2009 Third International Conference on Network and System Security

A New Particle Swarm Optimization Algorithm for Neural Network Optimization

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

Hung T. Nguyen,
Centre for Health Technologies,
Faculty of Engineering and Information Technology, University of Technology, Sydney, NSW, Australia
e-mail: Hung.Nguyen@uts.edu.au

K.Y. Chan
Digital Ecosystems and Business Intelligence Institute, Curtin University of Technology, WA, Australia
e-mail: kit.chan@curtin.edu.au

This paper presents a new particle swarm optimization (PSO) algorithm for tuning parameters (weights) of neural networks. The new PSO algorithm is called fuzzy logic-based particle swarm optimization with cross-mutated operation (FPSOCM), where the fuzzy inference system is applied to determine the inertia weight of PSO and the control parameter of the proposed cross-mutated operation by using human knowledge. By introducing the fuzzy system, the value of the inertia weight becomes variable. The cross-mutated operation is effectively force the solution to escape the local optimum. Tuning parameters (weights) of neural networks is presented using the FPSOCM. Numerical example of neural network is given to illustrate that the performance of the FPSOCM is good for tuning the parameters (weights) of neural networks.

I. INTRODUCTION

Particle swarm optimization (PSO) is a new approach which inspired by the social behaviors of animals like fish schooling and bird flocking [6]. Comparing with other population based stochastic optimization methods, such as evolutionary algorithms, PSO has comparable or even superior search performance for many hard optimization problems with faster and more stable convergence rates [7]. Furthermore, PSO has memory, previous visited best positions in PSO are remembered, while in evolution algorithms, which are forgotten once the current population changes. PSO has been used in different industrial areas such as power systems [1], parameters learning of neural networks [5], scheduling [10], etc. However, observations reveal that PSO converges sharply in the early stage of the searching process, but saturates or even terminates in the later stage. It behaves like the traditional local searching methods that trap in local optima.

Recently, different hybrid PSO methods have been proposed to overcome the drawback of trapping in local optima. The hybrid PSO has been first proposed in 1998 [2], in which a standard selection mechanism is integrated with PSO. A new hybrid gradient descent PSO (HGPSO), which is integrated with gradient information to achieve faster convergence without getting trapped in local minima is proposed [13]. However, the computational demand of HGPSO is increased by the process of the gradient descent. In addition, it is poor to handle the multimodel problem that contain many local minima. In [5], a hybrid PSO algorithm named HGAPSO is proposed, which incorporates GA's evolutionary operations of crossover, mutation and reproduction. In [1], a hybrid PSO named HPSOM is proposed, in which a constant mutating space is used in mutation. In both HGAPSO and HPSOM, the solution space can be explored by performing mutation operations on particles along the search, and premature convergence is more likely to be avoided. However, the mutating space is kept unchanged all the time throughout the search, and the space for the permutation of particles in PSO is also fixed. It can be improved by varying the mutating space along the search. A hybrid PSO with wavelet mutation operation (HPSOWM) is proposed [8], which the mutating space is varying by applying wavelet theory. The solution quality and solution reliability are improved.

In [4], an improved version of PSO where inertia weight factor is introduced. The aim of the inertia weight provides a balance between the global exploration and local exploitation and which governed by a linear characteristic function. However, not all the optimization problems are linear. Thus, in this paper, a hybrid fuzzy logic based PSO with cross-mutated operation is proposed. The main contributions of this paper are as follows: (i) an adaptive inertia weight is proposed which incorporate with fuzzy inference system. Fuzzy logic is good in representing some expect knowledge and experience in some linguistic rule which can be easily understood by human being. Using fuzzy inference to determine the inertia weight of PSO, the characteristic of function of inertia weight becomes nonlinear. It shows that the nonlinear characteristic of the inertia weight performs better solution quality in this paper; (ii) a new operation namely cross-mutated (CM) operation is introduced. The CM operation is effectively to solve the drawback of PSO that it is easier to trap in local optima. In CM operation, the control parameter is determined by fuzzy rules and the operation becomes adaptive; (iii) as numerical example, the proposed PSO is used to learn the neural network parameters for estimate the number of sunspots. The results will compared with other exiting hybrid PSO methods on these applications and we can see that the learning performance of these applications are improved by using proposed fuzzy-based PSO with CM operation.

