Continuous genetic algorithm-based fuzzy neural network for learning fuzzy IF–THEN rules

R.J. Kuo\textsuperscript{a,*}, S.M. Hong\textsuperscript{b}, Y. Lin\textsuperscript{a}, Y.C. Huang\textsuperscript{c}

\textsuperscript{a}Department of Industrial Engineering and Management, National Taipei University of Technology, No. 1, Section 3, Chung-Hsiao East Road, Taipei, Taiwan 106, ROC
\textsuperscript{b}Department of Business Administration, National Chengchi University, No. 64, Sec. 2, ZhiNan Rd., Wenshan District, Taipei City 11605, Taiwan, ROC
\textsuperscript{c}Department of Accounting, I-Shou University, No.1, Sec. 1, Syuecheng Rd., Dashu Township, Kaohsiung County 840, Taiwan, ROC

Abstract

This study proposes a fuzzy neural network (FNN) that can process both fuzzy inputs and outputs. The continuous genetic algorithm (CGA) is employed to enhance its performance. Both the simulation and real-world problem results show that the proposed CGA-based FNN can obtain the relationship between fuzzy inputs and outputs. CGA can not only shorten the training time but also increase the accuracy for the FNN.

Keywords: Fuzzy neural networks; Continuous genetic algorithms

1. Introduction

Previous researches [13,20] have presented fuzzy neural networks (FNNs) of different structures, yet most of them are only for crisp data instead of fuzzy numbers. Nonetheless, some observations of real-world problems can be crisp information obtained by measuring instruments, while others can be linguistic information given by the domain experts. The latter is especially critical for the field of social science in the knowledge economics, since most of the know-how is from the human experts.

Hence, this study proposes a continuous genetic algorithm (CGA)-based FNN, which is capable of learning fuzzy IF–THEN rules. Some researches [10,11,20,22] have proposed different schemes for processing linguistic information with fuzzy inputs, fuzzy weights, and fuzzy outputs. However, improving the computational performance is still necessary. Thus, real-coded GA is utilized to generate the fuzzy weights on the basis of FNN structure proposed by Kuo and Chen [13]. One advantage of CGA is that it is a kind of multi-point searching methods compared with the gradient steepest search method. The other merit is that it is not necessary to transform the data into binary values compared with the conventional genetic algorithm. This can dramatically increase both computational efficiency and accuracy.

The simulated data are first employed to verify the feasibility of the proposed FNN. The results show that the proposed CGA-based FNN can accurately learn the relations of fuzzy inputs and outputs compared with the FNN by Kuo and Chen [13] on the basis of mean square error. Moreover, the proposed network is applied to a real-world problem with respect to supplier selection. The purpose is to learn the qualitative factors that may influence the selection decision via the proposed FNN. Results of the real-world problem for a world-class OEM notebook computer company also indicate that the proposed FNN can really learn the relationship between the qualitative factors, or fuzzy factors, and their corresponding impacts.

The rest of this paper is organized as follows. Section 2 presents a brief review of the background, while the proposed method is illustrated in Section 3. Both the
simulation and real-world problem results are shown in Sections 4 and 5, respectively. Finally, the concluding remarks are made in Section 6.

2. Literature review

This Section will briefly describe the FNNs and genetic algorithms (GAs), respectively.

2.1. Fuzzy neural networks (FNNs)

Artificial neural networks (ANNs), fuzzy logic, and genetic algorithms constitute three independent research fields regarding sixth generation systems (SGS) as illustrated in the following. ANNs [23] and the fuzzy sets theory [19] have been widely applied but there are both advantages and disadvantages regarding the combinations. Therefore, successful combination of the two approaches merits further investigation. This kind of method is called fuzzy neural network.

2.1.1. Crisp FNNs

In traditional fuzzy systems, the construction of expert experience is not so objective. Consequently, Lee et al. [29] utilized the learning rule to revise bad fuzzy IF–THEN rules and attain better results. They exploited using FNN to train the fuzzy model and the IF–THEN rules. The result infers an effective conclusion in two-dimensional curved surface. Moreover, the problems of fuzzy model are the lack of flexibility in the decision-making of fuzzy numbers and which kind of fuzzy shapes can represent the expert experience. The answer emerges when the fuzzy parameters of IF–THEN rules are tuned up by the numerical information of ANNs. That is why the so-called FNN is developed. Kuo and Cohen [15,16] and Lin and Lee [21] proposed the so-called neural-network-based fuzzy logic control system (NN-FLCS), which is now widely applied. The methodology of this system is the mixed learning rule. First, a self-organization map is used for data clustering. After finding out the fuzzy parameter for IF–THEN rules ($\mu^L$, $\mu^R$), the fuzzy number in the inter rule is then adjusted by back-propagation to reduce the errors of functions.

2.1.2. Non-crisp FNNs

However, only numerical data fit the above-mentioned crisp FNNs and the knowledge of experts is always of fuzzy type. Thus, some studies focus on the method for overcoming this problem. Gupta and Qi [5] presented some models with fuzzy neurons but no learning algorithms within. However, a number of publications were cited in a survey paper [2], the authors discussed the learning algorithms and applications for FNNs with fuzzy inputs, weights and outputs [1,6]. Ishibuchi et al. [10,11] also proffered the learning methods of neural networks to utilize not only numerical data but also expert knowledge represented by fuzzy IF—THEN rules. In these works, triangular membership functions are applied. Lin [20] and Lin and Lu [22] presented an FNN capable of handling both fuzzy inputs and outputs. On the basis of Ishibuchi’s work, Kuo and Xue [18] presented an FNN, which carries not only the asymmetric fuzzy inputs and outputs but also the asymmetric fuzzy weights. The network was applied to the sales forecasting problem. Kuo and his colleagues also made the improvement for the above FNN by eliminating the unimportant fuzzy weights [17]. This really can accelerate the training speed and have the smaller training error, mean square error.

