Self-Organizing Polynomial Neural Networks Based on Genetically Optimized Multi-Layer Perceptron Architecture

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

Abstract: In this paper, we introduce a new topology of Self-Organizing Polynomial Neural Networks (SOPNN) based on genetically optimized Multi-Layer Perceptron (MLP) and discuss its comprehensive design methodology involving mechanisms of genetic optimization. Let us recall that the design of the "conventional" SOPNN uses the extended Group Method of Data Handling (GMDH) technique to exploit polynomials as well as to consider a fixed number of input nodes at polynomial neurons (or nodes) located in each layer. However, this design process does not guarantee that the conventional SOPNN generated through learning results in optimal network architecture. The design procedure applied in the construction of each layer of the SOPNN deals with its structural optimization involving the selection of preferred nodes (or PNs) with specific local characteristics (such as the number of input variables, the order of the polynomials, and input variables) and addresses specific aspects of parametric optimization. An aggregate performance index with a weighting factor is proposed in order to achieve a sound balance between the approximation and generalization (predictive) abilities of the model. To evaluate the performance of the GA-based SOPNN, the model is experimented using pH neutralization process data as well as sewage treatment process data. A comparative analysis indicates that the proposed SOPNN is the model having higher accuracy as well as more superb predictive capability than other intelligent models presented previously.

Keywords: Aggregate objective function, design procedure, GA-based SOPNN, Genetic Algorithms (GAs), Group Method of Data Handling (GMDH), Polynomial Neuron (PN), Self-Organizing Polynomial Neural Networks (SOPNN).

1. INTRODUCTION

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

When the complexity of the system to be modeled increases, experimental data as well as some degree of prior domain knowledge (conveyed by the model developer) are essential to the completion of an efficient design procedure.

It is also worth stressing that the nonlinear form of the model acts as a two-edged sword: while we gain flexibility to cope with experimental data, we are provided with an abundance of nonlinear dependencies that need to be exploited in a systematic manner. In particular, when dealing with high-order nonlinear and multivariable equations of the model, we require a vast amount of data necessary for estimating its complete range of parameters [1,2].

To help alleviate such problems, one of the first approaches along systematic design of nonlinear relationships between system's inputs and outputs comes into play, known as the Group Method of Data Handling (GMDH). GMDH was developed in the late 1960s by Ivakhnenko [3-6] as a vehicle for identifying nonlinear relations between input and output variables. While providing a useful systematic design procedure, GMDH also has some drawbacks. First, it tends to generate quite complex polynomials for relatively simple systems (data). Second, owing to its limited generic structure (quadratic two-variable polynomial), GMDH also tends to produce an overly complex

Manuscript received September 23, 2003; revised March 23, 2004; accepted April 26, 2004. Recommended by Editorial Board member Min-Jea Tahk under the direction of Editor Jin Bae Park. This work has been supported by KESRI (R-2003-B-274) which is funded by MOCIE (Ministry of Commerce, Industry and Energy).

Ho-Sung Park, Byoung-Jun Park, and Sung-Kwun Oh are with the Dept. of Electrical Electronic & Information Engineering, Wonkwang University, 344-2 Shinyong-dong, Iksan, Chonbuk 570-749, Korea (e-mails: {neuron, lcap, ohsk}@wonkwang.ac.kr).

Hyun-Ki Kim is with the Dept. of Electrical Engineering, Suwon University, San 2-2 Wau-ri, Bongdam-eup, Hwaseong-si, Gyeonggi-do 445-743, Korea (e-mail: hkkim@suwon.ac.kr).

* Corresponding author.

network (model) when it comes to highly nonlinear systems. Third, if there are less than three input variables, the GMDH algorithm does not generate a highly versatile structure.

To alleviate the problems associated with the GMDH, Self-Organizing Polynomial Neural Networks (SOPNN) were introduced by Oh et al. [7-9] as a new class of networks. In a nutshell, these networks come with a high level of flexibility as each node (processing element forming a PD (or PN)) can have a different number of input variables as well as exploit a different order of the polynomials (linear, quadratic, cubic, etc.). Although the SOPNN contains flexible model architecture with higher accuracy due to its systematic design procedure, it is difficult to obtain a structurally and parametrically optimized network because of the limited design of polynomial neurons (PNs) located in each layer of SOPNN. Accordingly, the SOPNN algorithm tends to produce overly complex networks as well as a repetitive computational load by the trial and error method and/or a repetitive parameter adjustment by the designer, as in the case of the original GMDH algorithm.

In this study, in solving the problems with the conventional SOPNN as well as the GMDH algorithm, we introduce a new design approach of GA-based SOPNN. Optimal design parameters available within the PN (viz. the number of input variables, the order of the polynomials, and input variables) lead to a structurally and parametrically optimized network, which is more flexible as well as simpler in architecture than the conventional SOPNN. Furthermore, we introduce an aggregate objective function that deals with training data and testing data, and elaborate on its optimization to produce a meaningful balance between approximation and generalization abilities of the model. In a nutshell, the objective of this study is to develop a general design methodology of GA-based SOPNN modeling, come up with a logic-based structure for such a model and propose a comprehensive evolutionary development environment in which the optimization of the models can be efficiently carried out both at the structural as well as at the parametric level [10].

To evaluate the performance of the proposed model, we exploit pH neutralization process data [19-26] as well as sewage treatment process data [15-18].

2. THE SOPNN ALGORITHM AND ITS GENERIC ARCHITECTURE

2.1. SOPNN algorithm

The SOPNN algorithm [7-9] is based on the GMDH method and utilizes a class of polynomials such as linear, quadratic, and modified quadratic (refer to Table 1). By choosing the most significant input variables and a certain order of the polynomials among the available variety of structures at our disposal, we can construct the best partial description (PD) as polynomial neuron (PN). The individual PNs are expressed as a second-order regression equation. In particular, when combining two inputs at each node as the generic structure we arrive at the following relationship;

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

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

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

$$y = A + \sum_{i=1}^{n} B_i X_i + \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij} X_i X_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk} X_i X_j X_k \cdots, \quad (2)$$

where X_i, X_j and X_k are the nodal input variables, and y is the output of an individual neuron (node). A, B_i, C_{ij}, and D_{ijk} are the coefficients of the Ivakhnenko polynomial.

