Genetically Optimized Hybrid Fuzzy Neural Networks: Analysis and Design of Rule-based Multi-layer Perceptron Architectures

Sung-Kwun Oh and Witold Pedrycz

Summary. 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 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. The optimization of the FNN is realized with the aid of a standard back-propagation learning algorithm and genetic optimization. We distinguish between two types of the fuzzy 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. Through the consecutive process of such structural and parametric optimization, an optimized PNN is generated in a dynamic fashion.

To evaluate the performance of the gHFNN, the models are experimented with several representative numerical examples. A comparative analysis demonstrates that the proposed gHFNN come with higher accuracy as well as superb predictive capabilities when comparing with other neurofuzzy models.

1 Introductory remarks

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 models should be able to take advantage of the existing

S.-K. Oh and W. Pedrycz: *Genetically Optimized Hybrid Fuzzy Neural Networks: Analysis and Design of Rule-based Multi-layer Perceptron Architectures*, Studies in Computational Intelligence (SCI) **82**, 23–57 (2008)
www.springerlink.com

domain knowledge (such as a prior experience of human observers or process operators) and augment it by available numeric data to form a coherent data-knowledge modeling entity. The omnipresent modeling tendency is the one that exploits techniques of Computational Intelligence (CI) by embracing fuzzy modeling [1–6], neurocomputing [7], and genetic optimization [8–10]. Especially the two of the most successful approaches have been the hybridization attempts made in the framework of CI [11,12]. Neuro-fuzzy systems are one of them [13–20]. 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 [21].

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 [19,20,34,35]) with genetic algorithms (GAs). The conventional HFNNs exhibits FNN architecture treated as the premise part while the PNN structures are used in common as the conclusion part of HFNNs. 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. Each of them is placed into the two main categories according to the type of fuzzy inference, namely the simplified and linear fuzzy inference. 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 [14,15] and GAs [8–10]. gPNN leads to the effective reduction of the depth of the networks as well as the width of the layer, and the avoidance of a substantial amount of time-consuming iterations for finding the most preferred networks in conventional PNN. 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 Figs. 1–3 [19,20,34,35]. Let us recall that the fuzzy inference (both simplified and linear) -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 (see Table 1), 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.

According to the alternative position of two connection points (interface) in case of th usage of FS_FNN shown in Fig. 2, we realize a different combination of FNN and PNN while forming the HFNN architecture. Especially when dealing with the interface of FNN realized by means of PNNs, we note that if input variables to PNN used in the consequence part of HFNN are less than three (or four), the generic type of HFNN does not generate a highly versatile structure. As visualized in Figs. 1–3, we identify also two types of the topology, namely a generic and advanced type. Observe that in Figs. 2–3, z_i'(Case 2) in the 2^{nd} layer or higher indicates that the polynomial order of

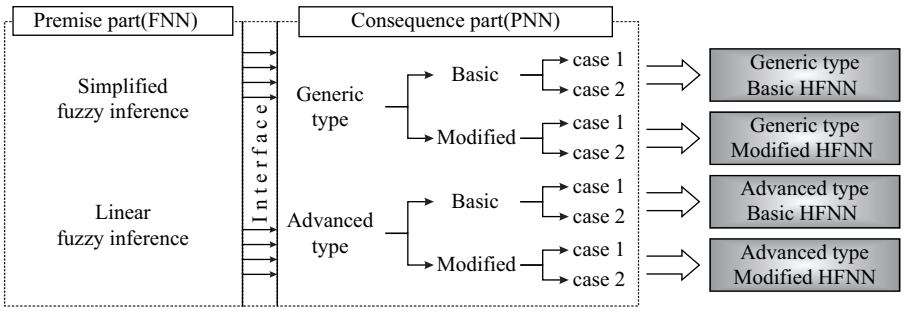
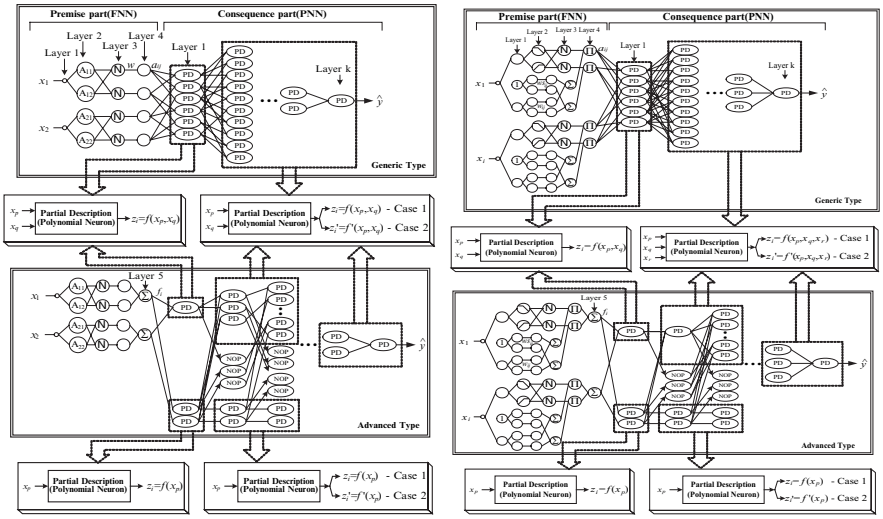
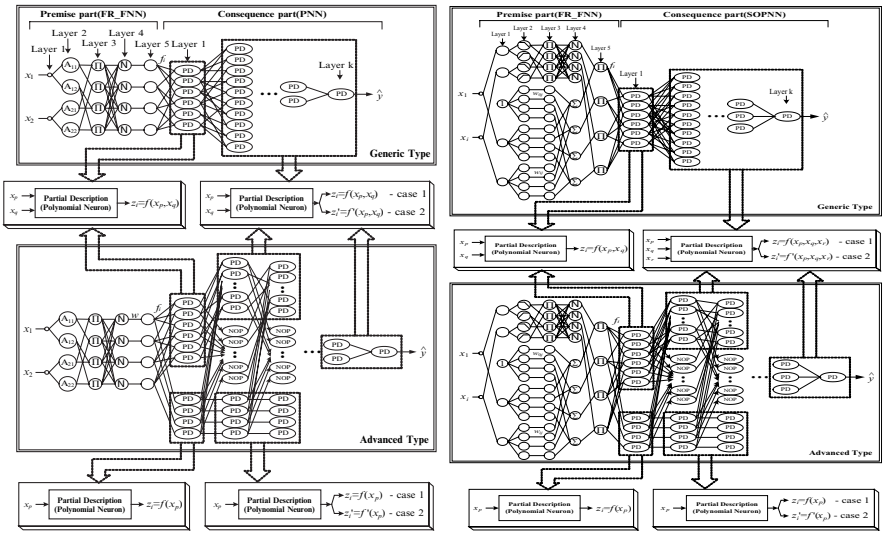


Fig. 1. Overall diagram for generating the conventional HFNN architecture



(a) Basic HFNN with simplified fuzzy inference (b) Modified HFNN with linear fuzzy inference

Fig. 2. FS_HFNN architecture combined with FS_FNN and PNN



(a) Basic HFNN with simplified fuzzy inference (b) Modified HFNN with linear fuzzy inference

Fig. 3. FR_HFNN architecture combined with FR_FNN and PNN

Table 1. Taxonomy of various PNN architectures

Layer of PNN	No. of input variables of polynomial	Order of Polynomial	PNN architecture
1^{st} layer	p	Type P	(1) p = q : Basic PNN a) Type P = Type Q: Case1 b) Type P≠Type Q: Case2
2^{nd} or higher layer	q	Type Q	(2) p≠q : Modified PNN a) Type P = Type Q: Case1 b) Type P ≠ Type Q: Case2

(p = 2,3,4,5, q = 2,3,4,5 ; P = 1,2,3, Q = 1,2,3)

the PD of each node has a different type in comparison with z_i of the 1^{st} layer. The "NOP" node states that the A^{th} node of the current layer is the same as the node positioned in the previous layer (NOP stands for "no operation"). An arrow to the NOP node is used to show that the same node moves from the previous layer to the current one.

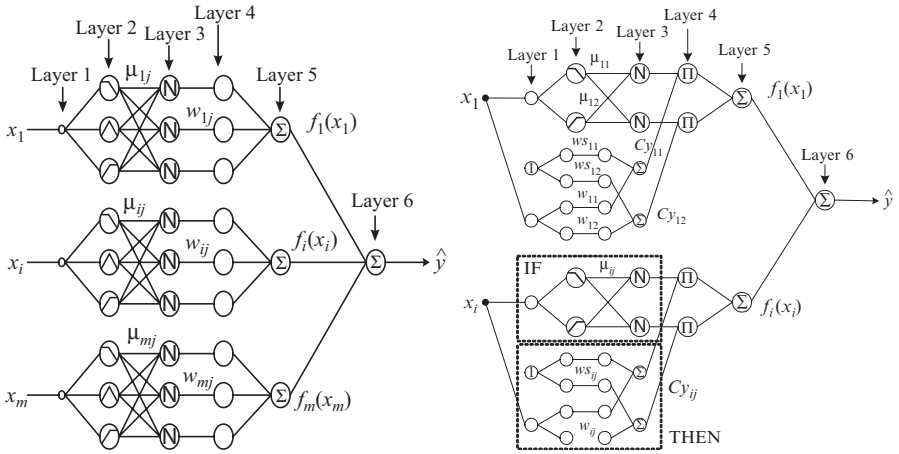
3 The architecture and development of genetically optimized HFNN (gHFNN)

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 [24,32] and PNN [14,15].

