GA-based Feed-forward Self-organizing Neural Network Architecture and Its Applications for Multi-variable Nonlinear Process Systems

Sung-Kwun Oh[1], Ho-Sung Park[1], Chang-Won Jeong[2] and Su-Chong Joo[2]
[1] Department of Electrical Engineering, University of Suwon,
San 2-2 Wau-ri, Bong-dam-eup, Hwaseong-si, Gyeonggi-do, 445-743, South Korea
[e-mail: {ohsk, parkhs}@suwon.ac.kr]
[2] Department of Computer Engineering, Wonkwang University,
344-2, Shinyong-Dong, Iksan, Chon-Buk, 570-749, South Korea
[e-mail: {mediblue, scjoo}@wku.ac.k]
*Corresponding author: Su-Chong Joo

Received May 3, 2009; revised June 10, 2009; accepted June 14, 2009; published June 22, 2009

Abstract

In this paper, we introduce the architecture of Genetic Algorithm (GA) based Feed-forward Polynomial Neural Networks (PNNs) and discuss a comprehensive design methodology. A conventional PNN consists of Polynomial Neurons, or nodes, located in several layers through a network growth process. In order to generate structurally optimized PNNs, a GA-based design procedure for each layer of the PNN leads to the selection of preferred nodes (PNs) with optimal parameters available within the PNN. To evaluate the performance of the GA-based PNN, experiments are done on a model by applying Medical Imaging System (MIS) data to a multi-variable software process. A comparative analysis shows that the proposed GA-based PNN is modeled with higher accuracy and more superb predictive capability than previously presented intelligent models.

Keywords: Feed-forward self-organizing neural networks (FSONN), genetic algorithms, group method of data handling (GMDH), medical imaging system

The priminary work of this paper was presented in PRICAI 2006: *Trends in Artificial Intelligence*, *9th Pacific Rim International Conference on Artificial Intelligence*, Guilin, China, August 7-11, 2006. This research was supported by the research fund of Wonkwang University in 2008.

DOI: 10.3837/tiis.2009.03.006

1. Introduction

Recently, much attention has been directed towards the advanced techniques of system modeling. The panoply of existing methodologies and ensuing detailed algorithms are confronted with nonlinear systems and extreme problem dimensionality on a quest for high accuracy and generalization capabilities of the ensuing models.

As the complexity of the system to be modeled increases, experimental data and some degree of prior domain knowledge conveyed by the model developer are essential to the completion of an efficient design procedure. It is also worth stressing that the nonlinear form of the model acts as a two-edged sword: while we gain flexibility to cope with experimental data, we are provided with an abundance of nonlinear dependencies that need to be exploited in a systematic manner. In particular, when dealing with high-order nonlinear and multivariable equations of the model, we require a vast amount of data to estimate its complete range of parameters [1][2]. To help alleviate such problems, Feed-forward Self-organizing Neural Networks (FSONN) were introduced by Oh et al. [3][4][5][6][18] as a new class of networks. These networks come with a high level of flexibility as each processing element forming a node can have a different number of input variables and exploit a different order of polynomials. Although the FSONN contains flexible model architecture with higher accuracy due to its systematic design procedure, it is difficult to obtain a structurally and parametrically optimized network because of the limited design of neurons located in each layer of FSONN. Accordingly, the FSONN algorithm tends to produce overly complex networks a repetitive computational load by the trial and error method and/or a repetitive parameter adjustment by the designer, as in the case of the original GMDH algorithm.

In this study, in solving the problems with the conventional FSONN as well as the GMDH algorithm, we introduce a new design approach of GA-based FSONN. Optimal design parameters available within the node lead to a structurally and parametrically optimized network, which is more flexible and simpler in architecture than the conventional FSONN. Furthermore, we introduce an aggregate objective function that deals with training data and testing data and elaborate on its optimization to produce a meaningful balance between the approximation and generalization abilities of the model. In a nutshell, the objective of this study is to develop a general design methodology of GA-based FSONN modeling, come up with a logic-based structure for such a model and propose a comprehensive evolutionary development environment in which the optimization of the models can be efficiently carried out both at the structural and at the parametric levels [7]. To evaluate the performance of the proposed model, we exploit MIS data [8] and pH neutralization process data [9].

2. The FSONN algorithm and its generic architecture

2.1 Polynomial Neuron (PN) based FSONN and its architecture

2.1.1 Polynomial Neuron (PN)

The PN-based FSONN algorithm [3][4][5] is based on the GMDH [10] method and utilizes a class of linear, quadratic, and modified quadraticpolynomals which can be seen in **Table 1**. By choosing the most significant input variables and a certain order of the polynomials among the available variety of structures at our disposal, we can construct the best partial description (PD) of a polynomial neuron (PN). The individual PNs are expressed as a second-order

regression equation. In particular, when combining two inputs at each node as the generic structure we arrive at the following relationship:

$$y = A + BX_i + CX_j + DX_i^2 + EX_j^2 + FX_iX_j \quad (1)$$

In the above expression, *A*, *B*, *C*, *D*, *E*, and *F* are parameters of the model, while *y* is the output of this model. *Xi* and *Xj* denote the two inputs.

The outputs obtained from each of these nodes are then combined to obtain a higher-degree polynomial. In this case, a complex Ivakhnenko polynomial is formed. This function usually takes on the form

$$y = A + \sum_{i=1}^{n} B_iX_i + \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_iX_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}X_iX_jX_k \cdots \quad (2)$$

where *Xi*, *Xj* and *Xk* are nodal input variables and *y* is the output of an individual neuron, or node. *A*, *Bi*, *Cij*, and *Dijk* are the coefficients of the Ivakhnenko polynomial.

The PN-based FSONN design activities have focused over the past years on the development of self-organizing, minimal polynomial networks with good generation capabilities. Searching for the optimal configuration in the space of all possible Feed-forward Self-Organizing Neural Networks is intractable and requires the application of certain heuristic rules. The PN-based FSONN leads to self-organizing heuristic hierarchical models of high degree equipped with an automatic elimination of undesirable variable interactions.

2.1.2 Polynomial Neuron (PN) based FSONN architecture

The PN-based FSONN based on a perceptron learning principle with neural network-type architecture is used to model the input-output relationship of a complex process system. The design of the PN-based FSONN structure continues and involves the generation of some additional layers. Each layer consists of nodes (PDs or PNs) for which the number of input variables could be the same as in the previous layers or may differ across the network as shown in **Fig. 1**.

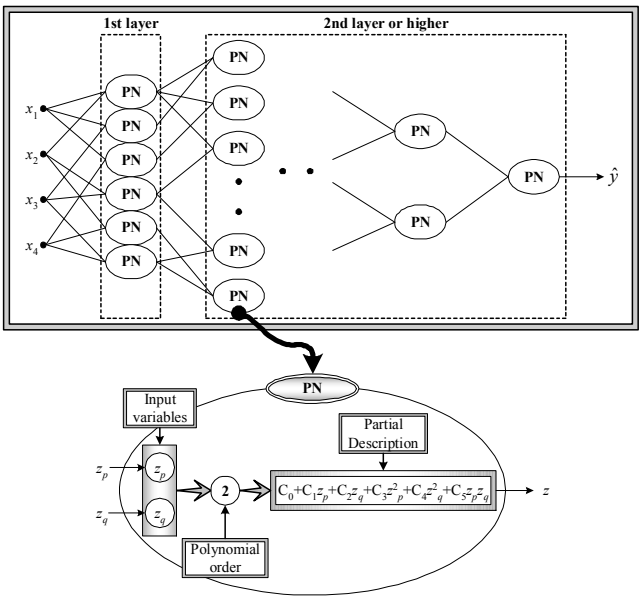


Fig. 1. A general architecture of PN-based FSONN

At each layer, new generations of complex equations are constructed from simple forms. The model obtained at each subsequent layer is progressively more complex than the model at the preceding layers. To avoid an overfit, the overall data set is divided into two segments. The first training set, which is used for the generation of several computing alternative models. The second is the testing set, which is used to test the accuracy of each model generated and for the selection of the best models at each layer. The number of layers is increased until the newer models begin to exhibit weaker predictability than their predecessors. This indicates the overfitting of the model. The final model is defined as a function of two, three, or four variables. The network result is a very sophisticated model obtained from a very limited data set.

We introduce two types of generic PN-based FSONN structures, namely the basic and the modified PN-based FSONN. Moreover, for each type of topology we identify two schemes [3][4][5]. In what follows, the PN-based FSONN emerges as a versatile architecture whose topology depends on the regression polynomial of a PN.

Table 1. Different forms of regression polynomials forming a PN

Order of the polynomial \ Number of inputs	1	2	3
1 (Type 1)	Linear	Bilinear	Trilinear
2 (Type 2)	Quadratic	Biquadratic-1	Triquadratic-1
2 (Type 3)	Quadratic	Biquadratic-2	Triquadratic-2

1 : Basic type 2 : Modified type

2.2 Fuzzy Polynomial Neuron (FPN) based FSONN and its topology

2.2.1 Fuzzy polynomial neuron (FPN)

As visualized in **Fig. 2**, the FPN consists of two basic functional modules. The first labeled F, is a collection of fuzzy sets ($\{A_l\}$ and $\{B_k\}$) that form an interface between the input and the processing segment realized by the neuron. The second module denoted here by P ,refer to function-based nonlinear (polynomial) processing. This nonlinear processing involves some input variables x_i and x_j. Quite commonly, we will be using the polynomial form of the nonlinearity, hence the name of the fuzzy polynomial processing unit. The use of polynomials is motivated by their generality. In particular, the ones used include constant and linear mappings. These are used quite often in rule-based systems.