2.2. Continuous genetic algorithms

The GA, a naturally evolutional heuristic methodology that can efficiently solve the problems of function optimization and control system was first put forward by Holland in 1970. The procedure of GA is a simulation from biological evolution behavior [25]. GA not only adopts the spirit of creature elimination rule but also finds the approximate optimal solution after the process of coding, decoding and constant operation (reproduction, crossover and mutation). In the late thirties, many scholars find that GA has the superior ability of finding the approximate optimal solution compared with other heuristic algorithms such as tabu search algorithm, simulated annealing and ant algorithm in some situations. GAs can also be applied to many areas, like optimization or classification. Fig. 1 presents the flow chart of general GAs.

Most of GA researches are binary oriented. Thus, a new algorithm called CGA is proposed for the global optimization of multi-minima functions [3]. In order to cover a wide domain of possible solutions, this algorithm first chooses
the initial population. Then it locates the most promising area of the solution space and continues the search through "intensification" inside this area. Reproduction, crossover and mutation are performed using the decimal code.

This algorithm uses real coding, which is as close as possible to Holland's approach using binary coding. This efficient algorithm modified the Michalewicz method [24] by proposing new crossover and mutation operators and by taking into account the size and distribution of the population in the entire solution space. The size of the population must be initially large enough to achieve a better convergence of the algorithm. To avoid a prohibitive CPU time, it is necessary to dynamically reduce the size of this population. The reductions in space area and in population size are performed after a given number of consecutive generations showed no improvement of the objective function. The variation steps of the crossover and mutation operators depend directly on the search space size. Thus, at each reduction of the search space size, these steps are reduced, too.

First, an iterative diversification using GA is performed to localize "promising areas," that are likely to contain a global minimum. After a specified number of successive generations show no improvement of the objective function, the best area is localized and diversification comes to an end. Then the intensification starts. It begins with reducing the search domain, the neighborhood of the individuals, the size of the population, the mutation probability and the perturbation steps in the crossover and mutation operations. Next, a new population is generated around the previously found best points inside the new search domain. The strategy of generating this new population is the same as the one used in diversification. Finally, the previous diversification module is employed again to perform the algorithm recursively.

2.3. Integration of GA with ANNs

Two main drawbacks of the above ANNs, no matter the conventional ANNs or FNNs, are long training time and local minimum problems. Thus, GA is utilized to prevent these two problems. In essence, most GA applications focus on binary GA [4,27]. Sexton et al. [26] compared the error back-propagation (EBP) type learning algorithm with GA. Results from seven testing examples revealed that GA is superior to EBP-type learning algorithm. Houck et al. [7] compared three approaches, namely binary GA, real-coded GA, and simulated annealing. The testing examples include non-linear, multi-modal, and non-convex and the simulation results showed that real-coded GA outperforms the other two methods, with the worst being simulated annealing.

In addition to conventional ANNs, FNNs also employ GA to enhance their performance. Integration of binary GA and the FNN together is proposed in order to derive better results both in speed and accuracy for the stock market forecasting problem [14]. Though the result is very promising, yet binary GA has to transform the real value to binary value for computation. This may decrease the computational efficiency and accuracy. Thus, this FNN is also combined with real-coded GA for order selection system [13]. This study proved that integration of real-coded GA with FNN can provided better result than integration of binary GA with FNN.

3. Methodology

According to the above two sections for reviewing FNNs and GAs, this section proposes a novel learning algorithm for formulating the weights of FNN.

3.1. Fuzzy IF–THEN rules construction

Aiming at the qualitative factors, the first step is to analyze the questionnaire and get the fuzzy number for each factor. The characteristic of fuzzy number is shown by sigma ($\sigma^I$), mean ($\mu$) and sigma ($\sigma^R$) to acquire the database of fuzzy rules.

In this research, the procedure of producing the weight is as follows:

(a) Collect all the possible factors that may affect the supplier selection. The domain experts select important factors and offer each a fuzzy number. This is the first questionnaire survey.
(b) Formulate the second questionnaire, and which contains a set of IF–THEN rules.
(c) Fuzzify the second questionnaire that is returned by the managers and determine the pessimistic index, optimistic index and average index. The formulations are as follows:

1. Pessimistic (minimum) index:

$$l = \frac{l_1 + l_2 + \cdots + l_n}{n},$$

where $l_i$ is the pessimistic index of the $i$th expert and $n$ is the number of experts.

2. Optimistic (maximum) index:

$$u = \frac{u_1 + u_2 + \cdots + u_n}{n},$$

where $u_i$ is the optimistic index of the $i$th expert.

3. Average (most appropriate) index:

For each interval ($l_i, u_i$), calculate the mid point, $m_i = (l_i + u_i)/2$ and then find

$$\mu = (m_1 \times m_2 \times \cdots \times m_n)^{1/n}.$$ (3)

(d) Establish fuzzy number $A = (\mu, \sigma^R, \sigma^L)$, which represents the mean, right width, and left width,
respectively, for an asymmetric bell-shaped function
\[
\sigma^R = \frac{l - \mu}{3},
\]  
\[
\sigma^L = \frac{u - \mu}{3}.
\]  
(e) Formulate the third questionnaire with the above indices and repeat until all experts reached a consensus for satisfying the formula [12] below. Otherwise, repeat the questionnaire.

\[
\delta(\bar{A}, \bar{B}) = \int_{0}^{1} \delta(\tilde{A}[z], \tilde{B}[z]) \, dz
\]
\[
= \frac{1}{2} \int_{0}^{1} \left[ \left( |\tilde{A}[z]| - |\tilde{B}[z]| \right) + \left( |\tilde{A}[z]| - |\tilde{B}[z]| \right) + \left( |\tilde{A}[z]| - |\tilde{B}[z]| \right) \right] \, dz
\]
\[
= \frac{1}{2} \left[ \int_{0}^{1} \left( |\tilde{A}[z]| - |\tilde{B}[z]| \right) \, dz \right]
+ \int_{0}^{1} \left( |\tilde{A}[z]| - |\tilde{B}[z]| \right) \, dz
\]
\[
= \frac{1}{2} \left[ \sum z (\mu_A - \mu_B) - (\sigma_A^R - \sigma_B^R)
\times (-2 \ln z^{1/2}) + \sum z (\mu_A - \mu_B)
\right.
+ (\sigma_A^L - \sigma_B^L)(-2 \ln z^{1/2})],
\]  
where \( \bar{A}, \bar{B} \) represents two asymmetrical fuzzy numbers and \( \delta(\bar{A}, \bar{B}) \) is the error between \( \bar{A} \) and \( \bar{B} \).

