Performance Comparison between Neural Network and Genetic Programming Using Gas Furnace Data

Hyeon Bae, Tae-Ryong Jeon, and Sungshin Kim †, *Member, KIMICS*

Abstract—This study describes design and development techniques of estimation models for process modeling. One case study is undertaken to design a model using standard gas furnace data. Neural networks (NN) and genetic programming (GP) are each employed to model the crucial relationships between input factors and output responses. In the case study, two models were generated by using 70% training data and evaluated by using 30% testing data for genetic programming and neural network modeling. The model performance was compared by using RMSE values, which were calculated based on the model outputs. The average RMSE for training and testing were 0.8925 (training) and 0.9951 (testing) for the NN model, and 0.707227 (training) and 0.673150 (testing) for the GP model, respectively. As concern the results, the NN model has a strong advantage in model training (using the all data for training), and the GP model appears to have an advantage in model testing (using the separated data for training and testing). The performance reproducibility of the GP model is good, so this approach appears suitable for modeling physical fabrication processes.

Index Terms—Gas furnace, modeling, estimation, neural network, genetic programming.

I. INTRODUCTION

INTEREST on modeling has been increasing for a long time in the several industries. Especially, to reduce expense and labor in experiments, the modeling techniques have been developed frequently in engineering fields. Nowadays, the system models are used for process optimization. Modeling types include white, gray, and black box models. A modeling method is determined according to the modeling purpose and the modeling performance can be different corresponding to the system feature. Thus, it is difficult to select the best modeling method for the better performance among the several methods. White box modeling such as mathematical and physical models shows the specific result in modeling performance, so it still remain an open question how to find the best modeling method among many data-driven methods. In this study, results of performance comparison are provided to evaluate both typical data-driven methods.

In the past, mathematical, theoretical, and physical methods have been actively applied to solar cell modeling. In these studies, curve fitting methods such as least square estimation were usually employed due to their ease and convenience [1]-[4]. In other early research, a statistical method was used. This approach gave good performance with uncertain measurement data [5]. In more recent research, soft computing methods such as genetic algorithms, neural networks, and neuro-fuzzy methods were broadly used in solar cell modeling. Soft computing methods are advantageous because detailed information and understanding of the system is not necessary for modeling using these approaches [6], [7].

Neural networks have been broadly applied for data driven modeling. Using this technique, the modeling results can be influenced significantly by the data conditions. For a physical process, it is often expensive and difficult to gather sufficient data to account for a variety of input conditions. Under these circumstances, it can be difficult to train accurate neural network models due to insufficient training data. To address this problem, a global method that has the ability to construct a data-driven model using data of limited quality and/or quantity is necessary. Genetic programming (GP) was selected as the global modeling method to be employed in this study. The performance of GP has been applied and confirmed in several application fields, including industrial, environmental, biological, and others [8].

In this study, typical data-driven methods for modeling applied broadly in several fields are implemented for performance evaluation using benchmark data such as gas furnace data. Parameter- and structure-based modeling methods are typical modeling types; therefore, the modeling performance can be compared by using the both methods. Process data used in this study are a benchmark data, which have been used in many studies, that is, gas furnace data. The process data include the non-linear feature for modeling performance.

The GP approach is a structural method that is less influenced by the characteristics and quality of data

Manuscript received September 6, 2008; revised December 1, 2008.

Hyeon Bae, Tae-Ryong Jeon, and Sungshin Kim are with the School of Electrical Engineering, Pusan National University, Busan 609-735, Korea (Tel: +82-51-510-2367, Fax: +82-51-513-0212, Email: sskim@pusan.ac.kr)

† Corresponding author

being modeled. Both the NN and GP methods were applied to design and evaluate the performance of models generated for gas furnace data from Box and Jenkins [9]. It was determined that GP has advantages in validation and reproducibility for this application.

II. DATA-DRIVEN MODELING METHODS

Artificial neural networks originated in the awareness that the process of computation in a human brain is quite different from that of a traditional digital computer. Neural networks also have the ability to organize neurons and calculate specific arithmetic problems faster than a digital computer [10], [11]. Neural networks have emerged as an alternative to physical models and statistical methods. Artificial neural networks possess many simple parallel processing units (called "neurons"), interconnected in such a way that knowledge is stored in the weight of the connections between them. The activation level of a neuron is determined by the weighted sum of its inputs, filtered by an activation function. This architecture endows the network with the ability to generalize with an added degree of freedom not available in statistical regression techniques.

