Integrating Design Stage of Fuzzy System using Genetic Algorithms

Lee, Michael A.
Computer Science Division, University of California

Takagi, Hideyuki
Computer Science Division, University of California

Integrating Design Stages of Fuzzy Systems using Genetic Algorithms[1]

Michael A. LEE and Hideyuki TAKAGI*

Computer Science Division, University of California, Berkeley, CA 94720 USA
lee@cnmat.berkeley.edu, takagi@diva.berkeley.edu, *FAX (510)642-5775

Abstract— **This paper proposes an automatic fuzzy system design method that uses a Genetic Algorithm and integrates three design stages; our method determines membership functions, the number of fuzzy rules, and the rule-consequent parameters at the same time. Because these design stages may not be independent, it is important to consider them simultaneously to obtain optimal fuzzy systems. The method includes a genetic algorithm and a penalty strategy that favors systems with fewer rules. The proposed method is applied to the classic inverted pendulum control problem and has been shown to be practical through a comparison with another method.**

1

1 Introduction

Fuzzy systems have become popular components of consumer products because they are inexpensive to implement, able to solve difficult non-linear control problems, and exhibit robust behavior. Designers are especially attracted to fuzzy systems because fuzzy systems allow them to capture domain knowledge quickly using rules that contain fuzzy linguistic terms. These attributes allow products with embedded fuzzy systems to be both cost effective and high performance.

While it is easy to describe human knowledge with fuzzy linguistic terms, it is not easy to define the terms by membership functions. In addition, fuzzy system design requires two other stages: determining the number of rules and determining the rule-consequent parameters. Some papers propose automatic methods using neural networks [5, 6, 7, 18], fuzzy clustering [4], genetic algorithms [8, 9, 10, 11, 12, 14, 16, 17, 22], or gradient methods[1, 13, 15]. Although these methods produce systems that perform better than systems designed by humans, they may be suboptimal because they treat only one or two of the three design stages.

This paper proposes an automatic fuzzy system design method that uses a Genetic Algorithm and integrates three design stages; our method determines membership functions, the number of fuzzy rules, and the rule-consequent parameters at the same time. As a sample fuzzy system, we use the Takagi-Sugeno-Kang (TSK) fuzzy model [19]. Rules in a TSK fuzzy model use traditional fuzzy variables for antecedents. However, the consequent values are computed by summing weighted combinations of the input values. We have formulated a TSK fuzzy model representation that parameterizes membership function shape and position and rule-consequent parameters. By combining our representation with the target application's boundary conditions, we can represent fuzzy systems with different numbers of rules. A genetic algorithm operates on this representation and optimizes the fuzzy system parameters with respect to performance and resource requirements. We chose a genetic algorithm optimization technique because genetic algorithms because genetic algorithms are robust, search many points simultaneously, and able to avoid local minima.

In the following sections we briefly review genetic algorithms and automatic fuzzy system design research. Next we discuss our TSK fuzzy model, how we incorporated genetic algorithms into the design process, and parameters of our design method. We demonstrate our method by deriving a four rule fuzzy system that balances an inverted pendulum. We conclude by comparing the performance of our controller with a controller derived by Another method.

2 Review

2.1 Genetic algorithms

A genetic algorithm is a probabilistically guided optimization technique modeled after the mechanics of genetic evolution. Unlike many classical optimization techniques, genetic algorithms do not rely on computing local derivatives to guide the search process. Genetic algorithms also include random elements, which helps avoid getting trapped in local minima.

Genetic algorithms explore a population of solutions in parallel. The size of the population is a free parameter, which trades off coverage of the search space against the time required to compute the next generation. Each solution in the population is

*The author is a Visiting Industrial Fellow at the University of California at Berkeley and a Senior Researcher for Central Research Laboratories, Matsushita Electric Industrial Co., Ltd.

