International Journal of Fuzzy Logic and Intelligent Systems, vol. 6, no. 1, March 2006 pp. 33-38

Genetically Optimized Fuzzy Polynomial Neural Network and Its Application to Multi-variable Software Process

In-Tae Lee*, Sung-Kwun Oh*, Hyun-Ki Kim*, and Witold Pedrycz**

*** Department of Electrical Engineering, The University of Suwon, San 2-2, Wau-ri, Bongdam-eup, Hwaseong-si, Gyeonggi-do, 445-743, South Korea**

**** Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2G6, Canada**

ohsk@suwon.ac.kr

Abstract

In this paper, we propose a new architecture of Fuzzy Polynomial Neural Networks(FPNN) by means of genetically optimized Fuzzy Polynomial Neuron(FPN) and discuss its comprehensive design methodology involving mechanisms of genetic optimization, especially Genetic Algorithms(GAs). The conventional FPNN developed so far are based on mechanisms ofself-organization and evolutionary optimization. The design of the network exploits the extended Group Method of Data Handling(GMDH) with some essential parameters of the network being provided by the designer and kept fixed throughout the overall development process. This restriction may hamper a possibility of producing an optimal architecture of the model. The proposed FPNN gives rise to a structurally optimized network and comes with a substantial level of flexibility in comparison to the one we encounter in conventional FPNNs. It is shown that the proposed advanced genetic algorithms based Fuzzy Polynomial Neural Networks is more useful and effective than the existing models for nonlinear process. We experimented with Medical Imaging System(MIS) dataset to evaluate the performance of the proposed model.

Key Words : Fuzzy Polynomial Neural Networks (FPNN), Fuzzy Polynomial Neuron (FPN), Genetic Algorithms based Fuzzy Polynomial Neural Networks(GAs-based FPNN), Genetic Algorithms (GAs), Group Method of Data Handling (GMDH), Multi-variable Software Process

1. Introduction

It is expected that efficient modeling techniques should allow for a selection of pertinent variables and a formation of highly representative datasets. Furthermore, the resulting models should be able to take advantage of the existing domain knowledge (such as a prior experience of human observers or operators) and augment it by available numeric data to form a coherent data-knowledge modeling entity. Most recently, the omnipresent trends in system modeling are concerned with a broad range of techniques of computational intelligence(CI) that dwell on the paradigm of fuzzy modeling, neurocomputing, and genetic optimization[1, 2].The list of evident landmarks in the area of fuzzy and neurofuzzy modeling [3, 4] is impressive. While the accomplishments are profound, there are still a number of open issues regarding structure problems of the models along with their comprehensive development and testing.

As one of the representative and advanced design approaches comes a family of fuzzy polynomial neuron (FPN)-based self organizing neural networks (abbreviated as FPNN or PNN and treated as a new category of neuro-fuzzy networks)[5-7]. The design procedure of FPNNs exhibits some tendency to produce overly complex networks as well as comes with a repetitive computation load caused by the trial and error method being a part of the development process. The latter is in essence inherited from theoriginal Group Method of Data handling (GMDH) [10] algorithm that requires some repetitive parameter adjustment by the designer.

In this study, in addressingthe above problems with the conventional SOPNN (especially, FPN-based PNN called "FPNN" [6, 9, 15]) as well as the GMDH algorithm, we introduce a new genetic design approach. Bearing this new design in mind, we will be referring to these networks as genetic algorithms based FPNN ("GAs-based FPNN"for brief). The determination of the optimal values of the parameters available within an individual FPN (viz. the number of input variables, the number of membership function (MFs), the order of the polynomial, and a collection of the specific subset of input variables) leads to a structurally and parametrically optimized network. The network is directly contrasted with several existing neurofuzzy models reported in the literature.

To assess the performance of proposed model, we experiment with well-know medical imaging system (MIS) [8] widely used in software engineering.

Manuscript received Dec. 6, 2005; revised Mar. 7, 2006
This work has been supported by KESRI (I-2004-0-074-0-00), which is funded by MOCIE(Ministry of Commerce, Industry and Energy)

2. The architecture and development of Fuzzy Polynomial Neural Networks (FPNN)

In this section, we elaborate on the architecture and a development process of the FPNN. This network emerges from the genetically optimized multi-layer perceptron architecture based on GA algorithms and the extended GMDH method.

