Improved Hybrid Particle Swarm Optimized Wavelet Neural Network for Modeling the Development of Fluid Dispensing for Electronic Packaging

S. H. Ling, *Member, IEEE*, H. H. C. Iu, *Senior Member, IEEE*, F. H. F. Leung, *Senior Member, IEEE*, and K. Y. Chan

***Abstract*—An improved hybrid particle swarm optimization (PSO)-based wavelet neural network (WNN) for Modeling the development of Fluid Dispensing for Electronic Packaging (MFD-EP) is presented in this paper. In modeling the fluid dispensing process, it is important to understand the process behavior as well as determine the optimum operating conditions of the process for a high-yield, low-cost, and robust operation. Modeling the fluid dispensing process is a complex nonlinear problem. This kind of problem is suitable to be solved by applying a neural network. Among the different kinds of neural networks, the WNN is a good choice to solve the problem. In the proposed WNN, the translation parameters are variables depending on the network inputs. Due to the variable translation parameters, the network becomes an adaptive one that provides better performance and increased learning ability than conventional WNNs. An improved hybrid PSO is applied to train the parameters of the proposed WNN. The proposed hybrid PSO incorporates a wavelet-theory-based mutation operation. It applies the wavelet theory to enhance the PSO in more effectively exploring the solution space to reach a better solution. A case study of MFD-EP is employed to demonstrate the effectiveness of the proposed method.**

***Index Terms*—Modeling, particle swarm optimization (PSO), wavelet neural network (WNN), wavelet theory.**

I. Introduction

RECENTLY, a new kind of neural networks known as wavelet neural networks (WNNs), which combine feedforward neural networks (FFNNs) with wavelet theory [6], [7], has been proposed [1]–[5]. The wavelet theory provides a multiresolution approximation for discriminate functions. The WNN can thus exhibit better performance in function learning than the conventional FFNNs. Researchers have successfully applied WNNs in function approximation [1], motor drive control [2], [3], robotics [4], and power systems [5]. Using neural networks to achieve learning [9], [10] usually involves two steps, i.e., designing a network structure and deriving an algorithm for the learning process. The structure of the neural network governs the nonlinearity of the modeled function. The learning algorithm determines the rules for optimizing the weight values of the network within the training period. A typical WNN structure offers a fixed set of weights after the learning process. This single set of weights is used to capture the characteristics of all input data. However, a fixed set of weights may not be enough to learn the data set if the data are separately distributed in a vast domain and/or the number of network parameters is too small.

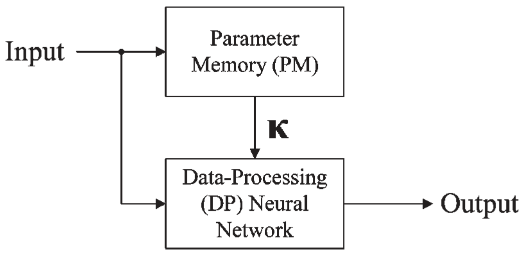


Fig. 1. Proposed architecture of the neural network.

In this paper, a variable translation WNN (VTWNN) is proposed. Wavelets are used as transfer functions in the hidden layer of the network. The network parameters, i.e., the translation parameters of the wavelets, are variable depending on the network inputs. Due to the variable translation parameters, the proposed VTWNN has the ability to model the input–output function with input-dependent network parameters. It works as if several individual neural networks are handling different sets of input data. Effectively, it becomes an adaptive network capable of handling different input patterns and exhibits a better performance. Fig. 1 shows the architecture of the proposed VTWNN, which consists of two units, namely, the parameter memory (PM) and the data processing (DP) neural network. The PM stores some parameters (κ) governing how the DP neural network handles the input data. By using this proposed neural network, some of the cases that cannot be handled by traditional neural networks with a limited number of parameters can now be tackled. To illustrate this point, Fig. 2 shows two sets of data (S1 and S2) separated far apart. (In practice,

Manuscript received August 20, 2007; revised March 3, 2008. Published August 29, 2008 (projected). This work was supported in part by The University of Western Australia and in part by The Hong Kong Polytechnic University under Project G-U414.

S. H. Ling is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576 (e-mail: elelsh@nus.edu.sg).

H. H. C. Iu is with the School of Electrical, Electronic, and Computer Engineering, University of Western Australia, Perth, WA 6009, Australia (e-mail: herbert@ee.uwa.edu.au).

F. H. F. Leung is with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: enfrank@inet.polyu.edu.hk).

K. Y. Chan is with the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: KitYan.Chan@inet.polyu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2008.922599

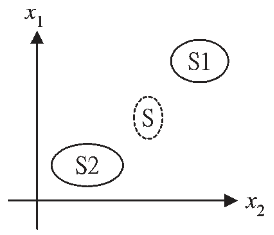


Fig. 2. Diagram showing two sets of data in a spatial domain.

even more data sets separated far apart can be present in a large domain.) If we model these data sets using a traditional neural network, the weights of the network are trained to minimize the error between the network output and the desired value. However, with a limited number of parameters, the network may only model the data set S instead, as shown in Fig. 2. In order to alleviate this problem, the VTWNN is proposed. Referring to Fig. 1, when the input data belong to S1, the PM will follow parameter set 1 to drive the DP neural network to handle the S1 data. Similarly, when the input data belong to S2, the parameters corresponding to S2 will be employed to drive the DP neural network to handle these input data.

Concerning the training of network parameters, one commonly used method is the gradient method [9]–[11], such as the Madaline Rule I (MRI), MRII, and MRIII rules, and the backpropagation technique, which adjusts the network parameters based on the gradient information of the fitness function in order to reduce the errors over all input patterns. Different backpropagation algorithms, such as backpropagation algorithms with momentum [10], backpropagation algorithms with variable learning rate [10], and conjugate gradient algorithm [11], have been proposed to improve the learning process. However, gradient methods may only converge to a local optimum and is sensitive to the values of the initial parameters. The function to be optimized needs to be differentiable, and the learning method may only be good to some specific network structure. Particle swarm optimization (PSO) [12] is one of the stochastic search algorithms. The error functions are less likely to be trapped in a local optimum and need not be differentiable or even continuous. Thus, PSO is more suitable for searching in a large, complex, nondifferentiable, and multimodal domain. It is a good training algorithm for neural or neural fuzzy networks [17]–[19] and fuzzy system [38]. The same PSO can be used to train many different networks, regardless of whether they are feedforward [9], [10], recurrent [9], wavelet [1], or other structure-type networks. This generally saves a lot of effort in developing the training algorithms for different types of networks. Lately, the real-life case studies of a neural network with stochastic search learning algorithm are developed [29]–[33]. PSO can solve multiobjective problems by using Pareto theory [34], [35] and penalty weight [36].

Recently, different hybrid PSOs have been proposed to overcome the drawback of possible trapping in the local optima. A new hybrid gradient descent PSO (HGPSO), which is integrated with gradient information to achieve faster convergence without getting trapped in the local minima, is proposed by Noel and Jannett [14]. However, the computational effort of the HGPSO is increased by the process of gradient descent. Juang [15] proposed a hybrid PSO algorithm named HGAPSO, which incorporates the genetic algorithm (GA)'s evolutionary operations of crossover, mutation, and reproduction into it. Ahmed *et al.* [13] proposed a hybrid PSO named HPSOM, in which a constant space is used for 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.

In GAs, the solution space is more likely to be explored in the early stage of the search by setting a larger mutating space, and it is more likely to be fine tuned for a better solution in the later stage of the search by setting a smaller mutating space based on the properties of the wavelet [6]. This idea can be applied when we introduce GA's mutation to the hybrid PSO. In this paper, a mutation with a dynamic mutating space by incorporating a wavelet function [6] is proposed. Wavelet is a tool to model seismic signals by combining dilations and translations of a simple oscillatory function (mother wavelet) of a finite duration. The PSO's mutating space is dynamically varying along the search based on the properties of the wavelet function.

Fluid dispensing is a manufacturing process by which fluid materials are delivered to substrates, boards, or work pieces in a controllable manner. This process is widely used in various packaging processes in the electronics and semiconductor manufacturing industry, such as integrated circuit encapsulation, die bonding, and surface mount technology. In the competitive market of today, this manufacturing process needs to be controlled at each of the many processing steps in the manufacturing line. The process directly affects the overall quality of the finished product as well as the throughput of the production line. All the variables controlling the desired outputs in a given process need to be understood and optimized for tight control. To achieve this, it is necessary to develop an accurate model for describing the process.

Li *et al.* [39] developed some analytical models to Model the Fluid Dispensing process for Electronic Packaging (MFD-EP). However, owing to the complex behavior of fluid dispensing and the high degree of uncertainties of the process in the real-world environment, an analytical model for fluid dispensing, which can provide accurate results, is generally difficult to be developed. Empirical modeling is another popular approach to developing process models using experimental data. Various techniques have been introduced in previous studies to develop process models based on an empirical modeling approach, such as applying statistical regression or artificial neural networks. The statistical regression method [40] is one of the most common empirical modeling techniques to develop process models. Statistical regression models are accurate over the range in which they are developed. As a result, statistical regression models can be applied only if the given data are distributed according to a statistical model, and the relation between dependent and independent variables is crisp. Ip *et al.* [41] proposed to use fuzzy regression to model the process, but the nonlinearity of the process was still not addressed.