[1]This paper appeared in the Proc. of the 2nd Int'l Conf. on Fuzzy Systems (FUZZ-IEEE'93), Vol.1, pp. 612-617 (1993)

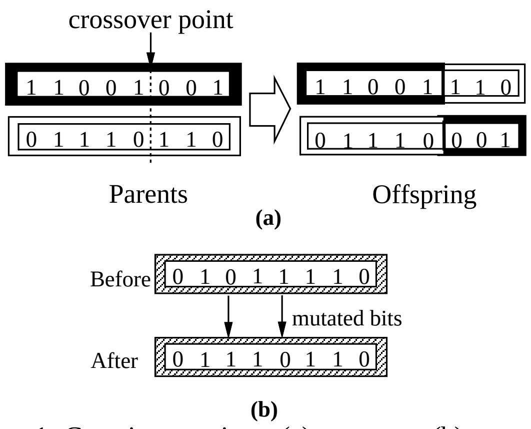

Figure 1: Genetic operations: (a) cross over (b) mutation

coded as a binary string or gene, and a collection of genes forms a generation. A new generation evolves by performing genetic operations, such as reproduction, crossover, and mutation, on genes in the current population and then placing the products into the new generation.

In a simple genetic algorithm, operations are performed in the following order; reproduction, crossover, and then mutation. Reproduction involves selecting two parent genes from the current population. Selection is based probabilistically on a gene's fitness value; the higher the fitness of a gene, the more likely it can reproduce. After selecting two parents, crossover is performed according to a crossover probability. If crossover is to be performed, offspring are constructed by copying portions of parent genes designated by random crossover points (single point crossover shown in Figure 1). Otherwise, an offspring copies its entire gene from one of the parents. As each bit is copied from parent to offspring, the bit has the probability of flipping, or mutating. Mutation is believed to help reinject any information that may have been lost in previous generations [3]. Variations of these operators are discussed in [2].

A gene's fitness is evaluated by decoding the gene's binary representation and then passing it through a fitness function. The fitness function is a means of ranking solutions in the population and can include penalty terms in addition to raw performance measures. In our method, we included a penalty strategy that favors production of systems with fewer membership functions and rules. More details are given in section 3. (See [3] for extensive discussions on constructing genetic objective functions.)

2.2 Automatic design of fuzzy systems

Several papers, mentioned in section 1 , have proposed automatic design methods. Much of the work has focused on tuning membership functions. For example, [18] uses neural networks as a membership values generator and [15] treats fuzzy systems as networks and use back-propagation techniques to adjust membership functions. Nodes in these networks perform parameterized functions. These parameters are tuned by computing derivatives of the network, with respect to these parameters, and then back-propagating the error as in traditional neural networks.

Other methods use genetic algorithms to determine the number of fuzzy rules [9, 20]. Karr has developed a method for determining membership functions and number of fuzzy rules using genetic algorithms [9]. In this paper, Karr's method first uses a genetic algorithm to determine the number of rules according to a predefined rule base. Following this stage, the method uses a genetic algorithm to tune the membership functions. Our method differs from Karr's in that we perform both operations simultaneously, as opposed to sequentially, and we include a penalty strategy that favors systems with fewer rules.

Although many of the proposed methods offer an improvement on human designed fuzzy systems, they usually combine only one or two of the design stages. Because these design stages may not be independent, it is important to consider them simultaneously to find the optimal solution.

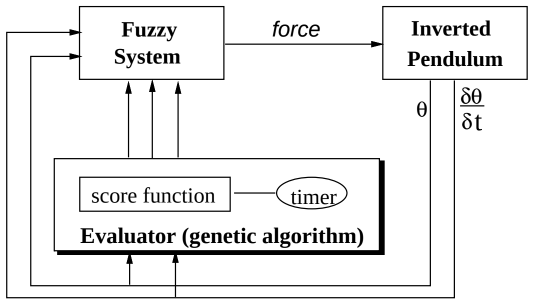

Figure 2: System diagram

3 Experimental Environment and Representations

The goal of our work is to develop an automatic fuzzy system design that uses minimal knowledge of the system to be controlled. As a sample fuzzy system, we chose the TSK fuzzy model, which is widely used in actual applications. In this section we first introduce our TSK fuzzy model representation used in our experiments. Second we present the inverted pendulum application used to illustrate our technique. Lastly we present our method for evaluating a fuzzy system's performance in our application context. A system diagram showing the relationship between the components discussed in this section is given in Figure 2.