The similarity of the two fuzzy numbers is determined according to whether \( \delta(\bar{A}, \bar{B}) \) is smaller than the tolerated error or not.

(f) Construct fuzzy IF–THEN rules and collect fuzzy numbers as training data in FNN.

3.2. General process of CGA

The main stages of CGA are initialization, generation of the initial population, production of the new population, intensification around the best points, and output of the best point found. These five steps can produce the new populations.

Once the offspring have been produced by reproduction, crossover, and mutation of individuals from the old population, the objective function values of the offspring are also determined. The new population is produced, and this process is reiterated. In the intensification module, the search domain, population size, neighborhood radius, and probability of mutation are reduced. Moreover, the new population is generated around the best point found and the genetic procedure inside this new search domain is restarted. The number of domain reductions depends on the required accuracy. The algorithm terminates when a given number of iterations is reached or a given accuracy relating to the individuals’ coordinates is obtained. In the parameter setting, the number of generations is set as many as the number of chromosomes and the number of iterations is equal to the population size. Fig. 2 illustrates the general flow chart of CGA.

3.2.1. Generation of initial population

To cover homogeneously the whole solution space and to avoid the risk of having too much individuals in the same region, the algorithm selects a large population, and defines a “neighborhood” for each selected individual. The type of neighborhood used, proposed in [8], uses the notion of “balls.” A ball \( B(s, \epsilon) \), centered on \( s \) with the radius \( \epsilon \), contains all points “\( s' \)” such as \( ||s' - s|| \leq \epsilon \).

3.2.2. Reproduction

The reproduction method used is a particular form of the roulette-wheel selection [24]. After having calculated the objective function value for each individual and detected the best value, calculate for each individual the difference, called fitness value, between its objective function value and this best-found value. Now the corresponding fitness of each individual is replaced by the fitness, obtained by adding its fitness to that of the previous ones. The individuals are mapped to contiguous segments of a line, such that each individual segment is equal in size to the associated fitness. A random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals is obtained.

3.2.3. Crossover

In real coding, to simulate as faithfully as possible this crossing operation involving two individuals, the research proceeds in the following two ways: determine the crossing point \( I \), and draw a random integer \( i \) with components between 0 and the dimension of the individual. All the components situated on the left of this crossing point are not affected and those situated beyond the crossing point are exchanged. For the components \( x(i) \) and \( y(i) \) of two individuals \( x \) and \( y \), respectively located at position \( I \), are operated as follows: deduce a quantity \( \Delta x \) from \( x(i) \) component, and add it to \( y(i) \) component, then deduce a quantity \( \Delta y \) from \( y(i) \) component, finally add it to \( x(i) \) component. These quantities are randomly determined: integer \( M \) is drawn between 1 and 1000 in random and compute \( \Delta x \) and \( \Delta y \) as follows [3]:
\[
\Delta x = \frac{x(i)}{M} \quad \text{and} \quad \Delta y = \frac{y(i)}{M}.
\]
The new coordinated values are

\[ x'(i) = x(i) + \Delta y - \Delta x \quad \text{and} \quad y'(i) = y(i) + \Delta x - \Delta y. \]  

This method allows enriching the population, contrary to Muhlenbein’s crossover operator, which simply consists of a permutation between components of two individuals [25].

3.2.4. Mutation

Usually the probability of mutation and the corresponding variation are large at the beginning of the search process, and they are decreased progressively while approaching the promising zone. In order to simulate as faithfully as possible this process, draw randomly for every individual one number between 0 and 1, and then compare
it with the probability of mutation, which decreases in the course of the treatment. If the drawn number is superior to this probability, the algorithm proceeds with the rest of the components; otherwise, the mutation is performed in the following way: select which component to be disturbed and the amount of disturbance, and draw a random integer \( i \) between 0 and the dimension of the individual, and another integer \( M \) between 1 and 10. Finally determine the following variation:

\[
\Delta x = \frac{\text{UpBd}(i) - \text{LowBd}(i)}{M},
\]

where \( \text{UpBd}(i) \) and \( \text{LowBd}(i) \) represent the superior bound and the inferior bound of \( x(i) \) component of individual \( x \) to be disturbed, respectively. So the new fitness function value.

3.3. FNN

FNN learns these fuzzy rules through the following two phases, initial weights generation via CGA and EBP-type learning algorithm.

3.3.1. Initial weights generation

The present study is conducted by integrating the CGA with FNN wherein. The CGA provides the initial weights for the FNN, and which not only can diminish the training time but can also prevent the network from getting stuck at the local minimum. The procedures of CGA used in this study are as follows [3]:

- **Step 1.** Generate \( n \) populations randomly and determine the number of generations and fitness function.
- **Step 2.** Assure the fitness function value for each chromosome.
- **Step 3.** Process the chromosome operators by selection, crossover, and mutation.0
- **Step 4.** Determine the new population.
- **Step 5.** Eliminate the chromosomes with lower fitness function value and add the new chromosomes with higher fitness function value.
- **Step 6.** If the stop criterion is satisfied or accuracy is reached, then stop; otherwise, go back to Step 3.