In this study, neural networks were applied to model the Box and Jenkins gas furnace data. Modeling was performed for performance comparison between the neural network and the genetic programming methods.

Genetic programming is a powerful method for modeling. The output a genetic programming is another computer program. In essence, this is the beginning of computer programs that program themselves [12]-[14]. Genetic programming works best for several types of problems. The first type is where there is no ideal solution. For example, there is no single optimal solution to driving a car. Some solutions drive safely at the expense of time, while others drive faster with a safety risk. Therefore, driving a car consists of making compromises between speed and safety, as well as many other variables. In this case, genetic programming will find a solution that attempts to compromise and identify the most efficient solution from a large list of variables. Genetic programming is also useful in finding solutions in which the variables are constantly changing. In the car example, the GP might find one solution for a smooth concrete highway, while it will find a totally different solution for a rough unpaved road.

A. Neural networks for modeling

Figure 1 depicts a feed-forward, multilayered neural network. Individual layers in the network receive, process, and transmit critical information regarding the relationships between the input and output pairs [15]. In the solar cell process model, the input layer of neurons corresponds to the four adjustable input parameters that were varied in the contact formation experiment. The network also incorporates one or more "hidden" layers of neurons, which can be viewed conceptually as representing fundamental internal state variables of the process being modeled.

Feed-forward neural networks used for semiconductor process modeling are typically trained via the error back-propagation algorithm, with a sigmoidal activation function in each neuron. In this algorithm, the calculated output is compared to the measured data, and the squared difference between these two vectors determines the system error. This error is minimized using the gradient descent approach, which adjusts the network weights by an amount proportional to the partial derivative of the accumulated system error.

In process modeling, performance is influenced by both the number of hidden layers and the number of neurons in each layer [16]. In optimizing the neural process models, the number of hidden layers, number of hidden neurons, and training tolerance were each considered. These parameters were optimized iteratively by minimizing the root-mean-squared error (RMSE). The optimized neural network model had two hidden layers with four neurons each, one input layer with four neurons, and an output layer with a single neuron (a 4-4-4-1 structure). The optimized model was established using the Object-Oriented Neural Network Simulator (ObOrNNS) software tool, which was developed by the Intelligent Semiconductor Manufacturing group at the Georgia Institute of Technology [17].

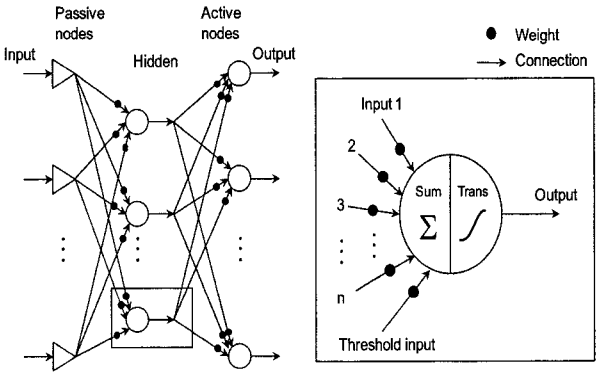


Fig. 1. Structure of the neural network [15].

B. Genetic programming for modeling

Genetic algorithms create a string of numbers that represent the solution to a given problem. GP uses four steps to solve problems, as shown in Figure 2.

1) Generate an initial population of random compositions of the functions and terminals of the problem (computer programs).
2) Execute each program in the population and assign it a fitness value according to how well it solves the problem.

3) Create a new population of computer programs by copying the best existing programs; creating new computer programs by mutation; and creating new computer programs by crossover (reproduction).
4) The best computer program that appeared in any generation, the "best-so-far solution," is designated as the result [12]-[14].

(1) Functions and terminals

The terminal and function sets are important components of genetic programming. They are defined as follows:

- The terminal set T is a set of variable atoms (sensors, detectors, state variables) or constant atoms (numbers, Boolean values, constants)
- The function set F is a set of arithmetic operations, mathematical functions, Boolean operations, conditional operators, functions causing iteration, functions causing recursion, or any other domain-specific functions

These components play an important role in the generation of trees in GP. In tree-based GP, we must construct syntactically valid trees. The main parameter for the initialization method is the maximum tree depth. This parameter is used to restrict the size of the initialized trees. Apart from the maximum tree depth parameter the initialization function needs a set of possible terminals and a set of possible internal nodes (non-terminals). For tree-based GP, there are two common methods to construct trees: the grow method and the full method [18]. The terminal set consists of variables and constants. In this study, the function is selected by the user, and the terminals are organized by the data variables and numerical values that are randomly selected within a specific range. Both the terminal and function sets are changed by subsequent iterative operations.