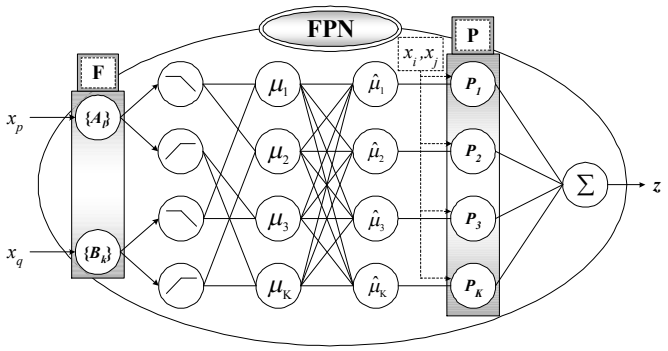


Fig. 2. The general architecture of the generic FPN module

In other words, FPN realizes a family of multiple-input single-output rules. Each rule has the form

$$IF\ x_p\ is\ A_l\ and\ x_q\ is\ B_k\ THEN\ z\ is\ P_{lk}(x_i, x_j, a_{lk}) \tag{3}$$

where a_{lk} is a vector of the parameters of the conclusion part of the rule while $P_{lk}(x_i, x_j, a_{lk})$ denotes the regression polynomial forming the consequence part of the fuzzy rule which uses several types of high-order polynomials in addition the constant function forming the simplest version of the consequence. You can see this in **Table 2**.

Alluding to the input variables of the FPN, especially a way in which they interact with the two functional blocks shown there, we use the notation FPN (x_p, x_q; x_i, x_j) to explicitly point at the variables. The processing of the FPN is governed by the following expressions that are in line with the rule-based computing existing in the literature [11][12].

2.2.2 Fuzzy Polynomial Neuron (FPN) based FSONN architecture

The topology of the FPN-based FSONN implies the ensuing learning mechanisms; in the description below we indicate some of these learning issues that permeate the overall architecture. First, the network is homogeneous in the sense that it is constructed with the use of FPNs. It is also heterogeneous in the sense that FPNs can be very different and this contributes to the generality of the architecture. The network may contain a number of hidden layers each of them of a different size, which is to say having different of nodes. The nodes may have a different number of inputs and this triggers a certain pattern of network connectivity. FPN itself promotes a number of interesting design options, see **Fig. 3**.

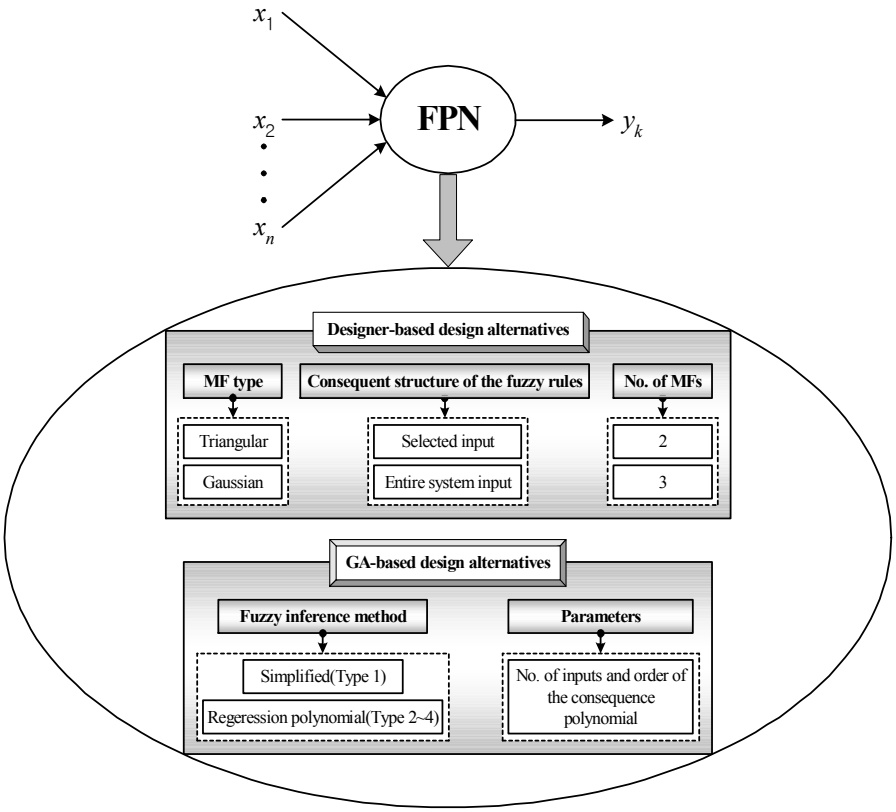


Fig. 3. The design alternatives available within a single FPN

These alternatives distinguish between two categories, a design-based category and a GA-based category. The design-based category concerns a choice of the membership function (MF) type, the consequent input structure of the fuzzy rules, and the number of MFs for each input variable. The latter is related to a choice of the number of inputs, the collection of a specific subset of input variables, and its associated polynomial order. This is done after realizing a consequence of the rules based on the fuzzy inference method.

Proceeding with the FPN-based FSONN architecture, essential design decisions have to be made with regard to the number of input variables, the order of the polynomial forming the conclusion of the rules, and a collection of the specific subset of input variables. The consequence segment can be expressed by linear, quadratic, or modified quadratic polynomial equations as mentioned previously. Especially for the consequence segment, we especially consider two kinds of input vector formats in the conclusion segment of the fuzzy rules of the 1st layer, namely i) selected inputs and ii) entire system inputs, as seen in **Table 3**.

i) The input variables of the consequence segment of the fuzzy rules are same as the input variables of premise segment.

ii) The input variables of the consequence segment of the fuzzy rules in a node of the 1st layer are same as the entire system input variables and the input variables of the consequence segment of the fuzzy rules in a node of the 2nd layer or higher are same as the input variables of premise segment.

Table 3. Polynomial type according to the number of input variables in the fuzzy rules conclusion segment

Type of the Consequence polynomial \ Input vector	Selected input varialbes in the premise part	Selected input varialbes in the consequence part	Entire system input variables
Type T	A	A	B
Type T*	a	b	B

Where notation A: Vector of the selected input variables (x_1, x_2,…, x_i), B: Vector of all system input variables (x_1, x_2, …x_i, x_j …), Type T: f(A)=f(x_1, x_2,…, x_i) - type of a polynomial function standing in the consequence segment of the fuzzy rules, Type T*: f(B)=f(x_1, x_2, …x_i, x_j …) - type of a polynomial function occurring in the consequence segment of the fuzzy rules

As shown in **Table 3**, A and B describe vectors of the selected input variables and the entire collection of input variables respectively. Proceeding with each layer of the FPN-based FSONN, the design alternatives available within a single FPN can be carried out with regard to the entire collection of the input variables or its selected subset as they occur in the consequence segment of fuzzy rules encountered at the 1st layer. Following these criteria, we distinguish between two fundamental types (Type T, Type T*), namely Type T- the input variables in the conditional part of fuzzy rules are kept as those in the conclusion segment of the fuzzy rules (Z_i of (A) FPN (x_p, x_q: x_i, x_j)). Type T*- the entire collection of the input variables is kept as input variables in the conclusion segment of the fuzzy rules (Z_i of (B) FPN (x_p, x_q: x_1, x_2, x_3, x_4)).

3. Genetic optimization of FSONN

When we construct the PNs or FPNs of each layer in the conventional FSONN, such parameters as the number of input variables, the order of polynomials, and input variables

available within a PN are selected in advance by the designer. That is, the designer must have pre-determined information related to the networks such as the number of system input variables and the polynomial order. Because the conventional FSONN is a heuristic method, it does not guarantee that the constructed FSONN is optimal network architecture. Accordingly, in order to generate a structurally and parametrically optimized FSONN network, such parameters need to be optimal.

In order to solve this problem, we use genetic algorithms that are a stochastic global search technique based on the principles of evolution, natural selection and genetic recombination by simulating survival of the fittest in a population of potential solutions to the problem at hand [13][14][15][16].

In this study, for the optimization of the FSONN model, GAs use the serial method of binary type, a roulette wheel in the selection operator, a one-point crossover in the crossover operator, and an invert in the mutation operator. As the roulette-wheel operator's stochastic characteristic, when creating a new generation by selection operators, we will choose the best chromosome from the last generation. To reduce the stochastic errors of roulette wheel selection, we use an elitist strategy [15]. The overall structural optimization process of FSONN using GAs is shown in **Fig. 4-5**.

As mentioned, when we construct PNs or FPNs of each layer in the conventional FSONN, such parameters as the number of input variables, the order of the polynomial and input variables available within a PN or a FPN are fixed in advance by the designer. This could have frequently contributed to the difficulties in the design of an optimal network. To overcome this apparent drawback, we resort to genetic optimization.

