# Genetically Optimized Hybrid Fuzzy Neural Networks Based on Linear Fuzzy Inference Rules

Sung-Kwun Oh, Byoung-Jun Park, and Hyun-Ki Kim

**Abstract:** In this study, we introduce an advanced architecture of genetically optimized Hybrid Fuzzy Neural Networks (gHFNN) and develop a comprehensive design methodology supporting their construction. A series of numeric experiments is included to illustrate the performance of the networks. The construction of gHFNN exploits fundamental technologies of Computational Intelligence (CI), namely fuzzy sets, neural networks, and genetic algorithms (GAs). The architecture of the gHFNNs results from a synergistic usage of the genetic optimization-driven hybrid system generated by combining Fuzzy Neural Networks (FNN) with Polynomial Neural Networks (PNN). In this tandem, a FNN supports the formation of the premise part of the rule-based structure of the gHFNN. The consequence part of the gHFNN is designed using PNNs. We distinguish between two types of the linear fuzzy inference rule-based FNN structures showing how this taxonomy depends upon the type of a fuzzy partition of input variables. As to the consequence part of the gHFNN, the development of the PNN dwells on two general optimization mechanisms: the structural optimization is realized via GAs whereas in case of the parametric optimization we proceed with a standard least square method-based learning. To evaluate the performance of the gHFNN, the models are experimented with a representative numerical example. A comparative analysis demonstrates that the proposed gHFNN come with higher accuracy as well as superb predictive capabilities when comparing with other neurofuzzy models.

**Keywords:** Genetically optimized hybrid fuzzy neural networks, computational intelligence, linear fuzzy inference rule-based fuzzy neural networks, genetically optimized polynomial neural networks, design procedure.

## 1. INTRODUCTION

Recently, a lot of attention has been devoted towards advanced techniques of modeling complex systems inherently associated with nonlinearity, high-order dynamics, time-varying behavior, and imprecise measurements. It is anticipated that efficient modeling techniques should allow for a selection of pertinent variables and in this way help cope with dimensionality of the problem at hand. The omnipresent modeling tendency is the one that exploits techniques of Computational Intelligence (CI)

Sung-Kwun Oh and Hyun-Ki Kim are with the Department of Electrical Engineering, The University of Suwon, San 2-2 Wau-Ri, Bongdam-Up, Hwasung-Si, Kyungki-Do 445-743, Korea (e-mails: {ohsk, hkkim}@suwon.ac.kr).

Byoung-Jun Park is with the Department of Electrical Electronic & Information Engineering, Wonkwang University, 344-2, Shinyong-Dong, Iksan, Chon-Buk 570-749, Korea (e-mail: lcap@wonkwang.ac.kr).

by embracing fuzzy modeling [2-6], neurocomputing [6], and genetic optimization [7-9]. Especially the two of the most successful approaches have been the hybridization attempts made in the framework of CI [10,11]. Neuro-fuzzy systems are one of them [12-17]. A different approach to hybridization leads to genetic fuzzy systems. Lately to obtain a highly beneficial synergy effect, the neural fuzzy systems and the genetic fuzzy systems hybridize the approximate inference method of fuzzy systems with the learning capabilities of neural networks and evolutionary algorithms [18].

In this study, we develop a hybrid modeling architecture, called genetically optimized Hybrid Fuzzy Neural Networks (gHFNN). In a nutshell, a gHFNN is composed of two main substructures driven by genetic optimization, namely a rule-based Fuzzy Neural Network (FNN) and a Polynomial Neural Network (PNN). From a standpoint of rule-based architectures (with their rules assuming the general form "if antecedent then consequent"), one can regard the FNN as an implementation of the antecedent (or premise) part of the rules while the consequent part is realized with the aid of PNN. The resulting gHFNN is

an optimized architecture designed by combining the conventional Hybrid Fuzzy Neural Networks (HFNN [17,27,28]) with genetic algorithms (GAs). In this study, the FNNs come with two kinds of network architectures, namely fuzzy-set based FNN and fuzzy-relation based FNN. The topology of the network proposed here relies on fuzzy partitions realized in terms of fuzzy sets or fuzzy relations that its input variables are considered separately or simultaneously. Moreover the PNN structure is optimized by GAs, that is, a genetically optimized PNN (gPNN) is designed and the gPNN is applied to the consequence part of gHFNN. The gPNN that exhibits a flexible and versatile structure is constructed on a basis of PNN [12,13] and Gas [7-9]. In this network, the number of layers and number of nodes in each layer are not predetermined (unlike in case of most neural-networks) but can be generated in a dynamic fashion. The design procedure applied in the construction of each layer of the PNN deals with its structural optimization involving the selection of optimal nodes (or PNs) with specific local characteristics (such as the number of input variables, the order of the polynomial, and a collection of the specific subset of input variables) and addresses specific aspects of parametric optimization.

The study is organized in the following manner. First, Section 2 delivers a brief introduction to the architecture of the conventional HFNN. In Section 3, we discuss a structure of the genetically optimized HFNN (gHFNN) and elaborate on the development of the networks. The detailed genetic design of the gHFNN model comes with an overall description of a detailed design methodology of the gHFNN presented in Section 4. In Section 5, we report on a comprehensive set of experiments. Finally concluding remarks are covered in Section 6.

## 2. THE ARCHITECTURE OF CONVENTIONAL HYBRID FUZZY NEURAL NETWORKS (HFNN)

The conventional HFNN architecture combined with the FNN and PNN is visualized in Fig. 1 [17,27,28]. Let us recall that the linear fuzzy inference (both fuzzy set and fuzzy relation) -based FNN is constructed
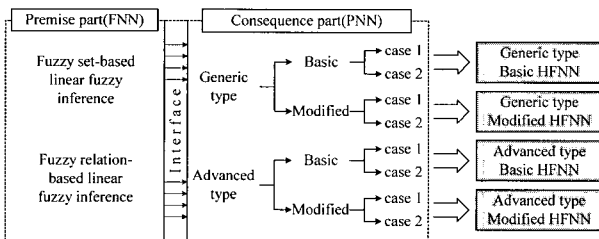
with the aid of the space partitioning realized by not only fuzzy set defined for each input variable but also fuzzy relations that effectively capture an ensemble of input variables. These networks arise as a synergy between two other general constructs such as FNN and PNN. Based on the different PNN topologies, the HFNN embraces two kinds of architectures, namely a basic and modified one. Moreover for each architecture of the HFNN, we identified two cases; refer to Fig. 1 for the overall taxonomy.

## 3. THE ARCHITECTURE AND DEVELOPMENT OF GENETICALLY OPTIMIZED HFNN (GHFNN)

In this section, we elaborate on the architecture and a development process of the gHFNN. This network emerges from the genetically optimized multi-layer perceptron architecture based on fuzzy set or fuzzy relation-based FNN, PNN and GAs. In the sequel, gHFNN is designed by combining the conventional Hybrid Fuzzy Neural Networks (HFNN) with GAs. These networks result as a synergy between two other general constructs such as FNN [21,25] and PNN [12,13].

First, we briefly discuss these two classes of models by underlining their profound features in Sections 3.1 and 3.2, respectively.
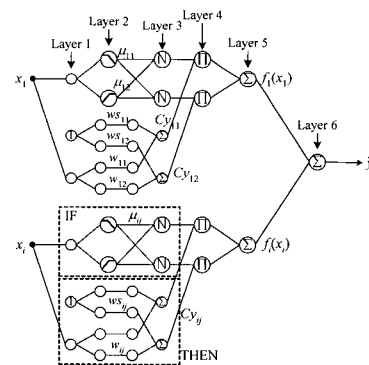


Fig. 2. Topology of FS_FNN by using space partitioning in terms of individual input variables.
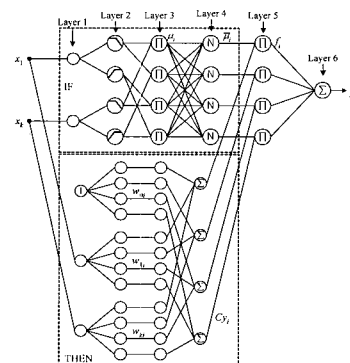


Fig. 3. Topology of FR_FNN by using space partitioning in terms of an ensemble of input variables.