(2) Initial structures

GP restricts the selection of the label for the root of the tree to the function set F (i.e., +, -, *, /, ^2). In the initial step of the algorithm, a hierarchical structure is generated. This structure creates a random program tree with the combined set, C=F∪T. It is initially assumed that combination C is defined by function set F {+, -, *, /, ^2} and terminal set T consisting of variables and constants.

The depth of the tree is defined as the length of the longest non-back-tracking path from its root to its endpoint. The depth is determined by initial random population methods such as the full, grow, and ramped half and half methods. The full method ensures that every non-back-tracking path in the tree is equal to a certain length. This is enforced by allowing only function nodes to be selected for all depths up to a maximum depth minus 1, and by selecting only terminal nodes at the lowest level. With the grow method, we create variable length paths by allowing a function or terminal to be placed at any level up to a maximum depth minus 1. In this study, the ramped half-and-half method was applied. In the ramped half-and-half method, we create trees using a variable depth from 2 up to the maximum depth. For each depth of tree, half are created using the full method, and the other half are created using the grow method. This is a mixed method that incorporates both the full and grows methods. For each branch of the tree, half is created using the full method, and the other half is created using the grow method. The ramped half and half method creates trees having a wide variety of sizes and shapes [12].

(3) Fitness function

The most important concept of genetic programming is the fitness function. The fitness function determines how well a program is able to solve the problem. Fitness is the driving force in Darwinian natural selection. Raw fitness, standardized fitness, adjusted fitness, and normalized fitness have been usually used in GP. In this study, the following adjusted fitness function was applied:

$$fitness = \frac{1}{1+error} \tag{1}$$

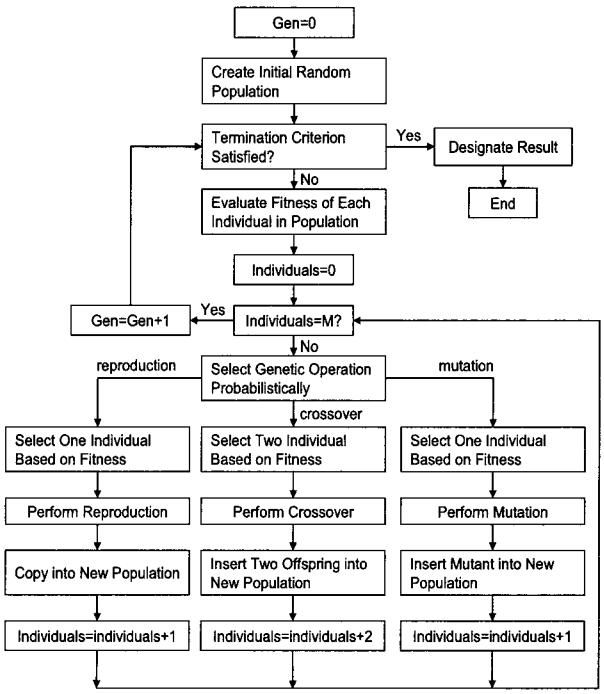


Fig. 2. Genetic programming flowchart.

(4) Genetic programming operations

The primary GP operators are crossover, reproduction, and mutation. Reproduction is just taking a copy of a good individual and placing it in a new population. Crossover takes two parents and creates two new

children by selecting a random point in each parent and swapping their sub-trees. Mutation randomly changes a portion of an individual solution.

(a) Selection

The four selection methods that have been employed in GP include fitness-proportionate selection, rank selection, tournament selection, and greedy over-selection. In this study, the greedy over-selection method was used. Koza made use of greedy over-selection [19] to reduce the number of generations required for a GP run. In this approach, individuals are selected based on their performance, but this method biases the selection towards the highest performers. All individuals are assessed, and their fitness values are calculated.

(b) Crossover

Koza considers crossover and reproduction to be the foremost genetic operations. Similar to its performance under genetic algorithms, crossover operates on two programs, and produces two child programs as shown in Fig. 3. Two random nodes are selected from within each program and then the resultant sub-trees are swapped, generating two new programs.