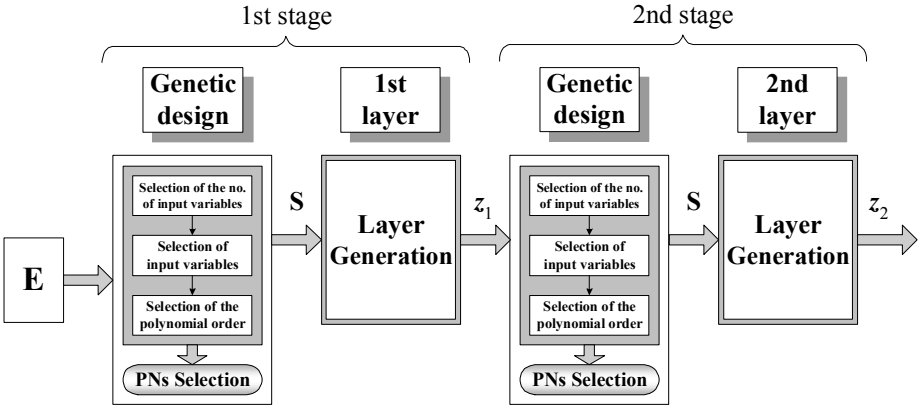
4. GA-based FSONN algorithm

The framework of the design procedure of Feed-forward PNNs (Editor's note: Already defined earlier, no need to do it again) consists of the following steps:

[Step 1] Determine system's input variables

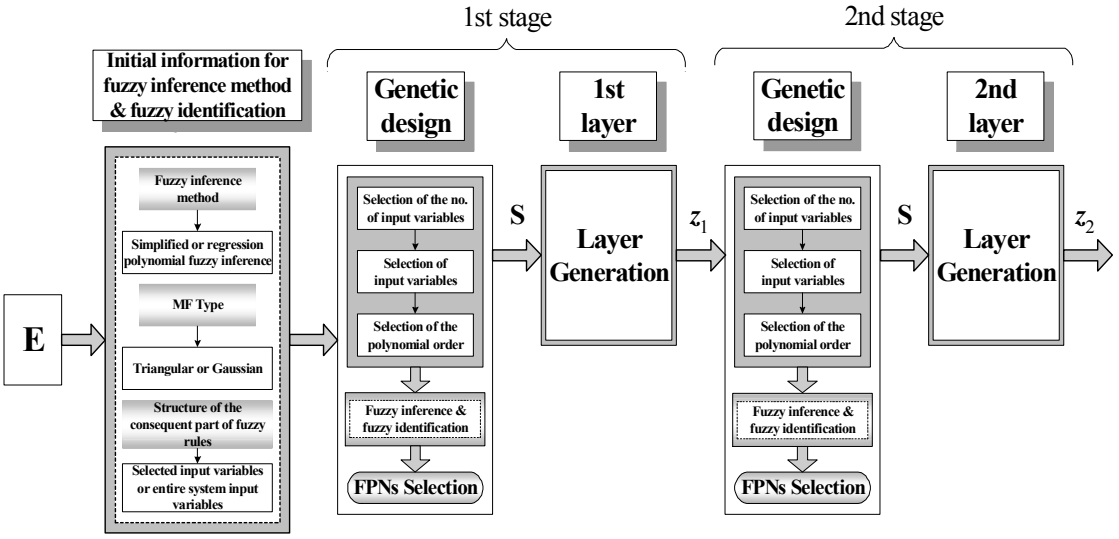
Define the system's input variables as $x_i(i=1, 2, \ldots, n)$ related to output variable y. If required, the data can be normalized as well.

[Step 2] Form training and testing data



E : Entire inputs, S : Selected PNs, z_i : Preferred outputs in the *i*th stage(z_i=z_{1i}, z_{2i}, …, z_{Wi})

Fig. 4. Overall structural optimization process of PN-based FSONN using Gas



E : Entire inputs, S : Selected FPNs, z_i : Preferred outputs in the *i*th stage(z_i=z_{1i}, z_{2i}, ..., z_{Wi})

Fig. 5. Overview genetically-driven structural optimization process of FPN-based FSONN

The input-output data set $(x_i, y_i)=(x_{1i}, x_{2i}, \ldots, x_{ni}, y_i)$, $i=1, 2, \ldots, N$ where N is the total amount of data is divided into two parts, that is, training and testing datasets. Denote their sizes by N_t and N_c respectively. Obviously we have $N=N_t+N_c$. The training data set is used to construct the FSONN model. Next, the testing data set is used to evaluate the quality of the model.

[Step 3] Determine the initial information for constructing the FSONN structure

We determine the initial information for the FSONN structure in the following manner:

a) According to the stopping criterion, two termination methods are exploited here:
- Comparison of a minimal identification error of the current layer with that of the previous layer of the network
- The maximum number of generations predetermined by the designer to achieve a balance between model accuracy and its complexity

b) The maximum number of input variables arriving at each node in the corresponding layer

c) The total number W of nodes to be retained (selected) at the next generation of the FSONN algorithm

d) The value of the weighting factor of the aggregate objective function

[Step 4] Decide a structure of the PN or FPN-based FSONN using genetic design

The concerns are with the selection of the number of input variables, the polynomial order, and the input variables to be assigned in each node of the corresponding layer. We determine the structure of the PN or FPN-based FSONN using genetic design.

When it comes to the organization of the chromosome representing the structure of the FSONN, we divide the chromosome to be used for genetic optimization into three sub-chromosomes as shown in **Fig. 6-7**. The 1st sub-chromosome contains the number of input variables, the 2nd sub-chromosome involves the order of the polynomial of the node, and the 3rd sub-chromosome contains input variables for the corresponding node. All these elements are optimized when running the GA.

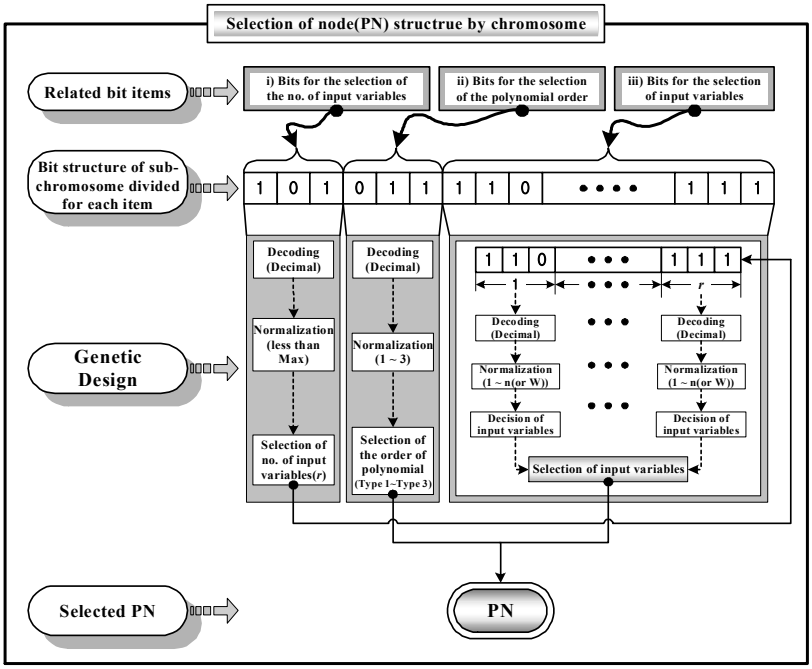


Fig. 6. The PN design available in FSONN architecture by using a GA chromosome

Each sub-step of the genetic design procedure of three kinds of parameters available within the PN or the FPN is structured as follows:

[Step 4-1] Selection of the number of input variables (1st sub-chromosome)

[Step 4-2] Selection of the order of polynomials (2nd sub-chromosome)

[Step 4-3] Selection of input variables (3rd sub-chromosome)

[Step 5] Estimate the coefficients of the polynomial corresponding to the selected node (PN or FPN)

[Step 5-1] In case of a PN

The vector of coefficients C_i is derived by minimizing the mean squared error between y_i and z_{mi}.

The vector of coefficients Ci is derived by minimizing the mean squared error between yi and zmi.

$$E = \frac{1}{N_{tr}} \sum_{i=0}^{N_{tr}} (y_i - z_{mi})^2 \tag{4}$$

Using the training data subset, this gives rise to the set of linear equations

$$\mathbf{Y} = \mathbf{X}_i \mathbf{C}_i \tag{5}$$

Evidently, the coefficients of the PN of nodes in each layer are determined by the standard least square method.

[Step 5-2] In case of a FPN

At this step, the regression polynomial inference is considered. The inference method deals with regression polynomial functions viewed as the consequents of the rules. Regression polynomials standing in the conclusion segment of fuzzy rules are given as different types of Type 1, 2, 3, or 4, see **Table 2**. In the fuzzy inference, we consider two types of membership functions, namely triangular and Gaussian-like membership functions.