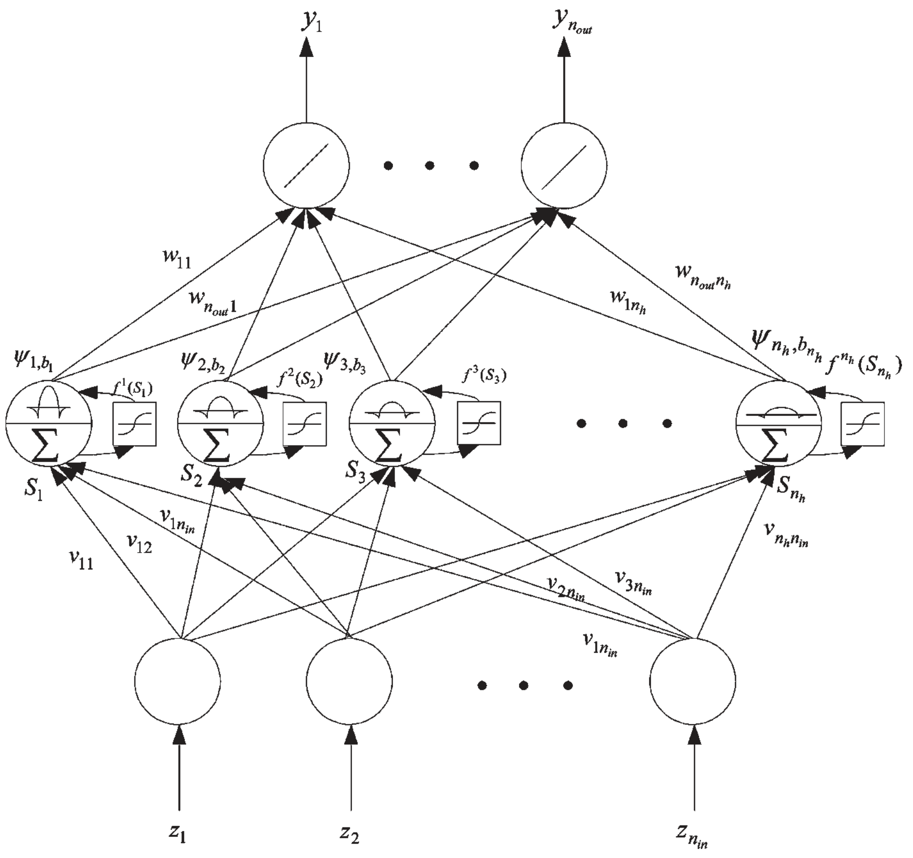


Fig. 3. Proposed VTWNN model.

Neural networks have been used to develop different models for various manufacturing processes, such as abrasive flow machining [23], classifying [24], machine condition monitoring [25], die casting [26], field programmable gate array [27], and switched reluctance motor [37]. Neural networks handle nonlinear mathematical models by using a black-box approach, providing a platform for the learning and generalization of the process nonlinearities. Kwong *et al.* [42] demonstrated that the classical neural network method can produce more accurate models for the epoxy dispensing process than the ones found by the statistical regression. In this paper, the VTWNN trained by the hybrid PSO with wavelet mutation (HPSOWM) is proposed for MFD-EP. By employing the proposed method on modeling the fluid dispensing process, smaller modeling errors with lower computational effort can be achieved as compared with other tested modeling methods.

This paper is organized as follows. The basic theory of wavelet is discussed in Section II. In Section III, the proposed VTWNN model is presented. The improved HPSOWM is discussed in Section IV. In addition, the training of the parameters of the proposed VTWNN using HPSOWM is presented. In Section V, the application on modeling the fluid dispensing process is given to show the merits of the proposed methodology. A conclusion is drawn in Section VI.

II. Basic Wavelet Theory

Certain seismic signals can be modeled by combining translations and dilations of an oscillatory function with a finite duration called a "wavelet." A continuous function $\psi(x)$ is a "mother wavelet" or "wavelet" if it satisfies the following properties.

Property 1:

$$\int_{-\infty}^{+\infty} \psi(x)dx = 0. \tag{1}$$

In other words, the total positive momentum of $\psi(x)$ is equal to the total negative momentum of $\psi(x)$.

Property 2:

$$\int_{-\infty}^{+\infty} |\psi(x)|^2 \, dx < \infty \tag{2}$$

where most of the energy of $\psi(x)$ is confined to a finite domain and is bounded.

In order to control the magnitude and position of $\psi(x)$, $\psi_{a,b}(x)$ is defined as

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right) \tag{3}$$

where a is the dilation parameter, and b is the translation parameter. It should be noted that $\psi_{a,b}(x)$ is scaled down as the dilation parameter a increases, and the location of the center of the wavelet is controlled by the translation parameter b.

III. Design and Analysis of VTWNN Model

In this section, the design and analysis of the VTWNN model will be presented. The WNN (WNN) can be considered as a particular case of FFNNs. The special point is that the transfer function of the WNN is a multiscaled wavelet function $\psi_{a,b}(x)$. In the proposed VTWNN, the translation parameter in the transfer function of the hidden nodes is variable and depends on the network inputs. With the variable translation parameters, the proposed VTWNN performs better and has a higher learning ability than the conventional WNN [1] and FFNN [10].

A. Design of Network Model

The proposed VTWNN has a three-layer structure with n_{in} nodes in the input layer, n_h nodes in the hidden layer, and n_{out} nodes in the output layer, as shown in Fig. 3. The input of the hidden layer S_j is given by

$$S_j = \sum_{i=1}^{n_{\text{in}}} z_i v_{ji}, \quad j = 1, 2, \ldots, n_h \tag{4}$$

where the z_i's, $i = 1, 2, \ldots, n_{\text{in}}$, are the input variables, and v_{ji} denotes the weight of the link between the ith input and the jth hidden nodes. In order to control the magnitude and the position of the wavelet, the multiscaled wavelet function $\psi_{a,b}(x)$ defined in (3) is used as the hidden node transfer function. The dilation parameter a of the first hidden node $(j = 1)$ is set as 1, i.e., $\psi_{1,b_1}(x) = \psi(x - b_1)$. For the second hidden node $(j = 2)$, the dilation parameter a is set as 2, i.e., $\psi_{2,b_2}(x) = (1/\sqrt{2})\psi((x - b_2)/2)$, where the output of the wavelet is scaled down by $1/\sqrt{2}$. Similarly, for the jth hidden node, the dilation parameter a is set as j. Hence, the output of the hidden layer of the proposed VTWNN is given by

$$\psi_{j,b_j} = \frac{1}{\sqrt{j}} \psi\left(\frac{S_j - b_j}{j}\right). \tag{5}$$

In this proposed network, the Maxican Hat function [36], as shown in Fig. 4, is used as the mother wavelet $\psi(x)$, which is defined as

$$\psi(x) = e^{-x^2/2}(1 - x^2). \tag{6}$$

Notice that $\psi(x)$ meets the requirements of Property 1 in (1) and Property 2 in (2). Referring to (5) and (6), we have

$$\psi_{j,b_j} = \frac{1}{\sqrt{j}} e^{\frac{-\left(\frac{S_j - b_j}{j}\right)^2}{2}} \left(1 - \left(\frac{S_j - b_j}{j}\right)^2\right). \tag{7}$$

The translation parameter b_j is set as a variable depending on the input S_j and is governed by a nonlinear function $f^j(\cdot)$ as

$$b_j = f^j(S_j). \tag{8}$$

We set

$$f^j(S_j) = 4j\left(\frac{2}{1 + e^{-\kappa_j \times S_j}} - 1\right) \tag{9}$$

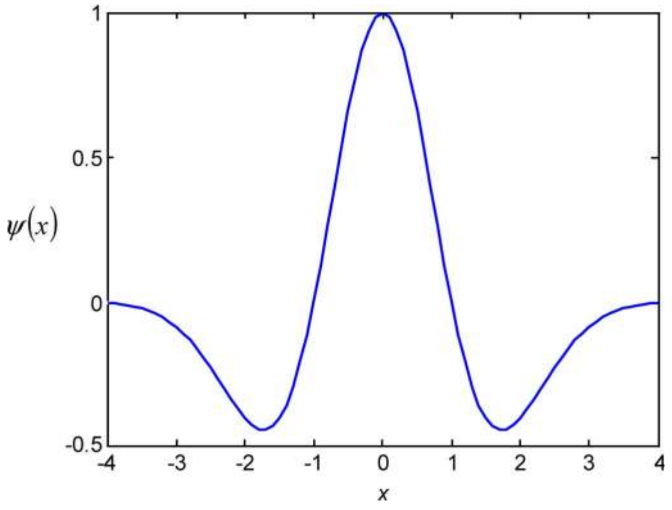


Fig. 4. Maxican Hat mother wavelet.