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

2.2. SOPNN architecture

The SOPNN based on the perceptron learning principle with neural network-type architecture is used to model the input-output relationship of a complex process system. The design of the SOPNN structure continues and involves the generation of some additional layers. Each layer consists of nodes (PDs or PNs) for which the number of input variables could be the same as in the previous layers or may differ across the network. At each layer, new generations of complex equations are constructed from simple forms. The model obtained at each layer is progressively more complex than the model at the preceding layers. To avoid an overfit, the overall data set is divided into

Table 1. Different forms of regression polynomials 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

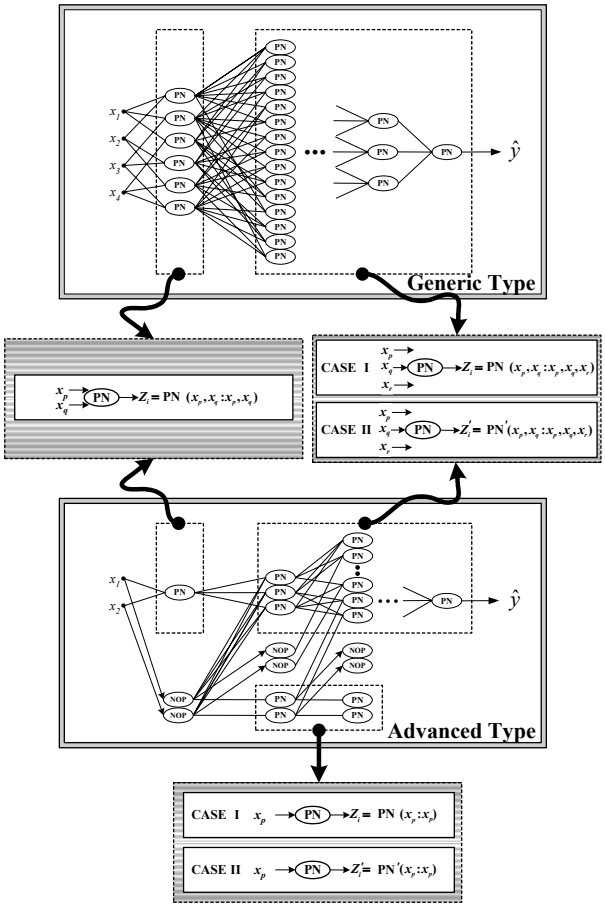


Fig. 1. An overall architecture of the conventional modified SOPNN.

a) the training set, which is used for the generation of several computing alternative models and b) the testing set, which is used to test the accuracy of each model generated and for the selection of the best models at each layer. The number of layers is increased until the newer models begin to exhibit weaker predictability than their predecessors. This indicates overfitting of the model. The final model is defined as a function of two, three, or four variables. The network result is a very sophisticated model obtained from a very limited data set.

We introduce two types of generic SOPNN structures, namely the basic and the modified SOPNN. Moreover, for each type of topology we identify two schemes [7-9]. The modified SOPNN architectures are shown in Fig. 1. In what follows, the SOPNN emerges

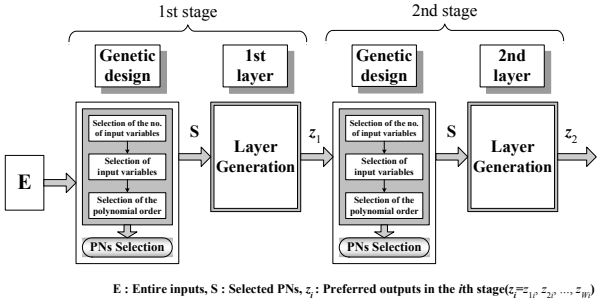


Fig. 2. Overall structural optimization process of SOPNN using Gas.

as a versatile architecture whose topology depends on the regression polynomial of a PN.

3. GENETIC OPTIMIZATION OF SOPNN

When we construct the PNs of each layer in the conventional SOPNN, such parameters as the number of input variables (nodes), the order of polynomials, and input variables available within a PN are fixed (selected) in advance by the designer. That is, the designer must have pre-determined information related to networks such as the number of system input variables and the polynomial order. Because the conventional SOPNN is a heuristic method, it does not guarantee that the constructed SOPNN is an optimal network architecture. Accordingly, in order to generate a structurally and parametrically optimized SOPNN network, such parameters need to be optimal.

In order to solve this problem, we use genetic algorithms that are a stochastic global search technique based on the principles of evolution, natural selection and genetic recombination by simulating "survival of the fittest" in a population of potential solutions (individuals) to the problem at hand [11-14].

In this study, for the optimization of the SOPNN model, GAs use the serial method of binary type, roulette-wheel in the selection operator, one-point crossover in the crossover operator, and invert in the mutation operator. As the roulette-wheel operator's stochastic characteristic, when creating new population of new generation by selection operator, we will choose the best chromosome from the last generation. To reduce the stochastic errors of roulette-wheel selection, we use elitist strategy [13]. The overall structural optimization process of SOPNN using GAs is shown in Fig. 2.

4. GA-BASED SOPNN ALGORITHM

The framework of the design procedure of the Self-Organizing Polynomial Neural Networks (SOPNN) consists of the following steps.

Step 1: *Determine system's input variables*

Define the system's input variables as $\boldsymbol{x}_i$(i=1, 2, …, n) related to output variable y. If required, normalization of input data can be completed as well.

Step 2: *Form training and testing data*
The input-output data set ($\boldsymbol{x}_i$, y_i)=(x_{1i}, x_{2i}, …, x_{ni}, y_i), i=1, 2, …, N (N: the total number of data) is divided into two parts, that is, a training and a testing dataset. Denote their sizes by N_t and N_c respectively. Obviously we have N=N_t+N_c. The training data set is used to construct the SOPNN model. Next, the testing data set is used to evaluate the quality of the model.

Step 3: *Determine initial information for constructing the SOPNN structure*
We determine initial information for the SOPNN structure in the following manner:
a) According to the stopping criterion, two termination methods are exploited here:
- Comparison of a minimal identification error of the current layer with that of the previous layer of the networks.
- The maximum number of generations predetermined by the designer to achieve a balance between model accuracy and its complexity.
b) The maximum number of input variables arriving at each node in the corresponding layer.
c) The total number W of nodes to be retained (selected) at the next generation of the SOPNN algorithm.
d) The value of the weighting factor of the aggregate objective function.