II. FUZZY LOGIC-BASED PARTICLE SWARM OPTIMIZATION WITH CROSS-MUTATED OPERATION (FPSOCM)

Particle swarm optimization (PSO) is a stochastic optimization method. It uses a number of particles that constitute a swarm. Each particle traverses the search space looking for the global optimum. Recently, an improved PSO is proposed [4], where constriction and inertia weight factors are introduced and the searching ability is improved compared with standard PSO [6]. In this paper, a fuzzy logic-based

978-0-7695-3838-9/09 $26.00 © 2009 IEEE
DOI 10.1109/NSS.2009.39

particle swarm optimization with cross-mutated operation namely FPSOCM is proposed. The details of both IPSO and FPSOCM will be discussed as follows.

A. PSO with constriction and inertia weight factors (IPSO)

Let $X(t)$ is a swarm at the t-th iteration. Each particle $\mathbf{x}^i(t) \in X(t)$ contains κ elements $x_j^i(t) \in \mathbf{x}^i(t)$ at the t-th iteration, 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. First, the particles of the swarm are initialized and then evaluated by a defined cost (objective) function. The evaluation cost value is represented by $f\left(\mathbf{x}^i(t)\right)$.In the same time, current generation number t is initially set at 0. The objective of PSO is to minimize the cost values of particles iteratively. The swarm evolves from iteration t+1 by repeating the processes until the terminal conditional occur.

In PSO, there is one important variable, velocity, which corresponds to the flight speed in the search space. The velocity $v_j^i(t)$ and the position $x_j^i(t)$ of the j-th element of the p-th particle at the t-th generation can be calculated using the following formulae:

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

and

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

where

$p^i = \left[p_1^i \quad p_2^i \quad \ldots \quad p_\kappa^i\right]$ and $g = \left[g_1 \quad g_2 \quad \ldots \quad g_\kappa\right]$; the best position of a particle i is represented as p^i; the position of the best particle among all the particles is represented as g; r_1 and r_2 return a uniform random number in the range of [0,1]. In [4], an improved version of PSO (IPSO) is presented, where the constriction factor and inertia weight factor are introduced. Here, when the PSO with constriction factor and inertia weight factor is used, (1) will be changed to:

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

where ω is an inertia weight factor; φ_1 and φ_2 are acceleration constants; k is a constriction factor derived from the stability analysis of equation (3) to ensure the system to be converged but not prematurely [4]. Mathematically, k is a function of φ_1 and φ_2 as reflected in the following equation:

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

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

IPSO utilizes p^i and g to modify the current search point in order to avoid the particles moving in the same direction, but to converge gradually toward p^i and g. A suitable selection of the inertia weight ω provides a balance between the global and local explorations. Generally, $\omega(t)$ is governed by the following equation:

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

```
begin
   t →0. // iteration number
   Initialize X (t). // X(t): swarm for iteration t
   Evaluate f(X (t)). // f( · ): cost function
while (not termination condition) do
      begin
      t→t + 1.
      Evaluate t/ T .
      Evaluate ς (t)/ ||ς (t)|| based on (6) and (7).
      Find the inertia weight w_k(t) by using fuzzy inference s  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) − (16).
      if R_cm > p_cm then
         Perform Cross-mutated operation based on (12) and (13).
      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).
      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
      Evaluate f(X (t)).
      end
end
end
```

Fig. 1 The pseudo code for FPSOCM

where t is the current iteration number, T is the total number of iteration, $\omega_{\max}$ and $\omega_{\min}$ are the upper and lower limits of the inertia weight, and normally set to 1.1 and 0.1 respectively [8, 13].

In (3), the particle velocity is limited by a maximum value $v_{\max}$. The parameter $v_{\max}$ determines the resolution with which regions are to be searched between the present position and the target position. This limit enhances the local exploitation of the problem space and it realistically simulates the incremental changes of human learning. If $v_{\max}$ is too high, particles might fly past good solutions. If $v_{\max}$ is too small, particles may not explore sufficiently beyond local solutions. From experience, $v_{\max}$ is often set at 10% to 20% of the dynamic range of the element on each dimension.