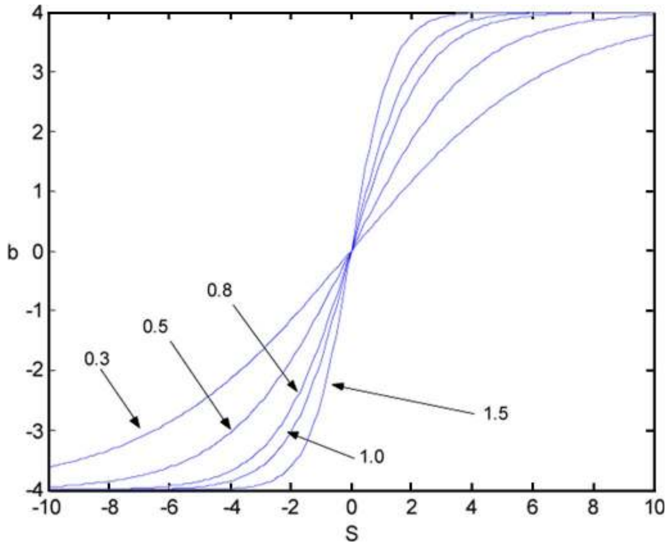


Fig. 5. Sample nonlinear functions with different values of parameter κ ($\kappa = 0.3, 0.5, 0.8, 1.0,$ and 1.5).

where κ_j is a tuned parameter that is used to control the shape of the nonlinear function $f^j(\cdot)$. The shape of $f^j(\cdot)$ with different κ_j is shown in Fig. 5. From (8) and (9), the value of the translation parameters b_j depends on the network inputs and the parameter κ_j. In other words, it operates such that the neural network will handle different input data with different network parameter b_j. Thus, the proposed VTWNN is an adaptive network.

The output of the proposed VTWNN is defined as

$$y_l = \sum_{j=1}^{n_h} \psi_{j,b_j}(S_j) \cdot w_{lj} \tag{10}$$

$$= \sum_{j=1}^{n_h} \psi_{j,b_j}\left(\sum_{i=1}^{n_{\text{in}}} z_i v_{ji}\right) \cdot w_{lj} \tag{11}$$

where w_{lj}, $j = 1, 2, \ldots, n_h$, and $l = 1, 2, \ldots, n_{\text{out}}$, denotes the weight of the link between the jth hidden and lth output

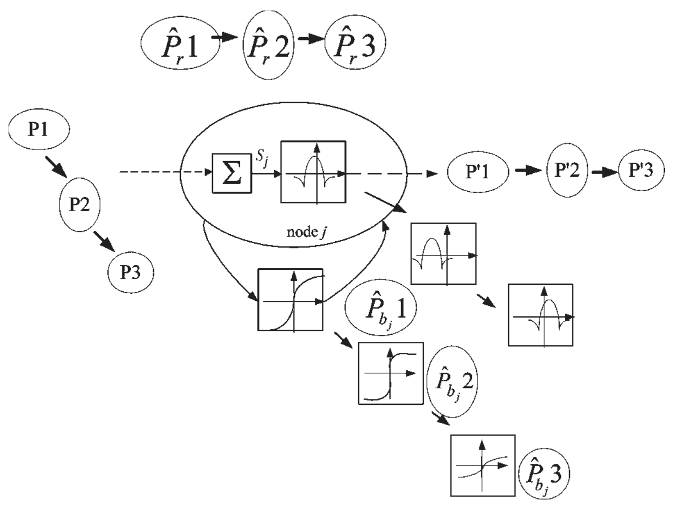


Fig. 6. Operation of the proposed neuron with three sets of data patterns.

nodes. The tuned parameters of the VTWNN are v_{ji}, w_{lj}, and κ_j. The number of parameters for v_{ji} is equal to $n_{\text{in}} \times n_h$; the number of parameters for w_{lj} is equal to $n_h \times n_{\text{out}}$; and the number of parameters for κ_j is equal to n_h. Thus, the total number of parameters of the proposed VTWNN is equal to $n_h(1 + n_{\text{in}} + n_{\text{out}})$.

B. *Interpretation of the Network*

Fig. 6 explains the operating principle of the proposed network and why it works well. In this figure, P1, P2, and P3 are three sets of input patterns. $\hat{P}_{b_j}1$, $\hat{P}_{b_j}2$, and $\hat{P}_{b_j}3$ are the input translation parameters for the corresponding input patterns. When the proposed neurons manipulate the input pattern P1, the shape of the wavelet transfer function is characterized by $\hat{P}_{b_j}1$, and the network eventually outputs the pattern P'1. Similarly, when the neurons manipulate the input pattern P2, the shape of the wavelet transfer function is characterized by $\hat{P}_{b_j}2$, and the network eventually outputs the pattern P'2. So, the activation function parameters are variable and dynamically dependent on the input patterns. Hence, the degree of freedom of the modeled function is increased. Comparing with the conventional wavelet and FFNNs, the VTWNN should be able to offer a better performance. All the parameters of the neural network can be tuned by an improved hybrid PSO that will be discussed in the next section.

C. *Parameter Design of VTWNN*

1) Number of Hidden Nodes (n_h)*:* The size of the hidden layer is a general question raised on designing multilayer FFNNs for real-life applications. An analytical method to determine the number of hidden nodes is difficult to obtain owing to the complexity of the network structure and the undetermined nature of the training process. Hence, the number of hidden nodes is experimentally found. In practice, the number of hidden nodes depends on the application and the dimension of the input space.

2) Parameter κ*:* The parameter κ is used to control the shape of the nonlinear function $f(\cdot)$ and governs the PM

```
begin
        $t \to 0$                      // iteration number
        Initialize $X(t)$       // $X(t)$: Swarm for iteration $t$
        Evaluate $f(X(t))$ // $f(\cdot)$: fitness function
while (not termination condition) do
        begin
        $t \to t+1$
        // Process of PSO //
        Update velocity $\mathbf{v}(t)$ and position of each particle $\mathbf{x}(t)$
        based on (1) and (2) respectively
          if $v(t) > v_{max}$
            $v(t) = v_{max}$
          end
          if $v(t) < -v_{max}$
            $v(t) = -v_{max}$
          end
        // End of the process of PSO //
          Reproduce a new $X(t)$
          Evaluate $f(X(t))$
        end
end
```

Fig. 7. Pseudocode for PSO.

```
begin
        $t \to 0$                      // iteration number
        Initialize $X(t)$       // $X(t)$: Swarm for iteration $t$
        Evaluate $f(X(t))$ // $f(\cdot)$: fitness function
while (not termination condition) do
        begin
        $t \to t+1$
        Perform the process of PSO (shown in Fig. 7)
        **Perform mutation operation**
        Reproduce a new $X(t)$
        Evaluate $f(X(t))$
        end
end
```

Fig. 8. Pseudocode for hybrid PSO with mutation operation.

(referring to Fig. 1). Fig. 5 shows the effect of the tuned parameter κ_j to b_j. In general, the range of κ is tuned within 0.3–1.5. We see that as $\kappa \to \infty$, the function reduces to a threshold function. Similarly, as $\kappa \to -\infty$, the function will become a constant line.

3) Network Parameters (Weights): A search method (hybrid PSO) is used to search the optimal values of the network parameters (weights) v_{ji} and w_{lj} in the VTWNN. The training process of the VTWNN with hybrid PSO is a minimization process of the error between the desired outputs and the actual ones.

IV. Training Network Parameters With HPSOWM

PSO is a novel optimization method developed by Kennedy and Eberhart [12]. It models the process of the sociological behavior associated with bird flocking and is one of the evolutionary computation techniques. It uses a number of particles that constitute a swarm. Each particle traverses the search space looking for the global optimum. The PSO process is shown in Fig. 7. In this paper, an HPSOWM is proposed and shown in Fig. 8. The details of both PSO and HPSOWM will be discussed as follows.

A. PSO

From Fig. 7, $X(t)$ denotes a swarm at the tth iteration. Each particle $\mathbf{x}^p(t) \in X(t)$ contains χ elements $x_j^p(t) \in \mathbf{x}^p(t)$ at the tth iteration, where $p = 1, 2, \ldots, \gamma$, and $j = 1, 2, \ldots, \chi$; γ denotes the number of particles in the swarm; and χ is the dimension of a particle. First, the particles of the swarm are initialized and then evaluated by a defined fitness function. The objective of PSO is to iteratively minimize the fitness values (cost values) of particles. The swarm evolves from iteration t to $t + 1$ by repeating the procedures as shown in Fig. 7. The PSO operations are discussed as follows.

The velocity $v_j^p(t)$ (corresponding to the flight speed in a search space) and the position $x_j^p(t)$ of the jth element of the pth particle at the tth generation can be calculated using [20]

$$v_j^p(t) = k \cdot \left\{ w \cdot v_j^p(t-1) + \varphi_1 \cdot rand() \cdot \left(pbest_j^p - x_j^p(t-1)\right) + \varphi_2 \cdot rand() \cdot \left(gbest_j - x_j^p(t-1)\right) \right\} \tag{12}$$

$$x_j^p(t) = x_j^p(t-1) + v_j^p(t) \tag{13}$$

where

$$pbest^p = \left[pbest_1^p \quad pbest_2^p \quad \cdots \quad pbest_\chi^p\right]$$
$$gbest = \left[gbest_1 \quad gbest_2 \quad \cdots \quad gbest_\chi\right]$$
$$j = 1, 2, \ldots, \chi.$$

The best previous position of the pth particle is recorded and represented as $pbest^p$; the best position of all the particles among all the times of iteration is represented as $gbest$; w is an inertia weight factor; φ_1 and φ_2 are acceleration constants; $rand()$ returns a uniform random number in the range of [0,1]; and k is a constriction factor derived from the stability analysis of (13) to ensure the system to be converged but not prematurely [8]. Mathematically, k is a function of φ_1 and φ_2 as reflected in

$$k = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{14}$$

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

PSO utilizes $pbest$ and $gbest$ to modify the current search point to avoid the particles moving in the same direction but to gradually converge toward $pbest$ and $gbest$. A suitable selection of the inertia weight w provides a balance between the global and local explorations. Generally, w can be dynamically set with [8]

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{T} \times t \tag{15}$$

where t is the current iteration number, T is the total number of iterations, and $w_{\max}$ and $w_{\min}$ are the upper and lower limits of the inertia weight, and are set to 1.2 and 0.1, respectively, in this paper.