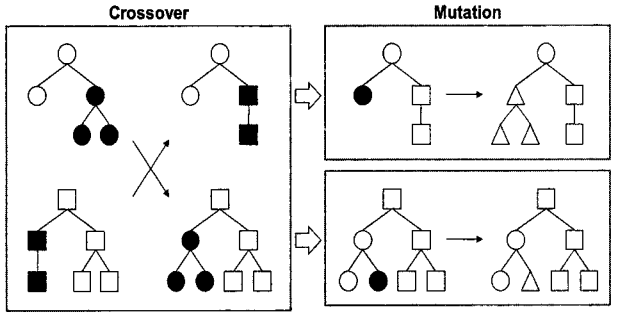


Fig. 3. Crossover and mutation for GP.

(c) Reproduction

Reproduction is performed by simply copying a selected member from the current generation to the next generation. In genetic programming, when an individual incestuously mates, the two resulting offspring will, in general, be different. The Darwinian reproduction operation [19] creates a tendency toward convergence; however, in GP, the crossover operation exerts a counterbalancing pressure away from convergence.

(d) Mutation

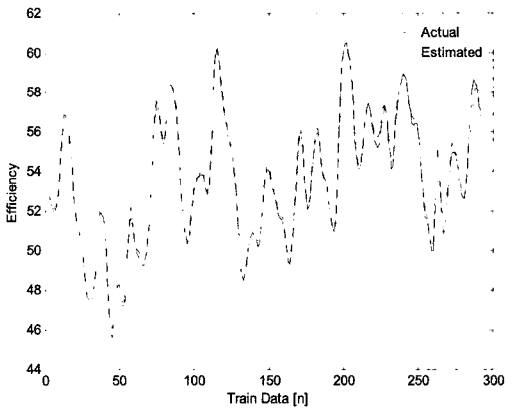
With genetic algorithms, mutation is an important operator that provides diversity in the population. However, mutation is relatively unimportant in the GP environment because the dynamic sizes and shapes of the individuals in the population already provide diversity, and as stated above, the population should not converge. Thus, mutation can be considered as a variation on the crossover operation.

III. PERFORMANCE COMPARISON : NNS AND GP

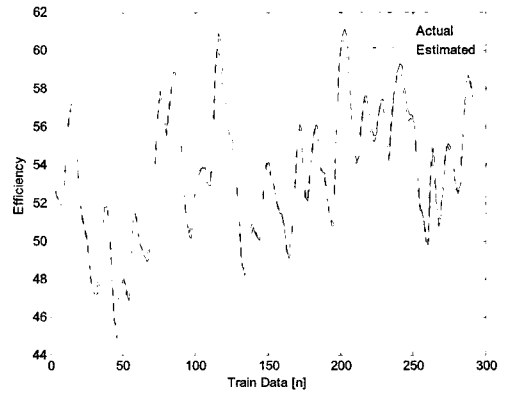
In this section, the performance of GP is compared to the performance of neural networks based on the gas furnace data. In past research [9], the entire set of gas furnace data was used for model training, and the resulting model was compared to models produced by other modeling methods. In this study, however, to evaluate the model performance based on training and testing conditions, the original gas furnace data were separated to training and testing sets. The modeling results were generated using the ObOrNNS neural network tool developed and operated by the Intelligent Semiconductor Manufacturing group at the Georgia Institute of Technology [17].

Table 1. Modeling results of NN and GP using all data

Modeling Method	Inputs	Data	RMSE
Neural Network	$y(t\text{-}1)$, $u(t\text{-}4)$	292	0.3913
Genetic Programming	$y(t\text{-}1)$, $u(t\text{-}4)$	292	0.6803



(a) Result of neural network



(b) Result of genetic programming

Fig. 4. Result of NN and GP using two inputs.

A. Modeling results of NN and GP using all data

In time-series modeling, it is very important to select the useful inputs for performance improvement. In this study, model inputs were determined based on referring the past research of gas furnace modeling. y(t-1) and u(t-4) were selected for model inputs. The result of this simulation is generated by using 292 data that are all process data. In the traditional research, modeling performance was compared by using all data that is the same with this study. As shown in Table 1, the result of the neural network model is better than that of the genetic programming. This result is caused by the modeling feature of the neural network, that is, neural network modeling shows available results in model training.

