membership functions (a) x-axis: $||\varsigma(t)||$,y-axis: $\mu_{N_1}(||\varsigma(t)||)$ (b) x-axis: t/T ,y-axis: $\mu_{N_2}(t/T)$.

element value; a control parameter $\beta(t)$ is introduced into the CM operation, which is governed by some fuzzy rules based on human knowledge. The maximum velocity and minimum velocity are therefore 0.25 of the range of particle element value. The value of 0.25 is chosen by trial and error through experiments. If this value is too large or too small, the searching performance might be degraded. In (12), the resulting velocity of particle element $\bar{v}_j^i(t)$ combines the information of $v_j^i(t)$ and $\tilde{v}_j^i(t)$, exhibiting the characteristic of the crossover operation. However, in (12), $\bar{v}_j^i(t)$ is changed individually the mutation operation. Therefore, it is called the cross-mutated (CM) operation.

In (12), the control parameter $\beta(t)$ provides a balance to control the resulting velocity $\bar{v}_j^i(t)$ converging toward $v_j^i(t)$ or $\tilde{v}_j^i(t)$. If $\beta(t)$ is approaching 0, $\bar{v}_j^i(t)$ will approach $v_j^i(t)$. Conversely, when $\beta(t)$ is approaching 1, $\bar{v}_j^i(t)$ will approach $\tilde{v}_j^i(t)$. Hence, $\tilde{v}_j^i(t)$ in (13) provides a means for the particle element to escape from a local optimum through a random movement governed by $\beta(t)$, of which the value is generated by the following fuzzy rules:

Rule j : IF $||\varsigma(t)||$ is N_1^j AND t/T is N_2^j, THEN $\beta(t) = \chi_j$,

$$j = \quad 1, \quad 2, \quad \ldots, \quad \varepsilon \tag{14}$$

where χ_j is a singleton to be determined. The final value of $\beta(t)$ is given by:

$$\beta(t) = \sum_{j=1}^{\varepsilon} m_j(t)\chi_j \tag{15}$$

where $m_j(t)$ is given by (10)

The output singletons use five terms, namely VL (Very Low), L (Low), M (Medium), H (High), and VH (Very High). As the control parameter of CM is in the range of 0.1 to 0.5. Thus, the values for these five terms are set at 0.1, 0.2, 0.3, 0.4, 0.5 respectively. Here, 9 linguistic IF-THEN fuzzy rules for determining $\beta(t)$ are used and listed as follows:

Rule 1: IF $||\varsigma(t)||$ is "L" AND t/T is "L", THEN $\beta(t)$ is "VH" (= 0.5)
Rule 2: IF $||\varsigma(t)||$ is "M" AND t/T is "L", THEN $\beta(t)$ is "H" (= 0.4)
Rule 3: IF $||\varsigma(t)||$ is "H" AND t/T is "L", THEN $\beta(t)$ is "VH" (= 0.5)
Rule 4: IF $||\varsigma(t)||$ is "L" AND t/T is "M", THEN $\beta(t)$ is "H" (= 0.4)
Rule 5: IF $||\varsigma(t)||$ is "M" AND t/T is "M", THEN $\beta(t)$ is "M" (= 0.3)
Rule 6: IF $||\varsigma(t)||$ is "H" AND t/T is "M", THEN $\beta(t)$ is "H" (= 0.4)
Rule 7: IF $||\varsigma(t)||$ is "L" AND t/T is "H", THEN $\beta(t)$ is "VL" (= 0.1)
Rule 8: IF $||\varsigma(t)||$ is "M" AND t/T is "H", THEN $\beta(t)$ is "L" (= 0.2)
Rule 9: IF $||\varsigma(t)||$ is "H" AND t/T is "H", THEN $\beta(t)$ is "L" (= 0.2)

The rationale for formulating the fuzzy rules is similar to that for formulating the rules governing the fuzzy inertia weight in section II.A1. As mentioned before, rules 1-3 with t/T being "L" correspond to the early searching process, and rules 7-9 correspond to the late searching process. The values of $\beta(t)$ in rules 1-3 are larger than those in rules 7-9. A more significant random velocity (higher value of $\beta(t)$ in (12)) provides more global exploration in the early stage. Conversely, the effect of the random velocity should be reduced in the late stage for more fine-tuning (local exploitation).

In the early stage, when $||\varsigma(t)||$ is "L", the locations of particles are close to one another. Hence, we have to set the value of $\beta(t)$ to be larger than that when $||\varsigma(t)||$ is "M" as the chance of trapping in a local optimum is high. Conversely, when the value of $||\varsigma(t)||$ is "L", a small $\beta(t)$ is used to fine-tune the solutions in the late stage.

III. Benchmark Test Function: Results and Analysis

A suite of 17 benchmark test functions [31]–[33] are used to test the performance of FPSOCM. Different landscapes of optimization problems are covered by these benchmark test functions. They can be divided into three categories. The first one is the category of unimodal functions, which involves a

symmetric model with a single minimum. The second one is the category of multimodal functions with a few local minima. The last one is the category of multimodal functions with many local minima. Out of the 17 functions, 5 are benchmark test functions [32] with shift and rotate in different categories. The expressions of these functions are tabulated in Table I. (The details of the parameter a, b, c, p in f_{kowa}, f_{hart} and the function $u()$ in f_{pen} are given in [33]).

Name	Test function $f(\mathbf{x})$	Domain range
f_{sphere}	$\sum_{i=1}^{30} x_i^2$	$-100 \le x_i \le 100$
f_{rosen}	$\sum_{i=1}^{9} \left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$-2.048 \le x_i \le 2.048$
f_{step}	$\sum_{i=1}^{100} \left(\lfloor x_i+0.5 \rfloor\right)^2$	$-10 \le x_i \le 10$
f_{quart}	$\sum_{i=1}^{10} i x_i^4 + random[0,1)$	$-2.56 \le x_i \le 2.56$
$f_{sch2.21}$	$\underset{i}{max}\left\{\lvert x_i \rvert\right\}, 1 \le i \le 30$	$-100 \le x_i \le 100$
$f_{sch2.22}$	$\sum_{i=1}^{30} \lvert x_i \rvert + \prod_{i=1}^{30} \lvert x_i \rvert$	$-10 \le x_i \le 10$
f_{kowa}	$\sum_{i=1}^{11} \left[a_i - \frac{x_1\left(b_i^2+b_i x_2\right)}{b_i^2+b_i x_3+x_4}\right]^2$	$-5 \le x_i \le 5$
f_{hart}	$-\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j-p_{ij}\right)^2\right]$	$0 \le x_i \le 1$
f_{pen}	$0.1\left\{\begin{array}{l}\sin^2\left(\pi 3 x_1\right) \\ +\sum_{i=1}^{29}\left(x_i-1\right)^2 \cdot\left[1+\sin^2\left(3\pi x_{i+1}\right)\right] \\ +\left(x_{30}-1\right)^2\left[1+\sin^2\left(2\pi x_{30}\right)\right]\end{array}\right\}+\sum_{i=1}^{30} u\left(x_i,5,100,4\right)$	$-50 \le x_i \le 50$
f_{rastri}	$\sum_{i=1}^{30}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	$-50 \le x_i \le 50$
f_{grie}	$\frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30}\cos\left(\frac{x_k}{\sqrt{i}}\right)+1$	$-600 \le x_i \le 600$
f_{ack}	$-20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right)-\exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos 2\pi x_i\right)+20+e$	$-32 \le x_i \le 32$
f_{sphere_sft}	$\sum_{i=1}^{30} x_i^2$	$-100 \le x_i \le 100$
$f_{sch1.2\_sft}$	$f(\mathbf{x})=\sum_{k=1}^{30}\sum_{i=1}^{k} x_i^2$	$-100 \le x_i \le 100$
f_{elli_sft}	$\sum_{i=1}^{10}\left(A^{\left(\frac{i-1}{9}\right)}\left(x_i^2\right)\right)$, where $A=1\times10^6$	$-100 \le x_i \le 100$
f_{rosen_sft}	$\sum_{i=1}^{30}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$-100 \le x_i \le 100$
f_{rast_sft}	$\sum_{i=1}^{30}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	$-5 \le x_i \le 5$

