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 for 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 through out 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.

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 achieve 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 $x^i(t) \in X(t)$ contains $\kappa$ elements $x^i_j(t) \in \mathbb{X}^i(t)$, where $i = 1, 2, \ldots, \gamma$ and $j = 1, 2, \ldots, \kappa$; $\gamma$ denotes the number of particles in the swarm and $\kappa$ is the dimension of a particle. At the beginning, the swarm particles are initialized and then evaluated by a defined fitness function $f(x^i(t))$. 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($X(t)$)**

```
\begin{align*}
& t \leftarrow 0 \\
& \text{Initialize } X(t) \\
& \text{output } (f(X(t))) \\
& \text{while } \neg \text{termination condition} > \left\{
\begin{array}{l}
  t \leftarrow t + 1 \\
  \text{Update velocity } \mathbf{v}(t) \text{ based on (3) \text{-- (5)}}. \\
  \text{Ensure the updated velocity is in the region of } v_{\text{max}} \\
  \text{if } v(t) > v_{\text{max}} \\
  \quad \text{then } v(t) = v_{\text{max}} \\
  \text{if } v(t) < -v_{\text{max}} \\
  \quad \text{then } v(t) = -v_{\text{max}} \\
  \text{do} \\
  \quad \text{Generate a new swarm } X(t) \text{ based on (2).} \\
  \quad \text{Ensure the updated } x^i_j(t) \text{ is inside the boundary} \\
  \quad \text{if } x^i_j(t) > p_{\text{max}} \\
  \quad \quad \text{then } x^i_j(t) = p_{\text{max}} \\
  \quad \text{if } x^i_j(t) < p_{\text{min}} \\
  \quad \quad \text{then } x^i_j(t) = p_{\text{min}} \\
  \quad \text{return } (g) \\
  \text{comment: } g \text{ is the best particle among all particles (solution)} \\
\end{array}
\right.
\end{align*}
```

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

\begin{equation}
\begin{aligned}
  v^i_j(t) &= 2 \cdot r_1 \cdot (p^i_j - x^i_j(t - 1)) \\
  &\quad + 2 \cdot r_2 \cdot (g_j - x^i_j(t - 1)) \\
  \text{(1)}
\end{aligned}
\end{equation}

and

\begin{equation}
\begin{aligned}
  x^i_j(t) &= x^i_j(t - 1) + v^i_j(t) \\
  \text{(2)}
\end{aligned}
\end{equation}

where $p^i = [p^i_1, p^i_2, \ldots, p^i_\kappa]$ is the best position of the particle $i$, and $g = [g_1, g_2, \ldots, g_\kappa]$ 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 \( v_j^t(t) \):

\[
v_j^t(t) = k \cdot \{\omega(t) \cdot v_j^t(t-1) + \varphi_1 \cdot r_1 \cdot (p_j^t - x_j^t(t-1)) + \varphi_2 \cdot r_2 \cdot (g_j - x_j^t(t-1))\}
\]  

where \( \omega(t) \) is the inertia weight factor; \( \varphi_1 \) and \( \varphi_2 \) 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 \( \varphi_1 \) and \( \varphi_2 \) as follows:

\[
k = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}
\]  

where \( \varphi = \varphi_1 + \varphi_2 \) and \( \varphi > 4 \).

The particle velocity is limited by a maximum value \( v_{\text{max}} \) (3). This parameter \( v_{\text{max}} \) 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 searchability between the present position and the target.

If \( v_j^t(t) > \rho_{\text{max}} \), the updated \( x_j^t(t) \) should be equal to \( \rho_{\text{max}} \). Similarly, if \( v_j^t(t) < \rho_{\text{min}} \), the updated \( x_j^t(t) \) should be equal to \( \rho_{\text{min}} \). Here, \( \rho_{\text{min}} = [\rho_{\text{min}_1}, \rho_{\text{min}_2}, \ldots, \rho_{\text{min}_n}] \) and \( \rho_{\text{max}} = [\rho_{\text{max}_1}, \rho_{\text{max}_2}, \ldots, \rho_{\text{max}_n}] \); \( \rho_{\text{min}_1} \) and \( \rho_{\text{min}_2} \) are the minimum and maximum values of \( x_j^t(t) \) respectively, and \( j = 1, 2, \ldots, n \).