Fig. 1. Overall diagram for generating the conventional HFNN architecture.

### 3.1. Fuzzy neural networks based on genetic optimization

We consider two kinds of FNNs (viz. FS_FNN and FR_FNN) based on linear fuzzy inference. The FNN is designed by using space partitioning realized in terms of the individual input variables or an ensemble of all variables. Its each topology is concerned with a granulation carried out in terms of fuzzy sets defined in each input variable or fuzzy relations that capture an ensemble of input variables respectively. The fuzzy partitions formed for each case lead us to the topologies visualized in Figs. 2-3.

The notation in these figures requires some clarification. The "circles" denote units of the FNN while "N" identifies a normalization procedure applied to the membership grades of the input variable $x_i$. The output $f_i(x_i)$ of the "$\Sigma$" neuron is described by some nonlinear function $f_i$. Not necessarily $f_i$ is a sigmoid function encountered in conventional neural networks but we allow for more flexibility in this regard. Finally, in case of FS_FNN, the output of the FNN $\hat{y}$ is governed by the following expression;

$$\hat{y} = f_1(x_1) + f_2(x_2) + \cdots f_m(x_m) = \sum_{i=1}^{m} f_i(x_i) \quad (1)$$

with $m$ being the number of the input variables (viz. the number of the outputs $f_i$'s of the "$\Sigma$" neurons in the network). As previously mentioned, FS_FNN is affected by the introduced fuzzy partition of each input variable. In this sense, we can regard each $f_i$ given by fuzzy rules as shown in Table 1. Table 1 represents the comparison of fuzzy rules, inference result and learning for two types of FNNs. In Table 1, $R^j$ is the $j$-th fuzzy rule while $A_{ij}$ denotes a fuzzy variable of the premise of the corresponding fuzzy rule and represents membership function $\mu_{ij}$. $ws_{ij}$ s a constant consequence and $w_{ij}$ is an input variable consequence of the rules. They express a connection (weight) existing between the neurons as we have already visualized in Fig. 2. Mapping from $x_i$ to $f_i(x_i)$ is determined by the fuzzy inferences and a standard defuzzification. The inference result for individual fuzzy rules follows a standard center of gravity aggregation. An input signal $x_i$ activates only two membership functions, so inference results can be written as outlined in Table 1[20,21]. The learning of FNN is realized by adjusting connections of the neurons and as such it follows a standard Back-Propagation (BP) algorithm [20,21]. The complete update formulas are covered in Table 1. Where $\eta$ is a positive learning rate and $\alpha$ is a positive momentum coefficient. The case of FR_FNN is carried out in the same manner as that of FS_FNN.

The task of optimizing any model involves two main phases. First, a class of some optimization algorithms has to be chosen so that it meets the requirements implied by the problem at hand.

Table 1. Comparison of fuzzy set with fuzzy relation-based FNNs.

| Structure | FS_FNN | FR_FNN |
|---|---|---|
| Fuzzy rules | $R^1 : If\ x_i\ is\ A_{i1}\ then\ Cy_{i1} = ws_{i1} + w_{i1} \cdot x_i$ <br><br> $\vdots \qquad\qquad \vdots$ <br><br> $R^j : If\ x_i\ is\ A_{ij}\ then\ Cy_{ij} = ws_{ij} + w_{ij} \cdot x_i$ <br><br> $\vdots \qquad\qquad \vdots$ <br><br> $R^z : If\ x_i\ is\ A_{iz}\ then\ Cy_{iz} = ws_{iz} + w_{iz} \cdot x_i$ | $R^1 : If\ x_1\ is\ A_1 \cdots and\ x_k\ is\ A_{k1}\ then\ Cy_1 = w_{01} + w_{11} \cdot x_1 + w_{k1} \cdot x_k$ <br><br> $\vdots \qquad\qquad \vdots$ <br><br> $R^i : If\ x_1\ is\ A_i \cdots and\ x_k\ is\ A_{ki}\ then\ Cy_i = w_{0i} + w_{1i} \cdot x_1 + w_{ki} \cdot x_k$ <br><br> $\vdots \qquad\qquad \vdots$ <br><br> $R^n : If\ x_1\ is\ A_{1n} \cdots and\ x_k\ is\ A_{kn}\ then\ Cy_n = w_{0n} + w_{1n} \cdot x_1 + w_{kn} \cdot x_k$ |
| Inference result | $f_i(x_i) = \dfrac{\sum_{j=1}^{z}(\mu_{ij}(x_i) \cdot (ws_{ij} + x_i w_{ij}))}{\sum_{j=1}^{z}\mu_{ij}(x_i)}$ <br><br> $= \mu_{ik}(x_i) \cdot (ws_{ik} + x_i w_{ik})$ <br><br> $+ \mu_{ik+1}(x_i) \cdot (ws_{ik+1} + x_i w_{ik+1})$ | $\hat{y} = \sum_{i=1}^{n} f_i$ <br><br> $= \sum_{i=1}^{n} \bar{\mu}_i \cdot (w_{0i} + w_{1i} \cdot x_1 + w_{ki} \cdot x_k)$ <br><br> $= \sum_{i=1}^{n} \dfrac{\mu_i \cdot (w_{0i} + w_{1i} \cdot x_1 + w_{ki} \cdot x_k)}{\sum_{i=1}^{n}\mu_i}$ |
| Learning | $\begin{cases} \Delta ws_{ij} = 2 \cdot \eta \cdot (y - \hat{y}) \cdot \mu_{ij} \\ \qquad + \alpha(ws_{ij}(t) - ws_{ij}(t-1)) \\ \Delta w_{ij} = 2 \cdot \eta \cdot (y - \hat{y}) \cdot \mu_{ij} \cdot x_i \\ \qquad + \alpha(w_{ij}(t) - w_{ij}(t-1)) \end{cases}$ | $\begin{cases} \Delta w_{0i} = 2 \cdot \eta \cdot (y - \hat{y}) \cdot \bar{\mu}_i \\ \qquad + \alpha(w_{0i}(t) - w_{0i}(t-1)) \\ \Delta w_{ki} = 2 \cdot \eta \cdot (y - \hat{y}) \cdot \bar{\mu}_i \cdot x_k \\ \qquad + \alpha(w_{ki}(t) - w_{ki}(t-1)) \end{cases}$ |

Secondly, various parameters of the optimization algorithm need to be tuned in order to achieve its best performance. Along this line, genetic algorithms (GAs) viewed as optimization techniques based on the principles of natural evolution are worth considering. GAs have been experimentally demonstrated to provide robust search capabilities in problems involving complex spaces thus offering a valid solution to problems requiring efficient searching. It is instructive to highlight the main features that tell GA apart from some other optimization methods: (1) GA operates on the codes of the variables, but not the variables themselves. (2) GA searches optimal points starting from a group (population) of points in the search space (potential solutions), rather than a single point. (3) GA's search is directed only by some fitness function whose form could be quite complex; we do not require its differentiability [7-9].

In order to enhance the learning of the FNN, we use GAs to adjust learning rate, momentum coefficient and the parameters of the membership functions of the antecedents of the rules [17,27,28].

### 3.2. Genetically optimized PNN (gPNN)

As underlined, the PNN algorithm is based upon the GMDH [19] method and utilizes a class of polynomials such as linear, quadratic, modified quadratic, etc. to describe basic processing realized there. By choosing the most significant input variables and an order of the polynomial among various types of forms available, we can obtain the best one – it comes under a name of a partial description (PD). It is realized by selecting nodes at each layer and eventually generating additional layers until the best performance has been reached. Such a methodology leads to an optimal PNN structure [12,13].