Step 4: *Determine PN structure using genetic design*
This concerns with the selection of the number of input variables, the polynomial order, and the input variables to be assigned in each node of the corresponding layer. We determine PN structure using genetic design.

The genetic design available in a PN structure by using a chromosome of GAs is illustrated in Fig. 3. As shown in Fig. 3, the design of optimal parameters available within the PN (viz. the number of input variables, the order of the polynomials, and input variables) at last leads to a structurally and parametrically optimized network, which is more flexible as well as simpler in architecture than the conventional SOPNN.

Each sub-step of the genetic design procedure of three kinds of parameters available within the PN is as follows:
[Step 4-1] ***Selection of the number of input variables (1st sub-chromosome)***
Sub-step 1) The first 3 bits in a chromosome given are assigned to the binary bits for the selection of the number of input variables.
Sub-step 2) The 3 bits randomly selected by using

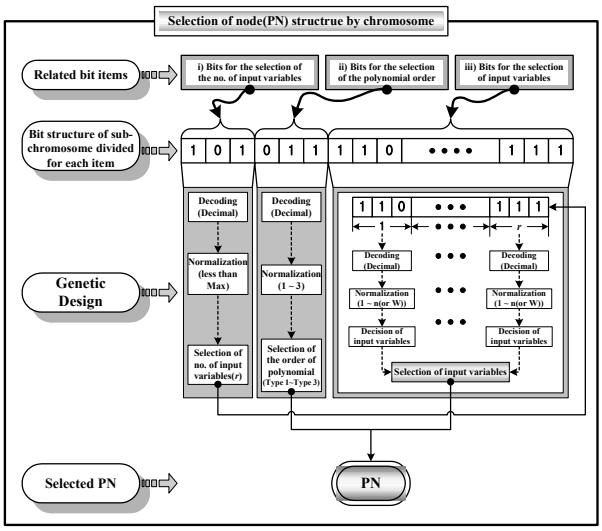


Fig. 3. The PN design available in SOPNN architecture by using a GA chromosome.

(3) are decoded in decimal.

$$\beta = (2^2 \times bit(3)) + (2^1 \times bit(2)) + (2^0 \times bit(1)), \qquad (3)$$

where, bit(1), bit(2) and bit(3) show the location of the 3 bits and are given as "0", or "1" respectively.
Sub-step 3) The decimal value β obtained by using (3) is normalized. We also round off the value obtained from (4).

$$\gamma = (\beta / \alpha) \times (\mathrm{Max} - 1) + 1, \qquad (4)$$

where Max is the maximum number of input variables arriving at the corresponding node (PN) and α is the decoded decimal value when all bits of the 1st sub-chromosome are 1's.
Sub-step 4) The normalized integer value is given as the number of input variables (or input nodes) arriving at the corresponding node.
[Step 4-2] ***Selection of the order of polynomials (2nd sub-chromosome)***
Sub-step 1) The 3 bits of the 2nd sub-chromosome following the 1st sub-chromosome are assigned to the binary bits for the selection of the order of polynomials.
Sub-step 2) The 3 bits randomly selected using (3) are decoded in decimal.
Sub-step 3) The decimal value β obtained using (4) is normalized. We also round off the value obtained from (4). We replace Max with 3 in (4). Therefore, the normalized integer value exists between 1 and 3 as in the following (refer to Table 1).
Sub-step 4) The normalized integer value is given as the selected polynomial order, when constructing each node of the corresponding layer.
[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 uniformly divided by the value obtained in step 4-1. If the remaining bits aren't uniformly divided, they are divided by the following rules.
***Sub-step 3*)** Each bit structure is decoded in decimal by using (3).
***Sub-step 4*)** Each decimal value obtained in sub-step 3 by using (4) is normalized. We also round off the values obtained from (4). We replace Max with the total number of inputs (viz. input variables or input nodes), n (or W) in the corresponding layer. Here, the total number of input variables means the number of entire system inputs, n, in the 1st layer, and the number of the selected nodes, W, as the output nodes of the preceding layer in the 2nd layer or higher.
***Sub-step 5*)** The normalized integer values are given as the selected input variables when constructing each node of the corresponding layer. Here, if the selected input variables are multiple-duplicated, the multiple-duplicated input variables (viz. same input numbers) are coped with as one input variable and the remainders (except for only one) are discarded (refer to Fig. 3).

Step 5: *Estimate the coefficients of the polynomial corresponding to the selected node (PN)*
The vector of coefficients C_i is derived by minimizing the mean squared error between y_i and z_{mi}.

$$E = \frac{1}{N_{tr}} \sum_{i=0}^{N_{tr}} (y_i - \hat{y}_i)^2. \tag{5}$$

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

$$Y = X_i C_i. \tag{6}$$

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

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

Step 6: *Select nodes (PNs) with the best predictive capability, and construct their corresponding layer*
As shown in Fig. 3, all nodes of the corresponding layer of SOPNN architecture are constructed by genetic optimization.
The generation process of PNs in the corresponding layer is described in detail as the design procedure of 9 sub-steps. A sequence of the sub-steps is as follows:
***Sub-step 1*)** We determine initial genetic information for generation of the SOPNN architecture. That is, the number of generations and populations, mutation rate, crossover rate, and the length of a chromosome.
***Sub-step 2*)** The nodes (PNs) are generated by genetic design as many as the number of populations in the 1st generation. Where, one population takes the same role as one node (PN) in the SOPNN architecture and each population is operated by GAs as shown in Fig. 3. That is, the number of input variables, the order of the polynomials, and the input variables as one individual (population) are selected by GAs. The polynomial parameters are produced by the standard least squares method.
***Sub-step 3*)** To evaluate the performance of PNs (nodes) in each population, we use an aggregate objective function that takes into account a sound balance between approximation and prediction capabilities of the one as shown in (7). And then, from the performance index obtained in (7), we calculate the fitness function of (8). The objective function (or cost function) is employed to decrease the error rate and to increase the predictability (generalization) capability of the model - that is, the objective function includes the performance index for training (PI), and the performance index for evaluation (EPI) that are combined by means of a weighting factor θ. The objective function (performance index) is a basic instrument guiding the evolutionary search in the solution space [15]. The objective function includes both the training data and the testing data and comes as a convex sum of two components.

$$f(PI, EPI) = \theta \times PI + (1- \theta) \times EPI. \tag{7}$$

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

$$F(fitness\ \ function) = \frac{1}{1 + f(PI,\ EPI)}. \tag{8}$$

PI and EPI denote the performance index for the training data and testing data (or validation data), respectively. Moreover θ is a weighting factor that allows us to strike a balance between the performance of the model for the training and testing data. The aggregate object function depends upon the values of a weighting factor. Both PI and EPI are considered and the proper selection of θ establishes the direction of optimization to maintain a balance between the approximation and generalization. Model selection is performed from the minimization of this aggregate objective function.
Sub-step 4*)** To produce the next generation, we carry out selection, crossover, and mutation operations using genetic initial information and the fitness values obtained from ***sub-step 3.