2.1 FPNN Based on Fuzzy Polynomial Neurons(FPNs)

We start with a fuzzy polynomial neuron(FPN). This neuron, regarded as a generic type of the processing unit, dwells on the concepts of fuzzy sets and neural networks. Fuzzy sets realize a linguistic interface by linking the external world numeric data with the processing unit. Neurocomputing manifests in the form of a local polynomial unit realizing some nonlinear processing. The FPN encapsulates a family of nonlinear "if-then" rules. FPN realizes a family of multiple-input single-output rules. Each rule, refer again to Fig. 1, reads in the form

if x_p is A_l and x_q is B_k then z is $P_{lk}(x_i, x_j, a_{lk})$ (1)

where a_{lk} is a vector of the parameters of the conclusion part of the rule while $P_{lk}(x_i, x_j, a_{lk})$ denotes the regression polynomial forming the consequence part of the fuzzy rule which uses several types of high-order polynomials besides the constant function forming the simplest version of the consequence, refer to Table 1. The activation levels of the rules contribute to the output of the FPN being computed as a weighted average of the individual condition parts (functional transformations) P_K (note that the index of the rule, namely "K" is a shorthand notation for the two indexes of fuzzy sets used in the rule (1), that is $K = (l, k)$).

$$z = \sum_{K=1}^{all\ rules} \mu_K P_K(x_i, x_j, a_K) \Big/ \sum_{K=1}^{all\ rules} \mu_K = \sum_{K=1}^{all\ rules} \tilde{\mu}_K P_K(x_i, x_j, a_K) \quad (2)$$

In the above expression, we use an abbreviated notation to describe an activation level of the "K"th rule to be in the form

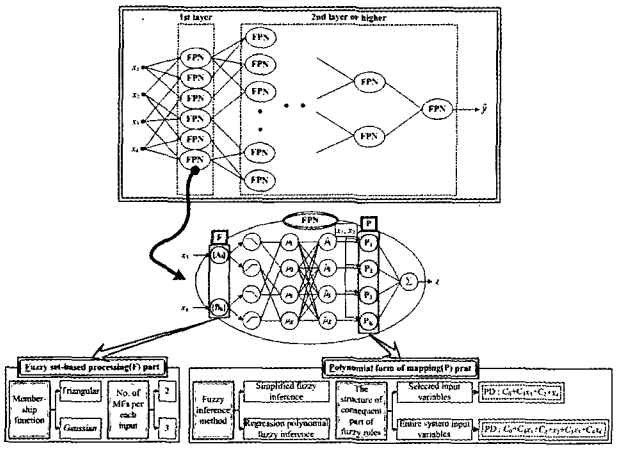


Fig. 1. A general topology of the FPN based FPNNalong with the structure of the generic FPN module (F: fuzzy set-based processing part, P: the polynomial form of mapping)

$$\tilde{\mu}_K = \frac{\mu_K}{\sum_{L=1}^{all\ rules} \mu_L} \quad (3)$$

Based on the genetically optimized number of the nodes (input variables) and the polynomial order, refer to Table 1, we construct the optimized self-organizing network architectures of the FPNNs.

Table 1. Different forms of the regression polynomials forming the consequence part of the fuzzy rules.

No. of input / Order of the Polynomial	1	2	3
0(Type 1)	Constant		
1(Type 2)	Linear	Bilinear	Trilinear
2(Type 3)	Quadratic	Biquadratic	Triquadratic
2(Type 4)		Modified Biquadratic	Modified Triquadratic

2.2 Genetic Algorithms based FPNN

In order to enhance the learning of the FPNN and augment its performance, we use genetic algorithms to obtain the structuraloptimization of the network by optimally selecting such parameters as the number of input variables(nodes), the order of polynomial, and input variables within a FPN. Here, GAs use serial method of binary type, roulette-wheel as the selection operator, one-point crossover, and an invert operation in the mutation operator[7].

In this study, for the optimization of the FPNN model, GA usesthe 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. The overall genetically-driven structural optimization process of FPNN is shown in Fig. 2.