In addressing the problems with the conventional PNN, we introduce a new genetic design approach; in turn we will be referring to these networks as genetically optimized PNN (to be called "gPNN"). When we construct PNs of each layer in the conventional PNN, such parameters as the number of input variables (nodes), the order of polynomial, and input variables available within a PN are fixed (selected) in advance by the designer. This could have frequently contributed to the difficulties in the design of the optimal network. To overcome this apparent drawback, we resort ourselves to the genetic optimization, see Fig. 5 of the next section for more detailed flow of the development activities. The overall genetically-driven structural optimization process of PNN is shown in Fig. 4. The determination of the optimal values of the parameters available within an individual PN (viz. the number of input variables, the order of the polynomial, and input variables) leads to a structurally and parametrically optimized network. As a result, this network is more



E : Entire inputs, S : Selected PNs, $z_i$ : Preferred outputs in the $i$th stage($z_i$=$z_{1i}$, $z_{2i}$, ..., $z_{Wi}$)
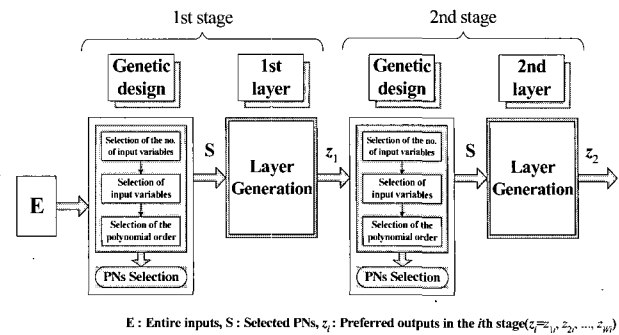
Fig. 4. Overall genetically-driven structural optimization process of PNN.

flexible as well as it exhibits simpler topology in comparison to the conventional PNN discussed in the previous research [12,13].

For the optimization of the PNN model, GAs uses the serial method of binary type, roulette-wheel used in the selection process, one-point crossover in the crossover operation, and a binary inversion (complementation) operation in the mutation operator. To retain the best individual and carry it over to the next generation, we use elitist strategy [7,8].

### 3.3. Optimization of gHFNN topologies

The topology of gHFNN is constructed by combining fuzzy set(or fuzzy relation)-based FNN for the premise part of the gHFNN with PNN being used as the consequence part of gHFNN. These networks emerge through a synergy between two other general constructs such as FNNs and gPNNs. In what follows, the gHFNN is composed of two main substructures driven by genetic optimization; see Fig. 5. The role of FNNs arising at the premise part is to support learning and interact with input as well as granulate the corresponding input space (viz. converting the numeric data into their granular representatives emerging at the level of fuzzy sets). Especially, two kinds of linear fuzzy inference-based FNN realized with the fuzzy partitioning of individual input variables or an ensemble of input variable are considered to enhance the adaptability of the hybrid network architecture. One should stress that the structure of the consequent gPNN is not fixed in advance but becomes dynamically organized during a growth process. In essence, the gPNN exhibits an ability of self-organization. The gPNN algorithm can produce an optimal nonlinear system by selecting significant input variables among dozens of those available at the input and forming various types of polynomials. Therefore, for the very reason we selected FNN and gPNN in order to design the gHFNN architecture.

One may consider some other hybrid network architectures such as a combination of FNNs and MLPs as well as ANFIS-like models combined with
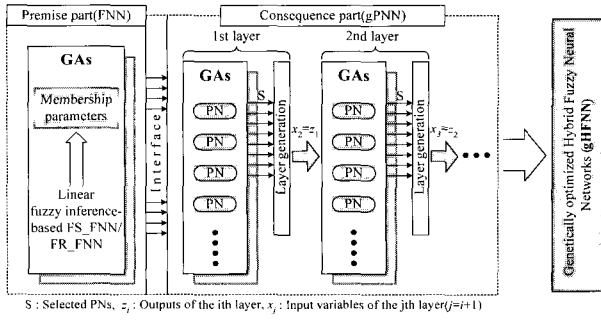
Fig. 5. Overall diagram for generation of gHFNN architecture.

MLPs. While attractive on a surface, such hybridization may lead to several potential problems: 1) The repeated learning and optimization of each of the contributing structure (such as ANFIS and MLP) may result in excessive learning time as well as generate quite complex networks for relatively simple systems and 2) owing to its fixed structure, it could be difficult to generate the flexible topologies of the networks that are required to deal with highly nonlinear dependencies.

## 4. THE ALGORITHMS AND DESIGN PROCEDURE OF GENETICALLY OPTIMIZED HFNN (GHFNN)

In this section, we elaborate on the algorithmic details of the design method by considering the functionality of the individual layers in the network architectures. The design procedure for each layer in the premise and consequence of gHFNN comprises of the following steps:

### 4.1. The premise of gHFNN: in case of FS_FNN

**Layer 1:** *Input layer:* The role of this layer is to distribute the signals to the nodes in the next layer.

**Layer 2:** *Computing activation degrees of linguistic labels:* Each node in this layer corresponds to one linguistic label (small, large, etc.) of the input variables in layer 1. The layer determines a degree of satisfaction (activation) of this label by the input.

**Layer 3:** *Normalization of a degree activation (firing) of the rule:* As described, a degree of activation of each rule was calculated in layer 2. In this layer, we normalize the activation level by using the following expression.

$$\bar{\mu}_{ij} = \frac{\mu_{ij}}{\sum_{j=1}^{n} \mu_{ij}} = \frac{\mu_{ik}}{\mu_{ik} + \mu_{ik+1}} = \mu_{ik}, \qquad (2)$$

where $n$ is the number of membership function for each input variable. An input signal $x_i$ activates only two membership functions simultaneously and the sum of grades of these two neighboring membership

functions labeled by $k$ and $k+1$ is always equal to 1, that is $\mu_{ik}(x_i) + \mu_{ik+1}(x_i) = 1$, so that this leads to a simpler format as shown in (2) [20,21].

**Layer 4:** *Multiplying a normalized activation degree of the rule by connection (weight):* The calculated activation degree at the third layer is now calibrated through the connections, that is

$$a_{ij} = \bar{\mu}_{ij} \times Cy_{ij} = \mu_{ij} \times Cy_{ij} \, (Here, \; Cy_{ij} = ws_{ij} + w_{ij} \cdot x_i) \,.(3)$$

If we choose CP (connection point) 1(viz. point between layer 4 and layer 5) for combining FS_FNN with PNN as shown in Fig. 2, $a_{ij}$ is given as the input variable of the PNN. If we choose CP 2(viz. point between layer 5 and layer 6), $f_i(x_i)$ corresponds to the input signal to the output layer of FNN viewed as the input variable of the PNN.

**Layer 5:** *Fuzzy inference for output of the rules:* Considering Fig. 2, the output of each node in the 5th layer of the premise part of gHFNN is inferred by the center of gravity method [20,21]. If we choose CP 2, $f_i$ is the input variable of gPNN that is the consequence part of gHFNN

$$f_i(x_i) = \frac{\sum_{j=1}^{z} a_{ij}}{\sum_{j=1}^{z} \mu_{ij}(x_i)} = \frac{\sum_{j=1}^{z} \mu_{ij}(x_i) \cdot (ws_{ij} + x_i w_{ij})}{\sum_{j=1}^{z} \mu_{ij}(x_i)} \qquad (4)$$

$$= \mu_{ik}(x_i) \cdot (ws_{ik} + x_i w_{ik})$$
$$+ \mu_{ik+1}(x_i) \cdot (ws_{ik+1} + x_i w_{ik+1}).$$

**[Output layer of FNN]** *Computing output of basic FNN:* The output becomes a sum of the individual contributions from the previous layer, see (1)

The design procedure for each layer in FR_FNN is carried out in a same manner as the one presented for FS_FNN.