In present study the fitness function is defined as

\[
F = \frac{1}{N} \sum_{i=1}^{N} (T_i - Y_i)^2,
\]

where \( N \) denotes the size of the population, \( T_i \) represents the \( i \)-th-desired output, and \( Y_i \) is the \( i \)-th actual output.

3.3.2. EBP-type learning algorithm

In this section, according to the fuzzy IF–THEN rules and fuzzy numbers in the above Section, the network training and learning are introduced. The operations of fuzzy numbers are presented first.

3.3.2.1. Operations of fuzzy numbers. In this study the real number is denoted by the lowercase letters (e.g. \( a, b, \ldots \) ) and the uppercase letters with a bar above (e.g. \( \bar{A}, \bar{B}, \ldots \) ) represent fuzzy numbers. To explicitly conduct learning algorithm in FNN, the definition of fuzzy numbers, addition, multiplication for fuzzy numbers and extension principles are introduced.

1. **Definition:** \( \bar{a} = (\sigma^L, \mu, \sigma^R)_{L-R} \) is asymmetric Gaussian fuzzy number and satisfy

\[
\bar{a} = \begin{cases} 
\exp \left( -\frac{(x-\mu)^2}{2\sigma^L} \right), & x < \mu, \\
1, & x = \mu, \\
\exp \left( -\frac{(x-\mu)^2}{2\sigma^R} \right), & \text{otherwise},
\end{cases}
\]

where \( \mu, \sigma^L \) and \( \sigma^R \) represent the mean, left-side sigma and right-side sigma, respectively.

The definition switches the triangle or trapezoid fuzzy number [10] to asymmetric Gaussian fuzzy number (Fig. 3). The purposes are to accelerate the network learning speed and to more fit the state of problems to obtain better outcome.

2. **Fuzzy addition rule:**

\[
\bar{Z}(z) = \bar{X}(x) + \bar{Y}(y) = \max \{ \bar{X}(x) \wedge \bar{Y}(y) | z = x + y \}. \tag{13}
\]

3. **Fuzzy multiplication rule:**

\[
\bar{Z}(z) = \bar{X}(x) \cdot \bar{Y}(y) = \max \{ \bar{X}(x) \wedge \bar{Y}(y) | z = x \cdot y \}. \tag{14}
\]

4. **Extension principles:**

\[
f(\text{Net}(x)) = \max \{ \text{Net}(x) | z = f(x) \}, \tag{15}
\]

where \( \bar{X}, \bar{Y}, \bar{Z}, \text{Net} \) are fuzzy numbers, \( (\cdot) \) denotes the membership function of each fuzzy number, \( \wedge \) is the minimum operator, and \( f(x) = (1 + \exp(-x))^{-1} \) is the activation function of hidden units and output units of the proposed FNN (Fig. 4).

The \( z \)-cut of fuzzy number \( \bar{X} \) is defined as:

\[
\bar{X}[z] = \{ x | \mu_{\bar{X}}(x) \geq z, \quad x \in \mathbb{R} \} \quad \text{for} \quad 0 < z \leq 1, \tag{16}
\]

where \( \bar{X}[z] \) represents \( \bar{X}[z] = [\bar{X}[z]^L, \bar{X}[z]^U] \) and \( \bar{X}[z]^L \) and \( \bar{X}[z]^U \) represent the lower and upper \( z \)-cuts of fuzzy number \( \bar{X} \), respectively.
3.3.2.2. Structure of FNN. Before discussing the algorithm, some assumptions have to be clarified as follows:

1. Fuzzify a three-layer feedforward neural network with \( n_1 \) input units, \( n_H \) hidden units, and \( n_O \) output units (i.e., input vectors, target vectors, connection bias and thresholds are fuzzified).
2. The input vectors are non-negative fuzzy numbers whose lower and upper bounds are larger than zero.
3. These fuzzy numbers are asymmetric Gaussian-shaped fuzzy numbers.

The basic structure of FNN is shown in Fig. 5 below with three parts, namely input layer, hidden layer, and output layer.

**Input layer:**
\[
\overline{\mathcal{O}}_{pi}[z] = \overline{\mathcal{X}}_{pi}[z], \quad i = 1, 2, \ldots, n_1,
\]

**Hidden layer:**
\[
\overline{\mathcal{O}}_{ph}[z] = f(\overline{\mathcal{N}}\overline{\mathcal{E}}_{ph}[z]), \quad h = 1, 2, \ldots, n_H,
\]
\[
\overline{\mathcal{N}}\overline{\mathcal{E}}_{ph}[z] = \sum_{i=1}^{n_1} \overline{W}_{hi}[z] \cdot \overline{\mathcal{O}}_{pi}[z] + \overline{\Theta}_{h}[z],
\]

**Output layer:**
\[
\overline{\mathcal{O}}_{pk}[z] = f(\overline{\mathcal{N}}\overline{\mathcal{E}}_{pk}[z]), \quad k = 1, 2, \ldots, n_O,
\]
\[
\overline{\mathcal{N}}\overline{\mathcal{E}}_{pk}[z] = \sum_{i=1}^{n_1} \overline{W}_{kh}[z] \cdot \overline{\mathcal{O}}_{ph}[z] + \overline{\Theta}_{k}[z],
\]
where \( \overline{\mathcal{X}}_{pi} \) denotes fuzzy input, \( \overline{W}_{hi} \) and \( \overline{W}_{kh} \) represent fuzzy weight, \( \overline{\Theta}_{h} \) and \( \overline{\Theta}_{k} \) are the fuzzy node bias.