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

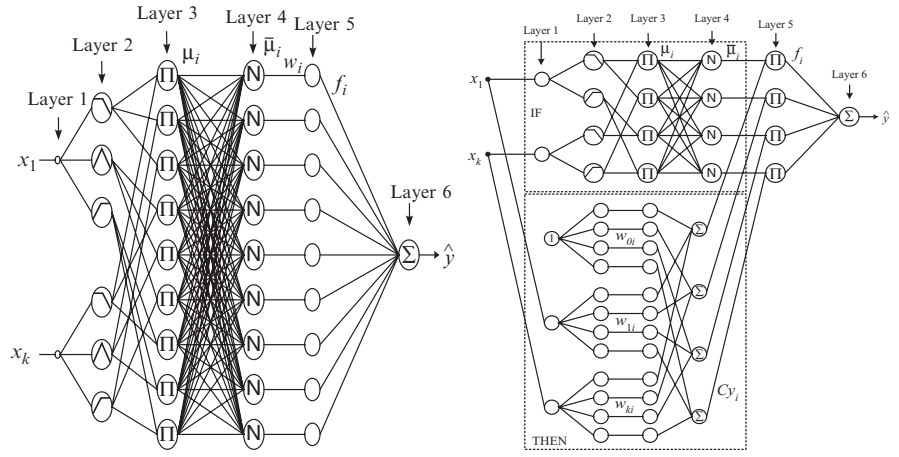
3.1 Fuzzy neural networks based on genetic optimization

We consider two kinds of FNNs (viz. FS_FNN and FR_FNN) based on two types of fuzzy inferences, namely, simplified and linear fuzzy inferences. The structure of the FNN is the same as the used in the premise of the conventional HFNN. 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. 4–5.



(a) Simplified fuzzy inference (Scheme I) (b) Linear fuzzy inference (Scheme II)

Fig. 4. Topology of FS_FNN by using space partitioning in terms of individual input variables



(a) Simplified fuzzy inference (Scheme I) (b) Linear fuzzy inference (Scheme II)

Fig. 5. Topology of FR_FNN by using space partitioning in terms of an ensemble of input variables

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 "$\sum$" 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 $\widehat{y}$ is governed by the following expression;

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

with m being the number of the input variables (viz. the number of the outputs f_i's of the "$\sum$" 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 2(a). Table 2(a) represents the comparison of fuzzy rules, inference result and learning for two types of FNNs. In Table 2(a), 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 μ_{ij}. In the simplified fuzzy inference, ω_{ij} is a constant consequence of the rules and, in the linear fuzzy inference, ωs_{ij} is a constant consequence and ω_{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. 4. 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 2(a) [23,24]. The learning of FNN is realized by adjusting connections of the neurons and as such it follows a standard Back-Propagation (BP) algorithm [23,24]. The complete update formulas are covered in Table 2(a). Where η is a positive learning rate and α is a positive momentum coefficient. The case of FR_FNN, see Table 2(b), is carried out in a same manner as outlined in Table 2(a) (the case 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. 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

Table 2. Comparison of simplified with linear fuzzy inference-based FNNs

(a) In case of using FS_FNN

Structure	Simplified fuzzy inference (Scheme I)	Linear fuzzy inference (Scheme II)
Fuzzy rules	R^1 : If x_i is A_{i1} then $\mathrm{Cy}_{i1} = \omega_{i1}$ $\vdots$ R^j : If x_i is A_{ij} then $\mathrm{Cy}_{ij} = \omega_{ij}$ $\vdots$ R^z : If x_i is A_{iz} then $\mathrm{Cy}_{iz} = \omega_{iz}$	$R^1 : \cdots$ then $\mathrm{Cy}_{i1} = \omega s_{i1} + x_i\omega_{i1}$ $\vdots$ $R^j : \cdots$ then $\mathrm{Cy}_{ij} = \omega s_{ij} + x_i\omega_{ij}$ $\vdots$ $R^z : \cdots$ then $\mathrm{Cy}_{iz} = \omega s_{iz} + x_i\omega_{iz}$
Inference result	$f_i(x_i) = \dfrac{\sum_{j=1}^{z} \mu_{ij}(x_i)\cdot\omega_{ij}}{\sum_{j=1}^{z}\mu_{ij}(x_i)}$ $= \mu_{ik}\cdot(x_i)\omega_{ik}$ $+ \mu_{ik}(x_i)\cdot\omega_{ik+1}$ $\widehat{y} = \sum_{i=1}^{n} f_i(x_i)$	$f_i(x_i) = \dfrac{\sum_{j=1}^{z}(\mu_{ij}(x_i)\cdot(\omega s_{ij} + x_i\omega_{ij}))}{\sum_{j=1}^{z}\mu_{ij}(x_i)}$ $= \mu_{ik}(x_i)\cdot(\omega s_{ik} + x_i\omega_{ik})$ $+ \mu_{ik+1}(x_i)$ $\cdot(\omega s_{ik+1} + x_i w_{ik+1})$ $\widehat{y} = \sum_{i=1}^{n} f_i(x_i)$
Learning	$\Delta\omega_{ij} = 2\cdot\eta\cdot(y_p - \widehat{y}_p)\cdot\mu_{ij}(x_i)$ $+ \alpha(\omega_{ij}(t) - \omega_{ij}(t-1))$	$\begin{cases}\Delta\omega s_{ij} = 2\cdot\eta\cdot(y-\widehat{y})\cdot\mu_{ij} \\ \quad + \alpha(\omega s_{ij}(t) - \omega s_{ij}(t-1)) \\ \Delta\omega = 2\cdot\eta\cdot(y-\widehat{y})\cdot\mu_{ij}\cdot x_i \\ \quad + \alpha(\omega_{ij}(t) - \omega_{ij}(t-1))\end{cases}$

(b) In case of using FR_FNN

Structure		fuzzy inference	Linear
Fuzzy rules	Premise part	R^1 : If x_1 is A_{11}, $\cdots$, and x_k is A_{k1} $\vdots$ R^i : If x_1 is A_{1i}, $\cdots$, and x_k is A_{ki} $\vdots$ R^n : If x_1 is A_{1n}, $\cdots$, and x_k is A_{kn}	
	Consequence part	then $\mathrm{C}y_1 = \omega_1$ $\vdots$ then $\mathrm{C}y_i = \omega_i$ $\vdots$ then $\mathrm{C}y_n = \omega_n$	then $\mathrm{C}y_1 = \omega_{01} + \omega_{11}\cdot x_1 + \cdots + \omega_{k1}\cdot x_k$ $\vdots$ then $\mathrm{C}y_i = \omega_{0i} + \omega_{1i}\cdot x_1 + \cdots + \omega_{ki}\cdot x_k$ $\vdots$ then $\mathrm{C}y_n = \omega_{0n} + \omega_{1n}\cdot x_1 + \cdots + \omega_{kn}\cdot x_k$

(*continued*)

Table 2. (Continued)

Inference result	$\widehat{y} = \sum_{i=1}^{n} f_i = \sum_{i=1}^{n} \bar{\mu_i} \cdot \omega_i = \sum_{i=1}^{n} \frac{\mu_i \cdot \omega_i}{\sum_{i=1}^{n} \mu_i}$	$\widehat{y} = \sum_{i=1}^{n} f_i = \sum_{i=1}^{n} \bar{\mu_i} \cdot (\omega_{0i} + \omega_{1i} \cdot x_1 + \omega_{ki} \cdot x_k) = \sum_{i=1}^{n} \frac{\mu_i \cdot (\omega_{0i} + \omega_{1i} \cdot x_1 + \omega_{ki} \cdot x_k)}{\sum_{i=1}^{n} \mu_i}$
Learning	$\Delta\omega_i = 2 \cdot \eta \cdot (y - \widehat{y}) \cdot \bar{\mu_i} + \alpha(\omega_i(t) - \omega_i(t-1))$	$\begin{cases} \Delta\omega_{0i} = 2 \cdot \eta \cdot (y - \widehat{y}) \cdot \bar{\mu_i} + \alpha(\omega_{0i}(t) - \omega_{0i}(t-1)) \\ \Delta\omega_{ki} = 2 \cdot \eta \cdot (y - \widehat{y}) \cdot \bar{\mu_i} \cdot x_k + \alpha(\omega_{ki}(t) - \omega_{ki}(t-1)) \end{cases}$

search is directed only by some fitness function whose form could be quite complex; we do not require its differentiability [8–10].