### 4.2. The consequence of gHFNN: in case of gPNN combined with FS_FNN

**Step 1:** *Configuration of input variables:* Define input variables $x_i$'s ($i$=1, 2, ..., $n$) to gPNN of the consequent structure of gHFNN. If we choose the first option to combine the structures of FNN and gPNN (CP 1), $a_{ij}$, which is the output of layer 4 in the premise structure of the gHFNN, is treated as the input of the consequence structure of gHFNN, that is, $x_1$=$a_{11}$, $x_2$=$a_{12}$,··· , $x_n$=$a_{ij}$ ($n$=$i$× $j$). For the second option of combining the structures (viz. CP 2), we have $x_1$=$f_1$, $x_2$=$f_2$,··· , $x_n$=$f_m$ ($n$=$m$).

**Step 2:** *Decision of initial information for constructing the gPNN structure:* We decide upon the design parameters of the PNN structure and they include that a) Stopping criterion, b) Maximum number of input variables coming to each node in the corresponding layer, c) Total number $W$ of nodes to be

retained (selected) at the next generation of the gPNN, d) Depth of the gPNN to be selected to reduce a conflict between overfitting and generalization abilities of the developed gPNN, and e) Depth and width of the gPNN to be selected as a result of a tradeoff between accuracy and complexity of the overall model. It is worth stressing that the decisions made with respect to (b)-(e) help us avoid building excessively large networks (which could be quite limited in terms of their predictive abilities).

**Step 3:** *Initialization of population:* We create a population of chromosomes for a PN, where each chromosome is a binary vector of bits. All bits for each chromosome are initialized randomly.

**Step 4:** *Decision of PN structure using genetic design:* This concerns 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. These important decisions are carried out through an extensive genetic optimization. When it comes to the organization of the chromosome representing a PN, we divide the chromosome into three sub-chromosomes as shown in Fig. 7. The $1^{st}$ sub-chromosome contains the number of input variables, the $2^{nd}$ sub-chromosome involves the order of the polynomial of the node, and the $3^{rd}$ sub-chromo-
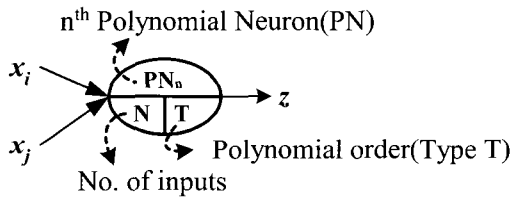


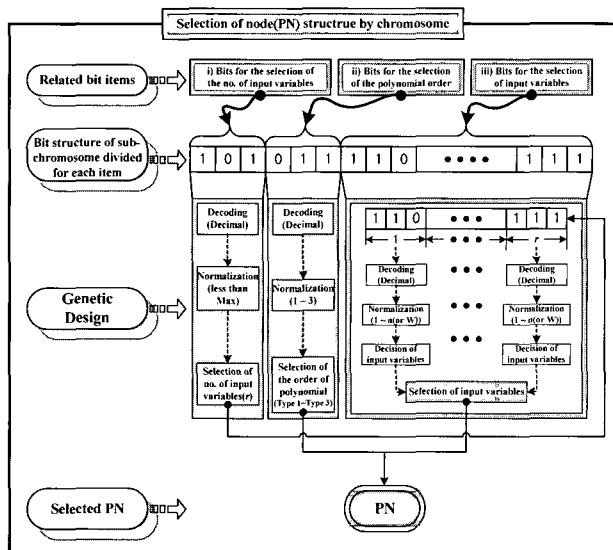Fig. 6. Formation of each PN in PNN architecture.



Fig. 7. The PN design used in the gPNN architecture – structural considerations and mapping the structure on a chromosome.

some (remaining bits) contains input variables coming to the corresponding node (PN). In nodes (PNs) of each layer of gPNN, we adhere to the notation of Fig. 6. '$PN_n$' denotes the $n^{th}$ PN (node) of the corresponding layer, 'N' denotes the number of nodes (inputs or PNs) coming to the corresponding node, and 'T' denotes the polynomial order in the corresponding node.

Each sub-step of the genetic design of the three types of the parameters available within the PN is structured as follows.

**Step 4-1:** Selection of the number of input variables (1st sub-chromosome)

*Sub-step 1)* The first 3 bits of the given chromosome are assigned to the binary bits for the selection of the number of input variables.

*Sub-step 2)* The selected 3 bits are decoded into a decimal.

*Sub-step 3)* The above decimal value is converted into [1 N] and rounded off. N denotes the maximal number of input variables entering the corresponding node (PN).

*Sub-step 4)* The normalized integer value is then treated as the number of input variables (or input nodes) coming to the corresponding node.

**Step 4-2:** Selection of the order of polynomial (2nd sub-chromosome)

*Sub-step 1)* The 3 bits of the 2nd sub-chromosome are assigned to the binary bits for the selection of the order of polynomial.

*Sub-step 2)* The 3 bits are decoded into a decimal format.

*Sub-step 3)* The decimal value obtained is normalized into [1 3] and rounded off.

*Sub-step 4)* The normalized integer value is given as the polynomial order.

**Step 4-3:** Selection of input variables (3rd sub-chromosome)

*Sub-step 1)* The remaining bits are assigned to the binary bits for the selection of input variables.

*Sub-step 2)* The remaining bits are divided by the value obtained in step 4-1.

*Sub-step 3)* Each bit structure is decoded into a decimal.

*Sub-step 4)* The decimal value obtained is normalized into [1 n (or W)] and rounded off. n is the overall system's inputs in the $1^{st}$ layer, and W is the number of the selected nodes in the $2^{nd}$ layer or higher.

*Sub-step 5)* The normalized integer values are then taken as the selected input variables while constructing each node of the corresponding layer. Here, if the selected input variables are multiple - duplicated, the multiple-duplicated input variables are treated as a single input variable.

**Step 5:** *Estimation of the coefficients of the polynomial assigned to the selected node and evaluation*

Table 2. Different forms of regression polynomial forming a PN.

| Number of inputs / Order of the polynomial | 2 | 3 | 4 |
|---|---|---|---|
| 1 (Type 1) | Bilinear | Trilinear | Tetralinear |
| 2 (Type 2) | Biquadratic-1 | Triquadratic-1 | Tetraquadratic-1 |
| 2 (Type 3) | Biquadratic-2 | Triquadratic-2 | Tetraquadratic-2 |

The following types of the polynomials are used;

- Bilinear $= c_0 + c_1 x_1 + c_2 x_2$ ,

- Biquadratic-1 (Basic)=Bilinear+ $c_3 x_1^2 + c_4 x_2^2 + c_5 x_1 x_2$ ,

- Biquadratic-2 (Modified)= Bilinear $+ c_3 x_1 x_2$

*of a PN:* The vector of coefficients is derived by minimizing the mean squared error between $y_i$ and $\hat{y}$ [12,13]. To evaluate the approximation and generalization capability of a PN produced by each chromosome, we use the following fitness function (the objective function is given in Section 5).

$$fitness\ function = \frac{1}{1 + Objective\ function} \qquad (5)$$

**Step 6:** *Elitist strategy and Selection of nodes (PNs) with the best predictive capability:* The nodes (PNs) obtained on the basis of the calculated fitness values $(F_1, F_2, \cdots, F_z)$ are rearranged in a descending order. We unify the nodes with duplicated fitness values (viz. in case that one node is the same fitness value as other nodes) among the rearranged nodes on the basis of the fitness values. We choose several PNs $(W)$ characterized by the best fitness values. For the elitist strategy, we select the node that has the highest fitness value among the generated nodes.

**Step 7:** *Reproduction:* To generate new populations of the next generation, we carry out selection, crossover, and mutation operation using genetic information and the fitness values.
Until the last generation, this step carries out by repeating steps 4-7.

**Step 8:** *Construction of a corresponding layer of consequence part of gHFNN:* Individuals evolved by GAs produce optimal PNs, $W$. The generated PNs construct their corresponding layer for the design of consequence part of gHFNN.

**Step 9:** *Check the termination criterion:* The termination condition builds a sound compromise between the high accuracy of the resulting model and its complexity as well as generalization abilities.