TABLE I
BENCHMARK TEST FUNCTIONS.

A. Experimental setup

For comparison purpose, the performance of HPSOWM [26], HPSOM [4], HGAPSO [11], IPSO [25], [27], and the proposed FPSOCM on solving the benchmark test functions is evaluated.

The following simulation conditions are used:

For All PSOs:

- Swarm size (γ): 50
- Number of runs: 50
- Acceleration constant φ_1: 2.05 [30]
- Acceleration constant φ_2: 2.05 [30]
- Maximum velocity v_{max}: 0.2 [26]
- Initial population: it is generated uniformly at random

For FPSOCM:

- Probability of cross-mutated operation (p_{cm}):
  p_{cm} = 0.001 for f_{sphere}, $f_{sch2.22}$, f_{pen}, f_{grie}, f_{ack}, f_{sphere_sft};
  p_{cm} = 0.005 for f_{rastri}, $f_{sch1.2\_sft}$, f_{elli_sft};
  p_{cm} = 0.01 for f_{rosen}, f_{step}, f_{quart}, $f_{sch2.21}$, f_{kowa}, f_{rosen_sft}, f_{rast_sft};
  p_{cm} = 0.1 for f_{hart};

For HPSOWM, HPSOM, and HGAPSO:

- Probability of mutation operation (p_m): it is chosen by trial and error through experiments for good performance for all functions. p_m= 0.2 for all unimodal functions, multimodal functions with many local optima, and functions with shift and rotate except p_m= 0.1 for f_{sphere}; p_m= 0.3 for f_{kowa}; p_m= 0.5 for f_{hart}

For HPSOWM:

- Shape parameter of the wavelet mutation: 2 (it is chosen by the trial and error through experiments for good performance for all functions)
- Parameter g of the wavelet mutation: 10000

For HGAPSO:

- Probability of crossover operation: 0.8

B. Results and analysis

In this section, the experimental results in terms of mean and best cost value, standard deviation and computational time for the 17 benchmark test functions are given to show the performance of FPSOCM. The experimental results are summarized in Tables II to V. A statistical evaluation (p-test) is given in Table VI.

1) Unimodal functions: Unimodal functions are symmetric with a single minimum. Six unimodal functions are used to test the searching algorithms. The optimal value of all the functions is equal to zero. The simulation results of unimodal function are shown in Table II. The function f_{sphere} depicts a sphere model, which is smooth and symmetric. For this function, the mean value, best value and standard deviation offered by FPSOCM are better than those of the other PSO methods. The mean value of FPSOCM is 0.00002882×10^{-6} (very near the optimal value), which is about 530 times better than that of HPSOWM (the second-best method). In addition, the standard deviation of FPSOCM is the smallest, meaning that the searched solution is the most reliable.

The function f_{rosen} is the generalized Rosenbrock's function, which is a non-separable function and the optimum is located in a very narrow ridge. The tip of the ridge is very sharp, and it runs around a parabola. From the table, FPSOCM gives the best mean cost value and the smallest standard deviation, which implies that the solution quality and reliability are improved. The function f_{step} is a step function which is a representation of flat surfaces. Flat surfaces are obstacles for optimization algorithms because they do not give any information about the search direction. Unless the algorithm has a variable step size, it can get stuck in one of the flat surfaces. All hybrid PSOs with the mutation operation and FPSOCM are good to tackle this function because it can generate a long jump by using the mutation or CM operation.

The function f_{quart} is a quadratic function padded with noise, which increases the difficulty for searching the minimum value because the function would not return the same value at the same point every time. The mean solution and reliability offered by FPSOCM are the best. The function $f_{sch2.21}$ is the Schwefel's problem 2.21. FPSOCM gives a better performance in terms of mean value, best value, and

standard deviation. The last unimodal function is the Schwefel's problem 2.22 $f_{sch2.22}$. FPSOCM indicates the good performance of the algorithm. In general, FPSOCM is the best to tackle unimodal functions when compared with the other methods. In terms of computational time, FPSOCM consumes almost the same amount of time as the other methods.

		FPSOCM	HPSOWM	HPSOM	HGAPSO	IPSO
	Mean	0.00002882	0.01534812	2.73156256	108348.08	3901000×10^{3}
f_{sphere}	Best	0.00000273	0.00399741	0.72431698	1211.2577	2359.8337
$(\times10^{-6})$	Std Dev	0.00003108	0.00969838	1.49258407	180205.71	3751042×10^{3}
$T=1000$	Time (s)	9.140	8.906	8.703	8.706	8.436
	Rank	1	2	3	4	5
	Mean	0.60950108	1.07276495	1.45500489	3.91451826	3.40327459
f_{rosen}	Best	0.08869805	0.66293636	0.77316192	3.08690784	2.45991577
$(\times10^{0})$	Std Dev	0.18512482	0.27233685	0.25291157	0.46018910	0.45763080
$T=1000$	Time (s)	8.070	7.906	7.750	7.875	7.657
	Rank	1	2	3	5	4
	Mean	0.1600	0.8400	3.2000	5.0200	32.0799
f_{step}	Best	0.0000	0.0000	0.0000	0.0000	13.0000
$(\times10^{0})$	Std Dev	0.4218	0.9116	2.6186	11.7603	24.1313
$T=500$	Time (s)	6.560	6.600	6.593	6.640	6.547
	Rank	1	2	3	4	5
	Mean	0.00435660	0.00512595	0.00784463	0.00549581	0.00706537
f_{quart}	Best	0.00215990	0.00178256	0.00240483	0.00293377	0.00275818
$(\times10^{0})$	Std Dev	0.00084899	0.00168712	0.00216227	0.00134085	0.00229472
$T=1000$	Time (s)	9.344	8.984	8.875	9.125	8.844
	Rank	1	2	5	3	4
	Mean	0.07381342	0.25872908	0.26844525	1.96894358	2.67772016
$f_{sch2.21}$	Best	0.02179127	0.10570235	0.05043046	1.02841405	1.53375349
$(\times10^{0})$	Std Dev	0.03575051	0.10697120	0.11293117	0.44177387	0.61486690
$T=1000$	Time (s)	8.469	8.484	8.187	8.266	8.156
	Rank	1	2	3	4	5
	Mean	0.00000192	0.00008720	0.00091393	19.5603257	18.1898148
$f_{sch2.22}$	Best	0.00000017	0.00002238	0.00027451	2.06753646	0.77553015
$(\times10^{-2})$	Std Dev	0.00000248	0.00004404	0.00053605	16.7876183	18.3568946
$T=1000$	Time (s)	8.759	8.859	8.735	8.797	8.656
	Rank	1	2	3	5	4
Overall Rank		**1**	**2**	**3**	**4**	**5**