B. NN and GP modeling with a 7/3 data ratio

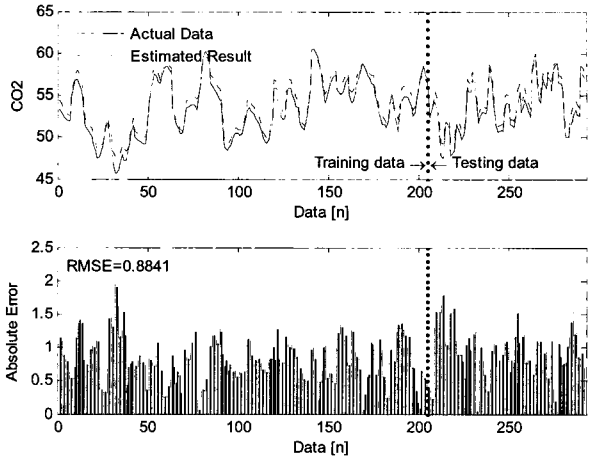
To evaluate and compare modeling performance, neural networks and genetic programming were applied to modeling the benchmark gas furnace data of Box and Jenkins [9]. For a fair comparison, the data were separated to training and testing sets under the same conditions. In the simulation, the input variable, u(t), and the output variable, y(t), represent the gas flow rate and the CO2 concentration, respectively. Two time-series inputs were used for model generation [20]. Past studies suggest that the best performance in gas furnace modeling can be obtained by using both inputs.

In previous modeling of the gas furnace data, there was no testing or evaluating result. Thus, the modeling results that were derived by using the modeling's own tools could not be compared directly with those results. In this simulation, the gas furnace data were separated into training/testing sets with a 7/3 ratio. The training data consisted of 205 data vectors, and the testing data consisted of 87 vectors. The structure of the network that was used had two input layer nodes, four nodes in the first hidden layer, four nodes in the second hidden layer, and one output layer node.

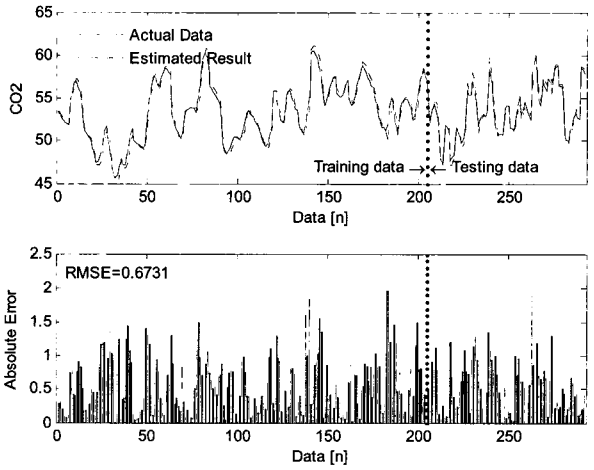
The modeling results are shown in Table 2. The GP results were better than those obtained using the neural network. The RMSE of the neural network model and the genetic programming models are 0.8841 and 0.6731 [% CO2], respectively. Figure 5 graphically depicts the training and testing results for the neural network and the genetic programming models, respectively. The upper graphs show the estimation results, and the lower graphs show the absolute errors. As shown in this figure, GP provides better performance in model training and testing. We believe this result is caused by the data selection method used in this simulation, where the data for training and testing were separated sequentially even though the original data was time-based. This approach facilitates a more reasonable estimation by GP because the training and testing errors that were obtained by using GP and lower than those obtained by using NNs.

Table 2. Results of the NN and GP using a 7:3 data ratio

Data	Inputs	Data	NN	GP
Training	$y(t\text{-}1), u(t\text{-}4)$	205	0.8336	0.7072
Testing	$y(t\text{-}1), u(t\text{-}4)$	87	0.8841	0.6731



(a) Estimation result and absolute error of NN



(b) Estimation result and absolute error of GP

Fig. 5. Result of GP using the separated data.

IV. CONCLUSIONS

This study was performed to compare process modeling using neural networks and genetic programming using gas furnace data. Modeling for such industrial processes is challenging, as data collection can be costly. Moreover, the reproducibility of the experimental data is a concern; that is, experimental results are often not reproduced even though the same process conditions are applied. Thus, neural network-based models can be generated with small training errors, but over-fitting can occur. To solve this problem, genetic programming has been employed. GP modeling was shown to be less susceptible to over-fitting given the limited amount of available training data. Therefore, this proposed approach appears suitable for semiconductor process modeling.

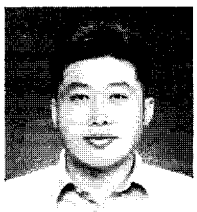
This study serves as a starting point for process optimization. In future work, optimized process conditions will be determined based on the process models developed in this study.