In (12), 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 exploration 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%–20% of the dynamic range of the element on each dimension.

B. HPSOWM Operation

We observe that PSO [14] works well in the early stage but often presents problems on reaching the near-optimal solution. If a particle's current position coincides with the global best position, the particle will only move away from this point if its inertia weight and velocity are different from zero. If their velocities are very close to zero, then all the particles will stop moving once they catch up with the global best particle, which may lead to a premature convergence, and no further improvement can be obtained. This phenomenon is known as *stagnation* [21].

Ahmed *et al.* [13] proposed to integrate GA's mutation operation into PSO, which aids to break through *stagnation*. Here, we called this hybrid PSO the HPSOM. The mutation operation starts with a randomly chosen particle in the swarm and moves to different positions inside the search area by using the mutation. The following mutation operation is used in HPSOM:

$$mut(x_j) = x_j - \omega, \quad r < 0 \tag{16a}$$

$$mut(x_j) = x_j + \omega, \quad r \geq 0 \tag{16b}$$

where x_j is a randomly chosen element of the particle from the swarm, and ω is randomly generated within the range $[0, 0.1 \times (para_{\max}^j - para_{\min}^j)]$, representing one-tenth of the length of the search space; r is a random number between $+1$ and -1; and $para_{\max}^j$ and $para_{\min}^j$ are the upper and lower boundaries of each particle element. The pseudocode of the hybrid PSO with mutation operation is shown in Fig. 8, in which the mutation on particles is performed after updating their velocities and positions. It can also be seen from Figs. 7 and 8 that the pseudocodes of both PSO methods are identical except the mutation operation is introduced.

However, it can be noticed from (16) that the mutating space in HPSOM is limited by ω. It may not be the best approach in fixing the size of the mutating space all the time along the search. Instead, a dynamic mutation operation in which the mutating space dynamically contracts along the search can be considered. We propose a wavelet mutation (WM) that varies the mutating space based on the wavelet theory. The resulting HPSOWM is identical to HPSOM except for the mutation operation used. The proposed WM is discussed in the following section.

C. WM

The mutation operation is used to mutate the elements of particles. In general, various methods like uniform mutation or nonuniform mutation [22] can be employed to realize the mutation operation. The proposed WM operation exhibits a

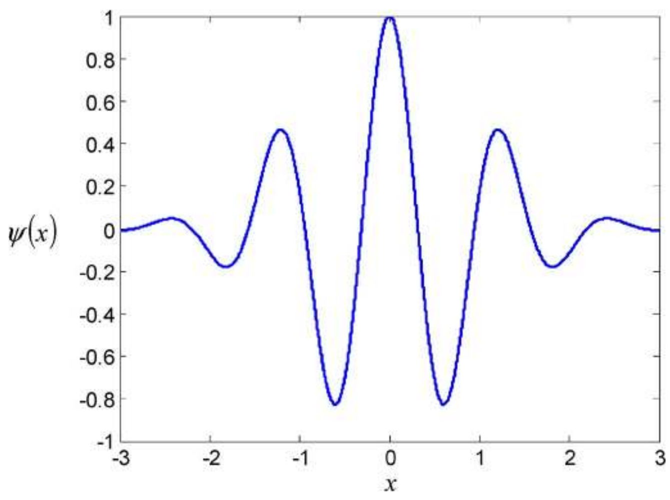


Fig. 9. Morlet wavelet.

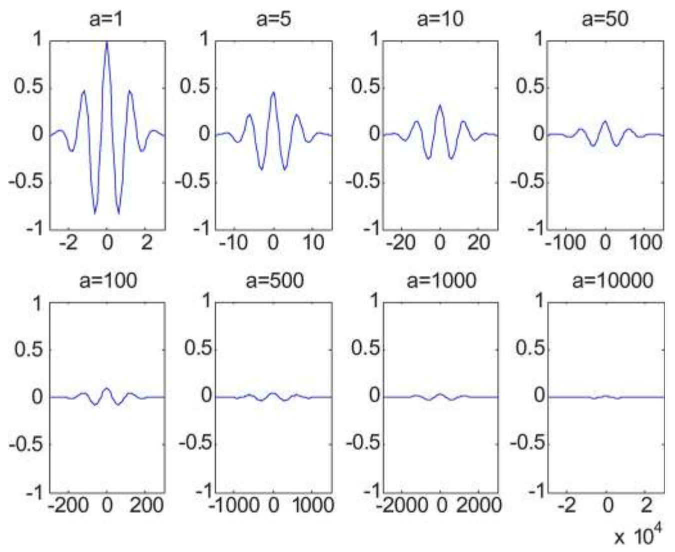


Fig. 10. Morlet wavelet dilated by different values of the parameter a (x-axis: x; y-axis: $\psi_{a,0}(x)$).

fine-tuning ability. The details of the operation are as follows. Every particle element of the swarm will have a chance to mutate, governed by a probability of mutation $p_m \in [0 \quad 1]$, which is defined by the user. For each particle element, a random number between 0 and 1 will be generated such that if it is less than or equal to p_m, the mutation will take place on that element. For instance, if $\mathbf{x}^p(t) = [x_1^p(t), x_2^p(t), \ldots, x_\chi^p(t)]$ is the selected pth particle, and the element of particle $x_j^p(t)$ is randomly selected for mutation (the value of $x_j^p(t)$ is inside the particle element's boundaries $[para_{\min}^j, para_{\max}^j]$), the resulting particle is given by $\bar{\mathbf{x}}^p(t) = [\bar{x}_1^p(t), \bar{x}_2^p(t), \ldots, \bar{x}_\chi^p(t)]$ and

$$\bar{x}_j^p(t) = \begin{cases} x_j^p(t) + \sigma \times \left(para_{\max}^j - x_j^p(t)\right), & \text{if } \sigma > 0 \\ x_j^p(t) + \sigma \times \left(x_j^p(t) - para_{\min}^j\right), & \text{if } \sigma \le 0 \end{cases} \tag{17}$$

where

$$\sigma = \psi_{a,0}(\varphi) \tag{18}$$

$$= \frac{1}{\sqrt{a}}\psi\left(\frac{\varphi}{a}\right). \tag{19}$$

By using the Morlet wavelet (as shown in Fig. 9) as the mother wavelet, we have

$$\psi(x) = e^{-x^2/2}\cos(5x) \tag{20}$$

$$\sigma = \frac{1}{\sqrt{a}} e^{-\left(\frac{\varphi}{a}\right)^2/2} \cos\left(5\left(\frac{\varphi}{a}\right)\right). \tag{21}$$

Fig. 10 shows different dilations of the Morlet wavelet. The amplitude of $\psi_{a,0}(x)$ will be scaled down as the dilation parameter a increases. This property is used to do the mutation operation in order to enhance the searching performance.

According to (17), if σ is positive approaching 1, the mutated element of the particle will tend to the maximum value of $x_j^p(t)$. Conversely, when σ is negative approaching -1, the mutated element of the particle will tend to the minimum value of $x_j^p(t)$. A larger value of $|\sigma|$ gives a larger searching space for $x_j^p(t)$. When $|\sigma|$ is small, it gives a smaller searching space for fine tuning. Referring to Property 1 of the wavelet, the sum of the positive σ is equal to the sum of the negative σ when the number of samples is large and φ is randomly generated. That is

$$\frac{1}{N}\sum_N \sigma = 0 \text{ for } N \to \infty \tag{22}$$

where N is the number of samples.

Hence, the overall positive mutation and the overall negative mutation throughout the evolution are nearly the same. This property gives better a solution stability (smaller standard deviation of the solution values upon many trials). As over 99% of the total energy of the mother wavelet function is contained in the interval $[-2.5, 2.5]$, φ can be randomly generated from $[-2.5, 2.5] \times a$. The value of the dilation parameter a is set to vary with the value of t/T in order to meet the fine-tuning purpose, where T is the total number of iteration, and t is the current number of iteration. In order to perform a local search when t is large, the value of a should increase as t/T increases to reduce the significance of the mutation. Hence, a monotonic increasing function governing a and t/T is proposed as

$$a = e^{-\ln(g)\times\left(1-\frac{t}{T}\right)^{\zeta_{wm}}+\ln(g)} \tag{23}$$

where ζ_{wm} is the shape parameter of the monotonic increasing function, and g is the upper limit of the parameter a. The effects of the various values of the shape parameter ζ_{wm} to a with respect to t/T are shown in Fig. 11. In this figure, g is set as 10 000. Thus, the value of a is between 1 and 10 000. Referring to (21), the maximum value of σ is 1 when the random number of $\varphi = 0$ and $a = 1(t/T = 0)$. Then, referring to (17), the element of particle $\bar{x}_j^p(t) = x_j^p(t) + 1 \times (para_{\max}^j - x_j^p(t)) = para_{\max}^j$. It ensures that a large search space for the mutated element is given. When the value t/T is near 1, the value of a is so large that the maximum value of σ will become very small. For example, at $t/T = 0.9$ and $\zeta_{wm} = 1$, the dilation parameter $a = 4000$; if the random value of φ is 0, the value of σ is 0.0158. With $\bar{x}_j^p(t) = x_j^p(t) + 0.0158 \times (para_{\max}^j - x_j^p(t))$, a small searching space for the mutated element is given for fine tuning.