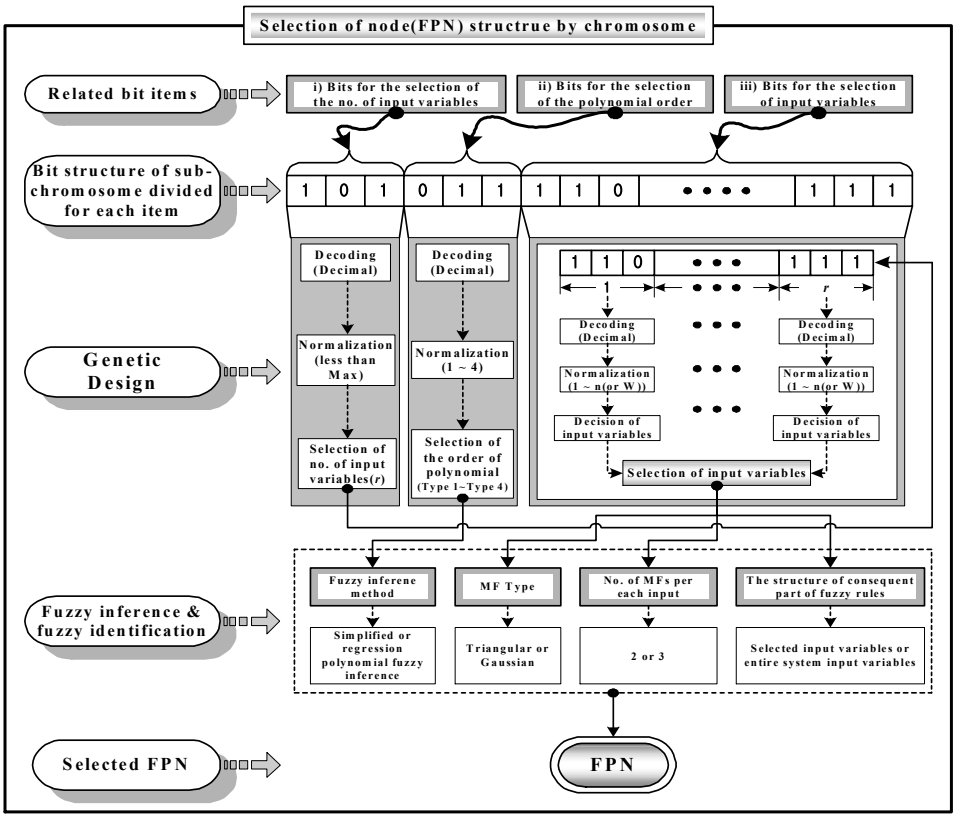
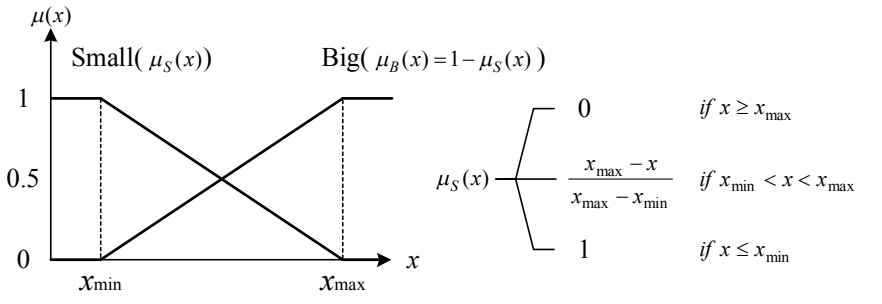
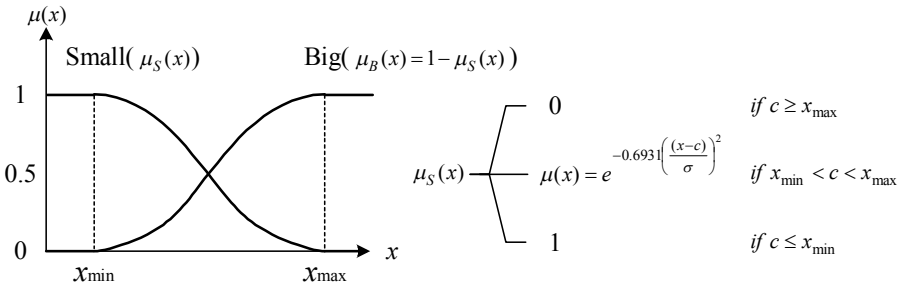


Fig. 7. The genetic FPN design used in FSONN architecture



(a) Triangular MF



(b) Gaussian-like MF

Fig. 8. Triangular and Gaussian-like membership function and their parameters

The regression fuzzy inference reasoning scheme is envisioned: The consequence segment can be expressed by constant, linear, quadratic, or modified quadratic polynomial equations as shown in **Table 2**. The use of the regression polynomial inference method gives rise to the expression

$$R^i : IF\ x_1\ is\ A_{i1}, \cdots, and\ x_k\ is\ A_{ik}\ THEN\ y_i = f_i(x_1, x_2, \cdots, x_k) \tag{6}$$

where R^i is the i-th fuzzy rule, $x_l(l=1, 2, \ldots, k)$ is an input variable, A_{ik} is a membership function of fuzzy sets, k denotes the number of the input variables, $f_i(\bullet)$ is a regression polynomial function of the input variables.

The calculation of the numeric output of the model is carried out in the well-known form

$$\hat{y} = \frac{\sum_{i=1}^{n} \mu_i f_i(x_1, x_2, \cdots, x_k)}{\sum_{i=1}^{n} \mu_i} = \sum_{i=1}^{n} \hat{\mu}_i f_i(x_1, x_2, \cdots, x_k) \tag{7}$$

where, n is the number of fuzzy rules, $\hat{y}$ is the inferred value, μ_i is the premise fitness of R^i and $\hat{\mu}_i$ is the normalized premise fitness of μ_i.

Here we consider the regression polynomial function of the input variables. The consequence parameters are produced by the standard least squares method.

This procedure is implemented repeatedly for all nodes of the layer and also for all FSONN layers, starting from the input layer and moving to the output layer.

[Step 6] Select nodes with the best predictive capability, and construct their corresponding layer

As shown in **Fig. 6-7**, all nodes of the corresponding layer of FSONN architecture are constructed by genetic optimization. The generation process of PNs or FPNs in the corresponding layer is described in detail as a design procedure of 9 sub-steps. The sequence of sub-steps is as follows:

Sub-step 1) We determine the initial genetic information for the generation of the FSONN architecture. That is, the number of generations and individuals, mutation rate, crossover rate, and the length of a chromosome.

Sub-step 2) The nodes are generated by genetic design. There are as many variations as the number of individuals in the 1st generation. One individual takes the same role as one node in the FSONN architecture and each individual is operated by GAs as shown in **Fig. 6-7**. That is, the number of input variables, the order of the polynomials, and the input variables as one individual are selected by GAs. The polynomial parameters are produced by the standard least squares method.

Sub-step 3) To evaluate the performance of PNs in each population, we use an aggregate objective function that takes into account a sound balance between approximation and prediction capabilities of the one as shown in Eq. (5). Then from the performance index obtained in Eq. (5), we calculate the fitness function of Eq. (6). The objective function, or cost function, is employed to decrease the error rate and to increase the predictability of the model - that is, the objective function includes the performance index for training (PI) and the performance index for evaluation (EPI), that are combined by means of a weighting factor θ. The objective function is a basic instrument guiding the evolutionary search in a solution space [12]. The objective function includes both the training data and the testing data and comes as a convex sum of two components.

$$f(PI, EPI) = \theta \times PI + (1-\theta) \times EPI \tag{8}$$

We define the fitness function of the genetic algorithm as follows:

$$F(\textit{fitness function}) = \frac{1}{1 + f(PI, EPI)} \tag{9}$$

PI and *EPI* denote the performance index for the training data and testing data respectively. Moreover, θ is a weighting factor that allows us to strike a balance between the performances of the models for training and testing data. The aggregate object function depends upon the values of a weighting factor. Both *PI* and *EPI* are considered and the proper selection of θ establishes the direction of optimization to maintain a balance between approximation and generalization. Model selection is performed from the minimization of this aggregate objective function.

Sub-step 4) To produce the next generation, we carry out selection, crossover, and mutation operations using initial genetic information and the fitness values obtained from sub-step 3.

Sub-step 5) The nodes are rearranged in descending order on the basis of the calculated fitness values (F_1, F_2, …, F_z). We unify the nodes with duplicated fitness values. In the case that more than one node has the same fitness value, among the rearranged nodes on the basis of the fitness values. We choose several nodes characterized by the best fitness values. Here, we use the pre-defined number *W* of nodes with better predictive capability that must be preserved for optimal operation of the next iteration in the PNN algorithm. The outputs of the retained nodes serve as inputs in the subsequent layer of the network. There are two cases as to the number of the retained nodes, that is,

(i) If $z<W$, then the number of the nodes retained for the next layer is equal to z.

(ii) If $z \geq W$, then for the next layer, the number of the retained nodes is equal to W.

Sub-step 6) For the elitist strategy, we select the node that has the highest fitness value among the selected nodes (W).

Sub-step 7) We generate new individuals of the next generation using operators of GAs obtained from sub-step 4. Then we use the elitist strategy. This sub-step carries out by repeating sub-steps 2-6. Especially in sub-step 5, we replace the node that has the lowest fitness value in the current generation with the node that has the highest fitness value in the previous generation obtained from sub-step 6.

Sub-step 8) We combine the nodes (W individuals) obtained in the previous generation with the nodes (W individuals) obtained in the current generation. In the sequel, W nodes that have higher fitness values among them ($2W$) are selected. That is, this sub-step carries out by repeating sub-step 5.

Sub-step 9) Until the last generation, this sub-step carries out by repeating sub-steps 7-8. The iterative process generates the optimal nodes of a layer in the FSONN model.