ACKNOWLEDGMENT

This work was supported for two years by Pusan National University Research Grant.

REFERENCES

[1] S. Mottet, "Solar cell modeling for computer-aided generator design and irradiation degradation computations," In ESA Photovoltaic Generators in Space, pp. 177-186, June 1980.
[2] G. Agostinellia, D. L. Batznerb, and M. Burgelmanc, "A theoretical model for the front region of cadmium telluride solar cells," Thin Solid Films, vol. 431-432, no. 1, pp. 407-413, May 2003.
[3] S. Michael, A. D. Bates, and M. S. Green, "Silvaco ATLAS as a solar cell modeling tool," Conference Record of the Thirty-first IEEE Photovoltaic Specialists Conference, no. 3-7, pp. 719-721, Jan. 2005.
[4] J. Appelbaum, A. Chait, and D. Thompson, "Parameterization of Solar Cells," NASA STI/Recon Technical Report N, Oct. 1992.
[5] D. C. Marvin, "Solar cell modeling and simulation," NASA STI/Recon Technical Report N, Feb. 1988.
[6] Joseph A Jervase, Hadj Bourdoucen and Ali Al-Lawati, "Solar cell parameter extraction using genetic algorithms," Measurement Science and Technology, vol. 12, pp. 1922-1925, 2001.
[7] M. AbdulHadi, A. M. Al-Ibrahim, and G. S. Virk, "Neuro-fuzzy-based solar cell model," IEEE Transaction on Energy Conversion, vol. 19, no. 3, pp. 619-624, Sept. 2004.
[8] www.genetic-programming.org
[9] G. E. P. Box and G. M. Jenkins, Time Series Analysis: Forecasting and Control. San Francisco, Holden-Day, 1976.
[10] Simon Haykin, Neural Networks. NJ: Prentice Hall, pp. 6-67, 1999.
[11] Lefteri H. Tsoukalas and Robert E. Uhrig, Fuzzy and Neural Approaches in Engineering. New York: John Wiley & Sons, Inc., pp. 238-246, 1997.
[12] John R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. MA: The MIT Press, December 11, 1992.
[13] John R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs. MA: The MIT Press, May 17, 1994.
[14] John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane, Genetic Programming III: Darwinian Invention and Problem Solving. CA: Morgan Kaufmann, May 15, 1999.
[15] D. Hammerslrom, "Working with neural networks," IEEE Spectrum, pp. 46-53, July 1993.
[16] B. Kim and G. May, "An Optimal Neural Network Process Model for Plasma Etching," IEEE Transaction on Semiconductor Manufacturing, vol. 7, pp. 12-21, Feb. 1994.
[17] Cleon Davis, Sang Jeen Hong, Ronald Setia, Rana Pratap, Terence Brown, Benjamin Ku, Greg Triplett, and Gary S. May, "An Object-Oriented Neural Network Simulator for Semiconductor Manufacturing Applications," IIIS 8th Ann. Multi-Conf. Systemics, Cybernetics Informatics 5, pp. 356-370, 2004.
[18] Matthew Walker, "Introduction to Genetic Programming," Technical report, Oct. 7, 2001.
[19] David E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Kluwer Academic Publishers, Boston, MA, 1989.
[20] Wael Farag and Ahmed Tawfik, "On Fuzzy Model Identification and the Gas Furnace Data," Proceedings of the IASTED International Conference Intelligent Systems and Control 2000, August 14-16, 2000.



Hyeon Bae (M'2006) received the B.S. degree from Gyeongsang National University, and M.S. and Ph.D. degrees from Pusan National University in Electrical Engineering, in 1999, 2001, and 2005, respectively. He was a post-doc at the School of Electrical and Computer Engineering of Georgia Institute of Technology in 2008. His research interests include intelligent control and system, fuzzy logic, process optimization, fault detection and diagnosis, and data mining.

Taeryong Jeon (M'2007) received the B.S. degree in computer science engineering from Silla University in 2007. He is currently pursuing the M.S. degree in the School of Electrical Engineering at Pusan National University. His research interests include data mining and intelligent control.

Sungshin Kim (M'2006) received the B.S. and M.S. degrees from Yonsei University, and the Ph.D. degree in electrical engineering from Georgia Institute of Technology, Atlanta, in 1984, 1986, and 1996, respectively. He is currently an Associate Professor in the School of Electrical Engineering, Pusan National University. His research interests include intelligent control, fuzzy logic control, manufacturing systems, and data mining.