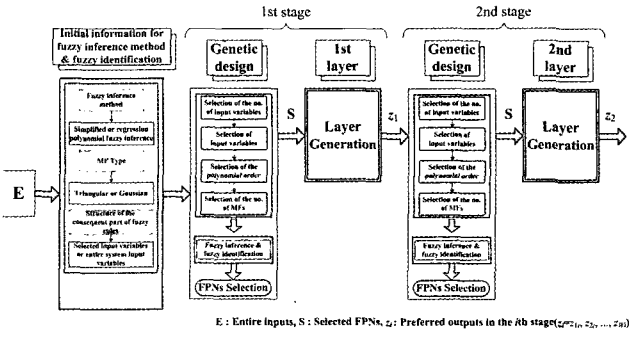


Fig. 2. Overall genetically-driven structural optimization process of FPNN

3. The design procedure of genetically optimized FPNN

We use FPNNs as the building blocks of the network. Each neuron of the network realizes a polynomial type of partial description(PD) of the mapping between input and output variables. The input-output relation formed by the FPNN algorithm can be described in the form

$$y = f(x_1, x_2, \cdots, x_n) \tag{4}$$

The estimated output $\hat{y}$ reads as the following polynomial

$$\hat{y} = \hat{f}(x_1, x_2, \cdots, x_n) = c_0 + \sum_{k1} c_{k1} x_{k1} + \sum_{k1k2} c_{k1k2} x_{k1} x_{k2} + \sum_{k1k2k3} c_{k1k2k3} x_{k1} x_{k2} x_{k3} + \cdots \tag{5}$$

where ck s are its coefficients.

The framework of the design procedureof the Fuzzy Polynomial Neural Networks (FPNN) based on genetically optimized multi-layer perceptron architecture comprises the following steps

[Step 1] *Determine system's input variables*

[Step 2] *Form training and testing data*

The input-output data set (xi, yi)=(x1i, x2i, , xni, yi), i=1, 2, , N (with N being the total number of data points) is divided into two parts, that is, a training and testing dataset.

[Step 3] *specify initial design parameters*

Here we decide upon the essential design parameters of the FPNN structure.

a) The decision ofinitial information for fuzzy inference method and fuzzy identification
   - Fuzzy inference method
   - MF type: Triangular or Gaussian-like MF
   - No. of MFs per each input of a node (or FPN)
   - Structure of the consequence part of fuzzy rules
b) The maximum number of input variables entering each node in the corresponding layer.
c) The total number (W) of nodes to be retained (selected) at the next generation of the FPNN algorithm.

[Step 4] *Decide FPN structure using genetic design*

When it comes to the organization of the chromosome representing (mapping) the structure of the FPNN, we divide the chromosome to be used for genetic optimization into three sub. The 1st sub-chromosome contains the number of input variables, the 2nd sub-chromosome involves the order of the polynomial of the node, the 3rd sub-chromosome (remaining bits) contains input variables coming to the corresponding node (FPN), and the 4th sub-chromosome contains the number of membership functions per each input variable. All these elements are optimized when running the GA.

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

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

[Step 4-2] Selection of the polynomial order of the consequent part of fuzzy rules (2nd sub- chromosome)

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

[Step 4-4] Selection of the number of membership functions(4th sub-chromosome)

[Step 5] Carry out fuzzy inference and coefficient parameters estimation for fuzzy identification in the selected node (FPN)

In each fuzzy inference, we consider two types of membership functions, namely triangular and Gaussian-like membership functions.

The consequence part can be expressed by linear, quadratic, or modified quadratic polynomial equation as mentioned previously.

Proceeding with each layer of FPNN, the designalternatives available within a single FPN can be made with regard to the selected input variables in the consequence part of fuzzy rules. Following these criteria, we distinguish between the two fundamental types of the rules

The use of the regression polynomial inference method gives rise to the expression

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

The numeric output $\hat{y}$ is determined in the same way as in the previous approach that is

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

where, $f_i(\bullet)$ is a regression polynomial function of the input variables.

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

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

To evaluate the performance of FPNs (nodes) constructed using the training dataset, the testing dataset is used. Based on this performance index, we calculate the fitness function.

The fitness function reads as

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

where EPI denotes the performance index for the testing data (or validation data). In this case, the model is obtained by the training data and EPIis obtained from the testing data (or validation data) of the FPNN model constructed by the training data.

The outputs of the retained nodes (FPNs) serve as inputs to the next layer of the network. There are two cases as to the number of the retained FPNs, that is

(i) If W*<W, then the number of the nodes (FPNs) retained for the next layer is equal to z. Here, W* denotes the number of the retained nodes in each layer that nodes with the duplicated fitness values are moved.

(ii) If W*>W, then for the next layer, the number of the retained nodes (FPNs) is equal to W.