After updated the velocity of all particles, a new swarm $X(t)$ is based on (2) to update. To ensure all particle element $x_j^i(t)$ in $X(t)$ falls within the range $\left[\rho_{\mathbf{min}}, \quad \rho_{\mathbf{max}},\right]$, the following conditions are considered. If $x_j^i(t) > \rho_{\max_j}$, then the updated $x_j^i(t)$should be equal to $\rho_{\max_j}$. Similarly, If $x_j^i(t) < \rho_{\min_j}$, then the updated $x_j^i(t)$ should be equal to $\rho_{\min_j}$. Here, $\rho_{\mathbf{min}} = \left[\rho_{\min x_1} \quad \rho_{\min_2} \quad \cdots \quad \rho_{\min_\kappa}\right]$ and $\rho_{\mathbf{max}} = \left[\rho_{\max_1} \quad \rho_{\max_2} \quad \cdots \quad \rho_{\max_\kappa}\right]$; $\rho_{\min_j}$ and $\rho_{\max_j}$ are minimum and maximum values of $x_j^i(t)$ respectively and j = 1, 2,..., κ.

B. Fuzzy logic-based PSO with cross-mutated operation

The pseudo code for FPSOCM is shown in Fig.1. In this proposed method, there are two main contributions that will enhance the performance of searching compared with other PSO methods. Firstly, an adaptive inertia weight $\omega_k(t)$ is

proposed to improve the solution quality of searching. Secondly, cross-mutated (CM) operation is given to solve the drawback of IPSO that it is easier to trap into local minima. From this figure, we can see that the adaptive inertia weight $\omega_k(t)$ and the control parameter $\beta(t)$ of CM operation are determined by a fuzzy inference system. The detail of the operation of FPSOCM is given as next section.

1) Adaptive inertia weight

In IPSO, inertia weight is used to provide a balance between the global exploration and local exploitation. A linear inertia weight is governed by (5). From this equation, when the value of *t/T* is smaller which implies that it is doing global exploration. Similarly, higher value of *t/T* implies that it is doing fine-tuning (local exploitation). We can see that the characteristic of this function is linear. However, some of the optimization problems are nonlinear. Thus, an adaptive (nonlinear) inertia weight is proposed to enhance the performance of the searching. In this paper, an adaptive inertia weight $\omega_k(t)$ is proposed which is incorporated with a fuzzy inference system. In fuzzy inference system, there are two inputs and two outputs. One of the output is adaptive inertia weight $\omega_k(t)$, the other one is control parameter $\beta(t)$ of CM (It will discussed in later). The inputs of the fuzzy system are $\varsigma(t)/\|\varsigma(t)\|$ and *t/T*. $\varsigma(t)/\|\varsigma(t)\|$ is a normalized standard deviation of cost value among all the particles. A larger value of $\varsigma(t)/\|\varsigma(t)\|$ implies that the variance between each cost value of particle $f(X^i(t))$ is larger. Conversely, when the value of $\varsigma(t)/\|\varsigma(t)\|$ is smaller, that means the variance between each cost value of particle $f(X^i(t))$ is smaller. In other words, the location of each particle is closer. The formulation of $\varsigma(t)/\|\varsigma(t)\|$ is defined as follows:

$$\varsigma(t)=\sqrt{\frac{1}{\gamma}\sum_{i}^{\lambda}\left(f(X^i(t))-\bar{f}(X^i(t))\right)^2}, \tag{6}$$

where

$$\bar{f}(X^i(t))=\frac{1}{\gamma}\sum_{i=1}^{\gamma}f(X^i(t)), \tag{7}$$

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

The adaptive inertia weight $\omega_k(t)$ is governed by the following fuzzy rules:

Rule *j*: IF $\varsigma(t)/\|\varsigma(t)\|$ is N_1^j AND *t/T* is N_2^j,

THEN $\omega_k(t)=\sigma_j$, $j=1, 2, \ldots, \varepsilon$, (8)

where N_1^j and N_2^j are fuzzy terms of rule *j*, ε denotes the number of rule, $\sigma_j\in[\omega_{\min}\ \ \omega_{\max}]$ is the singleton to be determined. In this paper, $\omega_{\min}$ and $\omega_{\max}$ are set at 0.1 and 1.1 respectively [15, 20]. The final value of $\omega_k(t)$ is given by:

$$\omega_k(t)=\sum_{j=1}^{\varepsilon}m_j(t)\sigma_j, \tag{9}$$

where

$$m_j(t)=\frac{\mu_{N_1^j}(\varsigma(t)/\|\varsigma(t)\|)\times\mu_{N_1^j}(t/T)}{\sum_{j=1}^{r_m}\mu_{N_1^j}(\varsigma(t)/\|\varsigma(t)\|)\times\mu_{N_1^j}(t/T)}, \tag{10}$$