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

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

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

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

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

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

Sub-step 9*)** Until the last generation, this sub-step carries out by repeating ***sub-steps 7-8.

The iterative process generates the optimal nodes of a layer in the SOPNN model.

Step 7: *Check the termination criterion*

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

$$F_1 \leq F_*, \tag{9}$$

where, F_1 denotes a maximal fitness value occurring at the current layer whereas F_* stands for a maximal fitness value that occurred at the previous layer. As far as the depth of the network is concerned, the generation process is stopped at a depth of less than five layers. This size of the network has been experimentally found to form a sound compromise between the high accuracy of the resulting model and its complexity as well as generalization abilities.

In this study, we use a performance index that is Euclidean.

$$E(PI_s \ \ or \ \ EPI_s) = \frac{1}{N}\sum_{p=1}^{N}(y_p - \hat{y}_p)^2, \tag{10}$$

where y_p is the *p-th* target output data and $\hat{y}_p$ stands for the *p-th* actual output of the model for this specific data point. N is training (PI_s) or testing (EPI_s) input-output data pairs and E is an overall (global) performance index defined as a sum of the errors for the N.

Step 8: *Determine new input variables for the next layer*

If (9) has not been satisfied, the model has to be expanded. The outputs of the preserved nodes (z_{1i}, z_{2i}, …, z_{Wi}) serves as new inputs to the next layer (x_{1j}, x_{2j}, …, x_{Wj})($j=i+1$). This is captured by the expression

$$x_{1j} = z_{1i}, \ x_{2j} = z_{2i}, \ \ldots, \ x_{wj} = z_{wi}. \tag{11}$$

The SOPNN algorithm is carried out by repeating steps 4-8 consecutively.

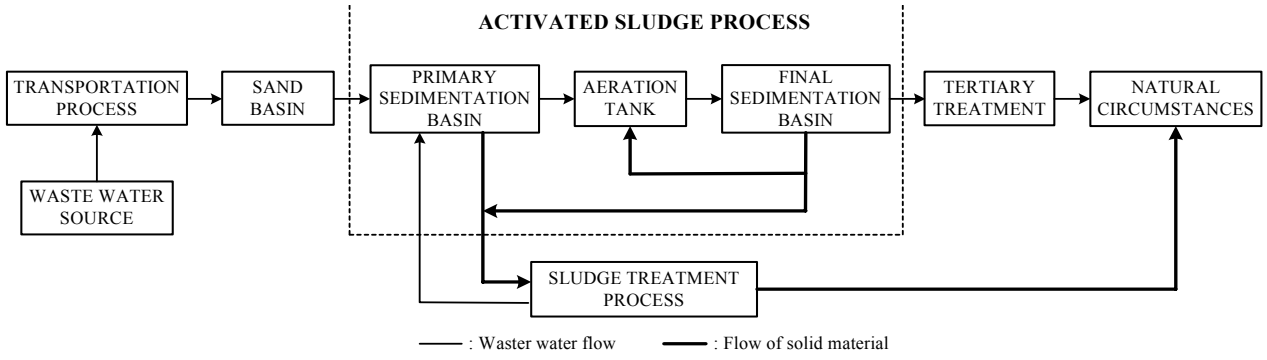


Fig. 4. Configuration of a sewage treatment system.

5. EXPERIMENTAL STUDIES

5.1. Sewage treatment process

Sewage treatment generally uses the activated sludge process that consists of sand basin, primary sedimentation basin, aeration tank and final sedimentation basin (see Fig. 4).

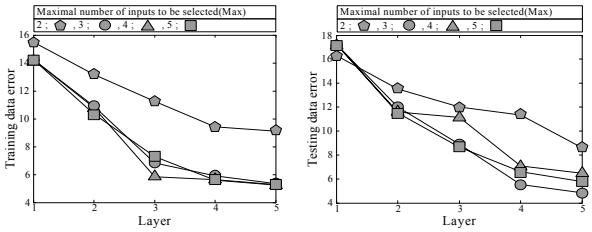
In this experiment, we use a data set coming from the sewage treatment system plant in Seoul, Korea. The proposed model is carried out using 52 pairs of input (MLSS, WSR, RRSP, and DOSP)-output (SS) data obtained from the activated sludge process [15]. Table 2 summarizes the parameters of the optimization environment and computational effort. In each layer, we use 100 generations, 60 populations, 36 bits as a string, crossover rate equal to 0.65, and mutation probability equal to 0.1. A chromosome used in the genetic optimization consists of a string including 3 sub-chromosomes. Bits per chromosome are assigned to 3, 3, and 30 respectively. The population size being selected from total population size, 60, is given to 30. That is, 60 nodes (PNs) are generated in each layer of the network through GA operation. All nodes generated in each layer are estimated and evaluated using the training and testing data set according to the change of a weighting factor of the aggregate objective function. These values are then compared and several PNs are chosen by a predefined number, 30 that provides better performance than the remaining PNs of the current layer from the viewpoint of aggregate performance index taking into account both *PI* and *EPI*. The number of inputs to be selected is confined to a maximum of 5 inputs. The form of the polynomial order is given as 3 types- Type 1, Type 2, and Type 3. The value of a weighting factor is considered as 5 cases such as 0.0, 0.25, 0.5, 0.75, and 1.0.