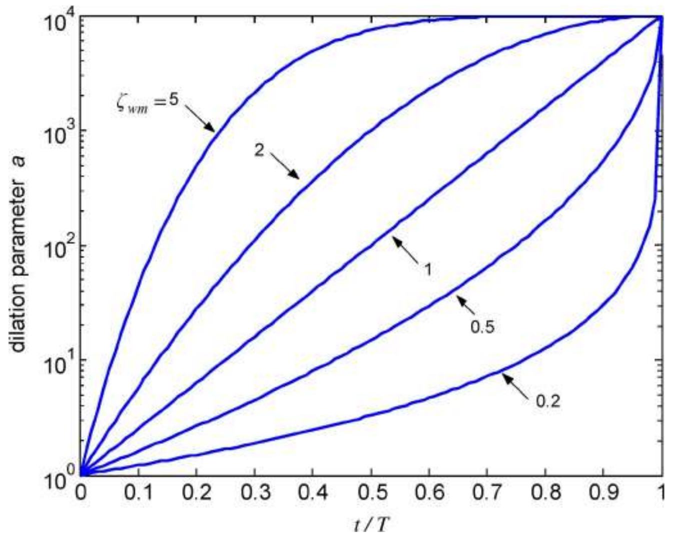


Fig. 11. Effect of the shape parameter ζ_{wm} to a with respect to t/T.

After the operation of WM, a new swarm is generated. This new swarm will repeat the same process. Such an iterative process will be terminated if a predefined number of iteration has been met.

D. Choosing the HPSOWM Parameters

HPSOWM is seeking 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 the HPSOWM, is governed by the right choices of the control parameters, e.g., swarm size (γ), the probability of mutation (p_m), and the shape parameter of WM (ζ_{wm}). Some views about these parameters are given as follows.

1) Increasing the swarm size (γ) will increase the diversity of the search space and reduce the probability that the HPSOWM prematurely converges to a local optimum. However, it also increases the time required for the population to converge to the optimal region in the search space.
2) Increasing the probability of mutation (p_m) tends to transform the evolutionary search into a random search such that when $p_m = 1$, all genes will mutate. This probability gives us an expected number $(p_m \times \gamma \times \chi)$ of elements of particles that undergo mutation. In other words, the value of p_m depends on the desired number of element of particles that undergo the mutation operation.
3) Changing the parameter ζ_{wm} will change the characteristics of the monotonic increasing function of WM. The dilation parameter a will take a value to perform fine tuning faster as ζ_{wm} is increasing. It is chosen by trial and error, which depends on the kind of optimization problem. When ζ_{wm} becomes larger, the decreasing speed of the step size (σ) of the mutation becomes faster. In general, if the optimization problem is smooth and symmetric, it is easier to find the solution, and the fine tuning can be done in early iteration. Thus, a larger value of ζ_{wm} can be used to increase the step size of the early mutation.

E. Tuning of the Network Parameters

One superior characteristic of the hybrid PSO is that the detailed information of the nonlinear system, e.g., the derivative information of the fitness function, is not necessarily known. Hence, it is suitable to handle some complex optimization problems. In this paper, the proposed HPSOWM is employed to optimize a fitness function, which is characterized by the parameters of the VTWNN. The fitness function is a mathematical expression that quantitatively measures the performance of the hybrid PSO tuning process. A larger fitness value indicates a better tuning performance. By adjusting the values of the network parameters, the fitness value is maximized by using the hybrid PSO. During the tuning process, particles with better fitness values are reproduced. In addition, the effect of the proposed mutation operation gradually decreases in the search domain with respect to the iteration number. This helps the convergence of the searching process of the network parameters. After the tuning process, the obtained network parameter values will be used by the proposed neural network. As the proposed neural network is a feedforward one, the outputs are bounded if its inputs are bounded, which happens for most of the real-life applications. Consequently, no convergence problem is present for the neural network itself.

The proposed VTWNN can be used to learn the input–output relationship of an application using the hybrid PSO. The input–output relationship can be described by

$$\mathbf{y}^d(t) = g\left(\mathbf{z}^d(t)\right), \quad t = 1, 2, \ldots, n_d \tag{24}$$

where $\mathbf{z}^d(t) = [z_1^d(t) \quad z_2^d(t) \quad \cdots \quad z_{n_{\text{in}}}^d(t)]$ and $\mathbf{y}^d(t) = [y_1^d(t) \quad y_2^d(t) \quad \cdots \quad y_{n_{\text{out}}}^d(t)]$ are the given inputs and the desired outputs of an unknown nonlinear function $g(\cdot)$, respectively, and n_d denotes the number of input–output data pairs. The fitness function of the PSO depends on the application, which is defined as

$$fitness = err. \tag{25}$$

In (25), err can be the mean square error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), etc., where MSE is defined as

$$err = \frac{\sum_{t=1}^{n_d}\sum_{k=1}^{n_{\text{out}}}\left(y_k^d(t) - y_k(t)\right)^2}{n_d n_{\text{out}}} \tag{26a}$$

MAE is defined as

$$err = \frac{\sum_{t=1}^{n_d}\sum_{k=1}^{n_{\text{out}}}\left|y_k^d(t) - y_k(t)\right|}{n_d n_{\text{out}}} \tag{26b}$$

and MAPE is defined as

$$err = \frac{\sum_{t=1}^{n_d}\sum_{k=1}^{n_{\text{out}}}\frac{\left|y_k^d(t) - y_k(t)\right|}{y_k^d(t)}}{n_d n_{\text{out}}}. \tag{26c}$$

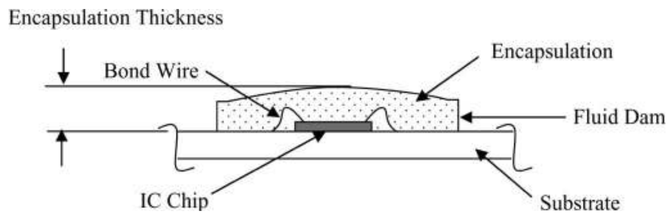


Fig. 12. Encapsulation of microchip.

The objective is to minimize the fitness value of (25) using the PSO by coding the particles of the swarm to be $[v_{ji} \quad w_{lj} \quad \kappa_j]\ \forall\, i,j,l$. The fitness value of (25) $\in [0,1]$.

V. MFD-EP

In this paper, modeling the fluid dispensing for microchip encapsulation is studied. Normally, silicon chips are covered using an X–Y numerically controlled dispensing system that delivers fluid encapsulant through a needle. The material is commonly dispensed in a pattern, working from the center out. A fluid dam around the die site and second wire bond points can be made to contain the flow material and make a uniform-looking part, as shown in Fig. 12.

Modeling the fluid dispensing process is critical for understanding the process behavior and achieving process optimization. The process is controlled by a number of parameters such as the diameter of the needle, the temperature of the epoxy, compressed air pressure, the viscosity of the epoxy resin, the pump motor speed, the distance between the needle and the substrate, the substrate temperature, and the path of dispensing. Based on the advice from the supporting company, three significant process parameters, namely, the compressed air pressure, the distance between the needle and the substrate, and the pump motor speed, were studied. The process parameter "compressed air pressure" refers to the pressure of the compressed air imposing on the epoxy resin, which is in the storage device of a dispensing system. The "pump motor speed" is to control the amount of epoxy to be dispensed. The "distance between the needle and the substrate" is controlled by using a stepping motor. Therefore, the distance is specified as the number of steps in the parameter setting. The three process parameters and their corresponding normal operating ranges are shown as follows:

1) compressed air pressure (1–4 bar) x_1;
2) pump motor speed (400–1000 r/min) x_2;
3) distance between substrate and needle (250–2000 steps) x_3.

Two quality characteristics (named as the outcome variables) were also identified, i.e., encapsulation weight (in milligrams) y and encapsulation thickness (in millimeters) z.

Ninety-six experiments are carried out based on a full factorial design with four levels in compressed air pressure (x_1), six levels in pump motor speed (x_2), and four levels in the height between the substrate and the needle (x_3).

A. Modeling With WNN

A VTWNN is used to model the fluid dispensing process. Its structure, shown in Fig. 3, consists of an input layer in which the input vectors (i.e., the process parameters x_1, x_2, and x_3) are fed, an output layer that produces the output response (either one of the quality characteristics y or z), and one hidden layer in between.

According to (11), the input–output relationship of the proposed three-layer neural networks for the encapsulation weight y and the encapsulation thickness z can be written as

$$y = \sum_{j=1}^{n_h} \psi_{j,b_j}\left(\sum_{i=1}^{3} x_i v_{ji}\right)\cdot w_{1j} \tag{27}$$

$$z = \sum_{j=1}^{n'_h} \psi'_{j,b'_j}\left(\sum_{i=1}^{3} x'_i v'_{ji}\right)\cdot w'_{1j} \tag{28}$$

where n_h (or n'_h) denotes the number of hidden nodes; w_{1j} (or w'_{1j}), $j = 1,2,\ldots,n_h$ (or n'_h), denotes the weight of the link between the jth hidden node and the output node; v_{ji} (or v'_{ji}), $i = 1,2,3$, and $j = 1,2,\ldots,n_h$ (or n'_h), denotes the weight between the ith input node and the jth hidden node; ψ_{j,b_j} (or ψ'_{j,b'_j}) denotes the wavelet function; and x_i (or x'_i) denotes the input data for the encapsulation weight and the encapsulation thickness, respectively.