TABLE II
COMPARISON BETWEEN DIFFERENT PSO METHODS FOR BENCHMARK TEST FUNCTIONS (UNIMODAL FUNCTIONS). ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS. (RANK: 1-BEST, 5-WORST)

2) Multimodal functions with a few local minima: There are two multimodal functions with a few local minima used to do the testing. The results are tabulated in Table III . The function f_{kowa} is the Kowalik's function and the optimal value of this function is around 0.03075×10^{-2}. For this function, we obtain statistically different results from the proposed FPSOCM and other PSO methods. Although the standard deviation of FPSOCM is slightly larger than that of HPSOWM, it is better than that of other PSO methods. Furthermore, FPSOCM performs better in terms of mean and best cost values. Another test function is the Hartman's family II function (f_{hart}), of which the optimal value is around -3.32. From the results, no statistically significant difference among the PSO methods is seen. They all can reach or get near to the optimal value. In general, FPSOCM is the best to tackle multimodal function with a few local minima when compared with the other methods.

3) Multimodal functions with many local minima: There are four multimodal functions with many local minima to be put to the test; the dimension of each function is larger than that of f_{kowa} and f_{hart} in Section III.B2. The results are tabulated in Table IV, where f_{pen}, f_{rastri}, f_{grie}, and f_{ack} are the Generalized penalized function, the Generalized Rastrigin function, the Generalized Griewank's function and the Ackley's function respectively. The optimal value of these functions are all zero. Table IV shows that if the PSO method does not involve any cross-mutated operation or mutation

		FPSOCM	HPSOWM	HPSOM	HGAPSO	IPSO
	Mean	0.07911638	0.10829255	0.43140062	0.30502802	0.14730902
f_{kowa}	Best	0.03074859	0.05844088	0.06485530	0.04208881	0.04040836
$(\times10^{-2})$	Std Dev	0.28391398	0.21739258	0.98862379	0.64609080	0.39021836
$T=500$	Time (s)	3.922	4.188	3.766	3.812	3.733
	Rank	1	2	5	4	3
	Mean	−3.28870509	−3.28394937	−3.25779282	−3.27443792	−3.28186516
f_{hart}	Best	−3.32199517	−3.32199517	−3.32199516	−3.32199517	−3.32199517
$(\times10^{0})$	Std Dev	0.05392485	0.05602387	0.05985758	0.05883683	0.05773915
$T=100$	Time (s)	0.859	0.890	0.829	0.844	0.828
	Rank	1	2	5	4	3
Overall Rank		**1**	**2**	**5**	**4**	**3**

TABLE III
COMPARISON BETWEEN DIFFERENT PSO METHODS FOR BENCHMARK TEST FUNCTIONS (MULTIMODAL FUNCTION WITH A FEW LOCAL MINIMA). ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS. (RANK: 1-BEST, 5-WORST)

operation (FPSOCM, HPSOWM and HPSOM), it easily taps at some local minimum. From the results, the mean cost value, best cost value, and standard deviation of FPSOCM are better than those of the other methods. FPSOCM is able to provide more reliable and high-quality solutions. In general, FPSOCM is the best to tackle multimodal functions with many local minima when compared with the other methods.

		FPSOCM	HPSOWM	HPSOM	HGAPSO	IPSO
	Mean	0.17272×10^{-15}	0.84681×10^{-9}	0.97485×10^{-7}	0.057175	739.9839
f_{pen}	Best	0.00099×10^{-15}	0.06136×10^{-9}	0.32841×10^{-7}	0.000703	0.002429
$(\times10^{0})$	Std Dev	0.33395×10^{-15}	0.58306×10^{-9}	0.89291×10^{-7}	0.065261	2242.842
$T=1000$	Time (s)	9.312	9.344	8.954	9.015	8.875
	Rank	1	2	3	4	5
	Mean	8.26120642	10.2854056	16.5168697	19.3510055	20.7363037
f_{rastri}	Best	3.98014140	3.98283334	5.96984825	10.1299218	12.6419625
$(\times10^{0})$	Std Dev	2.78186223	3.35215436	5.83979208	5.54214275	5.38951031
$T=1000$	Time (s)	7.641	7.734	7.234	7.422	7.204
	Rank	1	2	3	4	5
	Mean	0.00002232	0.10521360	2.83079844	462481.940	4404184.04
f_{grie}	Best	0.00000034	0.02530853	0.38603262	15624.9889	826.156570
$(\times10^{-8})$	Std Dev	0.00002843	0.08540789	3.17252049	482276.158	10251732.3
$T=1000$	Time (s)	7.875	7.968	7.662	7.656	7.609
	Rank	1	2	3	4	5
	Mean	0.00090173	10.6067650	138.680594	228843.746	498939.484
f_{ack}	Best	0.00007619	4.91853722	66.5416850	10222.0192	1536.83379
$(\times10^{-6})$	Std Dev	0.00069501	3.12032110	38.2603380	390456.528	627243.145
$T=1500$	Time (s)	13.703	13.656	13.062	13.328	12.766
	Rank	1	2	3	4	5
Overall Rank		**1**	**2**	**3**	**4**	**5**

TABLE IV
COMPARISON BETWEEN DIFFERENT PSO METHODS FOR BENCHMARK TEST FUNCTIONS (MULTIMODAL FUNCTIONS WITH A MANY LOCAL MINIMA). ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS. (RANK: 1-BEST, 5-WORST)

4) Functions with shift or rotate: Some benchmark test functions [31]–[33] have drawbacks [32]. For instance, all variables might have the same numerical value at the global optimum for different dimensions owing to their symmetry. Typically, the global optimum is at the origin lying in the center of the search domain. Some algorithms simply converge to the center of the search domain that happens to be the global optimum. Hence, these benchmark test functions might not be good enough to test an optimization algorithm. To avoid these drawbacks, five more benchmark test functions [32], [34] with shift or rotate are used. In these functions, the variables are made to have different numerical values at the global optimum point, which is not lying in the center of the search domain. The details of these functions are introduced in [32], [34]. The five functions with shift or rotate are: the Shifted sphere function (f_{sphere_sft}), the Shifted Schwefel's problem 1.2 ($f_{sch1.2\_sft}$), the Shifted rotated high conditioned elliptic function (f_{elli_sft}), the Shifted Rosenbrock's function (f_{rosen_sft}) and the Shifted Rastrigin's function (f_{rast_sft}).