[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 a 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,

$$F_1 \le F_* \tag{9}$$

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.

In this study, we use two measures (performance indexes) that is theMean Squared Error (MSE)

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

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) or testing(EPI) 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 met, the model is expanded. The outputs of the preserved nodes (zli, z2i, , zWi) serves as new inputs to the next layer (x1j, x2j, , xWj)(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 FPNN algorithm is carried out by repeating steps 4-8.

4. Experimental studies

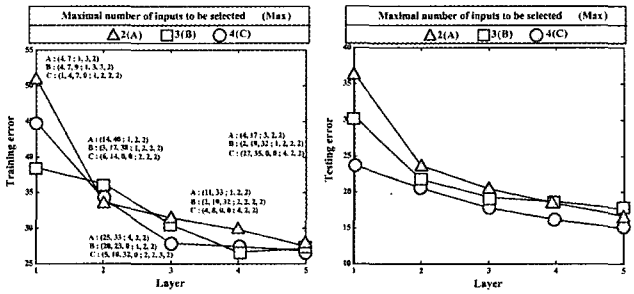
In this section, we illustrate the development of the GAs-based FPNN and show its performance for well known and widely used datasets in software engineering. That is medical imaging sysem (MIS)[8] data.

We consider a medical imaging system (MIS) subset of 390 software modules written in Pascal and FORTRAN for modeling. These modules consist of approximately 40,000 line of code. To design an optimal model from the MIS, we study 11 system input variables such as LOC, CL, .TChar, TComm, MChar, DChar, N, NE, NF, V(G) and BW. The output variable of the model is the number of changes changes made to the software module during its development. In the case of the MIS data, the performance index is defined as the mean squared error (MSE) as in (10)

Table 2 summarizes the list of parameters used in the genetic optimization of the network. Fig. 3 depicts the performance index of each layer of GAs-based FPNN according to the increase of maximal number of inputs to be selected. In Fig. 3, A(.)- C(.) denote the optimal node numbers at each layer of thenetwork, namely those with the best predictive performance.

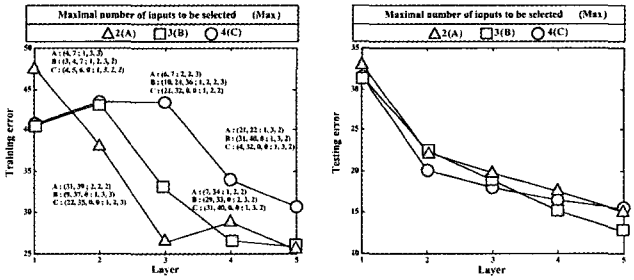
Table 2. Summary of the parameters of the genetic optimization

	Parameters	1st~3rd
GA	Maximum gen	300
	Total population size	150
	Selected population size	40
	Crossover rate	0.65
	Mutation rate	0.1
	String length	3+3+30+5
FPNN	Maximal no. of inputs to be selected(Max)	$1 \le l \le Max(2\sim4)$
	polynomial Type(Type T) of the consequent part of rules	$1 \le T^* \le 4$
	Membership Function(MFs) type	Gaussian Triangular
	No. of MFs per each input	2 or 3



(a-1) Training error (a-2) Testing error

(a) Triangular MF



(b-1) Training error (b-2) Testing error

(b) Gaussian-like MF

Fig. 3. Performance index of GAs-based FPNN with respect to the increase of number of layers

Fig. 4 illustrates the detailed optimal topologies of GAs-based FPNN for onelayer when using Max=3 and Gaussian-like MF: the results of the network have been reported as PI=25.804 and EPI=12.637. As shown in Fig 4, the proposed network enables the architecture to be a structurally more optimized and simplified network than the conventional FPNN. In nodes (FPNs) of Fig. 4, 'FPNn' denotes the nth FPN (node) of the corresponding layer, numeric values with rectangle form before a node(nuron) mean number of membership functions per each input variable, the number of the left side denotes the number of nodes (inputs or FPNs) coming to the corresponding node, and the number of the

right side denotes the polynomial order of conclusion part of fuzzy rules used in the corresponding node. Figs. 5-6 show output comparison and identification errors for the optimal network architecture visualized in Fig. 4.