3.3.2.3. Algorithm of FNN. From Eqs. (17)–(21), the \( \alpha \)-cut sets of the fuzzy output \( \overline{\mathcal{O}}_{pk} \) are calculated from the fuzzy inputs, fuzzy weights, and fuzzy biases. If the \( \alpha \)-cut set of fuzzy outputs \( \overline{\mathcal{O}}_{pk} \) is required, the above relation can be rewritten as follows [13,14,17,18]:

\[
\overline{\mathcal{O}}_{pi}[z] = [\overline{\mathcal{O}}_{pi}[z]^i, \overline{\mathcal{O}}_{pi}[z]^u] = [\overline{\mathcal{X}}_{pi}[z]^i, \overline{\mathcal{X}}_{pi}[z]^u],
\]

\[i = 1, 2, \ldots, n_1.\]

**Hidden layer:**
\[
\overline{\mathcal{O}}_{ph}[z] = [\overline{\mathcal{O}}_{ph}[z]^i, \overline{\mathcal{O}}_{ph}[z]^u] = [f(\overline{\mathcal{N}}\overline{\mathcal{E}}_{ph}[z]^i), f(\overline{\mathcal{N}}\overline{\mathcal{E}}_{ph}[z]^u)],
\]

\[h = 1, 2, \ldots, n_H.\]

The lower and upper fuzzy function numbers are:
\[
\begin{align*}
\overline{\mathcal{N}}\overline{\mathcal{E}}_{ph}[z]^i &= \sum_{i=1}^{n_1} \overline{W}_{hi}[z]^i \cdot \overline{\mathcal{O}}_{pi}[z]^i + \sum_{i=1}^{n_1} \overline{W}_{hi}[z]^u \cdot \overline{\mathcal{O}}_{pi}[z]^u + \overline{\Theta}_{h}[z]^i, \\
\overline{\mathcal{N}}\overline{\mathcal{E}}_{ph}[z]^u &= \sum_{i=1}^{n_1} \overline{W}_{hi}[z]^i \cdot \overline{\mathcal{O}}_{pi}[z]^i + \sum_{i=1}^{n_1} \overline{W}_{hi}[z]^u \cdot \overline{\mathcal{O}}_{pi}[z]^u + \overline{\Theta}_{h}[z]^u.
\end{align*}
\]

**Output layer:**
\[
\overline{\mathcal{O}}_{pk}[z] = [\overline{\mathcal{O}}_{pk}[z]^i, \overline{\mathcal{O}}_{pk}[z]^u] = [f(\overline{\mathcal{N}}\overline{\mathcal{E}}_{pk}[z]^i), f(\overline{\mathcal{N}}\overline{\mathcal{E}}_{pk}[z]^u)],
\]

\[k = 1, 2, \ldots, n_O.\]

The lower and upper fuzzy function numbers are:
\[
\begin{align*}
\overline{\mathcal{N}}\overline{\mathcal{E}}_{pk}[z]^i &= \sum_{k=1}^{n_O} \overline{W}_{kh}[z]^i \cdot \overline{\mathcal{O}}_{ph}[z]^i + \sum_{k=1}^{n_O} \overline{W}_{kh}[z]^u \cdot \overline{\mathcal{O}}_{ph}[z]^u + \overline{\Theta}_{k}[z]^i, \\
\overline{\mathcal{N}}\overline{\mathcal{E}}_{pk}[z]^u &= \sum_{k=1}^{n_O} \overline{W}_{kh}[z]^i \cdot \overline{\mathcal{O}}_{ph}[z]^i + \sum_{k=1}^{n_O} \overline{W}_{kh}[z]^u \cdot \overline{\mathcal{O}}_{ph}[z]^u + \overline{\Theta}_{k}[z]^u.
\end{align*}
\]

The objective to minimize the cost function is defined as
\[
E_p = \sum_{z} \sum_{k=1}^{n_O} \alpha(E^L_{k(z)} + E^U_{k(z)}) = \sum_{z} E_p(z),
\]
where
\[
E_p(z) = \sum_{k=1}^{n_O} \alpha(E^L_{k(z)} + E^U_{k(z)}).
\]
4. Simulation

This section will verify the feasibility of FNN and compare the advantages of CGA with the real-coded genetic algorithm. Finally, the better solution process and parameter settings are going to be chosen as the basis of the case study. The simulated program is written in Visual C++.

4.1. Fuzzy neural network

This section will show the feasibility of the proposed FNN. There are two examples given. The first example simulates the simple linear fuzzy rules, while the second example verifies the capability of inference to IF–THEN rules of FNN.

4.1.1. Example 1—linear fuzzy rules

Three training pairs, or rules, \((\vec{x}_p, \vec{y}_p)\) where \(p = 1, 2, 3\), in the two-dimensional space are used. The mean value is as \(y_{i\mu} = (x_{i\mu})\), while the left width and right width are \(y_{iL} = 2x_{i\sigma}\) and \(y_{iR} = 3x_{i\sigma}\), respectively. The training data are listed in Table 1.

The required parameters are as follows:

1. Number of hidden nodes: 3.
2. Number of hidden layers: 1.
3. \(\alpha\) cut set: \(\alpha = 0.1, 0.3, 0.5, 0.7\) and 0.9.
4. Number of training epochs: 25000
5. Training rate: \(\eta=0.3\).
6. Momentum: \(\beta=0.6\).

The training error (MSE) curve is illustrated in Fig. 6 and the MSE value is 0.000052, while Fig. 7 describes the graphical relation of fuzzy inputs and outputs as \(\alpha\) is equal to 0.1, 0.3, 0.5, 0.7 and 0.9 for training pairs. Fig. 8 illustrates the training results of the input-output relations.

Table 2 shows the four testing samples and each corresponding outputs, while the two testing results of input-output relations are illustrated in Fig. 9. The MSE value is 0.000112, which proves that the proposed FNN really has the capability of inference with the linear fuzzy rules.

4.1.2. Example 2—artificial fuzzy rules

The following three rules are assumed:

Rule 1. If temperature \((X)\) is low (L), then the speed of engine \((Y)\) is slow (S).
Rule 2. If temperature \((X)\) is medium (M), then the speed of engine \((Y)\) is medium (M).