The first four functions are unimodal functions and the last is multimodal functions. The experimental results are shown in Table V. From the table, we can see that the proposed FPSOCM, HPSOWM, HPSOM and HGAPSO show better performance than IPSO. When the PSO has no cross-mutated or mutation operation, it is hard to solve the optimization problems with the global optimum points shifted and rotated, no matter the problem is unimodal or multimodal. Comparing FPSOCM, HPSOWM, HPSOM and HGAPSO, the performance of FPSOCM is the best in tackling functions with shift or rotate.

5) P-test: The p-test is a statistical method to evaluate the significant difference between two algorithms. When the p-value is less than the significance level ($p < 0.05$), the result is said to be statistically significant and the performance is significantly better than the other methods with a 95% confidence level. The p-values between FPSOCM and the other optimization methods are shown in Table VI. We see that around 86% p-values in this table are less than 0.05. 72% p-value are less than 0.0001 which means the difference between two methods is considered to be statically very significant. One reason that some p-values being higher than 0.05 is that all the hybrid PSO methods can reach or nearly reach the optimal point of the function, for example, f_{hart}. There is 92% of p-values in this table being less than 0.05 when f_{hart} is not included to evaluate the algorithms. In general, the performance of FPSOCM is significantly better than the other PSO methods with a 95% confidence level ($p < 0.05$).

		FPSOCM	HPSOWM	HPSOM	HGAPSO	IPSO
	Mean	0.00001038	0.00688587	1.23955015	100.531580	3843.10×10^{9}
f_{sphere_sft}	Best	0.00000006	0.00049868	0.10838914	12.8393453	130.040×10^{9}
$(\times10^{-9})$	Std Dev	0.00002508	0.00650427	1.40738585	180.55471	2970.66×10^{9}
$T = 1000$	Time (s)	24.460	24.188	23.991	24.141	23.857
	Rank	1	2	3	4	5
	Mean	0.19941990	2.28726897	3.98412058	4.80494487	225.465158
$f_{sch1.2\_sft}$	Best	0.02864735	0.06011202	0.53766698	1.09813575	225.465158
$(\times10^{0})$	Std Dev	0.12470094	1.38115143	3.94147119	2.53962887	110.157119
$T = 1500$	Time (s)	43.520	44.047	43.985	44.297	43.189
	Rank	1	2	3	4	5
	Mean	0.99057319	1.24225443	1.2902117	1.30362559	2.65637576
f_{elli_sft}	Best	0.01313945	0.03160666	0.10001893	0.15561725	0.09915978
$(\times10^{5})$	Std Dev	0.91490110	1.08794832	1.15561726	0.83779842	4.30836229
$T = 2000$	Time (s)	80.875	79.578	79.343	80.983	79.031
	Rank	1	2	3	4	5
	Mean	0.16560902	0.41886823	1.27705404	1.78652083	452295.412
f_{rosen_sft}	Best	0.00031570	0.01480565	0.00721032	0.01287281	15.0019863
$(\times10^{3})$	Std Dev	0.32912649	0.53806929	3.48283223	4.11598323	605375.474
$T = 500$	Time (s)	12.672	12.766	12.439	12.578	12.462
	Rank	1	2	3	4	5
	Mean	55.89766296	110.844452	102.598313	136.327427	122.281957
f_{rast_sft}	Best	14.3241485	9.22564214	23.125384	16.8775564	67.5286722
$(\times10^{0})$	Std Dev	40.7798978	51.4620399	55.2994743	54.8496199	34.7743748
$T = 1000$	Time (s)	40.045	38.811	38.719	38.813	38.546
	Rank	1	3	2	5	4
Overall Rank		**1**	**2**	**3**	**4**	**5**

TABLE V
COMPARISON BETWEEN DIFFERENT PSO METHODS FOR BENCHMARK TEST FUNCTIONS (FUNCTIONS WITH SHIFT AND ROTATE). ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS. (RANK: 1-BEST, 5-WORST)

Function	p-value between FPSOCM and HPSOWM	p-value between FPSOCM and HPSOM	p-value between FPSOCM and HGAPSO	p-value between FPSOCM and IPSO
f_{sphere}	<0.0001	<0.0001	<0.0001	<0.0001
f_{rosen}	<0.0001	<0.0001	<0.0001	<0.0001
f_{step}	<0.0001	<0.0001	<0.0001	<0.0001
f_{quart}	0.0059	<0.0001	<0.0001	<0.0001
$f_{sch2.21}$	<0.0001	<0.0001	<0.0001	<0.0001
$f_{sch2.22}$	<0.0001	<0.0001	<0.0001	<0.0001
f_{kowa}	0.5646	0.0193	0.0283	0.3222
f_{hart}	0.6691	0.0092	0.2136	0.5447
f_{pen}	<0.0001	<0.0001	<0.0001	0.0240
f_{rastri}	0.0019	<0.0001	<0.0001	<0.0001
f_{grie}	<0.0001	<0.0001	<0.0001	0.0038
f_{ack}	<0.0001	<0.0001	0.0001	<0.0001
f_{sphere_sft}	<0.0001	<0.0001	<0.0001	<0.0001
$f_{sch1.2\_sft}$	<0.0001	<0.0001	<0.0001	<0.0001
f_{elli_sft}	0.2172	0.1562	0.0813	0.0103
f_{rosen_sft}	0.0066	0.0290	0.0077	<0.0001
f_{rast_sft}	<0.0001	<0.0001	<0.0001	<0.0001

TABLE VI
P-VALUE BETWEEN FPSOCM AND THE OTHER PSO METHODS.

C. Comparison between fuzzy inertia weight and linear inertia weight

In this section, we give an analysis based on experimental results to illustrate the improvement brought by the fuzzy inertia weight $\tilde{\omega}$. The experimental settings are the same as before, except the probability of cross-mutated operation p_{cm} is set at 0. By using this setting, no particle will undergo the CM operation. Hence, the performance of the fuzzy inertia weight can be evaluated. For comparison purpose, IPSO uses a linear inertia weight ω which is employed without using any hybrid operation. The experimental results (mean cost values) on using the fuzzy inertia weight and the linear inertia weight for all 17 benchmark test functions are summarized in Table VII. We can see that except $f_{sch2.21}$, all functions using the fuzzy inertia weight give better mean cost values. In conclusion, the searching performance of PSO with the fuzzy inertia weight is improved. We can see also the effectiveness of the CM operation. Comparing Table II to V (mean values), the performance of fuzzy PSO with CM operation is better than that without CM operation.