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 [19,20,34,35].

3.2 Genetically optimized PNN (gPNN)

As underlined, the PNN algorithm is based upon the GMDH [22] 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 [14,15].

In addressing the problems with the conventional PNN (see Fig. 6), 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 Figs. 8–9 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. 7. The determination of the optimal values of the parameters available within an individual PN (viz. the number of input variables,

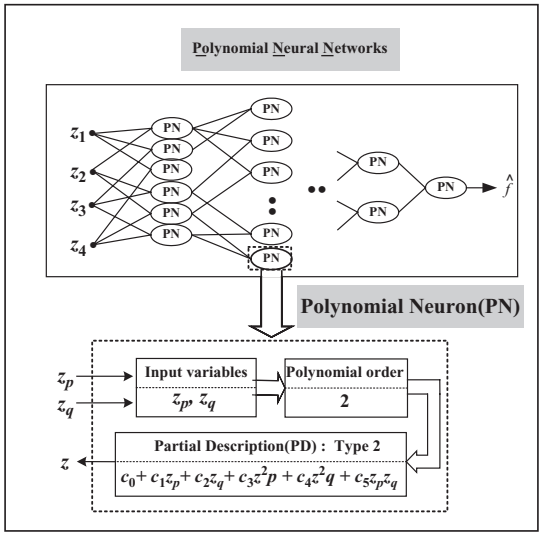


Fig. 6. A general topology of the PN-based PNN: note a biquadratic polynomial occurring in the partial description

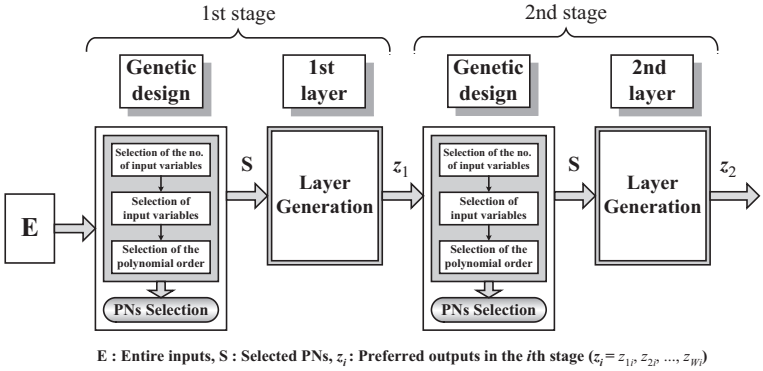


Fig. 7. Overall genetically-driven structural optimization process of PNN

the order of the polynomial, and input variables) leads to a structurally and parametrically optimized network. As a result, this network is more flexible as well as it exhibits simpler topology in comparison to the conventional PNN discussed in the previous research [14,15].

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 [8,9].

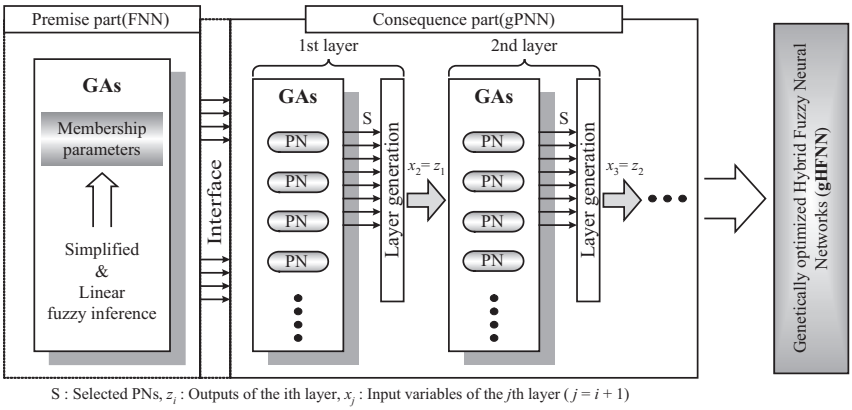


Fig. 8. Overall diagram for generation of gHFNN architecture

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 Figs. 8–9. 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 types of fuzzy inferences-based FNN (viz. FS-FNN or FR-FNN) realized with the fuzzy partitioning of individual input variables or an ensemble of input variables 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 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.

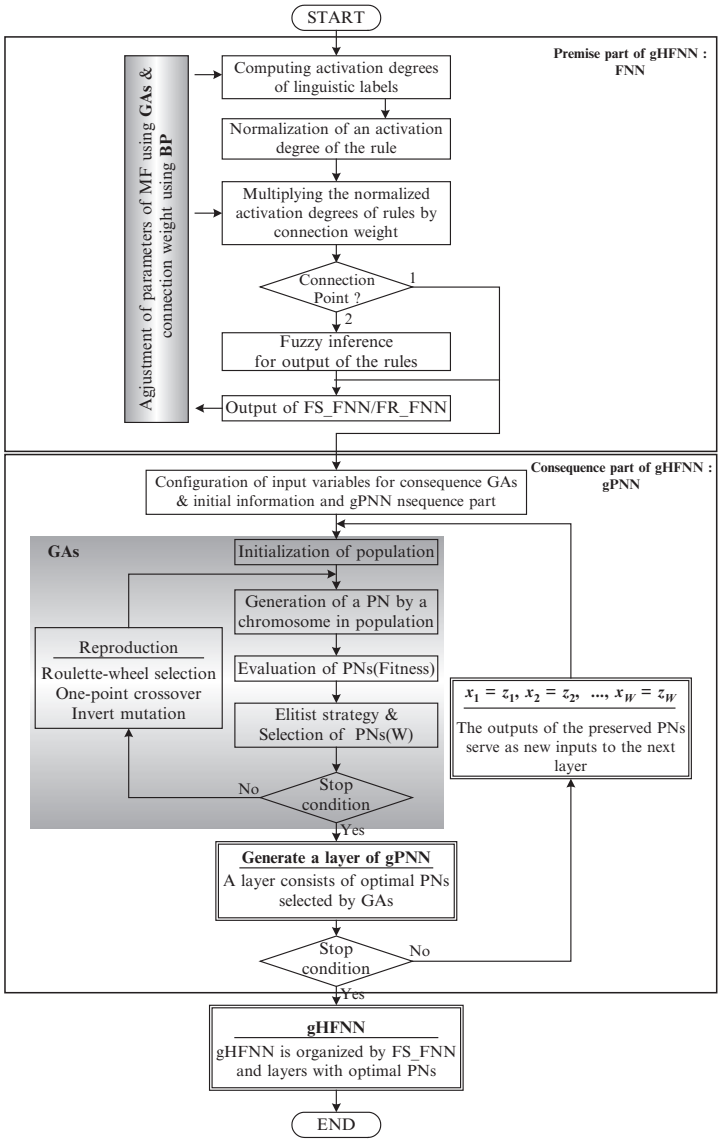


Fig. 9. Overall design flowchart of the gHFNN architecture

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.

$$\bar{\mu}_{ij} = \frac{\mu_{ij}}{\sum_{j=1}^{n} \mu_{ij}} = \frac{\mu_{ik}}{\mu_{ik} + \mu_{ik+1}} = \mu_{ik} \tag{2}$$

where n is the number of membership function for each input variable. An input signal xi 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) [23,24].
[**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

$$a_{ij} = \bar{\mu}_{ij} \times Cy_{ij} = \mu_{ij} \times Cy_{ij} \tag{3}$$

$$\begin{cases} Simplified : Cy_{ij} = \omega_{ij} \\ Linear : Cy_{ij} = \omega s_{ij} + \omega_{ij} \cdot x_i \end{cases} \tag{4}$$

If we choose CP (connection point) 1 for combining FS_FNN with PNN as shown in Fig. 10, a_{ij} is given as the input variable of the PNN. If we choose CP 2, $f_i(x_i)$ 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. 4, the output of each node in the 5th layer of the premise part of gHFNN is inferred

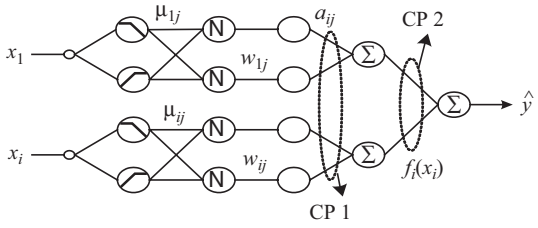


Fig. 10. Connection points used for combining FS_FNN (Simplified) with gPNN

by the center of gravity method [23,24]. If we choose CP 2, fi is the input variable of gPNN that is the consequence part of gHFNN

$$Simplified: f_i(x_i) = \frac{\sum_{j=1}^{z} a_{ij}}{\sum_{j=1}^{z} \mu_{ij}(x_i)} = \frac{\sum_{j=1}^{z} \mu_{ij}(x_i) \cdot \omega_{ij}}{\sum_{j=1}^{z} \mu_{ij}(x_i)} = \mu_{ik}(x_i) \cdot \omega_{ik} + \mu_{ik+1}(x_i) \cdot \omega_{ik+1} \tag{5}$$

$$Linear: f_i(x_i) = \frac{\sum_{j=1}^{z} a_{ij}}{\sum_{j=1}^{z} \mu_{ij}(x_i)} = \frac{\sum_{j=1}^{z} \mu_{ij}(x_i) \cdot (\omega s_{ij} + x_i \omega_{ij})}{\sum_{j=1}^{z} \mu_{ij}(x_i)} = \mu_{ik}(x_i) \cdot \omega_{ik} + \mu_{ik+1}(x_i) \cdot \omega_{ik+1} \tag{6}$$

[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, $\cdots$, 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, $x_1 = a_{11}, x_2 = a_{12}, \cdots x_n = a_{ij} (n = i \times j)$. For the second option of combining the structures (viz. CP 2), we have $x_1 = f_1, x_2 = f_2, \cdots, x_n = f_m (n = m)$.