**Step 10:** *Determine new input variables for the next layer:* If the termination criterion has not been met, the model is expanded. The outputs of the preserved nodes $(z_1, z_2, ..., z_W)$ serves as new inputs to the next layer $(x_1, x_2, ..., x_W)$. Repeating steps 3-10 carries out the gPNN.

## 5. EXPERIMENTAL STUDIES

In this section, the performance of the gHFNN is illustrated with the aid of well-known and widely used gas furnace dataset (Box-Jenkins dataset)[1-5,22-28]. The performance index (object function) was used as Mean Squared Error (MSE)

$$E(PI\ or\ EPI) = \frac{1}{n} \sum_{p=1}^{n} (y_p - \hat{y}_p)^2 . \qquad (6)$$

Genetic algorithms use binary type, roulette-wheel as the selection operator, one-point crossover, and an invert operation in the mutation operator. The crossover rate of GAs is set to 0.75 and probability of mutation is equal to 0.065. The values of these parameters come from experiments and are very much in line with typical values encountered in genetic optimization.

We illustrate the performance of the network and elaborate on its development by experimenting with data coming from the gas furnace process. The time series data (296 input-output pairs) resulting from the gas furnace process has been intensively studied in the previous literature [1-5,23-28]. The delayed terms of methane gas flow rate, $u(t)$ and carbon dioxide density, $y(t)$ are used as system input variables such as $u(t$-$3)$, $u(t$-$2)$, $u(t$-$1)$, $y(t$-$3)$, $y(t$-$2)$, and $y(t$-$1)$. We use two types of system input variables of FNN structure, Type I and Type II to design an optimal model from gas furnace data. Type I utilize two system input variables such as $u(t$-$3)$ and $y(t$-$1)$ and Type II utilizes 3 system input variables such as $u(t$-$2)$, $y(t$-$2)$, and $y(t$-$1)$. The output variable is $y(t)$.

Table 3 summarizes the computational aspects related to the genetic optimization of gHFNN. Design information for the optimization of gHFNN distinguishes between information of two networks such as the premise FNN and the consequent gPNN. First, a chromosome used in genetic optimization of the premise FNN contains the vertices of 2 membership functions of each system input (here, 2 (Type I) or 3 (Type II) system input variables have been used), learning rate, and momentum coefficient. The numbers of bits allocated to a chromosome are equal to 40(Type I)/60(Type II), 10, and 10, respectively, that is 10 bits is assigned to each one variable. The parameters such as learning rate, momentum coefficient, and membership parameters are tuned with the help of genetic optimization of the FNN as shown in Table 3. Next, in case of the consequent gPNN, a chromosome used in the genetic optimization consists of a string including 3 sub-chromosomes. The numbers of bits allocated to each sub-chromosome are equal to 3, 3, and 24, respectively. The population size being selected from the total population size, 60 are equal to 30. The process is realized as follows. 60 nodes (PNs) are

generated in each layer of the network. The parameters of all nodes generated in each layer are estimated and the network is evaluated using both the training and testing data sets. Then we compare these values and choose 30 PNs that produce the best (lowest) value of the performance index. The number of inputs to be selected is confined to a maximum of four entries. The order of the polynomial is chosen from three types, that is Type 1, Type 2, and Type 3.

Table 4 includes the results of the overall network reported according to various alternatives concerning various forms of FNN architecture, format of entire system inputs and location of the connection point. When considering the FS_FNN with Type I (4 fuzzy rules), the minimal value of the performance index, that is PI= 0.041 and EPI=0.267 are obtained. In case of the FS_FNN with Type II (6 fuzzy rules), the best results are reported with the performance index such that PI=0.0256 and EPI=0.143. When using FR_FNN, the best results (PI= 0.025 and EPI=0.265) were obtained for Type I(4 fuzzy rules) and Type II(8 fuzzy rules), respectively (in the second case we have PI= 0.033 and EPI=0.119).

Table 3. Computational aspects of the optimization of gHFNN.
(a) In case of using FS_FNN.

| | | Generation | 150 |
|---|---|---|---|
| GAs | | Population size | 60 |
| | | Elite population size(W) | 30 |
| | String length | Premise structure (FNN) | 10(per one variable) |
| | | Consequence structure (PNN) | 3+3+24 |
| gHFNN | Premise (FNN) | No. of entire system inputs | 2/3 |
| | | Learning iteration | 300 |
| | | Learning rate tuned | 0.0052 |
| | | Momentum Coefficient tuned | 0.0004 |
| | | No. of rules | 4/6 |
| | Consequence (gPNN) | No. of entire inputs CP 1 | 4/6 |
| | | CP 2 | 2/3 |
| | | Maximal layer | 5 |
| | | No. of inputs to be selected(N) | $1 \leq N \leq 4$(Max) |
| | | Type(T) | $1 \leq T \leq 3$ |

N, T : integer

(b) In case of using FR_FNN.

| | | Generation | 150 |
|---|---|---|---|
| GAs | | Population size | 60 |
| | | Elite population size(W) | 30 |
| | String length | Premise structure (FNN) | 10(per one variable) |
| | | Consequence structure (PNN) | 3+3+24 |
| gHFNN | Premise (FR_FNN) | No. of entire system inputs | 2/3 |
| | | Learning iteration | 500 |
| | | Learning rate tuned | 0.0144 |
| | | Momentum Coefficient tuned | 0.00064 |
| | | No. of rules | 4/8 |
| | Consequence (gPNN) | No. of entire inputs | 4/8 |
| | | Maximal layer | 5 |
| | | No. of inputs to be selected(N) | $1 \leq N \leq 4$(Max) |
| | | Type(T) | $1 \leq T \leq 3$ |

N, T : integer

Next the values of the performance index of output of the gHFNN depend on each connection point based on two forms such as FS_FNN and FR_FNN. The values of the performance index vis-à-vis choice of number of layers of gHFNN related to the optimized architectures in each layer of the network are shown in Table 4. That is, according to the maximal number of inputs to be selected (Max=4), the selected node numbers, the selected polynomial type (Type T), and its corresponding performance index (PI and EPI) were shown when the genetic optimization for each layer was carried out. For example, in case when considering connection point 2 of FS_FNN with Type II in Table 4, let us investigate the $3^{rd}$ layer of the network (shadowed in Table 4). The fitness value in layer 3 attains its maximum for Max=4 when nodes 19, 30 (such as $z_{19}$, $z_{30}$) occur among preferred nodes

Table 4. Performance index of HFNN for the gas furnace.
(a) In case of using FS_FNN.

| Premise part | | | | C P | Consequence part | | | | | | | | | PI | EPI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fuzzy inference | No.of rules (MFs) | PI | EPI | | Layer | No.of inputs | Input No. | | | | | T | | PI | EPI |
| Linear | 4 (2+2) | 0.041 | 0.267 | 01 | 1 | 4 | 4 | 2 | 1 | 3 | | 3 | | 0.019 | 0.292 |
| | | | | | 2 | 4 | 7 | 12 | 2 | 10 | | 2 | | 0.018 | 0.271 |
| | | | | | 3 | 4 | 20 | 21 | 5 | 3 | | 2 | | 0.017 | 0.267 |
| | | | | | 4 | 3 | 22 | 13 | 29 | · | | 2 | | 0.016 | 0.263 |
| | | | | | 5 | 4 | 25 | 18 | 27 | 9 | | 3 | | 0.015 | 0.258 |
| | | | | 02 | 1 | 2 | 1 | 2 | · | · | | 3 | | 0.027 | 0.310 |
| | | | | | 2 | 3 | 4 | 6 | 5 | · | | 2 | | 0.021 | 0.279 |
| | | | | | 3 | 4 | 6 | 14 | 7 | 1 | | 2 | | 0.018 | 0.270 |
| | | | | | 4 | 3 | 15 | 3 | 2 | · | | 2 | | 0.018 | 0.263 |
| | | | | | 5 | 3 | 16 | 6 | 14 | · | | 2 | | 0.016 | 0.259 |
| Linear | 6 (2+2+2) | 0.0256 | 0.143 | 01 | 1 | 4 | 6 | 3 | 1 | 5 | | 3 | | 0.0218 | 0.136 |
| | | | | | 2 | 4 | 6 | 24 | 16 | 30 | | 3 | | 0.0197 | 0.124 |
| | | | | | 3 | 3 | 4 | 16 | 26 | · | | 1 | | 0.0196 | 0.121 |
| | | | | | 4 | 4 | 22 | 24 | 1 | 13 | | 1 | | 0.0193 | 0.119 |
| | | | | | 5 | 3 | 11 | 18 | 21 | · | | 3 | | 0.0191 | 0.117 |
| | | | | 02 | 1 | 3 | 1 | 2 | 3 | · | | 3 | | 0.0232 | 0.130 |
| | | | | | 2 | 4 | 12 | 15 | 13 | 6 | | 2 | | 0.0196 | 0.120 |
| | | | | | 3 | 2 | 19 | 30 | · | · | | 2 | | 0.0194 | 0.115 |
| | | | | | 4 | 4 | 2 | 21 | 11 | 5 | | 1 | | 0.0188 | 0.113 |
| | | | | | 5 | 4 | 13 | 3 | 26 | 25 | | 1 | | 0.0184 | 0.110 |