Function	fuzzy inertia weight	Linear inertia weight
f_{sphere}	2200.00337	3901.00079
f_{rosen}	0.9091225	3.40327459
f_{step}	29.9152	32.0799
f_{quart}	0.00635508	0.00706537
$f_{sch2.21}$	2.71231265	2.67772016
$f_{sch2.22}$	0.12485042	0.18189815
f_{kowa}	0.00051884	0.00147309
f_{hart}	−3.28201027	−3.28186516
f_{pen}	580.2509	739.9839
f_{rastri}	18.8930033	20.7363037
f_{grie}	0.00221422	0.04404187
f_{ack}	0.30051016	0.49893948
f_{sphere_sft}	2861.191	3843.100
$f_{sch1.2\_sft}$	192.317624	225.465158
f_{elli_sft}	1.443740×10^{5}	2.656375×10^{5}
f_{rosen_sft}	372857×10^{3}	452295×10^{3}
f_{rast_sft}	116.416818	122.281957

TABLE VII
COMPARISON OF MEAN COST VALUES BETWEEN PSO WITH FUZZY INERTIA WEIGHT AND PSO WITH LINEAR INERTIA WEIGHT. ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS.

D. Sensitivity of the parameter p_{cm}

FPSOCM seeks a balance between the exploration of new regions and the exploitation of the already sampled regions

in the search space. This balance, which critically affects the performance of FPSOCM, is governed by the right choice of the control parameter value: the probability of CM operation (p_{cm}). Increasing the probability of the CM operation (p_{cm}) tends to transform the search into a random search such that when p_{cm} = 1, all elements of particles in the swarm will mutate under the CM operation. This probability gives us an expected number ($p_{cm} \times \gamma \times \kappa$) of elements of particles that undergo the CM operation. In other words, the value of p_{cm} depends on the desired number of element of particles that undergo the CM operation, which is application dependent.

Normally, when the dimension of the search space is very low (say, the number of elements in a particle is less than 5), p_{cm} can be set at 0.1 to 0.2 (10-20% elements of particles will undergo the CM operation). When the dimension is around 5-10, p_{cm} can be set at 0.01 to 0.05 (1-5% elements of particles will undergo the CM operation). When the dimension is in the range of 11 to 100, p_{cm} can be set at 0.001 to 0.005 (0.1-0.5% elements of particles will undergo the CM operation). Lastly, when the dimension is in the range of 101 to 1000, p_{cm} can be set at 0.0001 to 0.0005 (0.01-0.05% elements of particles will undergo the CM operation). In principle, when the dimension is high, p_{cm} should be set to a small value. It is because if the dimension is high and p_{cm} is set to a larger value, the absolute number of elements of particles undergoing the CM operation will be large. It will increase the searching time and more importantly destroy the current information about the application in each time of iteration, as the elements of particles are randomly assigned.

Generally, by properly choosing the value of p_{cm}, the appropriate ratio of the number of elements of particles undergoing the CM operation to the swarm size can be maintained to prevent the searching process from turning to a random searching one. The choices of the values of p_{cm} for all the above benchmark functions and the following numerical examples in this paper are based on this selection criterion, and set by trial and error through experiments for good performance.

The mean cost values offered by FPSOCM with different values of p_{cm} for some benchmark test functions are tabulated in Table VIII. The functions are tested by using $p_{cm} = 0.05$, 0.01, 0.005, 0.001 and 0.0005. Results show that the value of the parameter p_{cm} is sensitive to the performance of the searching process. The dimension of the functions $f_{sch2.22}$, f_{pen}, f_{ack}, and $f_{sch1.2\_sft}$ is in the range of 11-100, we can see that the best mean cost value of these functions are given when p_{cm} is in the range of 0.001 to 0.005. It meets the selection criterion. On the other hand, the dimension of the functions f_{rosen} and f_{elli_sft} is in the range of 5-10, we can see that the best mean cost values of these two functions are given when p_{cm} is set at 0.01, meeting the selection criterion. It should be noted that no formal method is available to choose the value of the p_{cm}; its value depends on the characteristics of the optimization problem. We can see that the dimension of the function f_{rast_sft} is 30 and the best mean value is given when $p_{cm} = 0.01$; the selection criterion is not met. Thus, the selection criterion above just gives an idea to choose the range of the p_{cm} value for general optimization problems.

Function	$p_{cm} = 0.05$	$p_{cm} = 0.01$	$p_{cm} = 0.005$	$p_{cm} = 0.001$	$p_{cm} = 0.0005$
f_{rosen}	1.2879	0.6095	0.9638	2.4041	3.2133
$f_{sch2.22}$	0.4085×10^{-2}	0.0049×10^{-6}	0.155×10^{-6}	0.0192×10^{-6}	0.1756×10^{-2}
f_{pen}	0.2203	3.0092×10^{-6}	0.9167×10^{-9}	0.1727×10^{-15}	6.5220×10^{-15}
f_{ack}	0.4269	0.9920×10^{-3}	8.4530×10^{-6}	0.9017×10^{-9}	1.2786×10^{-9}
$f_{sch1.2\_sft}$	1.0236	0.8001	0.1994	3.2658	106.8590
f_{elli_sft}	1.4980×10^{5}	0.9906×10^{5}	1.2307×10^{5}	1.4598×10^{5}	1.6824×10^{5}
f_{rast_sft}	59.6823	55.8977	64.0543	83.4474	75.8785

TABLE VIII
SENSITIVITY OF p_{cm}.

IV. REAL WORLD APPLICATIONS

In this section, two real world applications namely economic load dispatch and email network services, are used to illustrate the performance and the applicability of the proposed FPSOCM.

A. Application I: Economic Load Dispatch

In this section, an industrial application example on economic load dispatch (ELD) was used to shown in the performance of different PSOs. The ELD aims to schedule power generator outputs with respect to the load demands, and to operate a power system economically [35]. Power generated for the ELD are nonlinear due to the valve-point loadings and rate limits. Hence, the problem of ELD is multimodal, discontinuous and highly nonlinear, and PSO has commonly been used to solve ELD problem.

1) Mathematical model for ELD: The economic load dispatch with a valve-point loading problem concerns the minimization of the following objective function:

$$Min \sum_{i=1}^{n} C_i(P_{L_i}) \tag{16}$$

where $C_i(P_{L_i})$ is the operation fuel cost of generator i, and n denotes the number of generators. The problem is subject to balance constraints and generating capacity constraints as follows:

$$D = \sum_{i=1}^{n} P_{L_i} - P_{Loss} \tag{17}$$

$$P_{L_i,min} \leq P_{L_i} \leq P_{L_i,max} \tag{18}$$

where D is the load demand, P_{L_i} is the output power of the i-th generator, P_{Loss} is the transmission loss, $P_{L_i,max}$ and $P_{L_i,min}$ are the maximum and minimum output powers of the i-th generator respectively. The operation fuel cost function with valve-point loadings of the generators is given by,

$$C_i(P_{L_i}) = a_i P_{L_i}^2 + b_i P_{L_i} + c_i + \|e_i \times \sin(f_i \times (P_{L_i,min} - P_{L_i}))\| \tag{19}$$

where a_i ,b_i, and c_i are coefficients of the cost curve of the i-th generator, e_i and f_i are coefficients of the valve-point loadings. (The generating units with multivalve steam turbines exhibit a greater variation in the fuel-cost functions. The valve-point effects introduce ripples in the heat-rate curves.)