To develop the neural-network-based model for the fluid dispensing process, the values of the neural network parameters (i.e., v_{ji}, w_{1j}, and κ_j with $i = 1,2,3$, and $j = 1,2,\ldots,n_h$) and the number of hidden nodes (n_h) used in the hidden layer need to be determined. These two settings are important because they affect the prediction accuracy of the neural-network-based process model.

To tune the network, we use hybrid PSO to minimize the MSE by setting the swarm particle to be $[v_{ji} \quad w_{1j} \quad \kappa_j]$ for all i and j. The MSEs for the encapsulation weight y and the encapsulation thickness z are defined as

$$\mathrm{MSE}_y = \frac{\sum_{k=1}^{n_{\mathrm{pat}}} \left(d_k^y - y_k\right)^2}{n_{\mathrm{pat}}} \tag{29}$$

$$\mathrm{MSE}_z = \frac{\sum_{k=1}^{n_{\mathrm{pat}}} \left(d_k^z - z_k\right)^2}{n_{\mathrm{pat}}} \tag{30}$$

where d_k^y and d_k^z denote the desired value of the encapsulation weight y and the encapsulation thickness z, respectively, and n_{pat} denotes the number of patterns. After training, the values of these network parameters will be fixed during the operation.

B. Results and Analysis

To illustrate the performance of the proposed method to this industrial application, a tenfold cross validation is considered. Cross validation [33] is the statistical practice of partitioning a sample of data into subsets such that the analysis is initially performed on a single subset, whereas the other subsets are

TABLE I
MEAN COST VALUES UNDER DIFFERENT VALUES OF THE SHAPE PARAMETER OF WM FOR ENCAPSULATION WEIGHT AND ENCAPSULATION THICKNESS (TRAINED WITH VTWNN)

	$\zeta_{wm}=0.2$	$\zeta_{wm}=0.5$	$\zeta_{wm}=1.0$	$\zeta_{wm}=2.0$	$\zeta_{wm}=5.0$
ENCAPSULATION WEIGHT	4.0212	3.9411	3.9126	**3.6492**	3.9303
ENCAPSULATION THICKNESS	0.5791×10^{-3}	0.5682×10^{-3}	0.5406×10^{-3}	$\mathbf{0.5251\times10^{-3}}$	0.5530×10^{-3}

TABLE II
COMPARISON BETWEEN DIFFERENT NEURAL NETWORK TOPOLOGIES WITH DIFFERENT PSO METHODS FOR MFD-EP (TRAINING). (a) ENCAPSULATION WEIGHT. (b) ENCAPSULATION THICKNESS. ALL RESULTS ARE AVERAGED OVER 50 RUNS (RANK: 1—BEST, 5—WORST)

(a)

T=2000		HPSOWM	HPSOM	HGAPSO	HGPSO	PSO
VTWNN	Mean	**3.6492**	5.7990	5.9217	9.3027	7.1167
	Best	**2.6033**	4.7461	4.3837	6.3027	5.5835
	Std Dev	0.1857	0.7673	0.7207	3.2056	0.9848
	t-value	N/A	19.26	21.59	12.45	24.47
	Run Time (s)	154.44	152.61	158.75	300.41	151.03
	Rank	1	2	3	5	4

T=2000		HPSOWM	HPSOM	HGAPSO	HGPSO	PSO
WNN	Mean	5.3277	6.1218	6.3601	10.5162	7.8121
	Best	4.4550	4.9230	4.9938	7.1204	5.9275
	Std Dev	0.1671	0.7110	0.7225	2.9121	0.9333
	t-value	N/A	7.69	9.84	12.58	18.53
	Run Time (s)	145.78	145.86	145.55	297.47	144.54
	Rank	1	2	3	5	4

T=2000		HPSOWM	HPSOM	HGAPSO	HGPSO	PSO
FFNN	Mean	6.3729	9.8213	8.7971	11.3102	9.6211
	Best	5.1207	5.8296	5.2797	8.2130	6.2011
	Std Dev	0.5987	1.7287	1.8020	4.7183	1.9156
	t-value	N/A	13.33	9.03	7.34	11.44
	Run Time (s)	101.76	100.56	101.36	206.94	100.03
	Rank	1	3	2	5	4

(b)

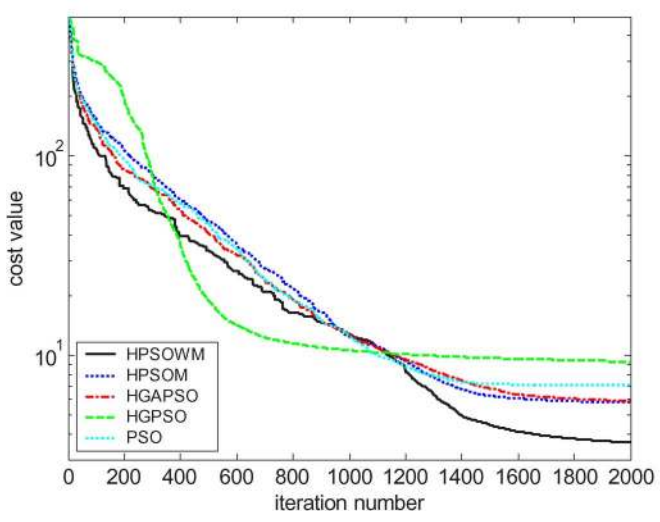
T=2000		HPSOWM	HPSOM	HGAPSO	HGPSO	PSO
VTWNN	Mean ($\times10^{-3}$)	**0.5251**	0.6171	0.6026	1.1310	0.6671
	Best ($\times10^{-3}$)	**0.4420**	0.5439	0.5488	0.6033	0.5814
	Std Dev ($\times10^{-3}$)	0.0322	0.0782	0.0703	0.1812	0.0588
	t-value	N/A	7.69	7.09	23.28	14.98
	Run Time (s)	167.33	166.52	167.17	345.66	165.79
	Rank	1	3	2	5	4

T=2000		HPSOWM	HPSOM	HGAPSO	HGPSO	PSO
WNN	Mean ($\times10^{-3}$)	0.6728	0.7323	0.7518	0.9035	0.8082
	Best ($\times10^{-3}$)	0.5012	0.6086	0.6096	0.6187	0.6267
	Std Dev ($\times10^{-3}$)	0.0238	0.0340	0.0512	0.1429	0.0559
	t-value	N/A	10.14	9.89	11.26	15.76
	Run Time (s)	150.88	148.12	149.37	305.16	147.61
	Rank	1	2	3	5	4

T=2000		HPSOWM	HPSOM	HGAPSO	HGPSO	PSO
FFNN	Mean ($\times10^{-3}$)	0.8542	0.9622	0.9127	1.1920	0.9452
	Best ($\times10^{-3}$)	0.6016	0.6299	0.6330	0.6071	0.6426
	Std Dev ($\times10^{-3}$)	0.0883	0.1609	0.1586	0.9722	0.1533
	t-value	N/A	4.16	2.28	2.45	3.64
	Run Time (s)	112.23	111.18	110.98	216.13	110.02
	Rank	1	4	2	5	3

retained for subsequent use in confirming and validating the initial analysis. In this paper, 96 experimental data of the encapsulation weight and encapsulation thickness are used. In the tenfold cross validation, the experimental data (samples) are partitioned into ten subsamples. Nine subsamples are used for training, and one subsample is used for testing (validation). The cross-validation process is then repeated ten times, with each of the ten subsamples used exactly once as the validation data. The ten results can be averaged to produce an average training and testing (validation) results.

For comparison purposes, a WNN [1] and a traditional FFNN [10] trained by HPSOWM, HPSOM [13], HGAPSO [15],



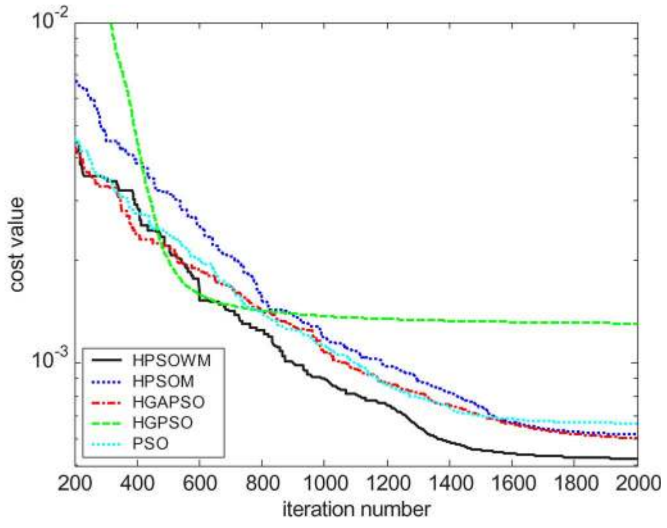
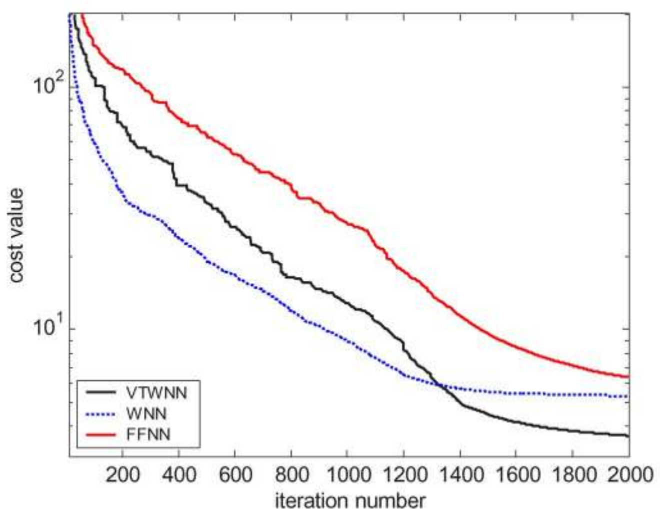


Fig. 13. (a) Comparisons between different PSO methods using VTWNN for MFD-EP (encapsulation weight). (b) Comparisons between different PSO methods using VTWNN for MFD-EP (encapsulation thickness).