Table 1

<table>
<thead>
<tr>
<th>Training data for Example 1</th>
<th>Input ((\vec{x}))</th>
<th>Output ((\vec{y}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma^L)</td>
<td>(\mu)</td>
<td>(\sigma^R)</td>
</tr>
<tr>
<td>1</td>
<td>0.45</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.72</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0.95</td>
<td>9</td>
</tr>
</tbody>
</table>

Fig. 6. Training error (MSE) curve of Example 1.
Rule 3. If temperature ($X$) is high (H), then the speed of engine ($Y$) is high (H).

The distribution of each function is shown in Figs. 10 and 11. The training error (MSE) curve is illustrated in Fig. 12 and the final MSE is 0.000015, while Fig. 13 describes the graphical relation of fuzzy inputs and outputs as $\alpha$ is equal to 0.1, 0.3, 0.5, 0.7 and 0.9 for training pairs. Fig. 14 illustrates the training results of the input–output relations.

The required parameters are as follows:

1. Number of hidden nodes: 3.
2. Number of hidden layers: 1.
3. $\alpha$-cut set: $\alpha = 0.1, 0.3, 0.5, 0.7$ and 0.9.
4. Number of training epochs: 25000.
5. Training rate: $\eta 0.3$.
6. Momentum: $\beta 0.6$.

After finishing network learning, two new IF–THEN rules are added (ML and MH) to prove that the proposed FNN really has the inference capability. Fig. 15 shows the function of IF rules with two new rules and the two testing results of input–output relations are illustrated in Fig. 16. Finally, the functions of THEN rules are illustrated in Fig. 17.
4.2. Continuous genetic algorithm—Example 3

This subsection applies the example as shown in Table 3 to testify the feasibility of GA, both real-coded and continuous GA. There are a total of nine fuzzy IF-THEN rules and the setup for the continuous genetic algorithm is as follows:

1. Initial size of population: 50.
2. Variation of population size: 4.
3. Crossover rate: 0.85.
4. Initial mutation rate: 0.9.
5. Number of generations: 12.
6. Number of iterations: 50.
7. Number of hidden nodes: 3, 4 and 5.
8. Range of each chromosome: 0–10.

In the aspect of parameter setting, the number of iterations is equal to the population size. In the setting of number of generations, if the number of generations is too large, it will lead to an embarrassing situation, making it difficult to proceed to the next step. However, if the
number of generations is too small, the process will seem meaningless. As a result, the trial-and-error method is utilized to determine the number of generations and the number is finally decided to be 12.

Table 4 shows the results of GA + FNN in which GA is employed to generate the initial weights being fine-tuned by the EBP-type learning algorithm. The computational results reveal that the CGA can converge better than the real-coded GA [13]. An interesting finding is that the results obtained from the CGA cannot be further improved, or fine-tuned by the EBP-type learning algorithm. This implies that only CGA is enough to formulate the relationship between the fuzzy inputs and outputs. Thus, even implementing the real-coded GA followed by the EBP-type learning algorithm will yield results still worse than those obtained using CGA only. This phenomenon indicates that our proposed CGA not only has better accuracy but also shorter computational time. According to the above discussion, CGA will be employed in the following section to generate the weights for the FNN. Both the continuous and the real-coded GA data as shown in Table 4 are the average of 10 implementations.

5. Case study

This section attempts to demonstrate the feasibility of CGA-based FNN for learning the real-world fuzzy rules obtained from the domain experts. Then, the results are integrated with the other crisp data for making the decision of supplier selection. This is quite critical in the area of global supply chain management.

5.1. Data collection

A world-class original equipment manufacturer (OEM) notebook computer company in Taiwan that requires PCB suppliers provides the evaluation data. The total number of samples is only 17 because in practice the candidate suppliers are few.

The method for deciding which factors are chosen depends on the reference of past surveys and the experience of domain experts. Eventually, six quantitative factors are determined and they are quality, price, location, finance, facility and productivity, and delivery deadline. The quantitative data are from the historical data of the suppliers of the case company. As for the qualitative data, this study selects the following four factors as the criteria after discussing with the domain experts. They include technical capability, quick response for requirement, etc.

Table 3
Nine fuzzy IF-THEN rules of Example 3

<table>
<thead>
<tr>
<th>Rule</th>
<th>(X_1)</th>
<th>(X_2)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>50</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>50</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>50</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>80</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>80</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>80</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4
FNN results using GA to generate initial weights (result unit: \(10^{-3}\))

<table>
<thead>
<tr>
<th>Network topology</th>
<th>Crossover rate</th>
<th>Mutation rate</th>
<th>FNN only</th>
<th>Different GA</th>
<th>Best</th>
<th>Second</th>
<th>Third</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-3-1</td>
<td>0.3</td>
<td>0.03</td>
<td>6.947</td>
<td>Continuous</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Real-coded</td>
<td>0.388</td>
<td>0.403</td>
<td>0.403</td>
</tr>
<tr>
<td>2-4-1</td>
<td>0.3</td>
<td>0.03</td>
<td>6.644</td>
<td>Continuous</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Real-coded</td>
<td>0.359</td>
<td>0.359</td>
<td>0.360</td>
</tr>
<tr>
<td>2-5-1</td>
<td>0.3</td>
<td>0.03</td>
<td>6.452</td>
<td>Continuous</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Real-coded</td>
<td>0.333</td>
<td>0.356</td>
<td>0.367</td>
</tr>
</tbody>
</table>

When the number in computer is smaller than \(10^{-6}\), it is ignored and regarded as zero.
managerial organization, and long-term relationship capability. The fuzzy questionnaire is designed and completed by the domain experts via the questionnaire survey presented in Section 3. Tables 5 and 6 illustrate these 27 fuzzy IF–THEN rules which are utilized to train the proposed CGA based FNN.

5.2. Construction of CGA based FNN

CGA based FNN construction is consisted of two parts, CGA and EBP-type learning. They are discussed in the following two subsections.