$\mu_{N_1^j}(\varsigma(t)/\|\varsigma(t)\|)$ and $\mu_{N_1^j}(t/T)$ are the membership function corresponding to N_1^j and N_2^j respectively. Noting that the value of $\omega_k(t)$ will replace the value of $\omega(t)$ in (3) to product a new $v_j^i(t)$ with adaptive inertia weight. Thus, the new velocity will be changed to

$$v_j^i(t)=k\cdot\{\omega_k(t)\cdot v_j^i(t-1)\}+\varphi_1\cdot r_1\left(p_j^i-x_j^i(t-1)\right)+\varphi_2\cdot r_2\left(g_j-x_j^i(t-1)\right), \tag{11}$$

In the proposed algorithm, there are 3 membership functions for each input and shown in Fig. 2. The 3 fuzzy terms are namely L (Low), M (Middle), and H (High). Based on the characteristic of $\varsigma(t)/\|\varsigma(t)\|$ and *t/T*, the 9 linguistic IF-THEN fuzzy rules for determine $\omega_k(t)$ are given as follows:

Rule 1: IF $\varsigma(t)/\|\varsigma(t)\|$ is "L" AND *t/T* is "L",THEN $\omega_k(t)$=1.1,

Rule 2: IF $\varsigma(t)/\|\varsigma(t)\|$ is "M" AND *t/T* is "L",THEN $\omega_k(t)$=0.9,

Rule 3: IF $\varsigma(t)/\|\varsigma(t)\|$ is "H" AND *t/T* is "L",THEN $\omega_k(t)$=1.1,

Rule 4: IF $\varsigma(t)/\|\varsigma(t)\|$ is "L" AND *t/T* is "M",THEN $\omega_k(t)$=0.6,

Rule 5: IF $\varsigma(t)/\|\varsigma(t)\|$ is "M" AND *t/T* is "M",THEN $\omega_k(t)$=0.5,

Rule 6: IF $\varsigma(t)/\|\varsigma(t)\|$ is "H" AND *t/T* is "M",THEN $\omega_k(t)$=0.7,

Rule 7: IF $\varsigma(t)/\|\varsigma(t)\|$ is "L" AND *t/T* is "H",THEN $\omega_k(t)$=0.1,

Rule 8: IF $\varsigma(t)/\|\varsigma(t)\|$ is "M" AND *t/T* is "H",THEN $\omega_k(t)$=0.2,

Rule 9: IF $\varsigma(t)/\|\varsigma(t)\|$ is "H" AND *t/T* is "H",THEN $\omega_k(t)$=0.3.

The rationale of the selected fuzzy rules for determine $\omega_k(t)$ is described as follows: the value of $\omega_k(t)$ is determined by fuzzy inputs $\varsigma(t)/\|\varsigma(t)\|$ and *t/T*. The value of *t/T* represents the iteration stage (smaller value of *t/T* represents the searching process in the early stage, a larger value of *t/T* represents the searching process in the later stage). The value of $\omega_k(t)$ should be higher as the value of *t/T* is smaller (in early stage) implies that a larger value of velocity of particle element is given for global searching. Similarly, a larger value of *t/T* implies that a higher value of velocity of particle element for local searching. Thus, the values of $\omega_k(t)$ of the fuzzy rules 1, 2, and 3 *t/T* is "L") are larger than the rules 4, 5, and 6 (*t/T* is "M"). For the same reason, the value of $\omega_k(t)$ of the fuzzy rules 4, 5, and 6 are larger than the rules 7, 8, and 9 (*t/T* is "H'). The value of $\omega_k(t)$ should be smaller as *t/T* increase in order to reduce the value of velocity of particle element for fine-tuning. As mention before, $\varsigma(t)/\|\varsigma(t)\|$ is a normalized standard deviation of cost value among all the particles. A larger value of $\varsigma(t)/\|\varsigma(t)\|$ implies that the variance between each cost value of particle $f(X^i(t))$ is larger. In other words, the location of each particle is far away.