[**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. 12.

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-chromosome (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. 11. '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 (1^{st} 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 (2^{nd} sub-chromosome)

Sub-step 1) The 3 bits of the 2^{nd} 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.

a) The normalized integer value is given as 1 ⇒ the order of polynomial is Type 1

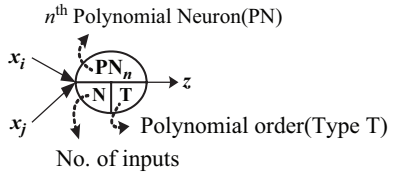


Fig. 11. Overall diagram for generation of gHFNN architecture

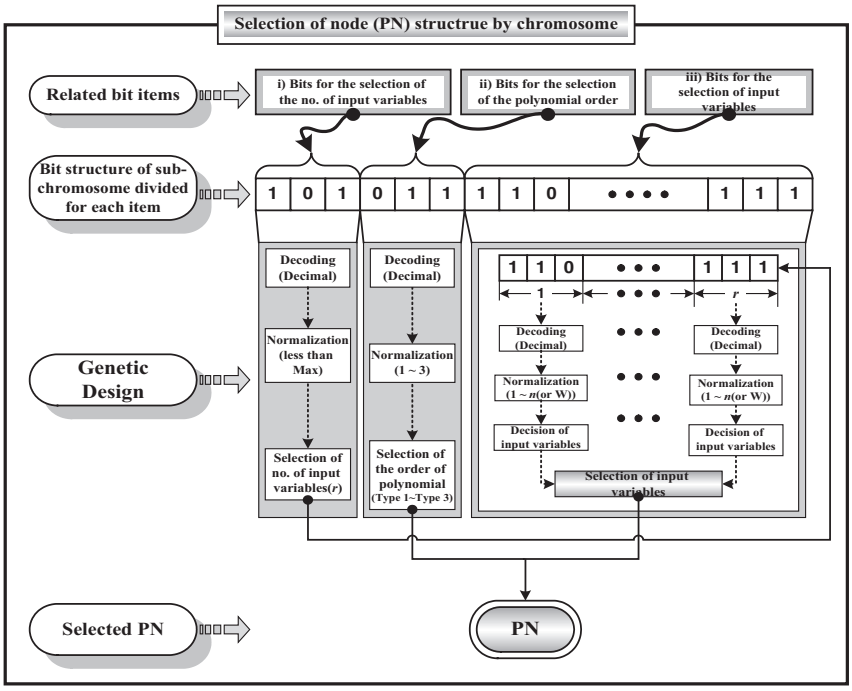


Fig. 12. Overall design flowchart of the gHFNN architecture

Table 3. Different forms of regression polynomial forming a PN

Order of the polynomial \ Number of inputs	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 $= c_0 + c_1x_1 + c_2x_2$
- Biquadratic-1 (Basic) = Bilinear $+c_3x_1^2 + c_4x_2^2 + c_5x_1x_2$,
- Biquadratic-2 (Modified) = Bilinear $+c_3x_1x_2$

b) The normalized integer value is given as 2 ⇒ the order of polynomial is Type 2
c) The normalized integer value is given as 3 ⇒ the order of polynomial is Type 3

[**Step 4-3**] Selection of input variables (3^{rd} sub-chromosome)

Sub-step 1) The remaining bits are assigned to the binary bits for the selection of input variables. binary bits for the selection of the number 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 assignedto the selected node and evaluation of a PN: The vector of coefficients is derived by minimizing the mean squared error between y_i and $\widehat{y}$ [14,15]. 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).

$$\textit{fitness function} = \frac{1}{1 + \textit{Objective function}} \tag{7}$$

[**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, $\cdots$, z_W) serves as new inputs to the next layer (x_1,x_2, $\cdots$, 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 some well-known and widely used datasets. In the first experiment, the network is used to model a three-input nonlinear function [1,13,19,25,28,29]. In the second simulation, an gHFNN is used to model a time series of gas furnace (Box-Jenkins data) [2–6,26,30–35]. Finally we use gHFNN for NOx emission process of gas turbine power plant [27,29,32]. The performance indexes (object function) used here are: (9) for the three-input nonlinear function and (8) for both gas furnace process and NOx emission process.
i) Mean Squared Error (MSE)

$$E(PI\ or\ EPI) = \frac{1}{n}\sum_{p=1}^{n}(y_p - \widehat{y}_p)^2 \tag{8}$$

ii) Mean Magnitude of Relative Error (MMRE)

$$E(PI\ or\ EPI) = \frac{1}{n}\sum_{p=1}^{n}\frac{|y_p - \widehat{y}_p|}{y_p} \times 100(\%) \tag{9}$$

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.

5.1 Nonlinear function

In this experiment, we use the same numerical data as in [1,13,19,25,28,29]. The nonlinear function to be determined is expressed as

$$y = (1 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5})^2 \tag{10}$$

We consider 40 pairs of the original input-output data. The performance index (PI) is defined by (9). 20 out of 40 pairs of input-output data are used as learning set; the remaining part serves as a testing set.

Table 4 summarizes the list of parameters related to the genetic optimization of the network. 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

Table 4. Parameters of the optimization environment and computational effort

(a) In case of using FS_FNN

Gas	Generation			100
	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 (FS_FNN)	No. of entire system inputs		3
		Learning iteration		1000
		Learning rate tuned	Simplified	0.039
			Linear	0.335
		Momentum Coefficient tuned	Simplified	0.004
			Linear	0.058
		No. of rules		6
	Consequence (gPNN)	No. of entire inputs	CP 1	6
			CP 2	3
		Maximal layer		5
		No. of inputs to be selected (N)		$1 \le N \le 4$ (Max)
		Type(T)		$1 \le T \le 3$

N, T : integer

(b) In case of using FR_FNN

Gas	Generation			150
	Population size			100
	Elite population size (W)			50
	String length	Premise structure (FNN)		10 (per one variable)
		Consequence structure (PNN)		$3+3+28$
gHFNN	Premise (FS_FNN)	No. of entire system inputs		3
		Learning iteration		1000
		Learning rate tuned	Simplified	0.309
			Linear	0.879
		Momentum Coefficient tuned	Simplified	0.056
			Linear	0.022
		No. of rules		8
	Consequence (gPNN)	No. of entire inputs		8
		Maximal layer		5
		No. of inputs to be selected (N)		$1 \le N \le 4$ (Max)
		Type (T)		$1 \le T \le 3$

N, T : integer

of the premise FNN contains the vertices of 2 membership functions of each system input (here, 3 system input variables have been used), learning rate, and momentum coefficient. The numbers of bits allocated to a chromosome are equal to 60, 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 4. 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 24/28, respectively. The population size being selected from the total population size, 60/100 is equal to 30/50. The process is realized as follows. 60/100 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 30/50 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.

Tables 5(a) and (b) summarize the results of the genetically optimized HFNN architectures when exploiting two kinds of FNN (viz. FS_FNN and FR_FNN) based on each fuzzy inference method. In light of the values of Table 5(a) reported there, we distinguish with two network architectures such as the premise FNN and the overall gHFNN. First, in case of the premise FNN, the network comes in the form of two fuzzy inference methods. Here, the FNN uses two membership functions for each input variable and has six fuzzy rules. In this case, as mentioned previously, the parameters of the FNN are optimized with the aid of GAs and BP learning. When considering the simplified fuzzy inference-based FNN, the minimal value of the performance index, that is PI = 5.217 and EPI = 5.142 are obtained. In case of the linear fuzzy inference-based FNN, the best results are reported in the form of the performance index such that PI = 2.929 and EPI = 3.45.

Next the values of the performance index of output of the gHFNN depend on each connection point based on the individual fuzzy inference methods. 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 5.