2) PSO for the ELD: To solve the ELD problem by PSO, the particle is defined as follows:

$$\mathbf{x} = [P_{L_1} P_{L_2} P_{L_3} ... P_{L_{n-1}}], \tag{20}$$

where n denotes the number of generators. From (17), we have,

$$P_{L_n} = D - \sum_{i=1}^{n-1} P_{L_i} + P_{Loss} \tag{21}$$

In this example, the power loss is not considered. Therefore,

$$P_{L_n} = D - \sum_{i=1}^{n-1} P_{L_i} \tag{22}$$

To ensure P_{L_n} falls within the range $[P_{L_i,min}\ P_{L_i,max}]$, the following conditions are considered:

$$\text{if } P_{L_n} > P_{L_n,max} \begin{cases} P_{L_1} = P_{L_1} + (P_{L_n} - P_{L_n,max}) \\ P_{L_n} = P_{L_n,max} \end{cases} \tag{23}$$

$$\text{if } P_{L_n} < P_{L_n,max} \begin{cases} P_{L_1} = P_{L_1} - (P_{L_n,min} - P_{L_n}) \\ P_{L_n} = P_{L_n,min} \end{cases} \tag{24}$$

It should be noted from (23) and (24) that if the value of P_{L_1} is also outside the constraint boundary. The exceeding portion of the power will be shared by other generators in order to make sure that all generators' output power is within the safety range.

3) Result and analysis: The PSO methods were applied to a 40-generator system, which was adopted as an example in [35]. The system is a very large one with nonlinearities. The load demand (D) is 10500MW. For comparison, FPSOCM, HPSOWM, HPSOM, HGAPSO, and IPSO were used to minimize the operation fuel cost in (19). The set values of the parameters of the PSO methods were basically the same as those in Section III(A). The number of iteration is 1000. The probability of mutation operation for HPSOWM, HPSOM and HGAPSO was set at 0.1. The statistical results are tabulated in Table IX, and the comparison of the convergence rate between different PSO methods is shown in Fig. 2. As can be seen from the table and figure, the mean and the best values offered by FPSOCM are the best. In addition, the convergence rate of FPSOCM is the highest. Furthermore, the smaller standard deviation implies a more reliable solution. Lastly, all p-values are less than 0.0001 which means that the performance of FPSOCM is statistically significant when compared with the other PSO methods. For example, FPSOCM gives a cost value that is statistically significant as compared with second best PSO method, HPSOWM (121790.16±77.37 vs 122608.60±524.60 ($p < 0.0001$)).

	FPSOCM	HPSOWM	HPSOM	HGAPSO	IPSO
Mean	121790.16	122608.60	123267.46	123542.48	124984.88
Best	121633.62	121697.24	121846.52	121764.34	122740.25
Std Dev	77.37	524.60	903.30	826.55	1043.33
p-value	–	< 0.0001	< 0.0001	< 0.0001	< 0.0001

TABLE IX
COMPARISON BETWEEN DIFFERENT PSO METHODS FOR ELD. ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS (p-VALUE ILLUSTRATE THE SIGNIFICANT DIFFERENCE BETWEEN FPSOCM AND THE OTHER METHODS).

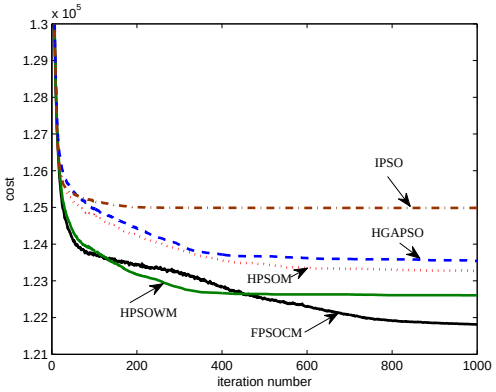


Fig. 2. Comparison of the convergence rate between different PSO methods.

B. Application II: self-provisioning of communication network services

1) Background: In this section, a telecommunication problem on self-provisioning of communication email services is used to further evaluate the performance of the FPSOCM. A four-layer networking structure is inspired from the information framework of Telecommunication Management Forum (TMF) [36]. A graph theory-based model is created to guide the email-box service configuration process. The managed network elements (NEs) are categorized into classes with a number of instantiated objects in the multiple layers, which are 1) email product layer (denoted as P below); 2) email component layer (denoted as C below); 3) email services layer (denoted as S below); and 4) email resources layer (denoted as R below). The problem is to make the service activation process become seamlessly self-sustained and as a result fewer human operators' interventions are required.

Essentially, from the operators and end-user's points of view, networks are designed and implemented for providing usable and configurable services [37]. Reducing the costs within the provisioning process is a major concern in the network management area. It is of vital importance to consume less cost to activate and configure services with necessary NEs. Provisioning of services requires a number of resources and processes to be coordinated. A number of sub-costs are therefore considered into the process. These costs are determined by many aspects, such as NEs' interdependencies, servers capacity, link utilization status, bandwidth, router status, CPU/Memory usage, and so on. These work together and exhibit a multimodal, nonlinear network behavior of high complexity with a large number of parameters to be considered, optimized and balanced.

2) Mathematical model for the self-provisioning of communication email network services: The cost-based email service configuration problem can be formulated with the following objective function:

$$\min Z = \sum_{l=1}^{\tilde{n}_r} \sum_{k=1}^{\tilde{n}_s} \sum_{i \in N_p, j=1}^{\tilde{n}_c} \begin{bmatrix} m_{ij} f_{\mathbf{P}_i \mathbf{C}_j}(\cdot) \\ +n_{j,k} f_{\mathbf{C}_j \mathbf{S}_k}(\cdot) \\ +o_{k,l} f_{\mathbf{S}_k \mathbf{R}_l}(\cdot) \end{bmatrix} \tag{25}$$

and subject to the following constraints:

$a) \sum_{i=1}^{\tilde{n}_p} f_{\mathbf{P}_i\mathbf{C}_j}(\cdot) \leq \sum_{j=1}^{\tilde{n}_c} \bar{\omega}_{n_j}$;

$b) \sum_{j=1}^{\tilde{n}_c} f_{\mathbf{C}_j\mathbf{S}_k}(\cdot) \leq \sum_{k=1}^{\tilde{n}_s} \bar{\omega}_{n_k}$;

$c) \sum_{k=1}^{\tilde{n}_s} f_{\mathbf{S}_k\mathbf{R}_l}(\cdot) \leq \sum_{l=1}^{\tilde{n}_r} \bar{\omega}_{n_l}$

where Z is the overall configuration cost across the four layer networking structure for NEs i, j, k and l; $\tilde{n}_r$, $\tilde{n}_s$, and $\tilde{n}_c$ denote the number of the corresponding NEs; $f_{\mathbf{P}_i\mathbf{C}_j}(\cdot)$, $f_{\mathbf{C}_j\mathbf{S}_k}(\cdot)$, and $f_{\mathbf{S}_k\mathbf{R}_l}(\cdot)$ denote the corresponding *total costs*, which consist of a number of subcosts including the Base Cost (BC), Variable Cost (VC), and $\varphi(\omega_{n,i,k}(t), R_{n,i,k}(t), C_{n,i}(t), \lambda_{j,k}) \in \mathbb{R}^n$ which is the link cost for the components in the resource layer. CPU/Memory usage, bandwidth, capacity are all factors considered into the calculation of this cost. Specifically, $\omega_{n,i,k}(t), R_{n,i,k}(t), C_{n,i}(t), \lambda_{j,k}$ are referring to the following costs respectively: 1) Traffic Intensity Condition, 2) Node Capacity Level, 3) Link Capacity Level and 4) Delay Time [38]–[40].