In rules 1 to 3, the searching process is in early stage t/T is "L"), when $\varsigma(t)/\|\varsigma(t)\|$ is "H", that implied the location of each particle is far away in early stage. Thus, a larger value of $\omega_k(t)$ should be given for global exploration. In a special case, when $\varsigma(t)/\|\varsigma(t)\|$ is "L", that implies the location of each particle is closed in early stage. We also set the value of $\omega_k(t)$ to larger, because there has a high chance that the solutions are trap into local optima (it is affected from the smaller value of $\varsigma(t)/\|\varsigma(t)\|$). Therefore, a larger value of $\omega_k(t)$ is given to force the particle escape the local optima. In rule 2, the value of $\varsigma(t)/\|\varsigma(t)\|$ is "M", we set the value of $\omega_k(t)$ is slight smaller than rules 1 and 3 in early stage t/T is "L").

In rule 4 to rule 6, the searching process is in middle stage (t/T is "M"). The rationale of the suggested value of $\omega_k(t)$ is similar to rules 1-3. However, there is one difference that when $\varsigma(t)/\|\varsigma(t)\|$ is "L', the value of $\omega_k(t)$ is smaller than $\varsigma(t)/\|\varsigma(t)\|$ is "H". It is because the optimal solution may be found in middle stage where a smaller value of $\omega_k(t)$ is given.

In rule 7 to rule 9, which the searching process is in later stage (t/T is "H"). In this stage, the searching process undergoes fine-tuning process (local exploitation) to find the optimal solution. Because of it, when the value of $\varsigma(t)/\|\varsigma(t)\|$ is "L" that implied the position of most particles are closed and near the best solution. Thus, a smallest value of $\omega_k(t)$ is given.

2) Cross-mutated operation

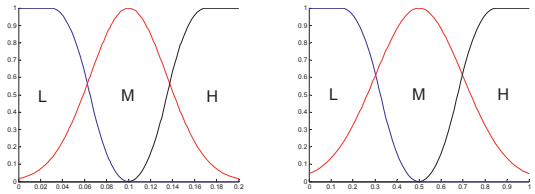
In this section, a new cross-mutated (CM) operation is introduced. This new CM operation is merged the idea of crossover and mutation operation of genetic algorithm. The aim of the CM operation is force the particle escape the local optima. Furthermore, a control parameter $\beta(t)$ is introduced into the CM operation, and the operation becomes more adaptive. This control parameter is governed by some fuzzy rules with human knowledge. By introducing the CM operation, the performance of the proposed PSO method is improved and good to tackle some multimodal optimization problems with many local minima.

The detail of the CM operation is as follows. Every velocity of particle element of the swarm will have a chance to undergo CM operation governed by a probability of cross-mutated operation, $p_{cm} \in [0 \quad 1]$,which is defined by user. For each velocity of particle element, a random number R_{cm} between 0 and 1 will be generated such that if it is less than or equal to p_{cm}, the CM operation will take place on that element. The sensitivity of the p_{cm} with experimental results and the analysis will be discussed in later. The choice of the p_{cm} will affect the quality of solution.

The resulting velocity of particle element under the CM operation is given by:

$$\bar{v}_j^i(t) = (1-\beta(t))v_j^i(t) + \beta(t)\tilde{v}_j^i(t), \tag{12}$$

where



$$\tilde{v}_j^i(t) = \frac{1}{4}\left\{r_3 \cdot \left(\rho_{\max_j} - \rho_{\min_j}\right) - \rho_{\max_j}\right\}, \tag{13}$$

where $v_j^i(t)$ is determined by (11), $\tilde{v}_j^i(t)$ is a random velocity of particle element and the value of this velocity is randomly generated and bounded with 0.25 of the dynamic range of the particle element. $r_3 \in [0 \quad 1]$ is an uniform random number. In (12), the resulting velocity of particle element $\bar{v}_j^i(t)$ is combined with the information of $v_j^i(t)$ and $\tilde{v}_j^i(t)$. This information exchanging process is like as crossover operation. However, in CM operation, $\bar{v}_j^i(t)$ is changed (mutated) one-by-one which is like as mutation operation. Therefore, we called it is cross-mutated (CM) operation.