[Step 7] Check the termination criterion

The termination condition that controls the growth of the model consists of two components, the performance index and the size of the network, expressed in terms of the maximal number of layers. As far as the performance index is concerned that reflects a numeric accuracy of the layers, termination is straightforward and comes in the form:

$$F_1 \leq F_* \tag{10}$$

Where, *F1* denotes a maximal fitness value occurring at the current layer whereas F^* stands for a maximal fitness value that occurred at the previous layer. As far as the depth of the network is concerned, the generation process is stopped at a depth of less than five layers. This

size of the network has been experimentally found to form a sound compromise between the high accuracy of the resulting model and its complexity and generalization abilities.
In this study, we use the Mean Squared Error (MSE) for measure of performance index.

$$E(PI_s \text{ or } EPI_s) = \frac{1}{N}\sum_{p=1}^{N}(y_p - \hat{y}_p)^2 \tag{11}$$

where y_p is the *p-th* target output data and $\hat{y}_p$ stands for the *p-th* actual output of the model for this specific data point. N is training (*PIs*) or testing (*EPIs*) input-output data pairs and E is an overall performance index defined as a sum of the errors for the N.
[Step 8] Determine new input variables for the next layer
If Eq. (10) has not been satisfied, the model has to be expanded. The outputs of the preserved nodes (z_{1i}, z_{2i}, …, z_{Wi}) serves as new inputs to the next layer (x_{1j}, x_{2j}, …, x_{Wj}) (j=i+1). This is captured by the expression.

$$x_{1j} = z_{1i}, x_{2j} = z_{2i}, \ldots, x_{wj} = z_{wi} \tag{12}$$

The GA-based FSONN algorithm is carried out by repeating steps 4-8 consecutively.

5. Experimental Studies

In this section, we illustrate the development of the GA-based FSONN and show its performance for a number of well-known and widely used datasets. The first one is a Medical Imaging System data which was studied previously in software process modeling [8]. The other one deals with pH neutralization process data [9].

5.1 Medical Imaging System Data

This section includes comprehensive numeric studies illustrating the design of the GA-based PNN model. We use a well-known MIS [8] data.
This data set concerns a MIS data set which involves 390 software modules written in Pascal and FORTRAN. These modules consist of approximately 40,000 lines of code. To design an optimal model from the MIS, we study 11 system input variables such as, Total lines of code including comments (*LOC*), Total code lines (*CL*), Total character count (*TChar*), Total comments (*TComm*), Number of comment characters (*MChar*), Number of code characters (*DChar*), Halstead's program length (N), Halstead's estimated program length ($\hat{N}$), Jensen's estimator of program length (N_F), McCabe's cyclomatic complexity ($V(G)$), and Belady's bandwidth metric (*BW*). The output variable of the model is the number of reported changes - Change Reports (*CRs*). In case of the MIS data, the performance index is defined as the mean squared error (MSE) as Eq. (11).
Table 4 summarizes the list of parameters used in the genetic optimization of the PN-based and the FPN-based FSONN.

Table 4. Computational aspects of the genetic optimization of PN-based and FPN-based FSONN

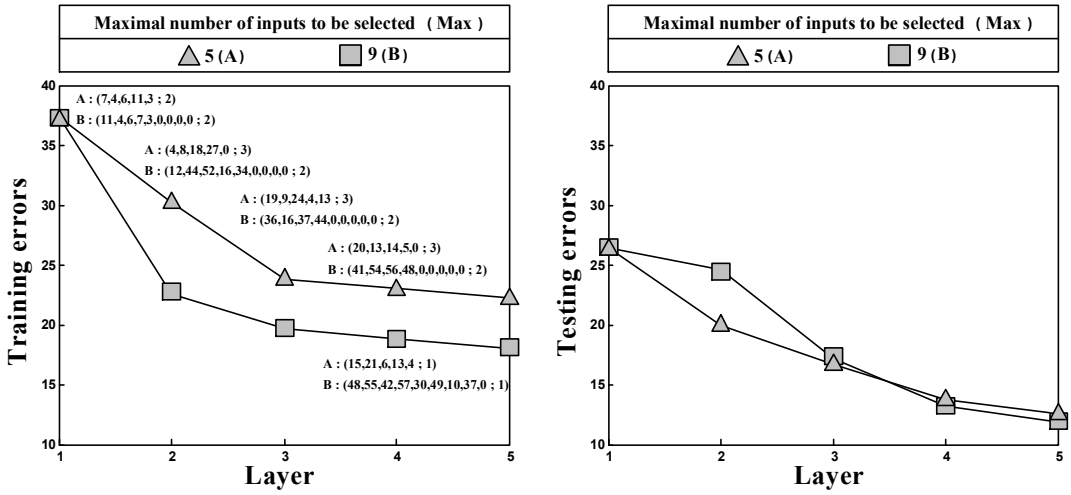
	Parameters		1st layer	2nd layer	3rd layer	4th layer	5th layer
GA	Maximum generation	PN	300				
		FPN	100				
	Total population size	PN	150				
		FPN	60				
	Selected population size (W)	PN	30(l=2~5), 60(l=9)				
		FPN	30(l=4)				

	String length	PN	3+3+30(l=2~5), 5+3+60 (l=9)
		FPN	3+3+30(l=4)
	Crossover rate		0.65
	Mutation rate		0.1
PN based FSONN	Maximal no. (Max) of inputs to be selected		1≤l≤Max (2~5, 9)
	Polynomial type (Type T) ($^{\#}$)		1≤T≤3
FPN based FSONN	Maximal no. (Max) of inputs to be selected		1≤l≤Max (2~4)
	Polynomial type (Type T) of the consequent part of fuzzy rules ($^{\#\#}$)		1≤T≤4
	Consequent input type to be used for Type T ($^{\#\#\#}$)	Type T*	Type T Type T
	Membership Function (MF) type		Triangular, Gaussian
	No. of MFs per input		2

l, T, Max: integers, $^{\#}$, $^{\#\#}$and $^{\#\#\#}$: refer to Tables 1-3 respectively.

5.1.1 Polynomial Neuron (PN) based FSONN

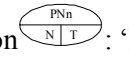
Fig. 9 depicts the performance index of a PN-based FSONN according to the maximal number of inputs to be selected when using θ=0.5 and Max=5, 9.



(a) Training data (b) Testing data

Fig. 9. Performance index of a PN-based FSONN according to the maximal number of inputs to be selected (θ=0.5 and Max=5, 9)

Fig. 10 illustrates the detailed optimal topologies of the PN-based FSONN for 3 layers. As shown in **Fig. 10**, the GA-based design procedure at each stage (layer) of PN-based FSONN leads to the selection of preferred nodes (or PNs) with local characteristics (such as the number of input variables, the order of the polynomial, and input variables) available within the PN-based FSONN. In the sequel, the proposed network enables the architecture to be a structurally more optimized and flexible network than the conventional PN-based FSONN.

Referring to **Fig. 10**, we adhere to the following notation 

: 'PNn' denotes the nth node (PN) of the corresponding layer, 'N' denotes the number of nodes (inputs or PNs) coming to the corresponding node, and 'T' denotes the polynomial order used in the corresponding node.

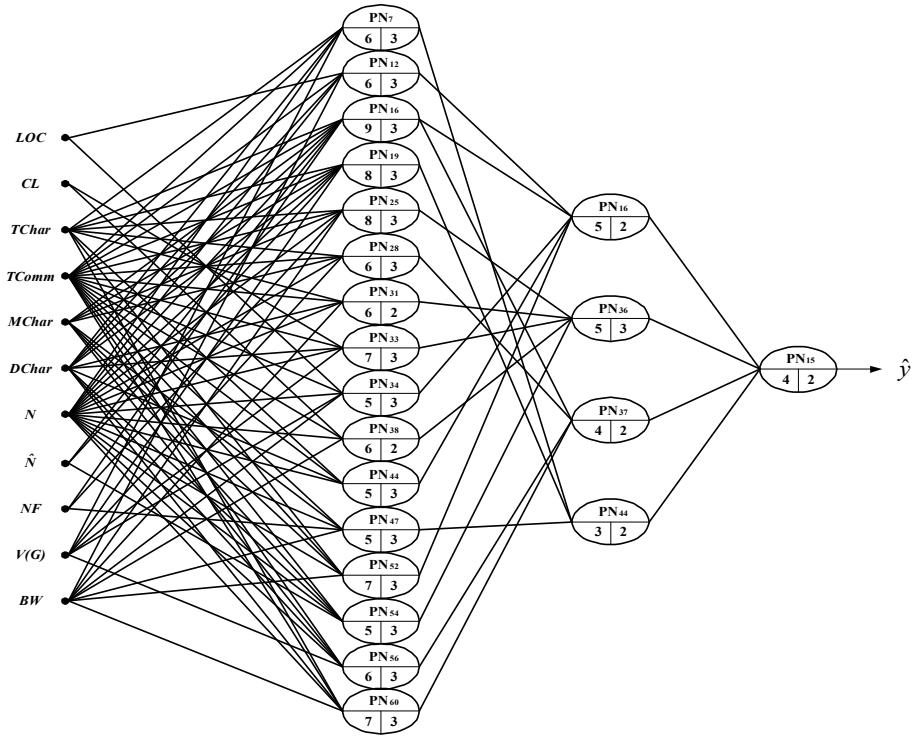
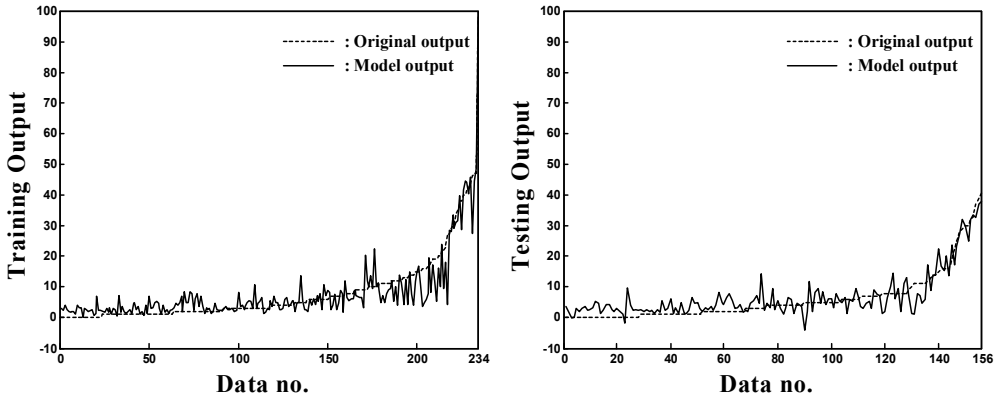