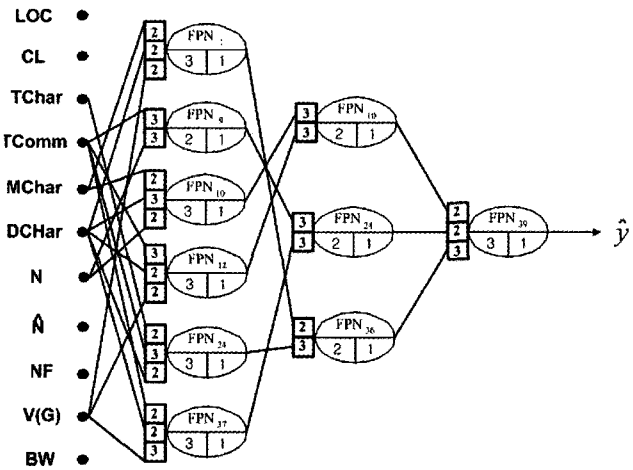
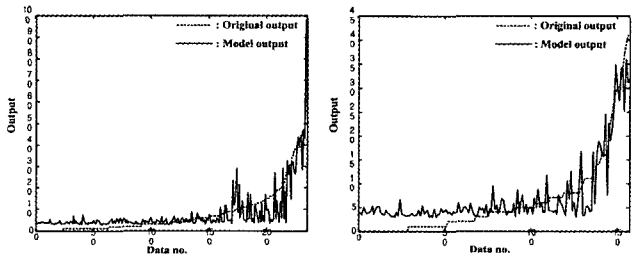
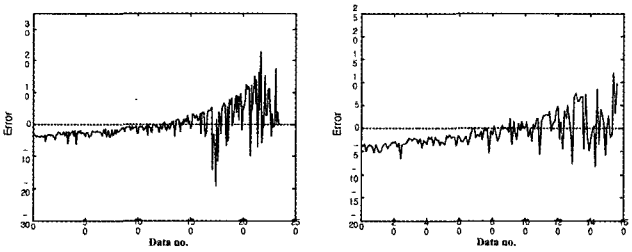


Fig. 4. Optimal networks structure of GAs-based FPNN (for 3 layers)



(a) Training data (b) Testing data

Fig. 5. Original output and model output of Medical Imaging System data(Max=3, Gaussian like MFs)

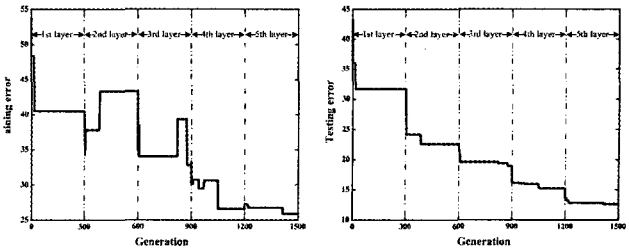


(a) Training data (b) Testing data

Fig. 6. Errors curve of GAs-based FPNN(Max=3, Gaussian like MFs)

Fig. 7 illustrates the optimization process by visualizing the values of the performance index obtained in successive generations of GA. It also shows the optimized network architecture when using the Gaussian-like membership functions(the maximal number (Max) of inputs to be selected is set to 3 with the structure composed of 5 layers).

Table 3 summarizes a comparative analysis of the performance of the network with other models. PIs(EPIs) is defined as the mean square errors (MSE) computed for the experimental data and the respective outputs of the network.



(a) Training error (b) Testing error

Fig. 7. The optimization process quantified by the values of the performance index

Table 3. Comparative analysis of the performance of the network; considered are models reported in the literature

Model	Sturcture				PI	EPI
SONFN[11]	Simplified	Generic Type	Basic Architecture	5th	40.753	17.898
	Linear	Genric Type	Basic Architecture	5th	35.745	17.807
FPNN[12]	Max	MF Type	T	Layer	PI	EPI
	2	T	2	5th	32.195	18.462
		G	1	5th	49.716	31.423
	3	T	1	5th	32.251	19.622
		G	1	5th	39.093	19.983
Our Model	2	T	3	5th	27.406	16.641
		G	1	5th	25.435	14.976
	3	T	1	5th	27.162	17.598
		G	1	5th	25.804	12.637
	4	T	4	5th	26.835	14.919
		G	1	5th	30.690	15.390

5. Concluding remarks

In this study, the GA-based design procedure of Fuzzy Polynomial Neural Networks (FPNN) along with its architectural considerations has been investigated. The GA-based design procedure applied at each stage (layer) of the FPNN leads to the selection of the preferred nodes (or FPNs) with optimal local characteristics (such as the number of input variables, the order of the consequent polynomial of fuzzy rules, input variables, and the number of membership functions) available within FPNN. These options contribute to the flexibility of the resulting architecture of the network. The design methodology emerges as a hybrid structuraloptimization (based on GMDH method and genetic optimization) and parametric learning being regarded as atwo phase design procedure. The GMDH method is comprised of both a structural phase such as a self-organizing and an evolutionary algorithm (rooted in natural law of survival of the fittest), and the ensuing parametric phase of the Least Square Estimation (LSE)-based learning.