(b) In case of using FR_FNN.

| Premise part | | | | Consequence part | | | | | | | | PI | EPI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fuzzy Inference | No. of rules (MFs) | PI | EPI | Layer | No. of inputs | Input No. | | | | | T | PI | EPI |
| Linear | 4 (2x2) | 0.025 | 0.265 | 1 | 4 | 4 | 1 | 3 | 2 | | 2 | 0.019 | 0.267 |
| | | | | 2 | 4 | 7 | 1 | 13 | 22 | | 3 | 0.026 | 0.251 |
| | | | | 3 | 2 | 1 | 28 | · | · | | 3 | 0.025 | 0.244 |
| | | | | 4 | 2 | 7 | 6 | · | · | | 3 | 0.025 | 0.243 |
| | | | | 5 | 3 | 29 | 22 | 17 | · | | 3 | 0.016 | 0.249 |
| Linear | 8 (2x2x2) | 0.033 | 0.119 | 1 | 4 | 6 | 5 | 2 | 8 | | 1 | 0.083 | 0.146 |
| | | | | 2 | 4 | 21 | 18 | 6 | 9 | | 2 | 0.028 | 0.116 |
| | | | | 3 | 4 | 4 | 24 | 5 | 6 | | 2 | 0.022 | 0.110 |
| | | | | 4 | 3 | 28 | 4 | 5 | · | | 2 | 0.021 | 0.106 |
| | | | | 5 | 3 | 21 | 18 | 25 | · | | 1 | 0.021 | 0.104 |

($W$) chosen in the previous layer (the $2^{nd}$ layer) are selected as the node inputs in the present layer. Furthermore 2 inputs of Type 2 (linear function) were selected as the results of the genetic optimization, refer to Fig. 8(b). In the "Input No." item of Table 4, a blank node marked by period ( · ) indicates that it has not been selected by the genetic operation. The best results for the proposed network related to the output node mentioned previously were reported as PI=0.0194 and EPI=0.115 for layer 3 (see Fig. 8(b)), and PI=0.0184 and EPI=0.110 for layer 5. In the sequel, the depth (the number of layers) and the width (the number of nodes) as well as the number of entire nodes (inputs) of the proposed genetically optimized HFNN (gHFNN) can be lower in comparison to the "conventional HFNN" (which immensely contributes to the compactness of the resulting network). In what follows, the genetic design procedure at stage (layer) of HFNN leads to the selection of the preferred nodes (or PNs) with optimal local characteristics (such as the number of input variables, the order of the polynomial, and input variables). In addition, when considering FR_FNN with Type II (8 fuzzy rules), the best results are reported in the form of the performance index such as PI=0.022 and EPI=0.110 for layer 3, and PI=0.021 and EPI=0.104 for layer 5. The optimal topology for layer 3 is shown in Fig. 9(b).

Tables 5-6 present the performance index of the corresponding node in three layers of genetically optimized PNN (gPNN) dynamically generated as the consequent part of HFPNN shown in Figs. 8-9.
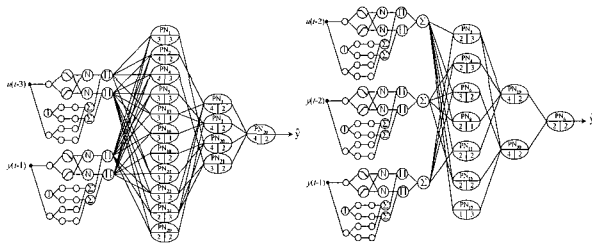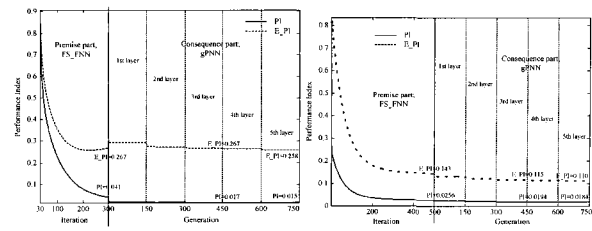


(a) In case of Type I.            (b) In case of Type II.

Fig. 8. Genetically optimized HFNN (gHFNN) with FS_FNN.
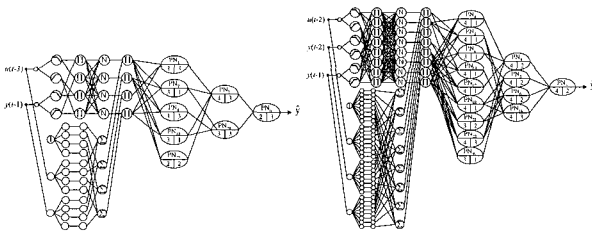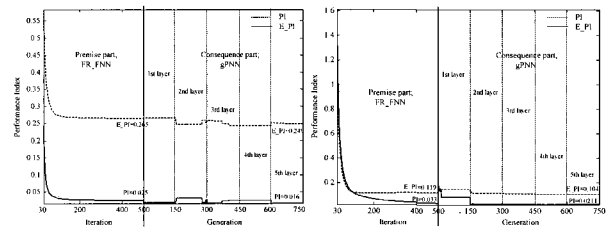


(a) In case of Type I.            (b) In case of Type II.

Fig. 9. Genetically optimized HFNN (gHFNN) with FR_FNN.



(a) FS_FNN with Type I.    (b) FS_FNN with Type II.

Fig. 10. Optimization procedure of FS_FNN based HFNN by BP learning and GAs.



(a) FR_FNN with Type I.    (b) FR_FNN with Type II.

Fig. 11. Optimization procedure of FR_FNN based HFNN by BP learning and GAs.

Table 5. Performance Index of the corresponding node in three layers of genetically optimized PNN combined with FS_FNN.