Fig. 10. Optimal networks structure of PN-based FSONN with 3 layers (θ=0.5 and Max=9)

Figs. 11-12 show output comparison and identification errors for the optimal network architecture visualized when using 5th layer, θ=0.5 and Max=9.

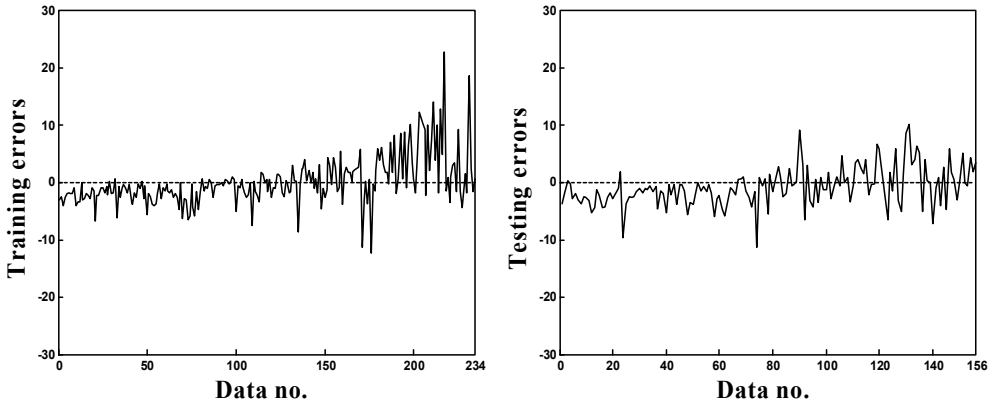
5.1.2 Fuzzy Polynomial Neuron (FPN) based FSONN

Table 5 summarizes the results when using Types T and T*: According to the maximal number of inputs to be selected (Max=2 to 5), the selected polynomial type (T), and its corresponding performance index (PI and EPI) were shown when the genetic optimization for each layer was carried out.



(a) Training data (b) Testing data

Fig. 11. Original output and model output of MIS data



(a) Training data (b) Testing data

Fig. 12. Errors curve of genetically designed PN-based FSONN

Table 5. Performance index of the network of each layer versus the increase of maximal number of inputs to be selected

(a) In case of Type T

(a-1) Triangular

Max	1st layer			2nd layer			3rd layer			4th layer			5th layer		
	T	PI	EPI	T	PI	EPI	T	PI	EPI	T	PI	EPI	T	PI	EPI
2	1	53.125	29.660	1	45.199	20.724	2	39.693	16.286	1	32.401	14.673	4	30.371	13.887
3	1	45.305	23.002	1	39.312	16.852	4	35.816	13.699	1	33.750	11.630	3	28.095	9.841
4	1	39.894	17.070	3	36.852	14.636	3	29.962	12.900	4	33.824	11.262	3	28.691	10.071
5	1	39.894	17.070	3	36.852	14.636	3	34.848	12.389	1	24.545	9.264	1	23.739	9.080

(a-2) Gaussian-like

Max	1st layer			2nd layer			3rd layer			4th layer			5th layer		
	T	PI	EPI	T	PI	EPI	T	PI	EPI	T	PI	EPI	T	PI	EPI
2	1	50.656	31.288	1	50.252	19.672	2	39.890	17.843	4	32.627	12.554	2	31.645	11.217
3	1	49.254	24.982	1	41.005	17.618	1	37.149	12.840	1	32.091	11.761	4	26.328	11.342
4	1	49.254	24.982	1	39.886	15.982	1	32.550	13.956	1	32.512	12.565	1	24.017	10.738
5	1	49.254	24.982	1	41.034	16.762	1	35.719	12.865	1	37.268	11.765	1	36.571	10.289

(b) In case of Type T*

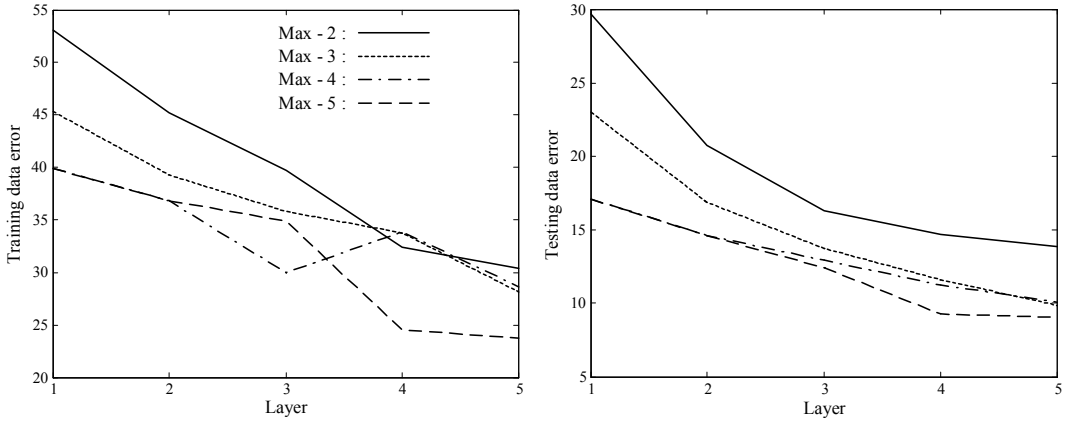
(b-1) Triangular

Max	1st layer			2nd layer			3rd layer			4th layer			5th layer		
	T	PI	EPI	T	PI	EPI	T	PI	EPI	T	PI	EPI	T	PI	EPI
2	2	39.499	20.481	1	36.989	13.236	1	36.877	12.800	3	31.181	11.624	1	29.812	10.926
3	2	39.499	20.481	1	24.322	12.916	1	24.303	12.175	1	24.229	11.063	1	24.086	10.979
4	1	39.894	17.070	1	36.989	13.236	1	26.614	10.601	1	26.583	10.418	4	26.521	10.164
5	1	39.894	17.070	1	36.989	13.236	1	36.756	12.683	3	24.888	10.828	1	25.187	10.392

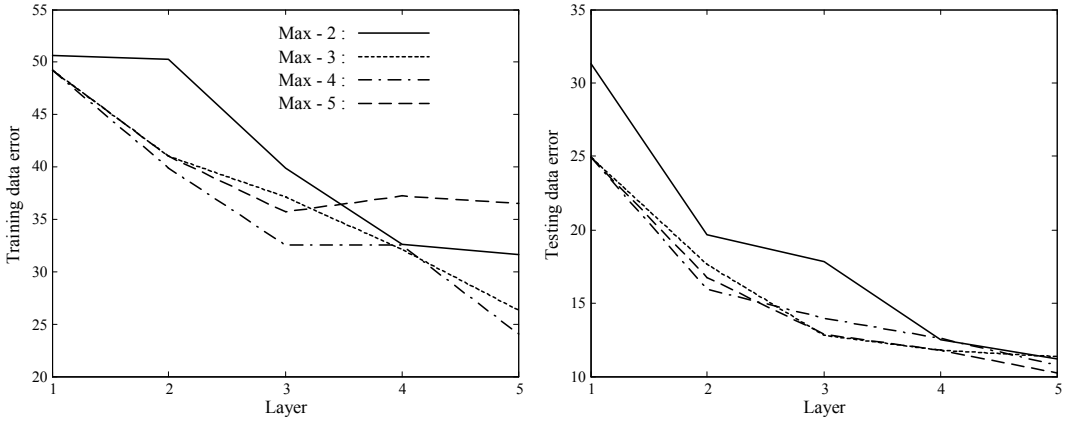
(b-2) Gaussian-like

Max	1st layer			2nd layer			3rd layer			4th layer			5th layer		
	T	PI	EPI	T	PI	EPI	T	PI	EPI	T	PI	EPI	T	PI	EPI
2	2	38.972	21.461	1	35.195	17.341	1	37.376	14.861	2	25.864	13.752	2	24.346	12.549
3	2	38.972	21.461	1	37.107	13.698	1	28.137	12.391	2	26.864	12.102	1	25.237	10.906
4	2	38.972	21.461	1	37.100	16.599	1	29.876	13.357	1	32.217	11.654	1	26.921	11.497
5	2	38.972	21.461	1	37.107	13.698	1	33.259	11.900	1	32.638	9.738	1	35.058	9.150

Fig. 13 shows the values of performance index vis-à-vis number of layers of the GA-based FSONN with respect to the maximal number of inputs to be selected as optimal architectures of each layer of the network included in **Table 5**(a)



(a) Triangular



(b) Gaussian-like

Fig. 13. Performance index of GA-based FSONN

Table 6 summarizes the results of comparative analysis of the proposed model with respect to other constructs.