In (12), the control parameter $\beta(t)$ provides a balance to control the resulting velocity $\bar{v}_j^i(t)$ converge toward $v_j^i(t)$ or $\tilde{v}_j^i(t)$. If $\beta(t)$ is approaching 0, the $\bar{v}_j^i(t)$ will tends to the $v_j^i(t)$. Conversely, when $\beta(t)$ is approaching 1,the $\bar{v}_j^i(t)$ will tends to the $\tilde{v}_j^i(t)$. The proposed random velocity $\tilde{v}_j^i(t)$ in (13) has ability to force the particle element to escape the local optima with a random movement. In this operation, the value of the $\beta(t)$ is governed by the following fuzzy rules:

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

THEN $\beta(t) = \chi_j, j = 1, 2, \ldots, \varepsilon,$ (14)

where N_1^j and N_2^j are fuzzy terms of rule j, ε denotes the number of rule, χ is the 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

$$m_j(t) = \frac{\mu_{N_1^j}\left(\varsigma(t)/\|\varsigma(t)\|\right) \times \mu_{N_1^j}(t/T)}{\sum_{j=1}^{r_m} \mu_{N_1^j}\left(\varsigma(t)/\|\varsigma(t)\|\right) \times \mu_{N_1^j}(t/T)}, \tag{16}$$

$\mu_{N_1^j}\left(\varsigma(t)/\|\varsigma(t)\|\right)$ and $\mu_{N_1^j}(t/T)$ are the membership function corresponding to N_1^j and N_2^j respectively. In this paper, 9 linguistic IF-THEN fuzzy rules for determine $\beta(t)$ are given as follows:

Rule 1: IF $\varsigma(t)/\|\varsigma(t)\|$ is "L" AND t/T is "L",THEN $\beta(t)$ =0.5,

Rule 2: IF $\varsigma(t)/\|\varsigma(t)\|$ is "M" AND t/T is "L",THEN $\beta(t)$ =0.4,

Rule 3: IF $\varsigma(t)/\|\varsigma(t)\|$ is "H" AND t/T is "L",THEN $\beta(t)$ =0.6,

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

The rationale of the selected fuzzy rules is similar to the rules of adaptive inertia weight in. The brief description of the choosing fuzzy rules for CM operation is given as follows: the value of $\beta(t)$ is determined by $\varsigma(t)/\|\varsigma(t)\|$ and t/T, and some fuzzy rules. As mention before, the value of the t/T represents the iteration stage. We can see from rules 1-3, where the searching process is in early stage and rules 7-9, where the searching process is in later stage, the values of $\beta(t)$ under rules 1-3 are larger than the values of $\beta(t)$ under rules 7-9. It is because a significant random velocity (higher value of $\beta(t)$ in (12) provides a global exploration in early stage. Conversely, the effect of the random velocity should be reduced in later stage for fine-tuning (local exploitation) with smaller value of $\beta(t)$.

Now, we have discuss about the effect of the $\varsigma(t)/\|\varsigma(t)\|$, the concept is same as the rules of adaptive inertia weight. In early stage, when $\varsigma(t)/\|\varsigma(t)\|$ is "L", that implied the location of each particle is closed in early stage. Thus, we need to set the value of $\beta(t)$ to larger compared with when $\varsigma(t)/\|\varsigma(t)\|$ is "M". It is because there have a higher chance that the solution is trapped into a local optimum. Conversely, in later stage, the searching process undergoes fine-tuning process (local exploitation) to find the optimal solution. Because of it, when the value of $\varsigma(t)/\|\varsigma(t)\|$ is "L" that implies the position of most particles are closed (similar) and near the best solution. Thus, a smallest value of $\beta(t)$ is given.

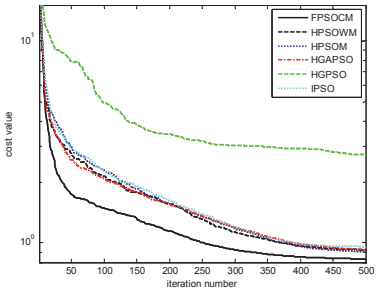
After the CM operation, an updated swarm is generated. This swarm will repeat the same process. Such an iterative process will be terminated when a defined number of iteration is met.