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 simplified fuzzy inference and connection point 2 of FS_FNN in Table 5(a), let us investigate the 2^{nd} layer of the network (shadowed in Table 5(a)). The fitness value in layer 2 attains its maximum for Max = 4 when nodes 8, 7, 4, 11 (such as z_8, z_7, z_4, z_{11}) occur among preferred nodes (W) chosen in the previous layer (the 1^{st} layer) are selected as the node inputs in the present layer. Furthermore 4 inputs of Type 2 (linear function) were selected as the results of the genetic optimization, refer to Fig. 13(b). In the "Input No." item of Table 5, a blank node marked by period (·) indicates that it has not been selected by the genetic operation. The performance of the conventional HFNN (called "SOFPNN" in the literature [19], which is composed of FNN and PNN with 4 inputs-Type 2 topology) was quantified

Table 5. Parameters of the optimization environment and computational effort

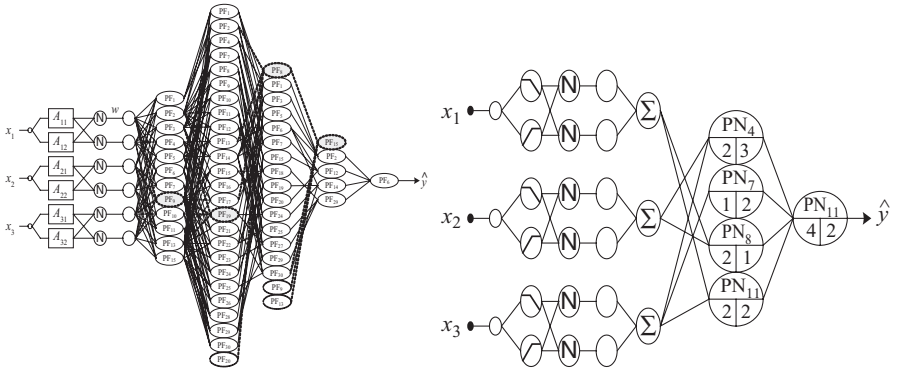
(a) In case of using FS_FNN

Premise part				CP	Consequence part							PI	EPI
Fuzzy Inference	No. of rules (MFs)	PI	EPI		Layer	No. of inputs	Input No.				T		
Simplified	6 (2 + 2 + 2)	5.21	5.14	01	1	4	6	2	5	3	2	2.070	2.536
					2	4	4	9	7	15	3	0.390	0.896
					3	3	28	7	18	·	3	0.363	0.642
					4	4	5	1	7	19	1	0.350	0.539
					5	2	5	3	·	·	2	0.337	0.452
				02	1	3	3	1	2	·	2	2.706	3.946
					2	4	8	7	4	11	2	0.299	0.517
					3	3	15	5	14	·	3	0.299	0.467
					4	3	5	3	24	·	3	0.299	0.412
					5	3	14	25	1	·	2	0.299	0.398
Linear	6 (2 + 2 + 2)	2.92	3.45	01	1	4	3	2	6	5	2	0.667	0.947
					2	4	1	7	28	15	2	0.087	0.315
					3	4	28	9	1	14	2	0.0029	0.258
					4	4	16	23	3	22	2	0.0014	0.136
					5	4	11	12	20	7	1	0.0014	0.112
				02	1	3	2	1	3	·	2	0.908	1.423
					2	4	16	1	2	7	2	0.113	0.299
					3	4	15	2	28	24	3	0.029	0.151
					4	4	5	15	20	24	3	0.010	0.068
					5	3	12	3	7	·	3	0.0092	0.056

(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		
Simplified	8 (2 × 2 × 2)	3.997		1	4	5	6	4	7	1	8.138	10.68
				2	4	2	45	49	25	2	0.382	2.316
			3.269	3	4	42	9	40	49	3	0.313	1.403
				4	3	36	16	33	·	1	0.311	0.734
				5	3	43	45	28	·	2	0.309	0.610
Linear	8 (2 × 2 × 2)	2.069		1	4	6	1	5	7	2	0.423	4.601
				2	4	49	4	44	34	3	0.184	2.175
			2.518	3	2	43	2	·	·	2	0.105	1.361
				4	3	33	44	5	·	3	0.063	0.761
				5	3	41	26	42	·	4	0.039	0.587

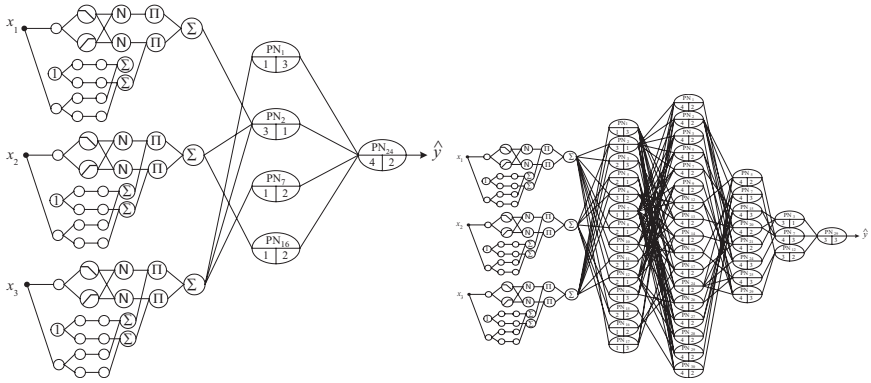
by the values of PI equal to 0.299 and EPI given as 0.555, whereas under the condition given as similar performance, the best results for the proposed network related to the output node mentioned previously were reported as PI = 0.299 and EPI = 0.517. In the sequel, the depth (the number of layers) as well as the width (the number of nodes) 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), refer to Fig. 13. In what follows, the genetic design procedure at stage (layer) of HFNN leads to the selection of the preferred nodes (or PNs)



(a) Conventional optimized HFNN (called "SOFPNN")

(b) Genetically optimized HFNN with FS_FNN

Fig. 13. Comparison of the proposed model architecture (gHFNN) and the conventional model architecture (SOFPNN [19])

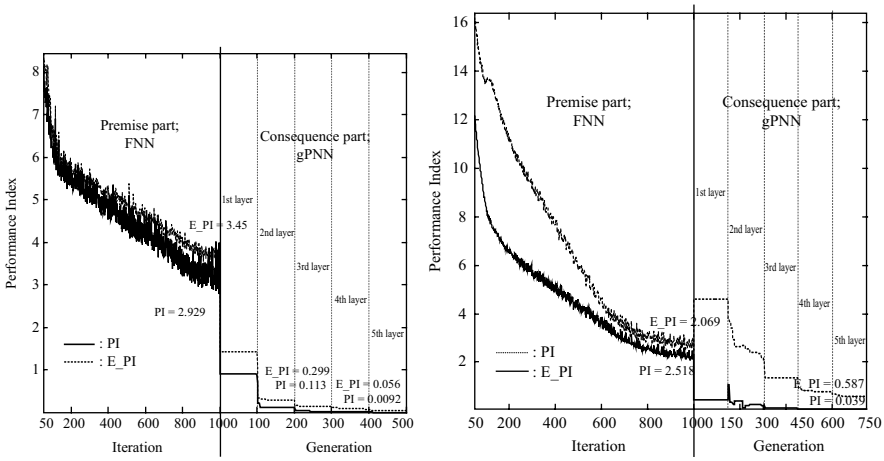


(a) 2 layers (PNN)

(b) 5 layers (PNN)

Fig. 14. Optimal topology of genetically optimized HFNN for the nonlinear function (In case of using FS_FNN)

with optimal local characteristics (such as the number of input variables, the order of the polynomial, and input variables). In addition, when considering linear fuzzy inference and CP2, the best results are reported in the form of the performance index such as PI = 0.113 and EPI = 0.299 for layer 2, and PI = 0.0092 and EPI = 0.056 for layer 5. Their optimal topologies are shown in Figs. 14 (a) and (b). Figs. 15(a) and (b) depict the optimization process by showing the values of the performance index in successive cycles of both BP learning and genetic optimization when using each linear fuzzy inference-based FNN. Noticeably, the variation ratio (slope) of the performance of the network changes radically around the 1^{st} and 2^{nd} layer. Therefore, to effectively reduce



(a) In case of FS_FNN (linear) (b) In case of FR_FNN (linear)

Fig. 15. Optimization procedure of HFNN by BP learning and GAs

a large number of nodes and avoid a substantial amount of time-consuming iterations concerning HFNN layers, the stopping criterion can be taken into consideration. Referring to Figs. 14 and 15 it becomes obvious that we can optimize the network up to maximally the 2^{nd} layer.

Table 6 covers a comparative analysis including several previous models. Sugeno's model I and II were fuzzy models based on linear fuzzy inference method while Shin-ichi's models formed fuzzy rules by using learning method of neural networks. The study of literature [29] is based on fuzzy-neural networks using HCM clustering and evolutionary fuzzy granulation. SOFPNN [19] is a network being called the "conventional HFNN" in this study. The proposed genetically optimized HFNN (gHFNN) comes with higher accuracy and improved prediction capabilities.