This four layer operational structure has N_p email product instances for c_p classes of products; $\tilde{n}_c$ email component instances for c_c classes of product components; $\tilde{n}_s$ email service instances for c_s classes of services; and $\tilde{n}_r$ email resource instances for c_r classes of resources. The cost elements among instances between layers are depicted as $(\tilde{n}_p \times \tilde{n}_c), (\tilde{n}_c \times \tilde{n}_s), (\tilde{n}_s \times \tilde{n}_r)$.

$\mathbf{P}_i\mathbf{C}, \mathbf{C}_j\mathbf{S}_k$,and $\mathbf{S}_k\mathbf{R}_l$ describe the connections between the layers as described above. $m_{ij}, n_{j,k}, o_{k,l}$ are the weights for the cost across the layers P to C, layers C to S, and layers S to R respectively. They are used as a way to further adjust the overall cost within constrained communication links. The objective is to minimize this configuration cost, and therefore find a suitable configuration path with necessary NEs activated for services.

The PSO particle is defined as $\mathbf{X} = [M, N, O]$, where M, N, O are the vector representation for individual weights $m_{ij}, n_{j,k}, o_{k,l}$ respectively. As indicated in (25), the parameters $m_{ij}, n_{j,k}$,and $o_{k,l}$ are to be tuned by PSOs. The objective is to find the optimal parameters of $m_{ij}, n_{j,k}$, and $o_{k,l}$ to minimize the overall configuration cost Z.

3) Result and analysis: We applied FPSOCM to minimize the configuration cost value as given in (25). To simplify the experiment, we assume each layer has only 4 classes, and each class only has a maximum of four instantiated objects, i.e. $N_p = 4$. The following parameter values are used in this application: $\tilde{n}_r = 16, \tilde{n}_s = 16, \tilde{n}_c = 16$, $c_p = 1, c_c = 4, c_s = 4$, and $c_r = 4$. For comparison HPSOWM, HPSOM, HGAPSO, and IPSO are used to minimize the overall configuration cost given in (25). The set values of the parameters of the PSO methods are basically the same as those in Section III(A). The number of iteration used in all PSO methods is 500. The probability of mutation operation for HPSOWM, HPSOM and HGAPSO is set at 0.5. The statistical results are tabulated in Table X, and the comparison of the convergence rate between different PSO methods is shown in Fig. 3. They show that the mean and the best values offered by FPSOCM are the best. It gives the smallest overall configuration cost. Referring to Fig. 3, FPSOCM gives a faster convergence than the other methods. Furthermore, the p-values are less than 0.005 for HPSOM, HGAPSO, and IPSO which means that the result of FPSOCM is very statistically significant. To compare with HPSOWM, FPSOCM gives a cost value that is statistically significant (27 ± 1.56 vs 27.56 ± 1.45 ($p = 0.0718$)) with a 93% confidence level. Hence, FPSOCM is the best method among all the tested PSO methods for solving this telecommunication problem.

	FPSOCM	HPSOWM	HPSOM	HGAPSO	IPSO
Mean	27.00	27.56	28.00	28.44	28.24
Best	21	22	23	24	24
Std Dev	1.56	1.45	1.59	1.68	1.60
p-value	–	0.0718	0.0049	< 0.0001	0.0003

TABLE X
COMPARISON BETWEEN DIFFERENT PSO METHODS FOR THE SELF-PROVISIONING OF COMMUNICATION NETWORK SERVICES. ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS (p-VALUE ILLUSTRATES THE SIGNIFICANT DIFFERENCE BETWEEN FPSOCM AND THE OTHER METHODS).

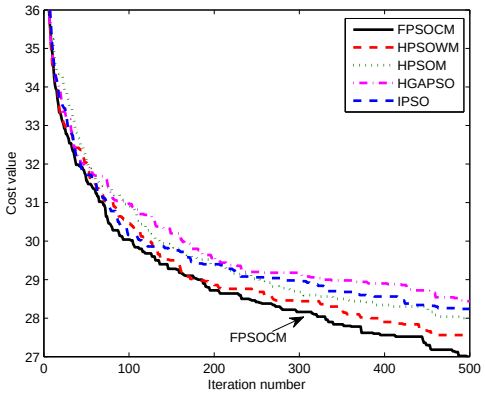


Fig. 3. Comparison of the convergence rate between different PSO methods.

V. CONCLUSION

In this paper, we have proposed a novel particle swarm optimization named FPSOCM that incorporates an adaptive inertia weight and a new cross-mutated operation. The value of the weight and the parameter for the cross-mutated operation are determined by a set of fuzzy rules. The new cross-mutated operation is used to force the particle escaping from the local optimum. With the fuzzy engine added, the solution quality obtained by the PSO is improved. On solving a suite of benchmark test functions, FPSOCM gives better results than the other recently developed PSO methods, including HPSOWM, HPSOM, HGAPSO, HGPSO and IPSO. Also, FPSOCM provides a faster convergence than all these PSO methods. On applying to real world applications, FPSOCM is found to be successful to 1) minimize the cost for economic load dispatch and 2) minimize the overall configuration cost for email network services. FPSOCM also outperforms the other PSO methods. Hence, better and more robust service quality can be generated using the proposed FPSOCM compared with the other tested PSO algorithms. These results also

indicate that the proposed FPSOCM is a good candidate on optimizing the quality and robustness on product or service engineering design which are generally multi-optima or non-convex natures. One limitation in this study is that choosing the suitable parameter values for the PSO is quite difficult. Most parameter values are determined by trial and error through experiments. The dynamic cross-mutated rate could be further studied to reduce the time for selecting a suitable rate for applications. In addition, an online fuzzy inference system can be developed to determine the fuzzy rules dynamically during learning.

ACKNOWLEDGMENT

The work described in this paper was partially supported by a grant from University of Technology Sydney (Activity code: 2032019), and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Account Code G-Y563)

REFERENCES

[1] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*. San Fransisco, USA: Morgan Kaufmann Publishers, 2001.

[2] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming* VII: proc. 7th ann. conf. on evolutionary conf., vol. 1447, pp. 611–616, 1998.

[3] S. K. Moon, K. J. Park, and T. W. Simpson, "Platform design variable identification for a product family using multi-objective particle swarm optimization," *Research in Engineering Design*, vol. 25, no. 2, pp. 95–108, Apr 2014.

[4] A. A. E. Ahmed, L. T. Germano, and Z. C.Antonio, "A hybrid particle swarm optimization applied to loss power minimization," *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 859–866, May 2005.

[5] Y. Fu, M. Ding, and C. Zhou, "Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV," *IEEE Trans. Syst. Man and Cybern. A*, vol. 42, no. 2, pp. 511–526, 2012.