5.2 pH Neutralization Process Data

To demonstrate the high modeling accuracy of the proposed model, we apply it to a highly nonlinear of pH neutralization of a weak acid and a strong base. This model can be found in a variety of practical areas including wastewater treatment, biotechnology processing, and chemical processing [9]. pH is the measurement of the acidity or alkalinity of a solution containing a proportion of water.

The system inputs of the proposed model structure consist of the delayed terms of $F_b(t)$ and $y_{pH}(t)$ which are input and output of the process, i. e.

$$\hat{y}_{pH}(t) = \varphi(F_b(t-3), F_b(t-2), F_b(t-1), y_{pH}(t-3), y_{pH}(t-2), y_{pH}(t-1)) \qquad (13)$$

where $\hat{y}_{pH}(t)$ and $y_{pH}(t)$ denote the proposed model output and the actual process output, respectively. **Table 7** summarizes the list of parameters used in the genetic optimization of the PN-based and the FPN-based FSONN.

Table 6. Comparison of identification error with previous models

Model				PI	PI_s	EPI_s
Regression model				36.13		
PNN				8.456		
SONFN[17]	Simplified	Generic	Basic		40.753	17.898
	Linear	Type	Architecture		35.745	17.807
FPNN[18]	No. of inputs: 2	Triangular	Type : 2		32.195	18.462
		Gaussian	Type : 1		49.716	31.423
	No. of inputs: 3	Triangular	Type : 1		32.251	19.622
		Gaussian	Type : 1		39.093	19.983
Our model	PN-based FSONN (θ=0.5)	max-inputs: 5	Type : 1≤T≤3		22.238	12.566
		max-inputs: 9		7.161	18.043	11.898
	FPN-based FSONN (θ=0.0)	Type T	Triangular		23.739	9.080
			Gaussian-like		24.017	10.738
		Type T*	Triangular		24.086	10.979
			Gaussian-like		25.237	10.906

Table 7. Computational aspects of the genetic optimization of PN-based and FPN-based FSONN

	Parameters	1st layer	2nd layer	3rd layer	4th layer	5th layer
GA	Maximum generation	100				
	Total population size	60				
	Selected population size (W)	30(l=2~5)				
	String length	3+3+30(l=2~5)				
	Crossover rate	0.65				
	Mutation rate	0.1				
PN based FSONN	Maximal no. (Max) of inputs to be selected	1≤l≤Max (2~5)				
	Polynomial type (Type T) (#)	1≤T≤3				
FPN based FSONN	Maximal no. (Max) of inputs to be selected	1≤l≤Max (2~4)				
	Polynomial type (Type T) of the consequent part of fuzzy rules (##)	1≤T≤4				
	Consequent input type to be used for Type T (###)	Type T*		Type T	Type T	
	Membership Function (MF) type	Triangular, Gaussian				
	No. of MFs per input	2				

l, T, Max: integers, #, ##and ###: refer to Tables 1-3 respectively.

5.2.1 Polynomial Neuron (PN) based FSONN

Table 8 summarizes the results about PN-based FSONN: According to the maximal number of inputs to be selected (Max=2 to 5), the selected polynomial type (Type T), and its corresponding performance index (PI) were shown when the genetic optimization for each layer was carried out.

Fig. 14 illustrates the optimization process by visualizing the performance index in successive generations of the genetic optimization with Max=5.

Table 8. Performance index of the PN-based FSONN viewed with regard to the increasing number of the layers

Max	1st layer		2nd layer		3rd layer		4th layer		5th layer	
	T	PI	T	PI	T	PI	T	PI	T	PI
2	2	1.0878	2	0.6401	2	0.5190	2	0.4080	2	0.2582
3	2	0.9811	2	0.3587	2	0.2472	2	0.1608	2	0.0949

4	2	0.8402	2	0.2927	2	0.1990	2	0.1051	2	0.0597
5	2	0.7553	2	0.3035	2	0.1449	2	0.0637	2	0.0325

Max	6th layer		7th layer		8th layer		9th layer		10th layer	
	T	PI	T	PI	T	PI	T	PI	T	PI
2	2	0.2170	2	0.1674	2	0.1402	2	0.1262	2	0.1144
3	2	0.0613	2	0.0365	2	0.0260	2	0.0199	2	0.0167
4	2	0.0289	2	0.0142	2	0.0088	2	0.0059	2	0.0040
5	2	0.0131	2	0.0063	2	0.0033	2	0.0020	2	0.0014

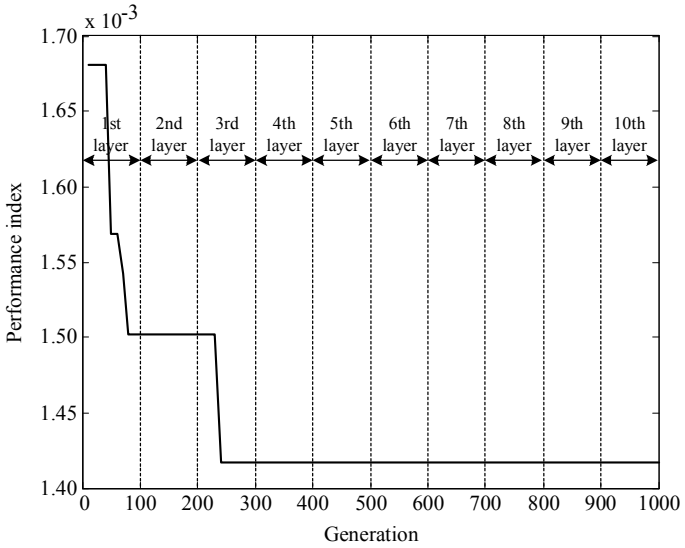


Fig. 14. The optimization process reported in term of performance index

5.2.2 Fuzzy Polynomial Neuron (FPN) based FSONN

Table 9 summarizes the performance of the 1st to 5th layer of the network when changing the maximal number of inputs to be selected; here Max was set up to 2 through 5.

Table 9. Performance index of the network of each layer versus the increase of maximal number of input to be selected

(a) In case of Type T

(a-1) Triangular

Max	1st layer		2nd layer		3rd layer		4th layer		5th layer	
	T	PI	T	PI	T	PI	T	PI	T	PI
2	3	0.5897	3	0.2445	3	0.0524	3	0.231	3	0.0142
3	3	0.2445	3	0.0111	3	0.0020	3	5.10e-4	3	2.40e-4
4	3	1.32e-4	3	1.25e-4	2	1.25e-4	3	1.25e-4	4	1.25e-4
5	3	1.30e-4	3	1.25e-4	2	1.25e-4	3	1.25e-4	2	1.25e-4

(a-2) Gaussian-like

Max	1st layer		2nd layer		3rd layer		4th layer		5th layer	
	T	PI	T	PI	T	PI	T	PI	T	PI
2	3	0.4972	3	0.0659	3	0.0088	4	0.0036	3	0.0015
3	3	0.0045	3	1.90e-4	3	1.40e-4	3	1.20e-4	4	1.20e-4
4	3	1.30e-4	3	1.25e-4	2	1.25e-4	4	1.25e-4	4	1.25e-4
5	3	1.30e-4	4	1.25e-4	2	1.25e-4	2	1.25e-4	2	1.25e-4

(b) In case of Type T*

(b-1) Triangular

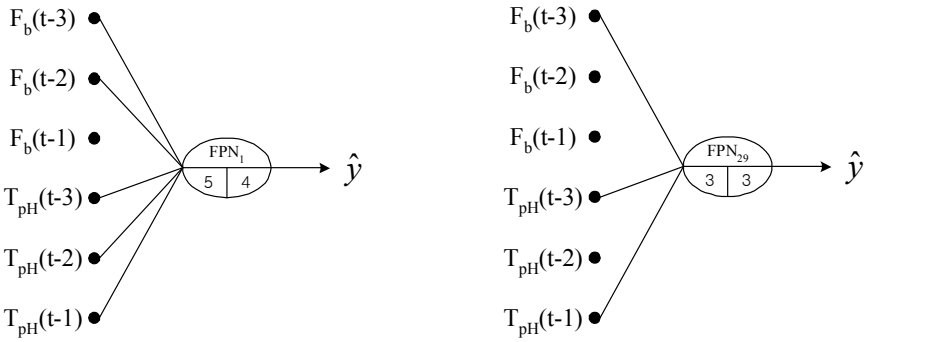
Max	1st layer	2nd layer	3rd layer	4th layer	5th layer

	T	PI	T	PI	T	PI	T	PI	T	PI
2	3	1.47e-4	4	1.36e-4	4	1.31e-4	3	1.30e-4	3	1.30e-4
3	3	1.30e-4	3	1.29e-4	4	1.25e-4	4	1.25e-4	3	1.25e-4
4	4	1.30e-4	3	1.25e-4	3	1.25e-4	3	1.25e-4	3	1.25e-4
5	4	1.29e-4	4	1.25e-4	2	1.25e-4	2	1.25e-4	2	1.25e-4

(b-2) Gaussian-like

Max	1st layer		2nd layer		3rd layer		4th layer		5th layer	
	T	PI	T	PI	T	PI	T	PI	T	PI
2	3	1.36e-4	3	1.31e-4	4	1.30e-4	3	1.30e-4	3	1.30e-4
3	3	1.30e-4	4	1.25e-4	3	1.25e-4	3	1.25e-4	3	1.25e-4
4	4	1.30e-4	3	1.25e-4	4	1.25e-4	2	1.25e-4	2	1.25e-4
5	3	1.30e-4	2	1.25e-4	3	1.25e-4	2	1.25e-4	2	1.25e-4

Fig. 15 (a) illustrate the detailed optimal topologies of FPN-based FSONN for 1 layer when using triangular MF: the results of the network have been reported as PI=1.29e-4 for Max=5 (see **Table 9**(b-1)). And also **Fig. 15** (b) illustrates the detailed optimal topologies of FPN-based FSONN for 1 layer in case of Gaussian-like MF: those are quantified as PI=1.30e-4 for Max=3 (see **Table 9**(b-2)).