(a) In case of using Type I.

| $1^{st}$ Layer | | $2^{nd}$ Layer | | | $3^{rd}$ Layer | | |
|---|---|---|---|---|---|---|---|
| No. | PI | E_PI | No. | PI | E_PI | No. | PI | E_PI |
| 1 | 0.022892 | 0.34449 | | | | | |
| 2 | 0.019152 | 0.29844 | | | | | |
| 4 | 0.019435 | 0.29263 | | | | | |
| 5 | 0.02283 | 0.34192 | 3 | 0.018712 | 0.27678 | | |
| 13 | 0.032813 | 0.31034 | 5 | 0.025978 | 0.26627 | 29 | 0.017446 | 0.26704 |
| 16 | 0.023313 | 0.32909 | 20 | 0.018295 | 0.27655 | | |
| 18 | 0.51323 | 0.5968 | 21 | 0.019051 | 0.27617 | | |
| 21 | 0.022597 | 0.34189 | | | | | |
| 23 | 0.022151 | 0.34656 | | | | | |
| 24 | 0.028435 | 0.30079 | | | | | |
| 29 | 0.023127 | 0.33766 | | | | | |

(b) In case of using Type II.

| $1^{st}$ Layer | | $2^{nd}$ Layer | | | $3^{rd}$ Layer · | | |
|---|---|---|---|---|---|---|---|
| No. | PI | E_PI | No. | PI | E_PI | No. | PI | E_PI |
| 3 | 0.095813 | 0.38431 | | | | | |
| 4 | 0.10422 | 0.21227 | | | | | |
| 6 | 0.021857 | 0.1429 | 19 | 0.019745 | 0.12828 | 6 | 0.019431 | 0.1155 |
| 8 | 0.42585 | 1.3038 | 30 | 0.019618 | 0.12456 | | |
| 11 | 0.091367 | 0.40557 | | | | | |
| 15 | 0.40244 | 1.346 | | | | | |
| 17 | 4.0107 | 5.9559 | | | | | |

When using FS_FNN, Fig. 8 illustrates the detailed optimal topology of the gHFNN with 3 layers of PNN; the networks come with the following values: PI=0.017, EPI=0.267 for Type I(4 fuzzy rules), and PI=0.194, EPI= 0.115 for Type II(6 fuzzy rules). In case of using FR_FNN, Fig. 9 depicts the detailed optimal topologies of the gHFNN with 3 layers of PNN, the minimal values of the performance index, that is PI=0.025, EPI=0.244 for Type I, and PI=0.022,

Table 6. Performance Index of the corresponding node in three layers of genetically optimized PNN combined with FR_FNN.

(a) In case of using Type I.

| 1st Layer | | | 2nd Layer | | | 3rd Layer | | |
|---|---|---|---|---|---|---|---|---|
| No. | PI | E_PI | No. | PI | E_PI | No. | PI | E_PI |
| 1 | 0.5143 | 1.5179 | 1 | 0.026649 | 0.25168 | 6 | 0.025519 | 0.24411 |
| 7 | 0.39659 | 0.97722 | 28 | 0.032416 | 0.2476 | | | |
| 8 | 0.50153 | 1.2104 | | | | | | |
| 13 | 0.5445 | 1.2858 | | | | | | |
| 22 | 2.7686 | 5.1946 | | | | | | |

(b) In case of using Type II.

| 1st Layer | | | 2nd Layer | | | 3rd Layer | | |
|---|---|---|---|---|---|---|---|---|
| No. | PI | E_PI | No. | PI | E_PI | No. | PI | E_PI |
| 4 | 0.083797 | 0.17231 | 4 | 0.031007 | 0.11729 | 5 | 0.022604 | 0.11023 |
| 5 | 0.21702 | 0.30624 | 5 | 0.029359 | 0.11921 | | | |
| 6 | 0.22268 | 0.30464 | 6 | 0.026414 | 0.11954 | | | |
| 13 | 0.25589 | 0.30755 | 24 | 0.030291 | 0.1171 | | | |
| 17 | 0.43082 | 0.58043 | | | | | | |
| 18 | 0.08356 | 0.14672 | | | | | | |
| 21 | 0.16308 | 0.14419 | | | | | | |
| 23 | 0.15087 | 0.13321 | | | | | | |
| 30 | 0.43268 | 0.55159 | | | | | | |

Table 7. Performance analysis of selected models.

| Model | | | PI | EPI | No. of rules |
|---|---|---|---|---|---|
| Kim, et al.'s model [23] | | | 0.034 | 0.244 | 2 |
| Lin and Cunningham's mode [24] | | | 0.071 | 0.261 | 4 |
| Fuzzy | Min-Max [4] | Simplified | 0.022 | 0.335 | 4(2×2) |
| | | | 0.022 | 0.336 | 6(3×2) |
| | | Linear | 0.024 | 0.358 | 4(2×2) |
| | | | 0.020 | 0.362 | 6(3×2) |
| | GAs [4] | Simplified | 0.023 | 0.344 | 4(2×2) |
| | | Linear | 0.018 | 0.264 | 4(2×2) |
| | Complex [1] | Simplified | 0.024 | 0.328 | 4(2×2) |
| | | Linear | 0.023 | 0.306 | 4(2×2) |
| | Hybrid [3] (GAs+Complex) | Simplified | 0.024 | 0.329 | 4(2×2) |
| | | Linear | 0.017 | 0.289 | 4(2×2) |
| | HCM [2] | Simplified | 0.755 | 1.439 | 6(3×2) |
| | | Linear | 0.018 | 0.286 | 6(3×2) |
| | HCM+GAs [2] | Simplified | 0.035 | 0.289 | 4(2×2) |
| | | | 0.022 | 0.333 | 6(3×2) |
| | | Linear | 0.026 | 0.272 | 4(2×2) |
| | | | 0.020 | 0.264 | 6(3×2) |
| Neural Networks [2] | | | 0.034 | 4.997 | |
| • Oh's Adaptive FNN [5] | | | 0.021 | 0.332 | 9(3×3) |
| | | | 0.022 | 0.353 | 4(2×2) |
| FNN [25] | Simplified | | 0.043 | 0.264 | 6(3+3) |
| | Linear | | 0.037 | 0.273 | 6(3+3) |
| Multi-FNN [26] | Simplified | | 0.025 | 0.274 | 6(3+3) |
| | Linear | | 0.024 | 0.283 | 6(3+3) |
| SOFPNN | Generic (FS_FNN) [27,28] | | 0.023 | 0.277 | 4 rules/5th layer(NA) |
| | | | 0.020 | 0.119 | 6 rules/5th layer (22 nodes) |
| | Advanced (FS_FNN) [28] | | 0.019 | 0.264 | 4 rules/5th layer(NA) |
| | | | 0.017 | 0.113 | 6 rules/5th layer (26 nodes) |
| Proposed model | gHFNN (FS_FNN) | | 0.017 | 0.267 | 4 rules/3rd layer (16 nodes) |
| | | | 0.019 | 0.115 | 6 rules/3rd layer (10 nodes) |
| | gHFNN (FR_FNN) | | 0.025 | 0.244 | 4 rules/3rd layer (8 nodes) |
| | | | 0.022 | 0.110 | 7 rules/3rd layer (14 nodes) |

NA: Not Available, layer (●): the number of entire nodes of the corresponding networks

EPI=0.110 for Type II are obtained. The proposed network enables the architecture to be a structurally optimized and gets simpler than the conventional HFNN. Figs. 10-11 illustrate the optimization process by visualizing the performance index in successive cycles of both BP learning and genetic optimization when using linear fuzzy inference-based FS_FNN or FR_FNN, refer to Tables 4-6 and Figs. 8-9. Table 7 contrasts the performance of the genetically developed network with other fuzzy and fuzzy-neural networks reported in the literature. It becomes obvious that the proposed genetically optimized HFNN architectures outperform other models both in terms of their accuracy and generalization capabilities.

## 6. CONCLUDING REMARKS

In this study, we have introduced a class of gHFNN driven genetic optimization regarded as a modeling vehicle for nonlinear and complex systems. The genetically optimized HFNNs are constructed by combining FNNs with gPNNs. The proposed model comes with two kinds of rule-based FNNs (viz. FS_FNN and FR_FNN based on linear fuzzy inferences) as well as a diversity of local characteristics of PNs that are extremely useful when coping with various nonlinear characteristics of the system under consideration. In what follows, in contrast to the conventional HFNN structures and their learning, the depth (the number of layers) and the width (the number of nodes) as well as the number of entire nodes (inputs) of the proposed genetically optimized HFNN (gHFNN) can be lower.