PSO utilizes \( p^* \) and \( g \) to affect the searching direction so that the particles will move in different manner. Also, the convergence can be made gradual towards \( p^* \) 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:

\[
\omega(t) = \omega_{\text{max}} - \frac{\omega_{\text{max}} - \omega_{\text{min}}}{T} \times t
\]

where \( T \) is the total number of iteration, \( t \) is the current iteration number, \( \omega_{\text{max}} \) and \( \omega_{\text{min}} \) 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 \( \omega \) 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:

\[
||\varsigma(t)|| = \sqrt{\frac{1}{\gamma} \sum_{i=1}^{\gamma} (||f(x^t_i(t))|| - ||\bar{f}(x^t_i(t))||)^2}
\]

where

\[
||\bar{f}(x^t_i(t))|| = \frac{1}{\gamma} \sum_{i=1}^{\gamma} ||f(x^t_i(t))||
\]

and \( ||\cdot|| \) denotes the \( l_2 \) vector norm.

Algorithm II.2: Pseudo code for FPSOCM(X(t))

\[
t \leftarrow 0
\]

Initialize \( X(t) \)

Define the probability of CM operation \( p_{cm} \)

output \( f(X(t)) \)

while <not termination condition>

\[
t \leftarrow t + 1
\]

output \( f(X(t)) \)

output \( (||\varsigma(t)|| \text{ based on (6) and (7)}) \)

Find the inertia weight \( \tilde{\omega}_k(t) \) by using fuzzy inference system based on (8) – (10).

Update velocity \( v(t) \) based on (11).

Find the control parameter \( \beta(t) \) 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 \( v_{\text{max}} \)

Generate a new swarm \( X(t) \) based on (2).

Ensure the updated \( x_j^t(t) \) 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 \( \tilde{\omega}(t) \):

\[
\begin{array}{l}
\left| \text{output} \right| = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 \text{ for } \tilde{\omega}(t) < 0.4 \\
\left| \text{output} \right| = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 \text{ for } \tilde{\omega}(t) > 0.6
\end{array}
\]
Rule $j$: IF $||z||$ is $N_j^1$, AND $t/T$ is $N_j^2$, THEN $\tilde{w}(t) = \sigma_j$, $j = 1, 2, \ldots, \varepsilon$  

where $N_j^1$ and $N_j^2$ are fuzzy terms of rule $j$, $\varepsilon$ is the number of rules, $\sigma_j \in \omega_{\min} \omega_{\max}$ is a singleton to be determined, with $\omega_{\min}$ and $\omega_{\max}$ being set at 0.1 and 1.1 respectively [26], [30]. The final value of $\tilde{w}(t)$ is given by:

$$\tilde{w}(t) = \sum_{j=1}^{\varepsilon} m_j(t) \sigma_j$$  

(9)

where

$$m_j(t) = \frac{\mu_{N_j^1}(||z(t)||) \times \mu_{N_j^2}(t/T)}{\sum_{j=1}^{\varepsilon} \left( \mu_{N_j^1}(||z(t)||) \times \mu_{N_j^2}(t/T) \right)}$$  

(10)

$\mu_{N_j^1}(||z(t)||)$ and $\mu_{N_j^2}(t/T)$ are the membership function values corresponding to $N_j^1$ and $N_j^2$ respectively. It should be noticed that $\tilde{w}(t)$ will replace $\omega(t)$ in (3) to produce $v_j^w(t)$:

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

(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 $\omega_{\min}$ and $\omega_{\max}$. For example, the output term is Very High, the value is equal to $\omega_{\max}$ (=1.1). If the output term is Medium, then the value is equal to $(\omega_{\min} + \omega_{\max})/2$ (=0.6). With $||z(t)||$ and $t/T$ as inputs, the 9 linguistic IF-THEN fuzzy rules for determining $\tilde{w}(t)$ are given as follows:

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

The rationale of the fuzzy rules for determining $\tilde{w}(t)$ is given as follows. The value of $t/T$ represents the evolution stage (a small $t/T$ represents an early stage.) The value of $\tilde{w}(t)$ 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, $\tilde{w}(t)$ 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 $||z||$ is the normalized standard deviation of fitness values among all the particles, a large value of $||z||$ 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 $||z||$ is “H”, the wide-spread particle velocity implies a larger value of $\tilde{w}(t)$ should be used for global exploration. When $||z||$ is “L” in the early stage, the value of $\tilde{w}(t)$ is also set large as the chance of the solution being trapped in a local optimum is high. In rule 2, the value of $\tilde{w}(t)$ is “M”, and we set the value of $\tilde{w}(t)$ 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 $\tilde{w}(t)$ is similar to that for rules 1-3. However, when $||z||$ is “H”, the value of $\tilde{w}(t)$ is smaller than that when $||z||$ is “H”. It is because the optimal solution may have been found in the middle stage when a smaller value of $\tilde{w}(t)$ 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 $||z||$ is “L”, the locations of particles are close to one another and near the optimal solution and the smallest value of $\tilde{w}(t)$ 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, $p_{cm} \in [0, 1]$, 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:

$$\tilde{v}_j^v(t) = \begin{cases} \frac{(1 - \beta(t)) v_j^v(t) + \beta(t) \tilde{v}_j^v(t), r_3 > 0.5}{(1 - \beta(t)) v_j^v(t) - \beta(t) \tilde{v}_j^v(t), r_3 \leq 0.5} \\
0.25 \left( r_4 \cdot (\rho_{\max} - \rho_{\min}) + \rho_{\min} \right) 
\end{cases}$$  

(12)

where

$$r_3, r_4 \in [0, 1]$$

is a random number, $v_j^v(t)$ is determined by (11), $\tilde{v}_j^v(t)$ is a random velocity of particle element and its value is bounded within 0.25 of the range of the particle
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 $\vec{v}_j(t)$ combines the 
information of $v_j(t)$ and $\vec{v}_j(t)$, exhibiting the characteristic 
of the crossover operation. However, in (12), $\vec{v}_j(t)$ 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 $\vec{v}_j(t)$ converging toward $v_j(t)$ 
or $\vec{v}_j(t)$. If $\beta(t)$ is approaching 0, $\vec{v}_j(t)$ will approach $v_j(t)$. 
Conversely, when $\beta(t)$ is approaching 1, $\vec{v}_j(t)$ will approach 
$\vec{v}_j(t)$. Hence, $\vec{v}_j(t)$ 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 $N_1$ AND $t/T$ is $N_2$, THEN $\beta(t) = \chi_j$, 

\[ j = 1, 2, ..., \varepsilon \]  

(14)

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

\[ \beta(t) = \sum_{j=1}^{N} m_j(t) \chi_j \]  

(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.A.1. 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 \( u(\cdot) \) in \( f_{pen} \) are given in [33]).