[6] Y. Fu, M. Ding, and C. Zhou, "Multi-objective optimal SVC installation for power system loading margin improvement," *IEEE Trans. Power Syst.*, vol. 27, no. 2, pp. 984–992, 2012.

[7] S. H. Ling, H. T. Nguyen, K. Y. Chan, F. H. F. Leung, and F. Jiang, "Intelligent fuzzy particle swarm optimization with cross-mutated operation," in *Proc. IEEE Congress on Evolutionary Computation (CEC 2012), 2012 IEEE World Congress on Computational Intelligence (WCCI 2012)*, (Australia), pp. 3009–3016, 2012.

[8] Y. Tang, H. Gao, J. Kurths, and J. Fang, "Evolutionary pinning control and its application in uav coordination," *IEEE Trans. Industrial Informatics*, vol. 8, no. 4, pp. 828–839, 2012.

[9] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning," *IEEE Trans. Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[10] A. Chatterjee, K. Pulasinghe, K. Watanabe, and K. Izumi, "A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems," *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1478–1489, Dec, 2005.

[11] C. F. Juang, "A hybrid genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst. Man and Cybern. B*, vol. 34, no. 2, pp. 997–1006, 2004.

[12] K. Y. Chan, C. K. Yiu, T. S. Dillon, S. Nordholm, and S. H. Ling, "Enhancement of speech recognitions for control automation using an intelligent particle swarm optimization," *IEEE Trans. Industrial Informatics*, vol. 8, no. 4, pp. 869–879, 2012.

[13] K. Ishaque and Z. Salam, "A deterministic particle swarm optimization maximum power point tracker for photovoltaic system under partial shading condition," *IEEE Trans. Industrial Electronics*, vol. 60, no. 8, pp. 3195–3206, 2013.

[14] K. Y. Chan, T. S. Dillon, and E. Chang, "An intelligent particle swarm optimization for short-term traffic flow forecasting using on-road sensor systems," *IEEE Trans. Industrial Electronics*, vol. 60, no. 10, pp. 4714–4725, 2013.

[15] P. P. San, S. H. Ling, and H. T. Nguyen, "Industrial application of evolvable block-based neural network to hypoglycemia monitoring system," *IEEE Trans. Industrial Electronics*, vol. 60, no. 12, pp. 5982–5901, 2013.

[16] P. P. San, S. H. Ling, N. Nuryani, and H. Nguyen, "Evolvable rough-block-based neural network and its biomedical application to hypoglycemia detection system," *IEEE Trans. on Cybernetics*, vol. 44, no. 8, pp. 1338–1349, 2014.

[17] S. H. Ling, H. H. C. Iu, F. H. F. Leung, and K. Y. Chan, "Improved hybrid PSO-based wavelet neural network for modelling the development of fluid dispensing for electronic packaging," *IEEE Trans. Ind. Electron.*, vol. 55, no. 9, pp. 3447–3460, Sep. 2008.

[18] K. Y. Chan, T. S. Dillon, and C. K. Kwong, "Modeling of a liquid epoxy molding process using a particle swarm optimization-based fuzzy regression approach," *IEEE Trans. Industrial Informatics*, vol. 7, no. 1, pp. 148–158, 2011.

[19] K. Y. Chan, T. S. Dillon, and C. K. Kwong, "Polynomial modeling for time-varying systems based on a particle swarm optimization algorithm," *Inf. Sci.*, vol. 181, no. 9, pp. 1623–1640, 2011.

[20] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Trans. Syst. Man and Cybern. B*, vol. 42, no. 3, pp. 627–646, 2012.

[21] L. D. S. Coelho, F. A. Guerra, and J. V. Leite, "Multiobjective exponential particle swarm optimization approach applied to hysteresis parameters estimation," *IEEE Trans. Magnetics*, vol. 48, no. 2, pp. 283–286, 2012.

[22] W. M. Lin, T. J. Su, and R. C. Wu, "Parameter identification of induction machine with a starting no-load low-voltage test," *IEEE Trans. Industrial Electronics*, vol. 59, no. 1, pp. 352–360, 2012.

[23] P. Angeline, "Using selection to improve particle swarm optimization," in *Proc. IEEE Congress on Evolutionary Computing*, (Anchorage), pp. 84–89, May 1998.

[24] M. M. Noel and T. C. Jannett, "Simulation of a new hybrid particle swarm optimization algorithm," in *Proc. 36th Southeastern Symposium on System Theory*, pp. 150–153, 1994.

[25] Y. Shi, "Empirical study of particle swarm optimization," in *Proc. of the 1999 Congress on Evolutionary Computation*, vol. 1945–1950, (Washington, US), 1999.

[26] S. H. Ling, H. H. C. Iu, K. Y. Chan, H. K. Lam, C. W. Yeung, and F. H. F. Leung, "Hybrid particle swarm optimization with wavelet mutation and its industrial applications," *IEEE Trans. Syst. Man and Cybern. B*, vol. 38, no. 3, pp. 743–763, 2008.

[27] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol Comput*, vol. 6, no. 1, pp. 58–73, 2002.

[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. 30th IEEE Conf. Decision and Control*, vol. 4, pp. 1942–1948, 1995.

[29] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. IEEE Congress on Evolutionary Computing*, vol. 1, pp. 84–88, Jul. 2000.

[30] N. Mo, Z. Y. Zou, K. W. Chan, and T. Y. G. Pong, "Transient stability constrained optimal power flow using particle swarm optimization," *IET Proceedings - Generation, Transmission and Distribution*, vol. 1, no. 3, pp. 476–483, May 2007.

[31] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*. USA: 3rd extended ed. Springer-Verlag, 1996.

[32] J. J. Lang, S. Baskar, P. Suganthan, and A. Qin, "Performance evaluation of multi agent genetic algorithm," *Natural Computing*, vol. 5, no. 1, pp. 83–96, 2006.

[33] X. Yao and Y. Liu, "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[34] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proc. IEEE Int. Swarm Intelligence Symposium*, pp. 68–75, 2005.

[35] P. H. Chen and H. C. Chang, "Large-scale economic dispatch by genetic algorithm," *IEEE Trans. Power Syst*, vol. 10, pp. 117–124, 2005.

[36] http://www.tmforum.org/browse.aspx, [Accessed on 03 Dec 2013].

[37] H. Nishiyama, H. Yamada, H. Yoshino, and N. Kato, "A cooperative user-system approach for optimizing performance in content distribution/delivery networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 2, pp. 476–483, 2012.

[38] J. Bühler and G. Wunder, "Traffic-aware optimization of heterogeneous access management," *IEEE Transactions on Communications*, vol. 58, no. 6, pp. 1737–1747, 2010.

[39] F. Chiang, R. Braun, and J. Hughes, "A hybrid genetic algorithm and particle swarm optimization for recurrent network design," *International*

Journal of Pervasive Computing and Communications, vol. 2, no. 3, pp. 261–276, 2006.

[40] H. M. Sigurdsson, S. E. Thorsteinsson, and T. K. Stidsen, "Cost optimization methods in the design of next generation networks," *IEEE Communications Magazine*, vol. 42, no. 9, pp. 118–122, 2004.