Fig. 5 depicts the performance index of a GA-based SOPNN according to an increase in the maximal number of inputs to be selected when using θ=0.5. Fig. 6 depicts the performance index of each layer of a GA-based SOPNN treated as a function of the weighting factor. Fig. 7 shows the performance index of a GA-based SOPNN according to the increase in the number of layers. Considering the training and testing data sets, the best results for the network in the 3rd layer are obtained when using θ=0.5 with Type 2 (Polynomial order: quadratic) and 3 node inputs (node numbers: 2, 11, 19), (that are quantified as PI=6.837, EPI=8.871). The best results for the network in the 5th layer coming with PI=5.365 and EPI=4.852 have been reported when using θ=0.5 with Type 2 (Polynomial order: quadratic) and 2 node inputs (node numbers: 18, 22). In Figs. 6-7, a(•)- e(•), and A(•)- E(•) denote the optimal node numbers of each layer of the network, namely those with the best predictive performance. Here, the node numbers of the 1st layer represent system input numbers, and the node numbers of each layer in the 2nd layer or higher represent the output node numbers of the preceding layer, as the optimal

Table 2. Computational overhead and list of parameters for a GA-based SOPNN.

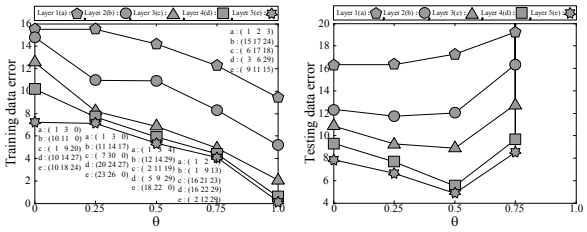
	Parameters	1st ~ 5th layer
GA	Maximum generation	100
	Total population size	60
	Selected population size (W)	30
	Crossover rate	0.65
	Mutation rate	0.1
	String length	3+3+30
SOPNN	Number of inputs to be selected (l)	$1 \le l \le$ Max (2~5)
	Type (T)	$1 \le T \le 3$
	Weighting factor (θ)	$0 \le \theta \le 1$

l,T: integer



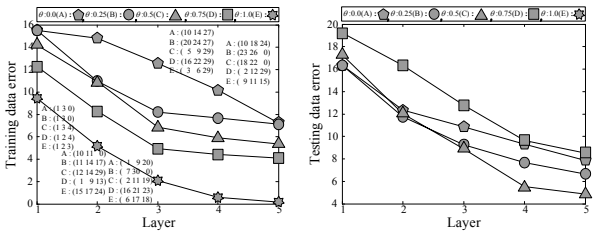
(a) PI. (b) EPI.

Fig. 5. Performance index of a GA-based SOPNN according to the increase in maximal number of inputs to be selected (θ=0.5).



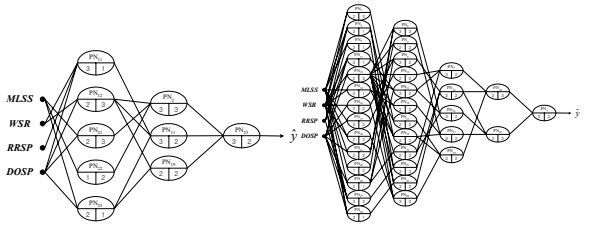
(a) PI. (b) EPI.

Fig. 6. Performance index of GA-based SOPNN treated as a function of the weighting factor.



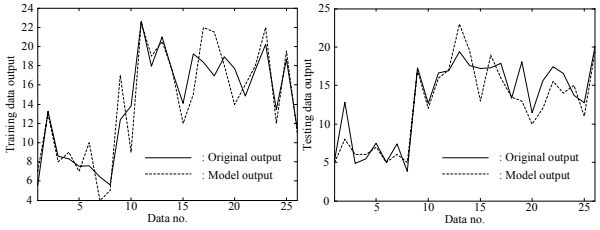
(a) PI. (b) EPI.

Fig. 7. Performance index of GA-based SOPNN according to the increase in the number of layers.



(a) Optimized SOPNN with 3 layers (θ=0.5). (b) Optimized SOPNN with 5 layers (θ=0.5).

Fig. 8. Genetically optimized SOPNN architecture (Max=3).



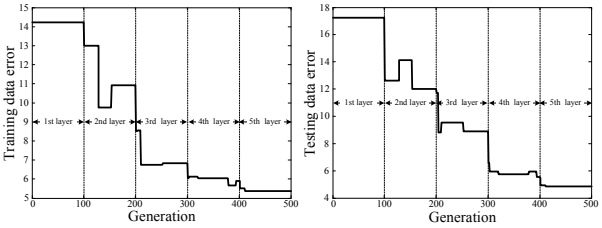
(a) Training data. (b) Testing data.

Fig. 9. Original output and model output of gas furnace process (in case of 5 layers and θ=0.5).

Fig. 10. Error curves of SOPNN (in case of 5 layers and θ=0.5).

node having the best output performance in the current layer. Fig. 8 illustrates the detailed optimal topologies of the GA-based SOPNN for 3 layers and 5 layers respectively. As shown in Fig. 8, the GA-based design procedure at each stage (layer) of SOPNN leads to the selection of preferred nodes (or PNs) with local characteristics (such as the number of input variables, the order of the polynomial, and input variables) available within the SOPNN. In the sequel, the proposed network enables the architecture to be a structurally more optimized and flexible network than the conventional SOPNN. Referring to Fig. 8, we adhere to the following notation 

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



(a) PI. (b) EPI.

Fig. 11. The optimization process of each performance index by the Gas.

Table 3. Comparison of performance with other modeling methods.

Model		Performance index	
		PI_s	EPI_s
Fuzzy model [15]	Simplified	13.726	16.206
	Linear	6.396	54.233
Hybrid Fuzzy model [16]	Simplified	12.403	12.200
	Linear	7.157	24.658
Fuzzy set-based FNN model[17]	Simplified	13.401	8.287
	Linear	12.307	9.828
Fuzzy relation-based FNN model [18]	Simplified	12.943	12.176
	Linear	10.584	12.108
Our model	θ=0.5 (3^{rd} layer)	6.837	8.871
	θ=0.5 (5^{th} layer)	5.365	4.852

PI_s - performance index on the training data, EPI_s - performance index on the testing data.

Figs. 9-10 show output comparison and identification errors for the optimal network architecture visualized in Fig. 8(b).

Fig. 11 illustrates the optimization process by visualizing the performance index in successive generations of the GA-based SOPNN. It also shows the optimized network architecture (the weighting factor θ is set at 0. 5 with 5 layers).