<table>
<thead>
<tr>
<th>Name</th>
<th>Test function / ( f(x) )</th>
<th>Domain range</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{sphere} )</td>
<td>( \sum_{i=1}^{30} x_i^2 )</td>
<td>(-100 \leq x_i \leq 100)</td>
</tr>
<tr>
<td>( f_{sch} )</td>
<td>( \sum_{i=1}^{30} [100 (x_i+1)^2 + (x_i-1)^2] )</td>
<td>(-2048 \leq x_i \leq 2048)</td>
</tr>
<tr>
<td>( f_{step} )</td>
<td>( \sum_{i=1}^{30} (x_i + 0.5)^2 )</td>
<td>(-10 \leq x_i \leq 10)</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>( \sum_{i=1}^{30} x_i^4 + \text{random}(0, 1) )</td>
<td>(-2.56 \leq x_i \leq 2.56)</td>
</tr>
<tr>
<td>( f_{kowa} )</td>
<td>( \sum_{i=1}^{30} x_i ) ( i \leq 5 ) ( \leq 30 )</td>
<td>(-100 \leq x_i \leq 100)</td>
</tr>
<tr>
<td>( f_{elli} )</td>
<td>( \sum_{i=1}^{30} x_i ) ( i \leq 10 ) ( \leq 30 )</td>
<td>(-5 \leq x_i \leq 5)</td>
</tr>
<tr>
<td>( f_{pen} )</td>
<td>( -\sum_{i=1}^{30} x_i ) ( \leq \sum_{i=1}^{30} x_i (x_i - p_i)^2 ) ( \leq 0 )</td>
<td>( \leq x_i \leq 1)</td>
</tr>
<tr>
<td>( f_{radt} )</td>
<td>( \sum_{i=1}^{30} x_i ) ( -10 \cos (2\pi x_i) + 10 )</td>
<td>(-50 \leq x_i \leq 50)</td>
</tr>
<tr>
<td>( f_{gos} )</td>
<td>( \sum_{i=1}^{30} x_i ) ( + 100 \cos (2\pi x_i) )</td>
<td>(-600 \leq x_i \leq 600)</td>
</tr>
<tr>
<td>( f_{radt} )</td>
<td>( -20 \exp \left( -0.2 + \left( \sum_{i=1}^{30} \left( \frac{x_i}{6} \right) \right) - \exp \left( \sum_{i=1}^{30} \cos 2\pi x_i \right) + 20 + e \right) )</td>
<td>(-32 \leq x_i \leq 32)</td>
</tr>
<tr>
<td>( f_{sphere} )</td>
<td>( \sum_{i=1}^{30} x_i^2 )</td>
<td>(-100 \leq x_i \leq 100)</td>
</tr>
<tr>
<td>( f_{scht} )</td>
<td>( f(x) = \sum_{i=1}^{30} x_i^4 )</td>
<td>(-100 \leq x_i \leq 100)</td>
</tr>
<tr>
<td>( f_{rosen} )</td>
<td>( \sum_{i=1}^{30} f(1 - \left( \frac{x_i}{10} \right)^2) ), ( \text{where} \ x = 1 \times 10^9 )</td>
<td>(-100 \leq x_i \leq 100)</td>
</tr>
<tr>
<td>( f_{gos} )</td>
<td>( \sum_{i=1}^{30} [100 (x_i+1)^2 + (x_i-1)^2] )</td>
<td>(-100 \leq x_i \leq 100)</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>( \sum_{i=1}^{30} [x_i^2 - 10 \cos (2\pi x_i) + 10] )</td>
<td>(-5 \leq x_i \leq 5)</td>
</tr>
</tbody>
</table>

TABLE I
BENCHMARK TEST FUNCTIONS.

**A. Experimental setup**

For comparison purpose, the performance of HPSOWM [26], HPSOM [4], HGAPO [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 (\( \gamma \)): 50
- Number of runs: 50
- Acceleration constant \( \varphi_1 \): 2.05 [30]
- Acceleration constant \( \varphi_2 \): 2.05 [30]
- Maximum velocity \( v_{\text{max}} \): 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 \( f_{sphere} \), \( f_{sch2.21} \), \( f_{pen} \), \( f_{grie} \), \( f_{ack} \), \( f_{sphere.sft} \);
- \( p_{cm} = 0.005 \) for \( f_{radt} \), \( f_{sch1.2.sft} \), \( f_{elli.sft} \);
- \( p_{cm} = 0.01 \) for \( f_{rosen} \), \( f_{step} \), \( f_{quart} \), \( f_{sch2.21} \), \( f_{kowa} \), \( f_{radt.sft} \), \( f_{elli.sft} \);
- \( p_{cm} = 0.1 \) for \( f_{hart} \);

For HPSOWM, HPSOM, and HGAPO:
- 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 HGAPO:
- 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 0.000002882 × 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 $2.22 f_{\text{sch2.22}}$. FPSOCM indicates the good performance of the algorithm. In general, FPSOCM is the best to tackle unimodal functions when compared with other methods. In terms of computational time, FPSOCM consumes almost the same amount of time as the other methods.

There are two multimodal functions with a few local minima used to do the testing. The results are tabulated in Table III. The function $f_{\text{kow}}$ 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 $f_{\text{hart}}$, 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 functions with a few local minima when compared with the other methods.

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 $f_{\text{kow}}$ 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 ($f_{\text{hart}}$), 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 functions 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 $f_{\text{kow}}$ and $f_{\text{hart}}$ in Section III.B2. The results are tabulated in Table IV, where $f_{\text{pen}}, f_{\text{fras}}, f_{\text{grie}},$ and $f_{\text{ack}}$ 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 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.

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 ($f_{\text{sphere} \_sft}$), the Shifted Schwefel’s problem 1.2 ($f_{\text{sch1.2} \_sft}$), the Shifted rotated high conditioned elliptic function ($f_{\text{elli} \_sft}$), the Shifted Rosenbrock’s function ($f_{\text{rosen} \_sft}$) and the Shifted Rastrigin’s function ($f_{\text{ras} \_sft}$).
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 statistically 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).