The comprehensive design methodology comes with the parametrically as well as structurally optimized network architecture. A few general notes are worth stressing: 1) as the premise structure of the gHFNN, the optimization of the rule-based FNN hinges on genetic algorithms and back-propagation (BP) learning algorithm: The GAs leads to the auto-tuning of vertexes of membership function, while the BP algorithm helps produce optimal parameters of the consequent polynomial of fuzzy rules through learning; 2) the gPNN that is the consequent structure of the gHFNN is based on the technologies of the extended GMDH algorithm and GAs: The extended GMDH method is comprised of both a structural phase such as a self-organizing and evolutionary algorithm and a parametric phase driven by the least square error (LSE)-based learning. Furthermore the PNN architecture is optimized by the genetic optimization that leads to the selection of the optimal nodes (or PNs) with local characteristics such as the number of input variables, the order of the polynomial, and a collection of the specific subset of input variables. In this sense, we have constructed a coherent development platform in which all

components of CI are fully utilized. In the sequel, a variety of architectures of the proposed gHFNN driven to genetic optimization have been discussed. The model is inherently dynamic - the use of the genetically optimized PNN (gPNN) of consequent structure of the overall network is essential to the generation process of the "optimally self-organizing" network by selecting its width and depth. The series of experiments helped compare the network with other models through which we found the network to be of superior quality.

## REFERENCES

[1] S.-K. Oh and W. Pedrycz, "Fuzzy identification by means of auto-tuning algorithm and its application to nonlinear systems," *Fuzzy Sets and Systems*, vol. 115, no. 2, pp. 205-230, 2000.

[2] B.-J. Park, W. Pedrycz, and S.-K. Oh, "Identification of fuzzy models with the aid of evolutionary data granulation," *IEE Proceedings -Control Theory and Application*, vol. 148, no. 5, pp. 406-418, 2001.

[3] S.-K. Oh, W. Pedrycz, and B.-J. Park, "Hybrid identification of fuzzy rule-based models," *International Journal of Intelligent Systems*, vol. 17, no. 1, pp. 77-103, 2002.

[4] B.-J. Park, S.-K. Oh, T.-C. Ahn, and H.-K. Kim, "Optimization of fuzzy systems by means of GA and weighting factor," *The Transactions of the Korean Institute of Electrical Engineers* (in Korean), vol. 48A, no. 6, pp. 789-799, June 1999.

[5] S.-K. Oh, C.-S. Park, and B.-J. Park, "On-line modeling of nonlinear process systems using the adaptive fuzzy-neural networks," *The Transactions of the Korean Institute of Electrical Engineers* (in Korean), vol. 48A, no. 10, pp. 1293-1302, 1999.

[6] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. on Neural Networks*, vol. 2, no. 2, pp. 252-262, March 1991.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 1989.

[8] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg, 1996.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbour, 1975.

[10] W. Pedrycz and J. F. Peters, *Computational Intelligence and Software Engineering*, World Scientific, Singapore, 1998.

[11] S.-K. Oh, *Computational Intelligence by Programming focused on Fuzzy, Neural Networks,*

and Genetic Algorithms (in Korean), Naeha, 2002.

[12] S.-K. Oh and W. Pedrycz, "The design of self-organizing polynomial neural networks," *Information Sciences*, vol. 141, no. 3-4, pp. 237-258, 2002.

[13] S.-K. Oh, W. Pedrycz, and B.-J. Park, "Polynomial neural networks architecture: analysis and design," *Computers and Electrical Engineering*, vol. 29, no. 6, pp. 653-725, 2003.

[14] T. Ohtani, H. Ichihashi, T. Miyoshi, and K. Nagasaka, "Orthogonal and successive projection methods for the learning of neurofuzzy GMDH," *Information Sciences*, vol. 110, pp. 5-24, 1998.

[15] T. Ohtani, H. Ichihashi, T. Miyoshi, and K. Nagasaka, "Structural learning with M-apoptosis in neurofuzzy GMHD," *Proc. of the 7th IEEE Iinternational Conference on Fuzzy Systems*, pp. 1265-1270, 1998.

[16] H. Ichihashi and K. Nagasaka, "Differential minimum bias criterion for neuro-fuzzy GMDH," *Proc. of 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing IIZUKA'94*, pp. 171-172, 1994.

[17] S.-K. Oh, W. Pedrycz, and B.-J. Park, "Self-organizing neurofuzzy networks based on evolutionary fuzzy granulation," *IEEE Trans. on Systems, Man and Cybernetics-A*, vol. 33, no. 2, pp. 271-277, 2003.

[18] L. Magdalena, O. Cordon, F. Gomide, F. Herrera, and F. Hoffmann, "Ten years of genetic fuzzy systems: current framework and new trends," *Fuzzy Sets and Systems*, vol. 141, no. 1, pp. 5-31, 2004.

[19] A. G. Ivahnenko, "The group method of data handling: a rival of method of stochastic approximation," *Soviet Automatic Control*, vol. 13, no. 3, pp. 43-55, 1968.

[20] T. Yamakawa, "A new effective learning algorithm for a neo fuzzy neuron model," *Proc. of 5th IFSA World Conference*, pp. 1017-1020, 1993.

[21] S.-K. Oh, K.-C. Yoon, and H.-K. Kim, "The design of optimal fuzzy-neural networks structure by means of GA and an aggregate weighted performance index," *Journal of Control, Automation and Systems Engineering* (in Korean), vol. 6, no. 3, pp. 273-283, 2000.

[22] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting, and Control*, 2nd edition Holden-Day, SanFransisco, 1976.

[23] E. Kim, H. Lee, M. Park, and M. Park, "A simply identified Sugeno-type fuzzy model via double clustering," *Information Sciences*, vol. 110, pp. 25-39. 1998.

[24] Y. Lin and G. A. Cunningham III, "A new

approach to fuzzy-neural modeling," *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 2, pp. 190-197, 1997.

[25] S.-K. Oh, W. Pedrycz, and H.-S. Park, "Hybrid identification in fuzzy-neural networks," *Fuzzy Sets and Systems*, vol. 138, no. 2, pp. 399-426, 2003.

[26] H.-S. Park and S.-K Oh, "Multi-FNN identification by means of HCM clustering and its optimization using genetic algorithms," *Journal of Fuzzy Logic and Intelligent Systems* (in Korean), vol. 10, no. 5, pp. 487-496, 2000.

[27] B.-J. Park, S.-K. Oh, and S.-W. Jang, "The design of adaptive fuzzy polynomial neural networks architectures based on fuzzy neural networks and self-organizing networks," *Journal of Control, Automation and Systems Engineering* (in Korean), vol. 8, no. 2, pp.126-135, 2002.

[28] B.-J. Park and S.-K. Oh, "The analysis and design of advanced neurofuzzy polynomial networks," *Journal of the Institute of Electronics Engineers of Korea* (in Korean), vol. 39-CI, no. 3, pp. 18-31, 2002.

**Sung-Kwun Oh** received his 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 held a position of a Postdoctoral Fellow in the Department of Electrical and Computer Engineering, University of Manitoba, Canada. He is currently a Professor in the Department of Electrical Engineering, University of Suwon, Korea. His research interests include fuzzy systems, fuzzy-neural networks, automation systems, advanced Computational Intelligence, and intelligent control. He is a member of IEEE. He currently serves as an Associate Editor of KIEE Transactions on Systems & Control, and Journal of Control, Automation & Systems Engineering of the ICASE. Korea.



**Byoung-Jun Park** received his B.S., M.S., and Ph.D. degrees in Control and Instrumentation Engineering from Wonkwang University, Korea in 1998, 2000, and 2003, respectively. He currently holds a position of a Postdoctoral Fellow in the Department of Electrical and Computer Engineering, University of Alberta, Canada, His research interests encompass fuzzy, neurofuzzy systems, genetic algorithms, computational intelligence, hybrid systems, and intelligent control



**Hyun-Ki Kim** received his B.S., M.S., and Ph.D. degrees in Electrical Engineering from Yonsei University, Seoul, Korea, in 1977, 1985 and 1991, respectively. During 1999-2003, he worked as Chairman at the Korea Association of Small Business Innovation Research. He is currently a Professor in the Dept. of Electrical Engineering, Suwon University, Korea. His research interests include system automation, and intelligent control. He currently serves as a Chief Editor for the Journal of Korea Association of Small Business Innovation Research.