(a) Triangular MF (1 layer and Max=5) (b) Gaussian-like MF (1 layer and Max=3)

Fig. 15. FPN-based FSONN architecture

Table 10 gives a comparative summary of the network with other models. The experimental results clearly reveal that it outperforms the existing models in terms of better approximation capabilities.

Table 10. Comparative analysis of the performance of the network; considered are models reported in the literature

Model				PI
PNN[3]	Basic type (Layer : 15)		Case 1	0.0015
			Case 2	0.0052
	Modified type (Layer : 10)		Case 1	0.0039
			Case 2	0.0124
Our model	PN-based FSONN (Layer : 10)	$1 \le l \le 2$	Type : $1 \le T \le 3$	0.1144
		$1 \le l \le 3$		0.0167
		$1 \le l \le 4$		0.0040
		$1 \le l \le 5$		0.0014
	FPN-based FSONN (Layer : 5)	Type T	Triangular	1.25e-4
			Gaussian-like	1.25e-4
		Type T*	Triangular	1.25e-4
			Gaussian-like	1.25e-4

6. Conclusions

In this study, the GA-based design procedure of Feed-forward Self-Organizing Neural Networks (FSONN) and its design methodology were proposed to construct optimal model architecture for nonlinear and complex system modeling. The design methodology comes with hybrid structural optimization and parametric learning viewed as two phases of modeling building. That is, the one phase (hybrid structural optimization) is realized via both GAs and a structural phase of an evolutionary algorithm as the main characteristics of the GMDH method while the other phase (parametric optimization) is carried out by a standard least square estimation (LSE)-based learning. The comprehensive experimental studies involving well-known datasets, MIS data and pH neutralization process data, quantify a superb performance of the network in comparison to the existing models. First of all, we could efficiently search for the optimal network architecture (structurally and parametrically optimized network) by the design methodology of GA-based FSONN in comparison to that of the conventional FSONN.

References

[1] V. Cherkassky, D. Gehring, F. Mulier, "Comparison of Adaptive Methods for Function Estimation from Samples," *IEEE Trans. Neural Networks*, Vol. 7, pp. 969-984, 1996.
[2] J.A. Dicherson, B. Kosko, "Fuzzy Function Approximation with Ellipsoidal Rules," *IEEE Trans. on Systems, Man and Cybernetics*, Part B. Vol. 26, pp. 542-560, 1996.
[3] S.K. Oh, W. Pedrycz, "The Design of Self-Organizing Polynomial Neural Networks," *Information Science*, Vol. 141, pp. 237-258, 2002.
[4] S.K. Oh, W. Pedrycz, B. J. Park, "Polynomial Neural Networks Architecture: Analysis and Design," Computers and Electrical Engineering, Vol. 29, No. 6, pp. 703-725, 2003.
[5] H.S. Park, B.J. Park, S.K. Oh, "Optimal Design of Self-Organizing Polynomial Neural Networks By Means of Genetic Algorithms," *Journal of the Research Institute of Engineering Technology Development*, Vol. 22, pp. 111-121, 2002.
[6] H.S. Park, K.J. Park, D.Y. Lee, S.K. Oh, "Advanced Self-Organizing Neural Networks Based on Competitive Fuzzy Polynomial Neurons," The Transaction on KIEE, Vol. 53, No. 3, pp. 135-144, 2004.
[7] W. Pedrycz, M. Reformat, "Evolutionary Optimization of Fuzzy Models in Fuzzy Logic: A framework for the New Millennium," *Studies in Fuzziness and Soft Computing*, Vol. 8, pp. 51-67, 1996.
[8] M.R. Lyu, "Handbook of Software Reliability Engineering," McGraw-Hill and IEEE Computer Society Press, 1995.
[9] C.L., Karr, E.J. Gentry, "Fuzzy Control of pH using Genetic Algorithms," *IEEE Trans. Fuzzy Systems*, Vol. 1, pp. 46-53, 1993.
[10] A.G. Ivakhnenko, "Polynomial Theory of Complex Systems," *IEEE Trans. on Systems, Man and Cybernetics*. Vol. 1, pp. 364-378, 1971.
[11] W. Pedrycz, "An Identification Algorithm in Fuzzy Relation System," *Fuzzy Sets and System*, Vol. 13, pp. 153-167, 1984.
[12] S.K. Oh, W. Pedrycz, "Identification of Fuzzy Systems by means of an Auto-Tuning Algorithm and Its Application to Nonlinear Systems, *Fuzzy sets and Systems*, Vol. 115, No. 2, pp. 205-230, 2000.
[13] J.H. Holland, "Adaptation in Natural and Artificial Systems," The University of Michigan Press, Ann Arbor, 1975.
[14] D.E. Goldberg, "Genetic Algorithm in Search, Optimization & Machine Learning" Addison Wesley, Boston, 1989.

[15] A. Kenneth, A. De, "Are Genetic Algorithms Function Optimizers?," Parallel Problem Solving from Nature 2, North-Holland Amsterdam, 1992.
[16] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," 3rd edition, Springer-Verlag, Berlin Heidelberg New York, 1996.
[17] S.K. Oh, W. Pedrycz, B.J. Park, "Relation-based Neurofuzzy Networks with Evolutionary Data Granulation," *Mathematical and Computer Modeling*, Vol. 40, No. 7-8, pp. 891-921, 2004.
[18] S.K. Oh, W. Pedrycz, "Fuzzy Polynomial Neuron-Based Self-Organizing Neural Networks," *Int. J. of General Systems*, Vol. 32, No. 3, pp. 237-250, 2003.

Sung-Kwun Oh received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Yonsei University, Seoul, Korea, in 1981, 1983, and 1993, respectively. During 1983-1989, he was a Senior Researcher of R&D Lab. of Lucky-Goldstar Industrial Systems Co., Ltd. From 1996 to 1997, he was a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada. He is currently a Professor with the Department of Electrical Engineering, University of Suwon, Suwon, South Korea. His research interests include fuzzy system, fuzzy-neural networks, automation systems, advanced computational intelligence, and intelligent control. He currently serves as an Associate Editor of the *KIEE Transactions on Systems and Control*, *International Journal of Fuzzy Logic and Intelligent Systems* of the KFIS, and *International Journal of Control, Automation, and Systems* of the ICASE, South Korea.

Ho-Sung Park received the BSc, MSc, and PhD. degrees in Control and Instrumentation Engineering from Wonkwang University, Korea, in 1999, 2001, and 2005, respectively. During 2005-2006, he was a full-time lecturer in the Department of Electrical Electronic and Information Engineering, Wonkwang University, Korea. From 2006 to 2007, he was a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is currently a Research Professor in the Industry Administration Institute, University of Suwon, Korea. His research interests include Fuzzy Inference System, Neural Network, Fuzzy-Neural Network, Genetically Optimization Algorithm, Granular Computing, Neuro-fuzzy Computing, and Evolutionary Computing.

Chang-Won Jeong received his BS degree in computer engineering from Wonkwang University in 1993, and M.S. and Ph.D. degrees at dept. of computer engineering from Wonkwang University in 1998 and 2003, respectively, in South Korea. Currently, he is a Post.Doc at dept. of computer engineering in Wonkwang University. His main research interests include distributed object computing, middleware and u-healthcare.

Su-Chong Joo received his BS degree in computer engineering from Wonkwang University in 1986, and M.S. and Ph.D. degrees at dept. of computer science and engineering from Chung-Ang University in 1988 and 1992, respectively, in South Korea. Currently, he is a professor at division of electrical, electronic and information engineering in Wonkwang University. His main research interests include distributed real-time computing, distributed object modeling, system optimization, multimedia database systems and healthcare applications. From Jul. 1993 to Aug. 1994, he was a Post-Doctoral Fellow at dept. of electrical and computer engineering in University of Massachusetts at Amherst. Also, from Dec. 2002 to Jan. 2005, he was a research professor at Dept. of Electrical Engineering and Computer Science in University of California at Irvine. He is a member of KISS, IASTED, IEEE and IEEE computer society.