### TABLE V

<table>
<thead>
<tr>
<th>Function</th>
<th>Mean FPSOCM</th>
<th>Mean HPSOM</th>
<th>Mean HPSOM</th>
<th>Mean HGAPSO</th>
<th>Mean IPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{sphere} )</td>
<td>2.872689</td>
<td>3.941711</td>
<td>2.804771</td>
<td>2.804771</td>
<td>2.804771</td>
</tr>
<tr>
<td>( f_{rosen} )</td>
<td>0.000180</td>
<td>0.000180</td>
<td>0.000180</td>
<td>0.000180</td>
<td>0.000180</td>
</tr>
<tr>
<td>( f_{ack} )</td>
<td>0.000513</td>
<td>0.000513</td>
<td>0.000513</td>
<td>0.000513</td>
<td>0.000513</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.001316</td>
<td>0.001316</td>
<td>0.001316</td>
<td>0.001316</td>
<td>0.001316</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.002105</td>
<td>0.002105</td>
<td>0.002105</td>
<td>0.002105</td>
<td>0.002105</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.000924</td>
<td>0.000924</td>
<td>0.000924</td>
<td>0.000924</td>
<td>0.000924</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.2172</td>
<td>0.2172</td>
<td>0.2172</td>
<td>0.2172</td>
<td>0.2172</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.000066</td>
<td>0.000066</td>
<td>0.000066</td>
<td>0.000066</td>
<td>0.000066</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

### TABLE VI

<table>
<thead>
<tr>
<th>Function</th>
<th>p-value between FPSOCM and HPSOM</th>
<th>p-value between FPSOCM and HPSOM</th>
<th>p-value between FPSOCM and HGAPSO</th>
<th>p-value between FPSOCM and IPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{sphere} )</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>( f_{rosen} )</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>( f_{ack} )</td>
<td>0.000513</td>
<td>0.000513</td>
<td>0.000513</td>
<td>0.000513</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.001316</td>
<td>0.001316</td>
<td>0.001316</td>
<td>0.001316</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.002105</td>
<td>0.002105</td>
<td>0.002105</td>
<td>0.002105</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.000924</td>
<td>0.000924</td>
<td>0.000924</td>
<td>0.000924</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.2172</td>
<td>0.2172</td>
<td>0.2172</td>
<td>0.2172</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.000066</td>
<td>0.000066</td>
<td>0.000066</td>
<td>0.000066</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

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 \( \bar{\omega}. \) 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 \( \omega \) 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 \( f_{sch.21}. \) 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.

### TABLE VII

<table>
<thead>
<tr>
<th>Function</th>
<th>fuzzy inertia weight</th>
<th>linear inertia weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{sphere} )</td>
<td>2200.00337</td>
<td>3901.00079</td>
</tr>
<tr>
<td>( f_{rosen} )</td>
<td>0.9091225</td>
<td>3.40327459</td>
</tr>
<tr>
<td>( f_{ack} )</td>
<td>0.09152</td>
<td>32.0799</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.0063556</td>
<td>0.00706537</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.72132615</td>
<td>2.67772016</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.2485042</td>
<td>0.18891851</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.00051884</td>
<td>0.00147309</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>3.28201027</td>
<td>3.28186516</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>580.2509</td>
<td>739.9839</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>18.3930033</td>
<td>20.7363037</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.00221422</td>
<td>0.04404187</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>0.30051016</td>
<td>0.49893948</td>
</tr>
<tr>
<td>( f_{hart} )</td>
<td>152.34764</td>
<td>32.281957</td>
</tr>
</tbody>
</table>

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 will undergo 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. 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 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 \(f_{sch2.2}, f_{pen}, f_{ack}, f_{sch1.2.sfl}\) 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 \(f_{elli.sfl}\) 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 \(f_{fast.sfl}\) 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.