III. Parameters Learning of Neural Network

Weight learning is one of the important issues of neural networks. The learning processing aims to find a set of optimal network parameters. PSO is a good training algorithm for neural networks. The same PSO method can be used to train many different networks regardless of whether they are feed-forward one, recurrent one, wavelet one, associative-memory or other special structure types. Also, PSO is good to tackle the problems which are nonlinear, non-differentiable and multimodal domain. In general, neural network can be used to learn the input-output relationship of an application using the PSO. The input-output relationship can be described by,

$$\mathbf{y}^d(t) = g\left(\mathbf{z}^d(t)\right),\ t = 1, 2, \ldots, n_d, \tag{17}$$

where $\mathbf{z}^d(t) = \left[z_1^d(t)\ \ z_2^d(t)\ \ \ldots\ \ z_{n_{in}}^d(t)\right]$ and $y^d(t) = \left[y_1^d(t)\ \ y_2^d(t)\ \ \ldots\ \ y_{n_{out}}^d(t)\right]$ are the given inputs and the desired outputs of an unknown nonlinear function $g(\cdot)$ respectively; n_d denotes the number of input-output data pairs. The objective is to minimize the error of neural network.



A. Numerical examples

A numerical example on forecasting of sunspot number [3] using wavelet neural network [9] will be given in this section. The cycles generated are nonlinear, non-stationary, and non-Gaussian which are difficult to model and predict. We use the variable translation wavelet neural network [9] for the sunspot forecasting. For this network, wavelets are used as the transfer functions in the hidden layer of the network. The one of the network parameters, translation parameters of wavelets, are depending on the network inputs. This wavelet neural network is tuned using proposed hybrid PSO. In this example, there are 3 inputs and 1 output. The inputs, z_i, of the network are defined as $z_1(t) = y_1^d(t-1)$, $z_2(t) = y_2^d(t-1)$ and $z_3(t) = y_3^d(t-1)$, where t denotes the year and $y_1^d(t)$ is the output which is sunspot numbers at the year t. The sunspot numbers of the first 230 years (i.e., $1703 \le t \le 1932$) are used to train the wavelet neural network. The output of the wavelet neural network used for sunspot forecasting is governed by [9]:

$$y_1(t) = \sum_{j=1}^{n_h} \psi_{j,b_j}(S_j) \cdot w_j\ , \tag{18}$$

where

$$S_j = \sum_{j=1}^{n_{in}} z_i(t) v_{ji}\ , \tag{19}$$

and

$$\psi_{j,b_j}(S_j) = \left(\frac{1}{\sqrt{j}} e^{\frac{-\left(\frac{S_j - b_j}{2}\right)^2}{2}}\right)\left(1 - \left(\frac{S_j - f^j(S_j)}{j}\right)^2\right), \tag{20}$$

where

$$f^j(S_j) = 4 \times j\left(\frac{2}{1 + e^{-\kappa^j \times S_j}} - 1\right). \tag{21}$$

v_{ji}, i=1, 2, …, n_{in}, j=1, 2, …, n_h denotes the weight of the link between i-th input node and j-th hidden node, n_{in} and n_h represent number of input and number of output respectively. In this example, n_{in} =3 and n_h =6. w_j denotes the weight of the link between j-th hidden node and single output. The tuned parameter of the network are v_{ji}, w_j and κ_j. Different PSO

methods (HPSOWM [8], HPSOM [1], HGAPSO [5], HGPSO [13], IPSO [4], and the proposed FPSOCM) are employed to tune the parameters of the network. The objective is to minimize the mean square error of the network. The objective function f_{obj} is defined as follow:

$$f_{obj} = \sum_{t=1703}^{1932} \frac{\left(y_1^d(t) - y_1(t)\right)^2}{230} \,. \tag{22}$$

The basic settings of the parameters of the PSOs are listed as follows:

-For All PSOs: i) Swarm Size γ: 50; ii) Number of runs: 50; iii) Acceleration constant φ_1 and φ_2: 2.05; iv) Maximum velocity $v_{\max}$: 0.2; v) Initial population is generated uniformly at random.

-For HPSOWM [8]: i) Parameter g of the wavelet mutation: 10000; ii) Shape parameter of the wavelet mutation: 2.

-For HGPSO [13]: i) Learning rate: 0.01.

-For HGAPSO [5]: i) Probability of crossover operation: 0.8.