Table 3 summarizes the results of comparative analysis of the proposed model with respect to other constructs.

5.2. pH neutralization process

To demonstrate the high modeling accuracy of the GA-based SOPNN, we apply it to a highly nonlinear pH neutralization consisting of a weak acid and a strong base. This model can be found in a variety of practical areas including wastewater treatment, biotechnology processing, and chemical processing [19-23]. pH is the measurement of the acidity or alkalinity of a solution containing a proportion of water. It is mathematically defined, for diluted solutions, as the negative decimal logarithm of the hydrogen ion concentration $[H^+]$ in the solution, that is

$$pH = - \log_{10}[H^+]. \quad (12)$$

Table 4. Parameters and initial values for pH process.

Variables	Meaning	Initial setting
V	Volume of tank	1000 cc
F_a	Flow rate of acid	81 cc/min
F_b	Flow rate of base	515 cc/min
C_a	Concentration of acid in F_a	0.32 mole/l
C_b	Concentration of base in F_b	0.05 mole/l
K_a	Acid equilibrium constant	$1.76*10^{-5}$
K_w	Water equilibrium constant	$1.0*10^{-14}$
$W_a(0)$	Concentration of acid	0.0435 mole/l
$W_b(0)$	Concentration of base	0.0432 mole/l

Table 5. Summary of the parameters of the optimization and computational effort.

	Parameters	$1^{st} \sim 5^{th}$ layer
GA	Maximum generation	100
	Total population size	60
	Selected population size (W)	30
	Crossover rate	0.65
	Mutation rate	0.1
	String length	4+3+60
SOPNN	Number of inputs to be selected (l)	$1 \le l \le$ Max (3,5,7,10)
	Type (T)	$1 \le T \le 3$
	Weighting factor (θ)	$0 \le \theta \le 1$

l,T : integer

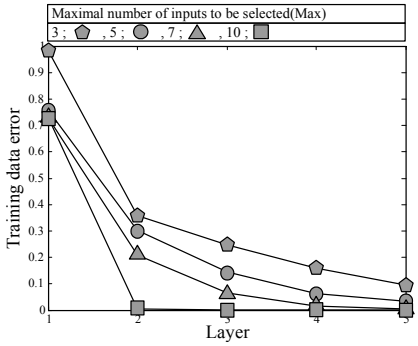


Fig. 12. Performance index of GA-based SOPNN according to the increase in the maximal number of inputs to be selected.

In a continuously stirred tank reactor (CSTR) [24,25] the investigated acetic acid (HAC) of concentration C_a flows into the tank at flow rate F_a, and is neutralized by sodium hydroxide (NaOH) of concentration C_b that flows into the tank at rate F_b. The equations of the CSTR can be described as follows (here we assume that the tank is perfectly mixed and isothermal, cf. [24]). The process equations for the CSTR are given by

$$\frac{Vdw_a}{dt} = F_a C_a - (F_a + F_b)W_a, \quad (13a)$$

$$\frac{Vdw_b}{dt} = F_b C_b - (F_a + F_b)W_b, \quad (13b)$$

where the constant V is the volume of the content in the reactor, w_a and w_b are the concentrations of the acid and the base, respectively.

The above equation describes how the concentrations of w_a and w_b change dynamically over time subject to the input streams F_a and F_b. To obtain the pH in the effluent, we need to find a relation between instantaneous concentrations w_a and w_b and pH values. This relationship can be described by a nonlinear algebraic equation known as the titration or characteristic curve. Depending on the chemical species used, the titration curve varies. Here we consider the case that a weak influent is neutralized by a strong reagent. The words strong and weak are used to characterize the degree of ionic dissociation in an aqueous solution. Strong reagents completely dissociate into their hydrogen or hydroxyl ions whereas weak reagents are only partially ionized.

Consider an acetic acid (weak acid) denoted by HAC being neutralized by a strong base NaOH (sodium bydroxide) in water. The reactions are

$$H_2O \Leftrightarrow H^+ + OH^-, \quad (14a)$$
$$HAC \Leftrightarrow H^+ + AC, \quad (14b)$$
$$NaOH \Rightarrow Na^+ + OH^-. \quad (14c)$$

According to the electroneutrality condition, the sum of the charges of all ions in the solution must be zero, i.e.

$$[Na^+]+[H^+] = [OH^-] + [AC^-], \quad (15)$$

where the symbol [X] denotes the concentration of the ion X.

On the other hand, the following equilibrium relationships hold for water and acetic acid:

$$K_a = [AC-][H+] / [HAC], \quad (16a)$$
$$K_w = [H+][OH-], \quad (16b)$$

where K_a and K_w are the dissociation constants of the acetic acid and water with $K_a=1.76\times10^{-5}$ and $K_w=10^{-4}$. Defining $w_a=[HAC]+[AC^-]$ as the total acetate and $w_b=[Na^+]$ and inserting Eqs. (16a) and (16b) into (15), we have

$$[H^+]^3+[H^+]^2\{K_a + w_b\}+[H^+]\{K_a(w_b - w_a)- K_w\}-K_aK_w=0. \quad (17)$$

Using (12), (17) becomes

$$W_b + 10^{-pH} - 10^{pH-pK_w} - \frac{W_a}{1+10^{pK_a-pH}} = 0, \quad (18)$$

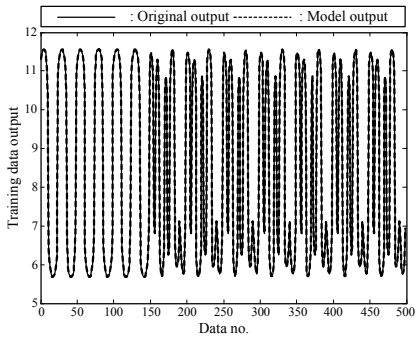


Fig. 13. Original output and model output of nonlinear function data.

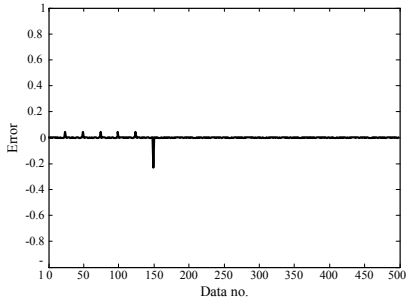


Fig. 14. Error curves of SOPNN.