5.2 Gas furnace process

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 [2-6,30-35]. 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 7 summarizes the computational aspects related to the genetic optimization of gHFNN.

Table 6. Performance analysis of selected models

Model			PI	EPI	No. of rules
Linear model [25]			12.7	11.1	
GMDH [25,28]			4.7	5.7	
Sugeno's [1,25]	Fuzzy model I		1.5	2.1	3
	Fuzzy model II		1.1	3.6	4
Shin-ichi's [13]	FNN Type 1		0.84	1.22	$8(2^3)$
	FNN Type 2		0.73	1.28	$4(2^2)$
	FNN Type 3		0.63	1.25	$8(2^3)$
FNN [29]	Simplified		2.865	3.206	$9(3+3+3)$
	Linear		2.670	3.063	$9(3+3+3)$
Multi-FNN [29]	Simplified		0.865	0.956	$9(3+3+3)$
	Linear		0.174	0.689	$9(3+3+3)$
SOFPNN [19]	BFPNN		0.299	0.555	6 rules/5^{th} layer
	MFPNN		0.116	0.360	8 rules/5^{th} layer
Proposed model	FS_FNN	Simplified	5.21	5.14	$6(2+2+2)$
		Linear	2.92	3.45	$6(2+2+2)$
	FR_FNN	Simplified	3.997	3.269	$8(2\times2\times2)$
		Linear	2.069	2.518	$8(2\times2\times2)$
	gHFNN (FS_FNN)	Simplified	0.299	0.517	6 rules/2^{nd} layer
			0.299	0.398	6 rules/5^{th} layer
		Linear	0.113	0.299	6 rules/2^{nd} layer
			0.0092	0.056	6 rules/5^{th} layer
	gFPNN (FR_FNN)	Simplified	0.309	0.610	8 rules/5^{th} layer
		Linear	0.039	0.587	8 rules/5^{th} layer

The GAs-based design procedure is carried out in the same manner as in the previous experiments. Table 8 includes the results of the overall network reported according to various alternatives concerning various forms of FNN architecture, types of fuzzy inference and location of the connection point. When considering the simplified fuzzy inference-based FS_FNN with Type I (4 fuzzy rules), the minimal value of the performance index, that is PI = 0.035 and EPI = 0.281 are obtained. In case of the linear fuzzy inference-based FS_FNN with Type I (4 fuzzy rules), the best results are reported with the performance index such that PI = 0.041 and EPI = 0.267. When using Type II (6 fuzzy rules), the best results (PI = 0.0248 and EPI = 0.126) were obtained for simplified fuzzy inference and linear fuzzy inference, respectively (in the second case we have PI = 0.0256 and EPI = 0.143). In case of using FR_FNN and Type II, the best results are given as the performance index such that PI = 0.026, EPI = 0.115 and PI = 0.033, EPI = 0.119 for simplified and linear fuzzy inference respectively.

When using FS_FNN and Type I, Fig. 16 illustrates the detailed optimal topology of the gHFNN with 3 layers of PNN; the network comes with the following values: PI = 0.017 and EPI = 0.267. The proposed network enables the architecture to be a structurally optimized and gets simpler than the

Table 7. Computational aspects of the optimization of gHFNN

(a) In case of using FS_FNN

GAs	Generation			150
	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
		Learning iteration		300
		Learning rate tuned	Simplified	0.0014
			Linear	0.0052
		Momentum Coefficient tuned	Simplified	0.0002
			Linear	0.0004
		No. of rules		4/6
	Consequence (Gpnn)	No. of system 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 and Type II

GAs	Generation			150
	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		3
		Learning iteration		500
		Learning rate tuned	Simplified	0.0524
			Linear	0.0144
		Momentum Coefficient tuned	Simplified	0.00086
			Linear	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

conventional HFNN. Fig. 17(a) illustrates the optimization process by visualizing the performance index in successive cycles (iteration and generation). It also shows the optimized network architecture when taking into consideration HFNN based on linear fuzzy inference and CP1, refer to Table 8(a) and Fig. 16. As shown in Figs. 17(a) and (b), the variation ratio (slope) of the performance of the network is almost the same around the 2^{nd} through 5^{th} layer.

Table 8. Performance index of HFNN for the gas furnace

(a) In case of using FS_FNN and Type I

Premise part				CP	Consequence part							PI	EPI
Fuzzy Inference	No. of rules (MFs)	PI	EPI		Layer	No. of inputs	Input No.				T		
Simplified	4 (2 + 2)	0.035	0.281	01	1	2	3	1	·	·	3	0.025	0.328
					2	4	24	11	30	18	2	0.024	0.269
					3	3	9	10	23	·	2	0.020	0.265
					4	4	7	13	27	20	2	0.019	0.262
					5	3	1	10	7	·	2	0.018	0.254
				02	1	2	1	2	·	·	3	0.024	0.328
					2	4	1	4	5	7	3	0.021	0.282
					3	3	29	27	26	·	2	0.020	0.270
					4	3	15	13	21	·	3	0.019	0.268
					5	4	8	4	20	13	2	0.018	0.265
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

(b) In case of using FR_FNN and Type II

Premise part				Consequence part							PI	EPI
Fuzzy Inference	No. of rules (MFs)	PI	EPI	Layer	No. of inputs	Input No.				T		
Simplified	8 (2 × 2 × 2)	0.026	0.115	1	4	2	5	6	7	2	0.021	0.124
				2	3	26	15	16	·	2	0.019	0.115
				3	3	21	3	27	·	2	0.018	0.114
				4	3	24	3	18	·	2	0.018	0.111
				5	4	13	12	29	20	1	0.018	0.109
Linear	8 (2 × 2 × 2)	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

Table 9 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.

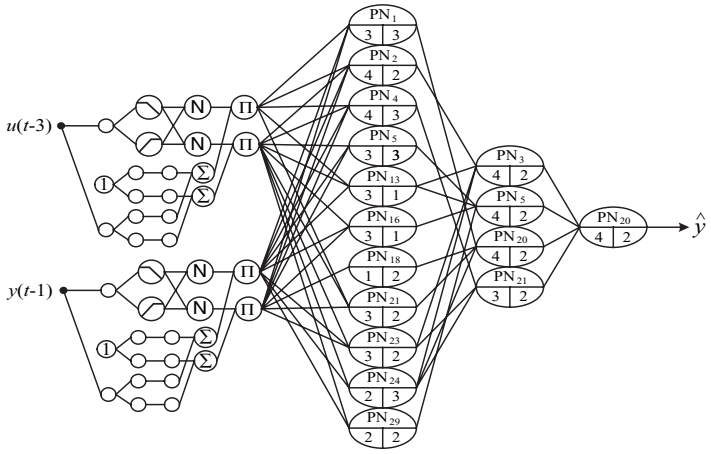
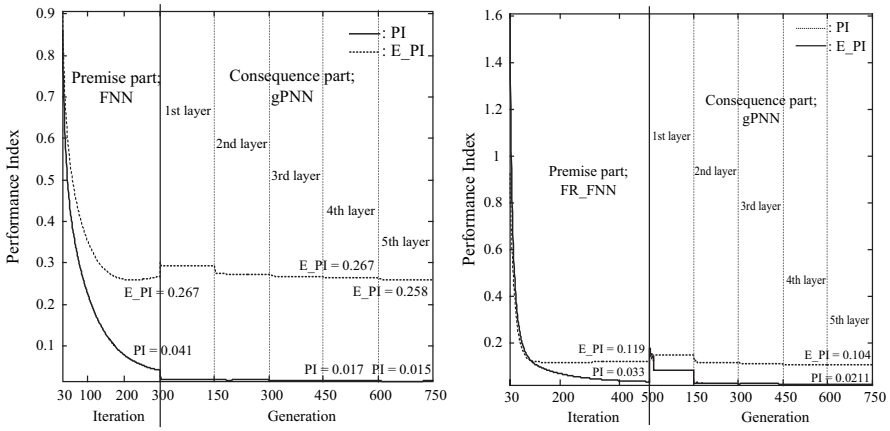


Fig. 16. Genetically optimized HFNN (gHFNN) with FS_FNN (linear)



(a) In case of using FS_FNN (linear) with Type I

(b) In case of using FR_FNN (linear) and Type II

Fig. 17. Optimization procedure of HFNN by BP learning and GAs

5.3 NOx emission process of gas turbine power plant

NOx emission process is modeled using the data of gas turbine power plant coming from a GE gas turbine power plant located in Virginia, US.