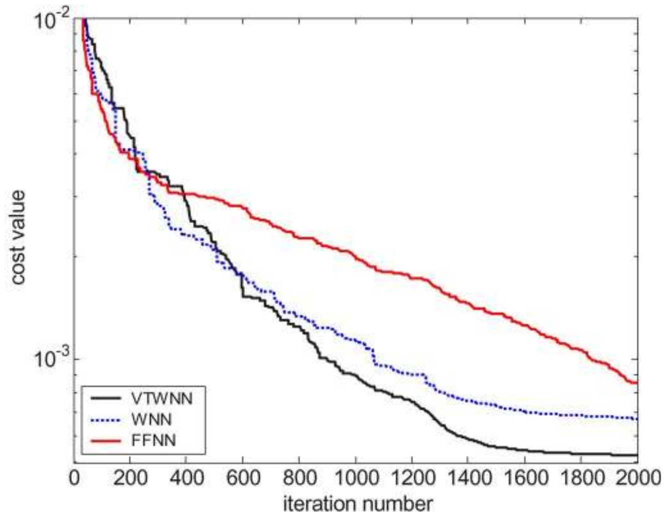


Fig. 14. (a) Comparisons between different neural network methods using HPSOWM for MFD-EP (encapsulation thickness). (b) Comparisons between different neural network methods using HPSOWM for MFD-EP (encapsulation weight).

HGPSO [14], and PSO [16] are also used to model the MFD-EP system. The basic settings of the parameters of the PSOs and the neural networks are shown as follows.

- Shape parameter of the WM (ζ_{wm}): 2. It is chosen by trial and error through experiments for good performance. The experimental result is given in Table I. In this table, different ζ_{wm}'s ($\zeta_{wm} = 0.2, 0.5, 1, 2,$ and 5) are tested. The best results are obtained when $\zeta_{wm} = 2$. Referring to Fig. 11, HPSOWM will go to perform fine tuning fairly fast when ζ_{wm} is set at 2. In this application, the sensitivity of this parameter is not high.
- Acceleration constants φ_1 and φ_2: 2.05 [16].
- Maximum velocity $v_{\max}$: 0.2 [16].
- Swarm size (γ): 50.
- Number of runs: 2000.
- Probability of mutation for HPSOWM, HPSOM, and HGAPSO (p_m): 0.1. It is chosen by trial and error through experiments for good performance. Different p_m's ($p_m = 0.01, 0.05, 0.1, 0.2,$ and 0.5) are tested.
- Probability of crossover for HGAPSO (p_c): 0.8.
- Initial population: it is generated uniform at random.
- Learning rate of HGPSO: 0.001.
- The initial ranges of the weights of the neural networks (VTWNN, WNN, and FFNN) for the encapsulation weight and the encapsulation thickness are bounded between -4 and 4.
- The initial ranges of κ_j for VTWNN are bounded between 0.1 and 1.5.
- The number of hidden nodes (n_h) of the neural network for the encapsulation weight and the neural network for the encapsulation thickness are set at 5 and 7, respectively.

The average training result comparisons between different neural network topologies (VTWNN, WNN, and FFNN) trained with different PSOs (HPSOWM, HPSOM, HGAPSO, HGPSO, and PSO) for the encapsulation weight and encapsulation thickness are tabulated in Table II(a) and (b), respectively. The convergence rates of different PSO methods using VTWNN for the encapsulation weight and thickness are

TABLE III
COMPARISON BETWEEN DIFFERENT NEURAL NETWORK TOPOLOGIES WITH DIFFERENT PSO METHODS FOR MFD-EP (TESTING). (a) ENCAPSULATION WEIGHT. (b) ENCAPSULATION THICKNESS. ALL RESULTS ARE AVERAGED OVER 50 RUNS (PSO1: HPSOWM, PSO2: HPSOM, PSO3: HGAPSO, PSO4: HGPSO, AND PSO5: PSO)

(a)

	VTWNN					WNN					FFNN				
	PSO1	PSO2	PSO3	PSO4	PSO5	PSO1	PSO2	PSO3	PSO4	PSO5	PSO1	PSO2	PSO3	PSO4	PSO5
Mean error	**6.41**	9.33	9.96	18.06	11.31	7.76	10.82	11.01	19.56	12.83	11.83	15.02	16.40	20.72	18.01
Std Dev	0.2711	1.4873	1.4902	5.8821	0.5621	0.4920	1.4930	1.7032	3.9842	1.5021	1.5832	3.6273	3.3322	11.2129	3.6250

(b)

	VTWNN					WNN					FFNN				
	PSO1	PSO2	PSO3	PSO4	PSO5	PSO1	PSO2	PSO3	PSO4	PSO5	PSO1	PSO2	PSO3	PSO4	PSO5
Mean error ($\times10^{-3}$)	**0.5280**	0.6231	0.6175	1.6821	0.7243	0.6892	0.7516	0.8043	1.0637	0.8102	0.8901	1.2631	1.0786	1.5525	1.1757
Std Dev ($\times10^{-3}$)	0.0454	0.0672	0.0623	0.3086	0.9231	0.0487	0.1623	0.2031	0.3965	0.2010	0.1678	0.3570	0.3065	1.4087	0.4503

given in Fig. 13. The comparisons between the different neural network topologies for encapsulation weight and thickness are given in Fig. 14. In these tables and figures, the convergence rate, mean value, best value, standard deviation, t-value, run time in second, and the rank are given. Comparing different PSO methods, the proposed HPSOWM provides a better solution quality and solution stability, and these improvements are reflected by the average mean cost values, the t values, and the standard deviations (a smaller standard deviation implies a more stable solution). Due to the wavelet properties, the stability of the results is improved. The t value for all approaches is larger than 2.15 (degree of freedom $= 49$), meaning that there is a significant difference between the HPSOWM and other PSOs with a 98% confidence level. Furthermore, comparing the run time, we can see that the HGPSO consumes more time. It is because the computational effort is increased by the process of gradient descent. The other methods consume almost the same amount of time. In general, PSO with mutation operations (HPSOWM and HPSOM) are better than other hybrid PSOs in terms of mean values. With the results shown in Fig. 13 and Tables II and III, we can see that HPSOWM gives better convergence and performance. It is because HPSOWM provides a fine-tuning ability in the later stage, whereas a large search step of mutation operation is given in the early stage. Comparing different neural network topologies, the VTWNN gives a better performance. The average mean error of the VTWNN trained with HPSOWM for the encapsulation weight is 3.6492, which implies 45% and 75% improvement over WNN (trained with HPSOWM) and FFNN (trained with HPSOWM), respectively. Similarly, the average mean error of the VTWNN trained with HPSOWM for the encapsulation thickness is 0.5251×10^3, which implies 28% and 63% improvement over WNN (trained with HPSOWM) and FFNN (trained with HPSOWM), respectively. VTWNN gives a better performance because the translation parameters of the WNN are variable. This makes the VTWNN capable of modeling the process with input-dependent network parameters. It works as if several individual networks are handling different sets of input data, which exhibit better performance. In order to test the generalization ability of the proposed network, the tenfold cross-validation process is used. The results in terms of the mean testing error and standard deviation are tabulated in Table III. In this table, the proposed VTWNN trained with HPSOWM gives a better testing result as compared with others.

VI. CONCLUSION

In this paper, an improved hybrid PSO has been proposed to optimize the parameters of the VTWNN. Due to the variable translation parameters in the network, the proposed VTWNN becomes adaptive and is able to improve the learning ability of the neural network. In addition, a hybrid PSO incorporated with WM has been proposed by applying the properties of the wavelet theory to enhance the PSO to explore the solution space more effectively on reaching the optimal solution. An industrial application on MFD-EP using the proposed PSO-based WNN has been discussed. Experimental results have been given to show the improved performance and solution stability of the VTWNN and the WM-based hybrid PSO. One limitation in this paper is that the choice of suitable parameters for the PSO and the neural network is quite difficult. Most parameters are determined by trial and error through experiments. Some possible future research directions can be identified. Multiobjective PSO could be studied, which are particularly good to handle some multiobjective optimization problems. In addition, dynamic mutation and shape parameter rate could be studied to reduce the time for selecting a suitable rate for application. Physical experiments will be conducted in order to further validate the effectiveness of the proposed predicted model.

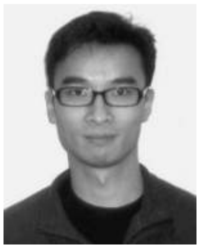
REFERENCES

[1] S. Yao, C. J. Wei, and Z. Y. He, "Evolving wavelet neural networks for function approximation," *Electron. Lett.*, vol. 32, no. 4, pp. 360–361, Feb. 1996.
[2] R. J. Wai and J. M. Chang, "Implementation of robust wavelet-neural-network sliding-mode control for induction servo motor drive," *IEEE Trans. Ind. Electron.*, vol. 50, no. 6, pp. 1317–1334, Dec. 2003.