In this application, the number of iteration is 500. The initial range of the v_{ji} and w_j are bounded between -5 and 5. The initial range of the κ_j is bounded between 0.3 and 1.5. The probability of mutation operation for HPSOWM, HPSOM and HGAPSO are set at 0.1. The total numbers of parameters is 37. The training results are tabulated in Table I, and the comparison between different PSO methods is shown in Fig. 3. The table shows that the performance of the FPSOCM is better than other PSO methods in term of the mean cost value, best cost value and the standard deviation. The mean cost value of FPSOCM is 0.83. The mean cost value of PSO methods with mutation operation (HPSOWM, HPSOM, and HGAPSO) is around 0.9 to 0.92. HSPSO performs poor because the gradient method cannot handle multimodel problem. Furthermore, with Fig. 4, FPSOCM gives a faster convergence rate. It reaches approximately 1.8 around 50 times of iteration, while the other PSO methods with mutation operation offer about 2.9. Furthermore, the *t*-values between FPSOCM and other PSO methods are higher than 1.645, and thus FPSOCM is significantly better with a 95% confidence level. The computational time of FPSOCM is near to that of the other PSO methods (HGPSO needs much more than because of the gradient processing). To evaluate the generalization ability of the PSO based neural network, the tuned network is used to forecast the sunspot number during the year 1933-1979. Table II shows the forecasting results of all PSO methods. It can be seen that the mean forecasting error of network tuned by FPSOCM is smaller. In short, the FPSOCM is a good tuning algorithm for neural network.

IV. Conclusion

In this paper, we have proposed a novel hybrid particle swarm optimization which incorporated with fuzzy inference system and a new cross-mutated operation. An adaptive inertia weight of PSO is presented. The value of the weight is determined by a set of fuzzy rules. Furthermore, a new operation namely cross-mutated operation is proposed which is used to force the particle escape the local optimum and push to global optimum. With these two new operations, the solution quality and solution reliability are improved. With numerical example on neural network application, FPSOCM are successful to tune the parameters of networks and outperform compared with other PSO methods.

TABLE I
COMPARISON THE TRAINING ERROR BETWEEN DIFFERENT PSO METHODS FOR FORECASTING OF SUNSPOT NUMBER USING WAVELET NEURAL NETWORK. ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS.

	FPSOCM	[8]	[1]	[5]	[13]	[4]
Mean	0.8313	0.9229	0.9030	0.9229	2.7220	0.9570
Best	0.7768	0.8156	0.8382	0.8013	0.9776	0.8697
Std Dev	0.0165	0.0521	0.0464	0.0668	1.8957	0.0779
t-value	N/A	11.86	10.30	9.42	7.05	11.16
Time (s)	21.531	22.422	22.500	22.266	47.437	21.45

TABLE II
COMPARISON THE TESTING ERROR BETWEEN DIFFERENT PSO METHODS FOR FORECASTING OF SUNSPOT NUMBER USING WAVELET NEURAL NETWORK. ALL RESULTS ARE AVERAGED ONES OVER 50 RUNS.

	FPSOCM	[8]	[1]	[5]	[13]	[4]
Mean	1.2965	1.4129	1.3575	1.4232	6.7050	1.5007
Best	0.9993	1.1785	1.1426	1.1454	1.4898	1.1574
Std Dev	0.2126	0.1641	0.2623	0.1807	5.2380	0.3522
t-value	N/A	3.07	1.28	3.21	7.30	3.51

References

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

[2] P.Angeline, "Using selection to improve particle swarm optimization," in *Proc. IEEE Congress on Evolutionary Computing*, Anchorahe, May 1998, pp. 84-89.

[3] T. J. Cholewo and J. M. Zurada, "Sequential network construction for time series prediction," in *Proc. Int. Conf. Neural Networks*, vol. 4, 1997, pp. 2034-2038.

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

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

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

[7] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligent*, Morgan Kaufmann Publishes, San Fransisco, USA, 2001.

[8] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," *IEEE Trans. Neural Nets*., vol. 14, no. 1, pp. 79-88, Jan 2003.

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

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

[11] B. Liu, L. Wang, and Y. H. Jin, "An effect PSO-based memetic algorithm for flow shop scheduling, *IEEE Trans. Syst. Man and Cyber. B*, vol. 37, no. 1, pp. 18-27, Feb 2007.

[12] R. Marinke, E. Araujo, Ld. S. Coelho, and I. Matiko, "Particle swarm optimization (PSO) applied to fuzzy modeling in a thermal-vacuum system," in *Proc. 5th Int. Cof. Hybrid Intelligent Systems*, Nov 2005, pp. 67-72.

[13] 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*, no. 1, vol. 3, pp. 476-483, May 2007.

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