The input variables include AT (Ambient Temperature at site), CS (Compressor Speed), LPTS (Low Pressure Turbine Speed), CDP (Compressor Discharge Pressure), and TET (Turbine Exhaust Temperature). The output variable is NOx [27,29,32]. The performance index is defined by (8). We consider 260 pairs of the original input-output data. 130 out of 260 pairs of input-output data are used as learning set; the remaining part serves as a

Table 9. Performance analysis of selected models

Model			PI	EPI	No. of rules
Kim, et al.'s model [30]			0.034	0.244	2
Lin and Cunningham's mode [31]			0.071	0.261	4
Fuzzy	Min-Max [5]	Simplified	0.022	0.335	$4(2 \times 2)$
			0.022	0.336	$6(3 \times 2)$
		Linear	0.024	0.358	$4(2 \times 2)$
			0.020	0.362	$6(3 \times 2)$
	GAs [5]	Simplified	0.023	0.344	$4(2 \times 2)$
		Linear	0.018	0.264	$4(2 \times 2)$
	Complex [2]	Simplified	0.024	0.328	$4(2 \times 2)$
		Linear	0.023	0.306	$4(2 \times 2)$
	Hybrid [4] (GAs+Complex)	Simplified	0.024	0.329	$4(2 \times 2)$
		Linear	0.017	0.289	$4(2 \times 2)$
	HCM [3]	Simplified	0.755	1.439	$6(3 \times 2)$
		Linear	0.018	0.286	$6(3 \times 2)$
	HCM+GAs [3]	Simplified	0.035	0.289	$4(2 \times 2)$
			0.022	0.333	$6(3 \times 2)$
		Linear	0.026	0.272	$4(2 \times 2)$
			0.020	0.2642	$6(3 \times 2)$
Neural Networks [3]			0.034	4.997	
Oh's Adaptive FNN [6]			0.021	0.332	$9(3 \times 3)$
			0.022	0.353	$4(2 \times 2)$
FNN [32]	Simplified		0.043	0.264	$6(3+3)$
	Linear		0.037	0.273	$6(3+3)$
Multi-FNN [33]	Simplified		0.025	0.274	$6(3+3)$
	Linear		0.024	0.283	$6(3+3)$
SOFPNN	Generic [34]		0.017	0.250	4 rules/5^{th} layer
	Advanced [35]		0.019	0.264	6 rules/5^{th} layer
Proposed model	FS_FNN	Simplified	0.035	0.281	$4(2+2)$
			0.024	0.126	$6(2+2+2)$
		Linear	0.041	0.267	$4(2+2)$
			0.025	0.143	$6(2+2+2)$
	FR_FNN	Simplified	0.024	0.329	$4(2 \times 2)$
			0.026	0.115	$8(2 \times 2 \times 2)$
		Linear	0.025	0.265	$4(2 \times 2)$
			0.033	0.119	$8(2 \times 2 \times 2)$
	gHFNN (FS_FNN)	Simplified	0.018	0.254	4 rules/5^{th} layer
			0.018	0.112	6 rules/5^{th} layer
		Linear	0.015	0.258	4 rules/5^{th} layer
			0.018	0.110	6 rules/5^{th} layer
	gHFNN (FR_FNN)	Simplified	0.017	0.250	4 rules/5^{th} layer
			0.018	0.109	8 rules/5^{th} layer
		Linear	0.016	0.249	4 rules/5^{th} layer
			0.021	0.104	8 rules/5^{th} layer

testing set. Using NOx emission process data, the regression model is

$$y = -163.77341 - 0.06709\,x_1 + 0.00322\,x_2 + 0.00235\,x_3 + 0.26365\,x_4 + 0.20893\,x_5 \quad (11)$$

And it comes with PI = 17.68 and EPI = 19.23. We will be using these results as a reference point when discussing gHFNN models. Table 10 summarizes

Table 10. Summary of the parameters of the optimization environment

(a) In case of using FS_FNN

GAs	Generation			150
	Population size			100
	Elite population size			50
	String length	Premise structure		10 (per one variable)
		Consequence structure	CP 1	$3+3+70$
			CP 2	$3+3+35$
gHFNN	Premise (FNN)	No. of entire system inputs		5
		Learning iteration		1000
		Learning rate tuned	Simplified	0.052
			Linear	0.034
		Momentum Coefficient tuned	Simplified	0.010
			Linear	0.001
		No. of rules		$10(2+2+2+2+2)$
	Consequence (gPNN)	No. of system inputs	CP 1	10
			CP 2	5
		Maximal layer		5
		No. of inputs to be selected (N)	CP 1	$1 \le N \le 10$ (Max)
			CP 1	$1 \le N \le 5$ (Max)
		Type (T)		$1 \le T \le 3$

N, T : integer

(b) In case of using FR_FNN and Type II

GAs	Generation			150
	Population size			100
	Elite population size (W)			50
	String length	Premise structure (FNN)		10 (per one variable)
		Consequence structure (PNN)		$3+3+105$
gHFNN	Premise	No. of entire system inputs		5
		Learning iteration		1000
		Learning rate tuned	Simplified	0.568
			Linear	0.651
		Momentum Coefficient tuned	Simplified	0.044
			Linear	0.064
		No. of rules		$32(2\times2\times2\times2\times2)$
	Consequence (Gpnn)	No. of entire inputs		32
		Maximal layer		5
		No. of inputs to be selected (N)		$1 \le N \le 15$ (Max)
		Type (T)		$1 \le T \le 3$

N, T : integer

the parameters of the optimization environment. The parameters used for optimization of this process modeling are almost the same as used in the previous experiments.

Table 11 summarizes the detailed results. When using FS_FNN, the best results for the network are obtained when using linear fuzzy inference and CP

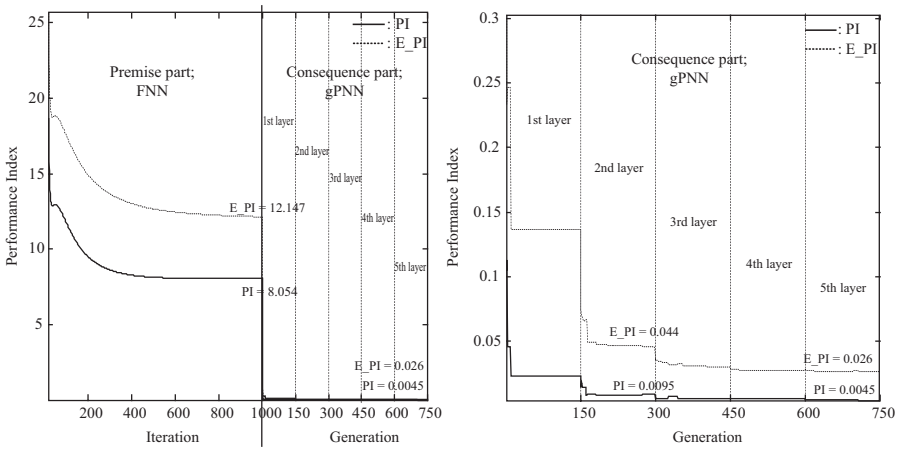
Table 11. Parameters of the optimization environment and computational effort

(a) In case of using FS_FNN

Premise part				CP	Consequence part			PI	EPI
Fuzzy Inference	No. of rules (MFs)	PI	EPI		Layer	No. of inputs	Type		
Simplified	10 $(2+2+2+2+2)$	22.331	19.783	01	1	7	2	0.916	2.014
					2	6	2	0.623	1.430
					3	9	1	0.477	1.212
					4	10	1	0.386	1.077
					5	4	2	0.337	1.016
				02	1	5	2	1.072	2.220
					2	5	2	0.176	0.291
					3	5	3	0.105	0.168
					4	3	2	0.060	0.113
					5	4	2	0.049	0.081
Linear	10 $(2+2+2+2+2)$	8.054	12.147	01	1	9	2	0.023	0.137
					2	5	2	0.0095	0.044
					3	9	1	0.0057	0.029
					4	2	2	0.0057	0.027
					5	9	1	0.0045	0.026
				02	1	5	2	2.117	4.426
					2	5	2	0.875	1.647
					3	5	2	0.550	1.144
					4	5	3	0.390	0.793
					5	3	2	0.340	0.680

(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	Type		
Simplified	32 $(2\times2\times2\times2\times2)$	0.711	1.699	1	13	2	0.149	0.921
				2	12	1	0.065	0.189
				3	11	1	0.046	0.134
				4	8	1	0.044	0.125
				5	3	3	0.041	0.111
Linear	32 $(2\times2\times2\times2\times2)$	0.079	0.204	1	10	2	0.205	1.522
				2	13	1	0.049	0.646
				3	3	3	0.028	0.437
				4	12	1	0.023	0.330
				5	3	3	0.019	0.286



(a) Premise and consequence part (b) Consequence part (extended)

Fig. 18. Optimal procedure of gHFNN with FS_FNN (linear) by BP and GAs

1 with Type 1 (linear function) and 9 nodes at input; this network comes with the value of PI equal to 0.0045 and EPI set as 0.026. In case of using FR_FNN and simplified fuzzy inference, the most preferred network architecture have been reported as PI = 0.041 and EPI = 0.111. As shown in Table 11 and Fig. 18, the variation ratio (slope) of the performance of the network changes radically at the 2^{nd} layer. Therefore, to effectively reduce a large number of nodes and avoid a large amount of time-consuming iteration of gHFNN, the stopping criterion can be taken into consideration up to maximally the 2^{nd} layer.