5.2.1. Utilize CGA to pursue approximate optimal solution

The network topology is composed of three layers: input, hidden and output layers. There are four input nodes representing long-term relationship capability, technical capability, technical capability, and managerial organization, respectively, while there is only one output, which is the degree of importance for the order. Regarding the number of hidden nodes, the trial-and-error method is used. Seven cases (3, 4, 5, 6, 7, 8 and 9 hidden nodes) are verified. As mentioned earlier, there are 27 fuzzy IF–THEN rules for training the FNN. Thus, the CGA is first employed to learn these 27 rules according to the above combinations. Then, three best results are adopted as the initial weights for the FNN in order to find the final results using the EBP-type learning algorithm. These procedures can establish an IF–THEN rule base. In this supplier case, because the range is localized within a small range (from 0 to 10), the

Table 5
Fuzzy numbers for each aspect

<table>
<thead>
<tr>
<th>Factors</th>
<th>Degree</th>
<th>$a^L$</th>
<th>$\mu$</th>
<th>$a^R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long-term relationship capability</td>
<td>Large</td>
<td>0.28539</td>
<td>2.356185</td>
<td>0.314605</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>0.51193</td>
<td>5.735799</td>
<td>0.5214</td>
</tr>
<tr>
<td></td>
<td>Small</td>
<td>0.26088</td>
<td>8.982649</td>
<td>0.27245</td>
</tr>
<tr>
<td>Technical capability</td>
<td>Large</td>
<td>0.3316</td>
<td>2.294803</td>
<td>0.368399</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>0.47701</td>
<td>5.731041</td>
<td>0.489653</td>
</tr>
<tr>
<td></td>
<td>Small</td>
<td>0.22755</td>
<td>8.982649</td>
<td>0.339117</td>
</tr>
<tr>
<td>Managerial organization</td>
<td>Large</td>
<td>0.28038</td>
<td>2.141127</td>
<td>0.319624</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>0.41003</td>
<td>5.230089</td>
<td>0.423304</td>
</tr>
<tr>
<td></td>
<td>Small</td>
<td>0.31373</td>
<td>8.841188</td>
<td>0.319604</td>
</tr>
<tr>
<td>Quick response for requirement</td>
<td>Large</td>
<td>0.30612</td>
<td>1.918361</td>
<td>0.327213</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>0.45834</td>
<td>5.875198</td>
<td>0.474934</td>
</tr>
<tr>
<td></td>
<td>Small</td>
<td>0.31533</td>
<td>9.045996</td>
<td>0.318001</td>
</tr>
</tbody>
</table>

Table 6
IF–THEN rules and fuzzy numbers

<table>
<thead>
<tr>
<th>Rule</th>
<th>Fuzzy input variables ( IF )</th>
<th>Fuzzy output (THEN)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Qualitative factors for supplier selection</td>
<td>Fuzzy numbers</td>
</tr>
<tr>
<td></td>
<td>Long-term relationship capability</td>
<td>Technical capability</td>
</tr>
<tr>
<td>1</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>2</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>3</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>4</td>
<td>Small</td>
<td>Medium</td>
</tr>
<tr>
<td>5</td>
<td>Small</td>
<td>Medium</td>
</tr>
<tr>
<td>6</td>
<td>Small</td>
<td>Medium</td>
</tr>
<tr>
<td>7</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>8</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>9</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>10</td>
<td>Medium</td>
<td>Small</td>
</tr>
<tr>
<td>11</td>
<td>Medium</td>
<td>Small</td>
</tr>
<tr>
<td>12</td>
<td>Medium</td>
<td>Small</td>
</tr>
<tr>
<td>13</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>14</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>15</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>16</td>
<td>Medium</td>
<td>Large</td>
</tr>
<tr>
<td>17</td>
<td>Medium</td>
<td>Large</td>
</tr>
<tr>
<td>18</td>
<td>Medium</td>
<td>Large</td>
</tr>
<tr>
<td>19</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>20</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>21</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>22</td>
<td>Large</td>
<td>Medium</td>
</tr>
<tr>
<td>23</td>
<td>Large</td>
<td>Medium</td>
</tr>
<tr>
<td>24</td>
<td>Large</td>
<td>Medium</td>
</tr>
<tr>
<td>25</td>
<td>Large</td>
<td>Large</td>
</tr>
<tr>
<td>26</td>
<td>Large</td>
<td>Large</td>
</tr>
<tr>
<td>27</td>
<td>Large</td>
<td>Large</td>
</tr>
</tbody>
</table>
step of reducing the search domain is ignored to avoid falling into the local optimal. Moreover, computing the radius consumes not only much ram space but also lots of time. That is, the radius criterion is not considered in this instance. The setup of genetic algorithm is as follows and continuous genetic algorithm results are shown in Table 7.

<table>
<thead>
<tr>
<th>Network topology</th>
<th>Crossover rate</th>
<th>Iteration</th>
<th>Best</th>
<th>Second</th>
<th>Third</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-3-1</td>
<td>0.85</td>
<td>50</td>
<td>27.23</td>
<td>27.23</td>
<td>27.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>21.03</td>
<td>21.13</td>
<td>21.13</td>
</tr>
<tr>
<td>4-4-1</td>
<td>0.85</td>
<td>50</td>
<td>148.55</td>
<td>150.21</td>
<td>150.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>139.10</td>
<td>139.44</td>
<td>139.44</td>
</tr>
<tr>
<td>4-5-1</td>
<td>0.85</td>
<td>50</td>
<td>369.24</td>
<td>369.25</td>
<td>369.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>351.97</td>
<td>364.38</td>
<td>364.38</td>
</tr>
<tr>
<td>4-6-1</td>
<td>0.85</td>
<td>50</td>
<td>184.76</td>
<td>329.02</td>
<td>329.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>164.73</td>
<td>187.90</td>
<td>187.90</td>
</tr>
<tr>
<td>4-7-1</td>
<td>0.85</td>
<td>50</td>
<td>125.23</td>
<td>125.24</td>
<td>125.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>124.46</td>
<td>124.46</td>
<td>124.46</td>
</tr>
<tr>
<td>4-8-1</td>
<td>0.85</td>
<td>50</td>
<td>189.72</td>
<td>191.86</td>
<td>191.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>173.63</td>
<td>173.75</td>
<td>173.75</td>
</tr>
<tr>
<td>4-9-1</td>
<td>0.85</td>
<td>50</td>
<td>142.04</td>
<td>142.06</td>
<td>142.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>141.05</td>
<td>141.07</td>
<td>141.07</td>
</tr>
</tbody>
</table>