The comprehensive experimental studies involving well-known multi-variable software process datasets quantify a superb performance of the network in comparison to the existing fuzzy and neuro-fuzzy models. Most importantly, through the proposed framework of genetic optimization we can effi-

ciently search for the optimal network architecture (being both structurally and parametrically optimized) and this design face-tbecomes crucial in improving the performance of the resulting model.

References

[1] W. Pedrycz, *Computational Intelligence: An Introduction,* CRC Press, Florida, (1998).

[2] J.F Peters and W. Pedrycz, Computational intelligence, *In Encyclopedia of Electrical and Electronic Engineering,* Volume 22, (Edited by J.G. Webster). John Wiley & Sons, New York, (1999).

[3] W. Pedrycz and J.F. Peters(Editers), *Computational Intelligence in Software Engineering,* World Scientific, Singapore, (1998).

[4] H. Takagi and I. Hayashi, NN-driven fuzzy reasoning, *Int. J. of Approximate Reasoning* 5 (3), 191-212, (1991).

[5] O. Cordon, et al., "Ten years of genetic fuzzy systems: current framework and new trends", *Fuzzy Sets and Systems,* Vol. 141, No. 1, pp. 5-31, 2004.

[6] S.-K. Oh and W. Pedrycz, "Self-organizing Polynomial Neural Networks Based on PNs or FPNs : Analysis and Design", *Fuzzy Sets and Systems*, volume 142, pp163-198, 1 March 2004.

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

[8] M.R. Lyu (Editor), Handbook of Software *Reliability Engineering* , McGraw-Hill, (1995).

[9] S.-K. Oh and W. Pedrycz, "Fuzzy Polynomial Neuron-Based Self-Organizing Neural Networks", *Int. J. of General Systems*, Vol. 32, No. 3, pp. 237-250, May, 2003.

[10] L. X. Wang, J. M. Mendel, "Generating fuzzy rules from numerical data with applications", *IEEE Trans. Systems, Man, Cybern.*, Vol. 22, No. 6, pp. 1414-1427, 1992.

[11] S. K. Oh, W. Pedrycz, and B. J. Park, "Relation-based Neurofuzzy Networks with Evolutionary Data Granulation", *Methematical and Computer Modeling,* 2003.

[12] S. K. Oh and W. Pedrycz, "Fuzzy Polynomial Neuron-Based Self-Organizing Neural Networks", I*nt. J. of General Systems*, Vol. 32, No. 3, pp. 237-250, May, 2003.

[13] S.-K. Oh, W. Pedrycz and T.-C. Ahn, "Self-organizing neural networks with fuzzy polynomial neurons", *Applied Soft Computing*, Vol. 2, Issue 1F, pp. 1-10, Aug. 2002.

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

[15] S.-K. Oh, W. pedrycz and T.-C. Ahn, "Self-organizing neural networks with fuzzy polynomial neurons", *Applied Soft Computiong*, Vol. 2, Issue 1F, pp. 1-10, Aug. 2002.

In-Tae Lee

He received the B.S degrees in electrical and electronics engineering from the Wonkwang University in 2005. He is now pursuing his Master degree in electrical engineering from University of Suwon. His research interests include FIS, Neural Networks, GAs, optimization theory, Computational Intelligence and Automation control.

Sung-kwun Oh

He received his BSc, MSc, and PhD. 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 Engineering, University of Suwon, Korea. His research interests include fuzzy systems, fuzzy-neural networks, automation systems, advanced Computational Intelligence, and intelligent control. He is a member of IEEE. He currently serves as an Associate Editor of the Korean Institute of Electrical Engineers (KIEE) and the Institute of Control, Automation & Systems Engineers (ICASE), Korea.

Hyun-Ki Kim

He received his B.Sc., M.Sc., and PhD. 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.