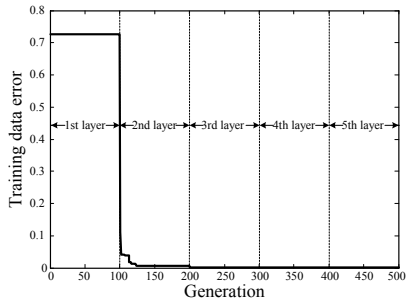


Fig. 15. The optimization process of each performance index by the genetic algorithms.

Table 6. Comparison of performance with other modeling methods.

Model			PI
Nie's model [26]	USOCPN		0.230
	SSOCPN		0.012
SOPNN [7]	Basic SOPNN (15th Layer)	Case 1	0.0015
		Case 2	0.0052
	Modified SOPNN (10th Layer)	Case 1	0.0039
		Case 2	0.0124
Our model (3rd Layer)			0.00019

where pKa=-$\log_{10}k_a$.

We consider the weak acid-strong base neutralization process described by (13a), (13b) and (18). By fixing the acid flow-rate F_a (81cc/min) at a specific value, the process is regarded as a single variable system with base flow-rate F_b and the pH in the effluent being the input and output, respectively. The (F_b, y_{pH}) data pairs were produced by using the process physical model with the values of the parameters given in Table 4.

The base flow rate F_b was given by

$$F_b = 515 + 51.5\sin(2\pi t/25), \quad for \quad t \le 150, \quad (19a)$$

$$F_b = 515 + 25.75\sin(2\pi t/25) + 25.75\sin(2\pi t/10), \quad for \quad t \rangle 150. \quad (19b)$$

To obtain such data pairs, we applied the Newton-Raphson method, which is given by

$$pH_{i+1} = pH_i - \frac{f(pH_i)}{f'(pH_i)}. \quad (20)$$

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

$$\hat{y}_{pH}(t) = \varphi(F_b(t-3),\ F_b(t-2),\ F_b(t-1),\ y_{pH}(t-3),\ y_{pH}(t-2), y_{pH}(t-1)), \quad (21)$$

where $\hat{y}_{pH}$ and y_{pH} denote the GA-based SOPNN model output and the actual process output, respectively. 500 data pairs are generated from (19a), (19b), and (20) where the total data are used for training. To come up with a quantitative evaluation of the network, we use the standard MSE performance index of (10).

Table 5 shows a summary of the parameters of the optimization and computational effort. The GA-based design procedure is carried out in a similar manner using the parameters of Table 4.

Fig. 12 depicts the performance index of GA-based SOPNN according to the increase in the maximal number of inputs to be selected. The best results for the network in the 3rd layer are obtained when using Max=10 with Type 2 (Polynomial order: quadratic) and 10 node inputs (node number: 2, 9, 10, 12, 18, 19, 21, 27, 28, 29), (that are quantified as PI=0.00019), The best results for the network in the 5th layer coming with PI=0.00013 have been reported when Max=10 with Type 2 (Polynomial order: quadratic) and 9 node inputs (node inputs: 1, 3, 5, 13, 14, 17, 21, 28, 30).

Figs. 13-14 show output comparison and identification errors for the optimal network architecture visualized when Max=10 with Type 2.

Fig. 15 illustrates the optimization process by visualizing the performance index in successive generations of the GA-based SOPNN. As shown in Fig. 15, the variation ratio (slope) of the performance of the GA-based SOPNN model changes radically at the 2nd layer. Therefore, to effectively reduce a large

number of nodes and avoid a large amount of time-consuming iteration of SOPNN layers, the stopping criterion can be taken into consideration up to maximally the 2nd or 3rd layer.

Table 6 depicts a comparison of identification errors with previous modeling methods. In the literature [26], which the unsupervised self-organizing counter-propagation network algorithm (USOCPN) and unsupervised self-organizing counter-propagation network algorithm (SSOCPN) were proposed, the number of fuzzy rules and mean square errors (MSEs) were 31 and 0.230, respectively for the case of the USOCPN, and 34 and 0.012, respectively for the case of the SSOCPN. Compared with that approach, Table 6 shows that the proposed method is much more superior.

6. CONCLUSIONS

In this study, the GA-based design procedure of Self-Organizing Polynomial Neural Networks (SOPNN) and its design methodology were proposed to construct optimal model architecture for nonlinear and complex system modeling. The design methodology comes with hybrid structural optimization and parametric learning viewed as two phases of modeling building. That is, the one phase (hybrid structural optimization) is realized via both GAs and a structural phase of a self-organizing and an evolutionary algorithm as the main characteristics of the GMDH method, while the other phase (parametric optimization) is carried out by a standard least square estimation-based learning. The essence of the GA-based SOPNN lies in polynomial based activation nodes and network architecture based on the GMDH and genetically optimized multi-layer perceptrons (MLPs). Accordingly, the architecture of the network is not predetermined (as in most existing neural networks) but becomes dynamically adjusted and optimized during the development process. The GA-based design procedure at each stage (layer) of the SOPNN leads to the selection of these preferred nodes (or PNs) with local characteristics (such as the number of input variables, the order of the polynomials, and input variables) available within the SOPNN. Then, based on these selections, we build flexible and optimized architecture of the GA-based SOPNN. The comprehensive experimental studies involving well-known datasets (sewage treatment process data and pH neutralization process) quantify a superb performance of the network in comparison to the existing fuzzy and neuro-fuzzy models. First of all, we could efficiently search for the optimal network architecture (structurally and parametrically optimized network) by the design methodology of GA-based SOPNN in comparison to that of the conventional SOPNN.

REFERENCES

[1] V. Cherkassky, D. Gehring, and F. Mulier, "Comparison of adaptive methods for function estimation from samples," *IEEE Trans. on Neural Networks*, vol. 7, pp. 969-984, July 1996.

[2] J. A. Dicherson and B. Kosko, "Fuzzy function approximation with ellipsoidal rules," *IEEE Trans. on Systems, Man and Cybernetics*, *Part B*, vol. 26, pp. 542-560, August 1996.

[3] A. G. Ivakhnenko, "Polynomial theory of complex systems," *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-1, pp. 364-378, 1971.