[3] R. J. Wai, R. Y. Duen, J. D. Lee, and H. H. Chang, "Wavelet neural network control for induction motor drive using sliding-mode design technique," *IEEE Trans. Ind. Electron.*, vol. 50, no. 4, pp. 733–748, Aug. 2003.
[4] S. J. Yoo, J. B. Park, and Y. H. Choi, "Adaptive dynamic surface control of flexible-joint robots using self-recurrent wavelet neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1342–1355, Dec. 2006.
[5] Y. C. Huang and C. M. Huang, "Evolving wavelet networks for power transformer condition monitoring," *IEEE Trans. Power Del.*, vol. 17, no. 2, pp. 412–416, Apr. 2002.
[6] I. Daubechies, "The wavelet transform, time–frequency localization and signal analysis," *IEEE Trans. Inf. Theory*, vol. 36, no. 5, pp. 961–1005, Sep. 1990.
[7] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 674–693, Jul. 1989.
[8] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. Congr. Evol. Comput.*, Jul. 2000, vol. 1, pp. 84–88.
[9] F. M. Ham and I. Kostanic, *Principles of Neurocomputing for Science & Engineering*. New York: McGraw-Hill, 2001.
[10] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, Sep. 1990.
[11] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 4, pp. 525–533, 1993.
[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, vol. 4, pp. 1942–1948.
[13] A. A. A. Esmin, G. Lambert-Torres, and A. C. Zambroni de Souza, "A hybrid particle swarm optimization applied to loss power minimization," *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 859–866, May 2005.
[14] M. M. Noel and T. C. Jannett, "Simulation of a new hybrid particle swarm optimization algorithm," in *Proc. 36th Southeastern Symp. Syst. Theory*, 2004, pp. 150–153.
[15] C. F. Juang, "A hybrid genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.
[16] N. Mo, Z. Y. Zou, K. W. Chan, and T. Y. G. Pong, "Transient stability constrained optimal power flow using particle swarm optimization," *Proc. IET–Generation, Transmission Distribution*, vol. 1, no. 3, pp. 476–483, May 2007.
[17] A. Chatterjee, K. Pulasinghe, K. Watanabe, and K. Izumi, "A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems," *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1478–1498, Dec. 2005.
[18] P. Melin and O. Castillo, "Comparison of nonuniform optimal quantizer designs for speech coding with adaptive critics and particle swarm," *IEEE Trans. Ind. Appl.*, vol. 43, no. 1, pp. 238–244, Jan./Feb. 2007.
[19] Y. Song, Z. Chen, and Z. Yuan, "New chaotic PSO-based neural network predictive control for nonlinear process," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 595–601, Mar. 2007.
[20] B. Zhao, C. X. Guo, and Y. J. Cao, "A multiagent-based particle swarm optimization approach for optimal reactive power dispatch," *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 1070–1078, May 2005.
[21] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming VII*, vol. 1447. New York: Springer-Verlag, 1998, pp. 611–616.
[22] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*, 2nd extended ed. New York: Springer-Verlag, 1994.
[23] K. L. Petri, R. E. Billo, and B. Bidanda, "A neural network process model for abrasive flow machining operations," *J. Manuf. Syst.*, vol. 17, no. 1, pp. 52–65, Jan. 1998.
[24] K. Saeed and M. K. Nammous, "A speech-and-speaker identification system: Feature extraction, description, and classification of speech-signal image," *IEEE Trans. Ind. Electron.*, vol. 54, no. 2, pp. 887–897, Apr. 2007.
[25] H. Su and K. T. Chong, "Induction machine condition monitoring using neural network modeling," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 241–249, Feb. 2007.
[26] K. D. V. Prasad and P. Yarlagadda, "Prediction of die casting process parameters by using an artificial neural network model for zinc alloys," *Int. J. Prod. Res.*, vol. 38, no. 1, pp. 119–139, Jan. 2000.
[27] H. Zhung, K. S. Low, and W. Y. Yau, "A pulsed neural network with on-chip learning and its practical applications," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 34–42, Feb. 2007.
[28] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, 1995, vol. 12, pp. 1137–1143. no. 2.
[29] S. B. Roh, W. Pedrycz, and S. K. Oh, "Genetic optimization of fuzzy polynomial neural networks," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 2219–2238, Aug. 2007.
[30] H. K. Lam and F. H. F. Leung, "Design and training for combinational neural-logic systems," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 612–619, Feb. 2007.
[31] K. W. Chau, "Particle swarm optimization training algorithm for ANNs in stage prediction of Shing Mun River," *J. Hydrol.*, vol. 329, no. 3/4, pp. 363–367, Oct. 2006.
[32] L. Y. Lin, C. T. Cheng, and K. W. Chau, "Using support vector machines for long-term discharge prediction," *Hydrol. Sci. J.*, vol. 51, no. 4, pp. 599–612, Aug. 2006.
[33] K. W. Chau, "Application of a PSO-based neural network in analysis of outcomes of construction claims," *Autom. Constr.*, vol. 16, no. 5, pp. 642–646, 2007.
[34] C. Veenhuis, M. Köppen, and R. Vicente-Garcia, "Evolutionary multi-objective optimization of particle swarm optimizers," in *Proc. Congr. Evol. Comput.*, Jul. 2007, pp. 2273–2280.
[35] U. Baumgartner, C. Magele, K. Preis, and W. Renhart, "Particle swarm optimisation for Pareto optimal solutions in electromagnetic shape design," *Proc. Inst. Electr. Eng.——Sci., Meas., Technol.*, vol. 151, no. 6, pp. 499–502, Nov. 2004.
[36] H. Q. Min, J. H. Zhu, and X. J. Zheng, "Obstacle avoidance with multi-objective optimization by PSO in dynamic environment," in *Proc. Mach. Learn. Cybern.*, Aug. 2005, vol. 5, pp. 2950–2956.
[37] Z. Lin, D. S. Reay, B. W. Williams, and X. He, "Online modelling for switched reluctance motors using B-spline neural networks," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3317–3322, Dec. 2007.
[38] L. dos Santos Coelho and B. M. Herrera, "Fuzzy identification based on a chaotic particle swarm optimization approach applied to a nonlinear yo-yo motion system," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3234–3245, Dec. 2007.
[39] H. X. Li, S. K. Tso, and H. Deng, "A concept approach to integrate design and control for the epoxy dispensing process," *Int. J. Adv. Manuf. Technol.*, vol. 17, pp. 667–682, 2001.
[40] G. A. F. Seber, *Linear Regression Analysis*. New York: Wiley, 2003.
[41] K. W. Ip, C. K. Kwong, H. Bai, and Y. C. Tsim, "The process modelling of epoxy dispensing for microchip encapsulation using fuzzy linear regression with fuzzy intervals," *Int. J. Adv. Manuf. Technol.*, vol. 22, no. 5/6, pp. 417–423, Oct. 2003.
[42] C. K. Kwong, K. Y. Chan, and H. Wong, "An empirical approach to modeling fluid dispensing for electronic packaging," *Int. J. Adv. Manuf. Technol.*, vol. 34, no. 1/2, pp. 111–121, Aug. 2007.

S. H. Ling (S'03–M'06) received the B.Eng., M.Phil., and Ph.D. degrees from The Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1999, 2002, and 2006, respectively.

During the period 2006–2007, he was a Postdoctoral Research Associate with the School of Electrical, Electronic and Computer Engineering, University of Western Australia, Perth, Australia. He is currently a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He has authored or coauthored over 50 international journal and conference papers on computational intelligence and its industrial applications. His current research interests include evolution computations, fuzzy logic, neural networks, hybrid systems, and job shop scheduling.



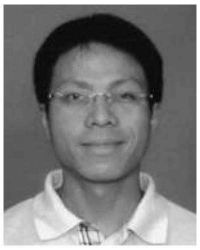
H. H. C. Iu (S'98–M'00–SM'06) received the B.Eng. (Hons) degree in electrical and electronic engineering from the University of Hong Kong, Kowloon, Hong Kong, in 1997, and the Ph.D. degree from The Hong Kong Polytechnic University, Kowloon, in 2000.

He is currently a Senior Lecturer with the School of Electrical, Electronic, and Computer Engineering, University of Western Australia, Perth, Australia. His research interests include nonlinear circuits and systems, power electronics, and TCP network dynamics. He has published over 60 papers in these areas. He currently serves as a Guest Editor for the *Australian Journal of Electrical and Electronics Engineering* and *Circuits, Systems and Signal Processing*.

Frank H. F. Leung (M'92–SM'03) was born in Hong Kong in 1964. He received the B.Eng. degree and the Ph.D. degree in electronic engineering from The Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1988 and 1992, respectively.

Since 1992, he has been with The Hong Kong Polytechnic University, where he is currently an Associate Professor in the Department of Electronic and Information Engineering. He is active in research and has published over 150 research papers on computational intelligence, control, and power electronics. He is currently involved in R&D on intelligent home automation. He has been serving as a reviewer for many international journals and helping the organization of many international conferences.

Prof. Leung is currently an Executive Committee member of the IEEE Hong Kong Chapter of Signal Processing. He is a Chartered Engineer and a corporate member of the Institution of Electrical Engineers, U.K.



K. Y. Chan received the M.Phil. degree in electronic engineering from the City University of Hong Kong, Kowloon, Hong Kong, and the Ph.D. degree in computing from London South Bank University, London, U.K.

He is currently a Postdoctoral Research Fellow with the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Kowloon. His research interests include computational intelligence and its applications in product design, signal processing, power systems, and operation researches.