5.2.2. EBP-type learning

In the above subsection, the initial weights for seven different hidden nodes networks have been obtained. Each adopts three best results. Totally, there are 21 combinations. The training rate was set as 0.3 and 0.6, respectively, while 0.3 and 0.6 are used for the momentum terms. Unfortunately, the MSE results after training by FNN is not better than those of the initial weight obtained by CGA. That is, it is reasonable to retain the CGA and abandon the FNN to prevent consuming too much.

5.3. Decision integration model (ANN)

This subsection emphasizes developing the integration model for considering both the qualitative and quantitative factors affecting supplier selection. Totally, 17 data are obtained from the Q Company in Taiwan. Fourteen data are used for training while the other three data are used for testing purpose. Each datum is composed of 10 components: (1) quality, (2) price, (3) location, (4) finance, (5) facility and productivity, (6) delivery deadline, (7) technical capability, (8) managerial organization, (9) long-term relationship capability, and (10) quick response for requirement. The first six components are crisp values, while the other four components are fuzzy numbers filled by the production managers.

5.3.1. Training ANN

For the 14 training samples, the qualitative factors, all fuzzy numbers, are all inputs to the CGA according to the FNN developed above. Then the FNN results, which are also fuzzy numbers, are \( \alpha \)-cut to obtain the crisp value before being integrated with the other six quantitative factors through a feedforward neural network with the EBP-type learning algorithm.

For the purpose of finding the best \( \alpha \)-cut levels, the following five sets are tested:

1. \( \alpha = \{0.1\} \).
2. \( \alpha = \{0.1, 0.3\} \).
3. \( \alpha = \{0.1, 0.3, 0.5\} \).
4. \( \alpha = \{0.1, 0.3, 0.5, 0.9\} \).
5. \( \alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\} \).

If the qualitative factors are not considered, there are six input nodes. But, if qualitative factors are put into account, each cut generates two crisp values. Thus, the numbers of input nodes can be \( 6 + 0 \), \( 6 + 2 \), \( 6 + 2 \ast 3 \), \( 6 + 2 \ast 4 \) and \( 6 + 2 \ast 5 \).

In order to find the best network, this study first tries different topologies (number of input nodes and number of hidden nodes). Then, different combinations of training rates and momentum terms are tested in order to determine the best training parameters. The network training will not terminate unless the MSE (mean square error) does not change over 500 epochs. The structure of 16-40-1 has the lowest MSE value, 0.00135.

5.3.2. Comparison

In order to illustrate the advantage of the proposed system, the same data are employed to determine the regression model and the MSE value is obtained as well. The results are illustrated in Table 8. It reveals that integrated ANN with both quantitative and qualitative factors outperforms the other three models.

6. Conclusions

In this study, we have presented a novel fuzzy neural network that applies continuous genetic algorithm to learn the relationship between fuzzy inputs and outputs. The FNN is capable of inferring the fuzzy IF-THEN rules. Unlike the traditional fuzzy inference system which needs
crisp inputs as well as crisp outputs, fuzzy numbers or real ones can be input in the FNN. In addition, the initial weights trained by CGA not only can prevent the solutions from falling into local optimal but also help to converge. The solution trained by CGA is more efficient and better than that trained by the real-coded genetic algorithm. This CGA runs with unfixed population size and mutation rate. The two changes accelerate the speed to converge and obtain better solution. In addition, the initial weights obtained by the CGA are better than those derived from the real-coded GA. The more promising finding is that only using the CGA without being fine-tuned by the EBP-type learning algorithm is enough. This is not like the real-coded GA proposed in [13] which has to combine both real-coded GA and EBP-type learning algorithm in order to obtain better results. Definitely, some other learning methods can be employed for the current FNN, like extreme learning machine (ELM) [9,28] which is able to prevent the local minima. In addition to the application of supplier selection, researchers can also apply it to other decision-making problems, such as determination of contractors and forecasting systems.

### Table 8

<table>
<thead>
<tr>
<th></th>
<th>MSE</th>
<th>Regression</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantitative factors only (6 input nodes)</td>
<td>0.004937</td>
<td>0.00135</td>
<td></td>
</tr>
<tr>
<td>Quantitative and qualitative factors (16 input nodes)</td>
<td>0.004391</td>
<td>0.00135</td>
<td></td>
</tr>
</tbody>
</table>

References

S.M. Hong received his Ph.D. degree in Industrial Engineering from Arizona State University, Tempe, AZ, in 1998. Prior to the academic services, he had several years of working experience in the fields of supply chain management, production management, and distribution systems. He is currently an Assistant Professor in the Department of Business Administration at National Chengchi University, Taiwan, ROC. His research interests include supply chain management, system simulation and analysis, production control system, and applications of operations research.

Y. Lin received the M.S. degree in Industrial Engineering and Management from National Taipei University of Technology, Taiwan in 2005. His research interests include architecture issues of neural networks, fuzzy sets and genetic algorithms and their application to supply chain management.

Y.C. Huang received the M.S. degree from Kaohsiung Polytechnic Institute, Taiwan, ROC in 1997 and the Ph.D. degree in Financial Management from the National Chung Cheng University, Taiwan, ROC in 2007. Currently, he is Instructor in the Department of Accounting, I-Shou University, Taiwan, ROC. His research interests include architecture issues of financial management, financial engineer, investment, futures and options, neural networks, fuzzy logic, and genetic algorithms.