[4] A. G. Ivakhnenko and H. R. Madala, *Inductive Learning Algorithms for Complex Systems Modeling*, CRC Press, London, 1994.

[5] A. G. Ivakhnenko and G. A. Ivakhnenko, "The review of problems solvable by algorithms of the group method of data handling (GMDH)," *Pattern Recognition and Image Analysis*, vol. 5, no. 4, pp. 527-535, 1995.

[6] A. G. Ivakhnenko, G. A. Ivakhnenko, and J.A. Muller, "Self-organization of neural networks with active neurons," *Pattern Recognition and Image Analysis*, vol. 4, no. 2, pp. 185-196, 1994.

[7] S.-K. Oh and W. Pedrycz, "The design of self-organizing polynomial neural networks," *Information Science*, vol. 141, pp. 237-258, 2002.

[8] 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. 703-725, 2003.

[9] H.-S. Park, B.-J. Park, and S.-K. Oh, "Optimal design of self-organizing polynomial neural networks by means of genetic algorithms," *Journal of the Research Institute of Engineering Technology Development* (in Korean), vol. 22, pp. 111-121, 2002.

[10] W. Pedrycz and M. Reformat, "Evolutionary optimization of fuzzy models in fuzzy logic: A framework for the new millennium," V. Dimitrov and V. Korotkich (eds.), *Studies in Fuzziness and Soft Computing*, Phsica-Verlag, vol. 8, pp. 51-67, September 1996.

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

[12] D. E. Goldberg, *Genetic Algorithm in Search, Optimization & Machine Learning*, Addison Wesley, 1989.

[13] K. A. De Jong, "Are genetic algorithms function optimizers?," *in Parallel Problem Solving from Nature* 2, Manner, R. and Manderick, B. eds., North-Holland, Amsterdam, 1992.

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

[15] S.-K. Oh and W. Pedrycz, "Identification of

fuzzy systems by means of an auto-tuning algorithm and its application to nonlinear systems," *Fuzzy Sets and Systems*, vol. 115, no. 2, pp. 205-230, 2000.
[16] S.-K. Oh, W. Pedrycz, and B.-J. Park, "Hybrid identification of fuzzy rule-based models," *Int. J. of Intelligent Systems,* vol. 17, no.1, pp. 77-103, Jan. 2002.
[17] S.-K. Oh, W. Pedrycz, and H.-S. Park, "Hybrid identification in fuzzy-neural networks," *Fuzzy Sets and Systems*, vol. 138, pp. 399-426, 2003.
[18] S.-K. Oh, W. Pedrycz, and H.-S. Park, "Fuzzy relation-based neural-networks and their hybrid identification," *IEEE Trans. on Instrumentation and Measurement*, 2004(*submitted*).
[19] F. G. Shinskey, *pH and pION Control in Proc. and Waste Streams*, Wiley, New York, 1973.
[20] R. C. Hall, and D. E. Seberg, "Modeling and self-tuning control of a multivariable pH neutralization process," *Proc. ACC*, pp. 1822-1827, 1989.
[21] T. J. McAvoy, "Time optimal and Ziegler-Nichols control," *Ind. Eng. Chem. Process Des. Develop*, vol. 11, no. 1, pp. 71-78, January 1972.
[22] G. A. Pajunen, "Comparison of linear and nonlinear adaptive control of a pH-process," *IEEE Control Systems Magazine*, vol. 7, no. 1, pp. 39-44, February 1987.
[23] C. L. Karr, and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. on Fuzzy Systems*, vol. 1, pp. 46-53, 1993.
[24] T. J. McAvoy, E. Hsu, and S. Lowenthal, "Dynamics of pH in controlled stirred tank reactor," *Ind. Engrg. Chem. Process Des. Develop,* vol. 11, no. 1, pp. 68-70, January 1972.
[25] T. K. Gustafsson and K. V. Waller, "Dynamic modeling and reaction invariant control of pH," *Chem. Engrg. Sci.*, vol. 38, pp. 389-398, 1983.
[26] J. Nie, A. P. Loh, and C. C. Hang, "Modeling pH neutralization processes using fuzzy-neural approaches," *Fuzzy Sets and Systems,* vol. 78, pp. 5-22, 1996.
[27] B.-J. Park, W. Pedrycz, and S.-K. Oh, "Fuzzy polynomial neural networks: hybrid architectures of fuzzy modeling," *IEEE Trans. on Fuzzy Systems,* vol. 10, no. 5, pp 607-621, October 2002.
[28] S.-K. Oh, J. F. Peters, W. Pedrycz, and T.-C. Ahn, "Genetically optimized rule-based fuzzy polynomial neural networks: synthesis of computational intelligence technologies," *Lecture Notes in Artificial Intelligence*, vol. 2639, pp. 437-444, May 2003.
[29] S.-K. Oh, W. Pedrycz, and B.-J. Park, "Self-organizing neurofuzzy networks based on evolutionary fuzzy granulation," *IEEE Trans. on SMC-A,* vol. 33, no. 2, pp. 271-277, March 2003.
[30] S.-K. Oh, *Fuzzy Model & Control System by C-Programming*, Naeha Press, 2002.
[31] S.-K. Oh, *Computational Intelligence by Programming focused on Fuzzy, Neural Networks, and Genetic Algorithms*, Naeha Press, 2002.

Ho-Sung Park received his B.S. and M.S. degrees in Control and Instrumentation Engineering from Wonkwang University, Korea in 1999 and 2001, respectively. He is currently a Ph.D. student at the same institute. His research interests include fuzzy and hybrid systems, neurofuzzy models, genetic algorithms, and computational intelligence. He is a member of KIEE and ICASE.

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. 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.

Sung-Kwun Oh received his B.S., M.S., and Ph.D. degrees in Electrical Engineering from Yonsei University, Seoul, Korea, in 1981, 1983 and 1993, respectively. During 1983-1989, he worked as the Senior Researcher in the R&D Lab. of Lucky-Goldstar Industrial Systems Co., Ltd. He was a Postdoctoral Fellow in the Department of Electrical and Computer Engineering at the University of Manitoba, Canada, from 1996 to 1997. He is currently a Professor in the School of Electrical, Electronic and Information Engineering, Wonkwang University, 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 for the Korean Institute of Electrical Engineers (KIEE) and the Institute of Control, Automation & Systems Engineers (ICASE), Korea.