Table 12 covers a comparative analysis including several previous fuzzy-neural network models. The experimental results clearly reveal that the proposed approach and the resulting model outperform the existing networks both in terms of better approximation 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. In contrast to the conventional HFNN structures and their learning, the proposed model comes with two kinds of rule-based FNNs (viz. FS_FNN and FR_FNN based on two types of 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.

Table 12. Performance analysis of selected models

Model			PI	EPI	No. of rules
Regression model [32]			17.68	19.23	
Ahn et al. [27]	FNN		5.835		
	AIM		8.420		
FNN [32]	Simplified		6.269	8.778	$30(6+6+6+6+6)$
	Linear		3.725	5.291	$30(6+6+6+6+6)$
Multi-FNN [33]	Simplified		2.806	5.164	$30(6+6+6+6+6)$
	Linear		0.720	2.025	$30(6+6+6+6+6)$
Proposed model	FS_FNN	Simplified	22.331	19.783	$10(2+2+2+2+2)$
		Linear	8.054	12.147	$10(2+2+2+2+2)$
	FR_FNN	Simplified	0.711	1.699	$32(2\times2\times2\times2\times2)$
		Linear	0.079	0.204	$32(2\times2\times2\times2\times2)$
	gHFNN (FS_FNN)	Simplified	0.176	0.291	10 rules/2^{nd} layer
			0.049	0.081	10 rules/5^{th} layer
		Linear	0.0095	0.044	10 rules/2^{nd} layer
			0.0045	0.026	10 rules/5^{th} layer
	gHFNN (FR_FNN)	Simplified	0.065	0.189	32 rules/2^{nd} layer
			0.041	0.111	32 rules/5^{th} layer
		Linear	0.049	0.646	32 rules/2^{nd} layer
			0.019	0.286	32 rules/5^{th} layer

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.

7 Acknowledgement

This work has been supported by KESRI(I-2004-0-074-0-00), which is funded by MOCIE (Ministry of commerce, industry and energy).

References

1. Kang G, Sugeno M (1987) Fuzzy modeling. Trans Soc Instrum Control Eng 23(6cr):106–108
2. Oh SK, Pedrycz W (2000) Fuzzy identification by means of auto-tuning algorithm and its application to nonlinear systems. Fuzzy Sets Syst 115(2):205–230
3. Park BJ, Pedrycz W, Oh SK (2001) Identification of fuzzy models with the aid of evolutionary data granulation. IEE Proc-Control Theory Appl 148(5):406–418
4. Oh SK, Pedrycz W, Park BJ (2002) Hybrid identification of fuzzy rule-based models. Int J Intell Syst 17(1):77–103
5. Park BJ, Oh SK, Ahn TC, Kim HK (1999) Optimization of fuzzy systems by means of GA and weighting factor. Trans Korean Inst Electr Eng 48A(6):789–799 (In Korean)
6. Oh SK, Park CS, Park BJ (1999) On-line modeling of nonlinear process systems using the adaptive fuzzy-neural networks. Trans Korean Inst Electr Eng 48A(10):1293–1302 (In Korean)
7. Narendra KS, Parthasarathy K (1991) Gradient methods for the optimization of dynamical systems containing neural networks. IEEE Trans Neural Netw 2:252–262
8. Goldberg DE (1989) Genetic algorithms in search, optimization & machine learning. Addison-wesley, Reading
9. Michalewicz Z (1996) Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin Heidelberg Newyork
10. Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbour
11. Pedrycz W, Peters JF (1998) Computational intelligence and software engineering. World Scientific, Singapore
12. Computational intelligence by programming focused on fuzzy neural networks and genetic algorithms. Naeha, Seoul (In Korean)
13. Horikawa S, Furuhashi T, Uchigawa Y (1992) On fuzzy modeling using fuzzy neural networks with the back propagation algorithm. IEEE Trans Neural Netw 3(5):801–806
14. Oh SK, Pedrycz W (2002) The design of self-organizing polynomial neural networks. Inf Sci 141(3–4):237–258
15. Oh SK, Pedrycz W, Park BJ (2003) Polynomial neural networks architecture: Analysis and Design. Comput Electr Eng 29(6):653–725
16. Ohtani T, Ichihashi H, Miyoshi T, Nagasaka K (1998) Orthogonal and successive projection methods for the learning of neurofuzzy GMDH. Inf Sci 110:5–24
17. Ohtani T, Ichihashi H, Miyoshi T, Nagasaka K (1998) Structural learning with M-Apoptosis in neurofuzzy GMHD. In: Proceedings of the 7th IEEE International Conference on Fuzzy Systems:1265–1270

18. Ichihashi H, Nagasaka K (1994) Differential minimum bias criterion for neuro-fuzzy GMDH. In: Proceedings of 3rd International Conference on Fuzzy Logic Neural Nets and Soft Computing IIZUKA'94:171–172
19. Park BJ, Pedrycz W, Oh SK (2002) Fuzzy polynomial neural networks: hybrid architectures of fuzzy modeling. IEEE Trans Fuzzy Syst 10(5):607–621
20. Oh SK, Pedrycz W, Park BJ (2003) Self-organizing neurofuzzy networks based on evolutionary fuzzy granulation. IEEE Trans Syst Man and Cybern A 33(2):271–277
21. Cordon O et al. (2004) Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets Syst 141(1):5–31
22. Ivahnenko AG (1968) The group method of data handling: a rival of method of stochastic approximation. Sov Autom Control 13(3):43–55
23. Yamakawa T (1993) A new effective learning algorithm for a neo fuzzy neuron model. 5th IFSA World Conference:1017–1020
24. Oh SK, Yoon KC, Kim HK (2000) The Design of optimal fuzzy- eural networks structure by means of GA and an aggregate weighted performance index. J Control, Autom Syst Eng 6(3):273–283 (In Korean)
25. Park MY, Choi HS (1990) Fuzzy control system. Daeyoungsa, Seoul (In Korean)
26. Box G.EP, Jenkins GM (1976) Time series analysis, forecasting, and control, 2nd edn. Holden-Day, SanFransisco
27. Ahn TC, Oh SK (1997) Intelligent models concerning the pattern of an air pollutant emission in a thermal power plant, Final Report, EESRI
28. Kondo T (1986) Revised GMDH algorithm estimating degree of the complete polynomial. Trans Soc Instrum Control Eng 22(9):928–934
29. Park HS, Oh SK (2003) Multi-FNN identification based on HCM clustering and evolutionary fuzzy granulation. Int J Control, Autom Syst 1(2):194–202
30. Kim E, Lee H, Park M, Park M (1998) A simply identified sugeno-type fuzzy model via double clustering. Inf Sci 110:25–39
31. Lin Y, Cunningham III GA (1997) A new approach to fuzzy-neural modeling, IEEE Trans Fuzzy Syst 3(2):190–197
32. Oh SK, Pedrycz W, Park HS (2003) Hybrid identification in fuzzy-neural networks. Fuzzy Sets Syst 138(2):399–426
33. Park HS, Oh SK (2000) Multi-FNN identification by means of HCM clustering and its optimization using genetic algorithms. J Fuzzy Logic Intell Syst 10(5):487–496 (In Korean)
34. Park BJ, Oh SK, Jang SW (2002) The design of adaptive fuzzy polynomial neural networks architectures based on fuzzy neural networks and self-organizing networks. J Control Autom Syst Eng 8(2):126–135 (In Korean)
35. Park BJ, Oh SK (2002) The analysis and design of advanced neurofuzzy polynomial networks. J Inst Electron Eng Korea 39-CI(3):18–31 (In Korean)
36. Park BJ, Oh SK, Pedrycz W, Kim HK (2005) Design of evolutionally optimized rule-based fuzzy neural networks on fuzzy relation and evolutionary optimization. International Conference on Computational Science. Lecture Notes in Computer Science 3516:1100–1103
37. Oh SK, Park BJ, Pedrycz W, Kim HK (2005) Evolutionally optimized fuzzy neural networks based on evolutionary fuzzy granulation. Lecture Notes in Computer Science 3483:887–895
38. Oh SK, Park BJ, Pedrycz W, Kim HK (2005) Genetically optimized hybrid fuzzy neural networks in modeling software data. Lecture Notes in Artificial Intelligence 3558:338–345

39. Zadeh NN, Darvizeh A, Jamali A, Moeini A (2005) Evolutionary design of generalized polynomial neural networks for modeling and prediction of explosive forming process. J Mater Process Technol 164(15):1561–1571
40. Delivopoulos E, Theocharis JB (2004) A modified PNN algorithm with optimal PD modeling using the orthogonal least squares method. Inf Sci 168(3):133–170