### 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 show in the performance of different PSOs. The ELD aims to schedule power generator outputs with respect to the load demands, but 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:

\[
\min \sum_{i=1}^{n} C_i(P_{Li})
\]

where \(C_i(P_{Li})\) 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:

\[
D = \sum_{i=1}^{n} P_{Li} - P_{Loss}
\]

\[
P_{Li,min} \leq P_{Li} \leq P_{Li,max}
\]

where \(D\) is the load demand, \(P_{Li}\) is the output power of the \(i\)-th generator, \(P_{Loss}\) is the transmission loss, \(P_{Li,min}\) and \(P_{Li,max}\) 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,

\[
C_i(P_{Li}) = a_i P_{Li}^2 + b_i P_{Li} + c_i + \| e_i \times \sin(f_i \times (P_{Li,min} - P_{Li})) \|
\]

where \(a_i, b_i, c_i\), and \(e_i, f_i\) are coefficients of the cost curve of the \(i\)-th generator, \(e_i, f_i\) 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:

\[ \mathbf{x} = [P_{L_1}, P_{L_2}, P_{L_3}, \ldots, P_{L_n}] \]

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

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

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 \([P_{L_i,\text{min}}, P_{L_i,\text{max}}]\), the following conditions are considered:

\[
\begin{align*}
\text{if } P_{L_i} > P_{L_i,\text{max}} & : P_{L_i} = P_{L_i} + (P_{L_i} - P_{L_i,\text{max}}) \\
\text{if } P_{L_i} < P_{L_i,\text{min}} & : P_{L_i} = P_{L_i} - (P_{L_i,\text{min}} - P_{L_i})
\end{align*}
\]

(23)

(24)

It should be noted from (23) and (24) that if the value of \( P_{L_i} \) 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 (\( p < 0.0001 \)).

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 service configuration problem can be formulated with the following objective function:

\[
\min Z = \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]

\[
= \sum_{l=1}^{n_c} \sum_{k=1}^{n_k} \sum_{i=1}^{n_c} \left[ m_{ij} f_p x_i + n_{jk} f_c y_j + c_{jk} z_k \right]
\]
and subject to the following constraints:

\[ a) \sum_{i=1}^{n_p} f_p C_i(\cdot) \leq \sum_{j=1}^{n_r} \omega_{n j}; \]
\[ b) \sum_{j=1}^{n_r} f_j C_j(\cdot) \leq \sum_{k=1}^{n_c} \omega_{n k}; \]
\[ c) \sum_{k=1}^{n_c} f_k R_k(\cdot) \leq \sum_{i=1}^{n_r} \omega_{n i}; \]

where Z is the overall configuration cost across the four layer networking structure for NEs i, j, k and l; \( \hat{n}_{ij} \), \( \hat{n}_{k} \), and \( \hat{n}_{c} \) denote the number of the corresponding NEs; \( f_p C_i(\cdot) \), \( f_j C_j(\cdot) \), and \( f_k R_k(\cdot) \) 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.

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

\( P, C, S, k \), and \( R_l \) describe the connections between the layers as described above. \( m_{i,j}, n_{j,k}, o_{k,l} \) 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 \( X = [M, N, O] \), where \( M, N, O \) are the vector representation for individual weights \( m_{i,j}, n_{j,k}, o_{k,l} \) respectively. As indicated in (25), the parameters \( m_{i,j}, n_{j,k}, \) and \( o_{k,l} \) are to be tuned by PSOs. The objective is to find the optimal parameters of \( m_{i,j}, n_{j,k}, \) 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: \( n_r = 16, n_s = 16, n_e = 16, c_p = 1, c_e = 4, c_s = 4, \) 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 \((27.56 \pm 1.56 vs 27.56 \pm 1.45 \ (p = 0.0718))\) with a 93% confidence level. Hence, FPSOCM is the best method among all the tested PSO methods for solving this telecommunication problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Best</th>
<th>Std Dev</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPSOCM</td>
<td>27.00</td>
<td>27.56</td>
<td>1.56</td>
<td>0.0718</td>
</tr>
<tr>
<td>HPSOWM</td>
<td>28.00</td>
<td>28.24</td>
<td>1.45</td>
<td>0.0049</td>
</tr>
<tr>
<td>HPSOM</td>
<td>28.44</td>
<td>28.24</td>
<td>1.59</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>HGAPSO</td>
<td>28.24</td>
<td>28.44</td>
<td>1.68</td>
<td>0.0003</td>
</tr>
<tr>
<td>IPSO</td>
<td>28.24</td>
<td>28.44</td>
<td>1.60</td>
<td></td>
</tr>
</tbody>
</table>

TABLE X

\( p \)-value 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.](image-url)

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