3.1 Fuzzy system representation for automatic design

We based our fuzzy system on the TSK fuzzy model. In this model the input variables are traditional fuzzy sets, however output variables are computed from linear combinations of the input values. For example, a typical rule in a TSK fuzzy system might be:

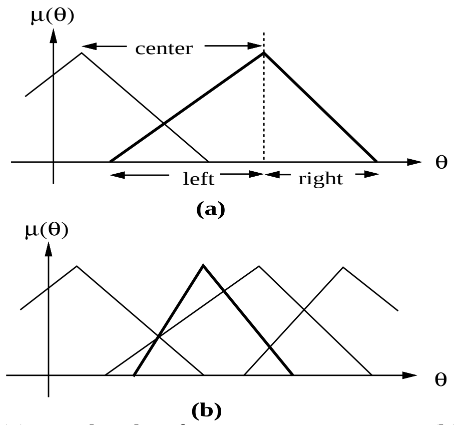

Figure 3: (a) membership function representation (b) possible member functions

$$\text{IF X is A and Y is B THEN C} = w_1\text{X} + w_2\text{Y} + w_3$$

where w_n are rule-consequent parameters.

The representation we formulated provides the flexibility to parameterize membership functions, number of fuzzy rules, and consequent parameters. Each membership function is triangular and parameterized by left base, right base, and distance from the previous center point (see Figure 3(a)). By encoding the centers as a distance from the previous center (the first center was given as an absolute position) and the base values as the distance from the corresponding center point (see Figure 3(a)), we can use the boundary conditions of the application to eliminate necessary membership functions, which has the direct effect of eliminating rules. For example, all θ membership functions sets with center positions greater than 90° can be eliminated.

Each membership function requires three parameters, and the consequent part of each fuzzy rule requires three parameters. Consider a two dimensional input space: one input dimension fuzzy-partitioned into m and the other into n. The number of membership functions are $m + n$ and the number of rules are nm. The total number of system parameters is $3(m+n)+3mn$.

Unlike most other methods, overlap restrictions were not placed on the sets in our system and the possibility of complete overlap existed (see Figure 3(b)). The final output values are computed by summing the weighted output of each rule according to its firing strength.

3.2 *Inverted pendulum*

For experimental purposes, we applied our method to the inverted pendulum problem. The inverted pendulum represents a classic non-linear control problem that can be described as the task of balancing a pole on a movable cart. In this simulation, the movement of both the pole and the cart is restricted to the vertical plane. The state of the system is described by the pole's angle and angular velocity (in this simulation, the cart is

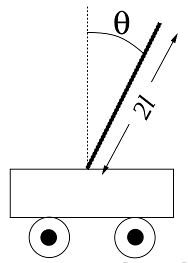

Figure 4: Inverted pendulum

allowed to move infinitely in the left or right direction). The controller can exert only a constant force on the cart in either the left or right direction. The objective of the controller is to balance the pole as quickly as possible (see Figure 2 for system diagram).

The state equations for the inverted pendulum can be expressed as [6]:

$$\begin{aligned}
\theta &= \frac{\delta\theta}{\delta t} \\
\ddot{\theta} &= H_2(\theta, \theta, force) = \frac{g\sin(\theta) + \cos(\theta)\left(\frac{-force - ml\theta^2\sin(\theta)}{m_c + m}\right)}{l\left(\frac{4}{3} - \frac{m\cos^2(\theta)}{m_c+m}\right)}
\end{aligned}$$

where g is 9.8 meters/sec^2, m_c is the mass of the cart, m is the mass of the pole, l is half the length of the pole, and $force$ is the applied force in Newtons.

In this simulation, the equations of motion are defined by differential equations. A two-step forward Euler integration can be used to approximate its state at $t + h$:

$$\begin{aligned}
\theta(t + \frac{h}{2}) &= \frac{h}{2}\theta(t) + \theta(t) \\
\theta(t + \frac{h}{2}) &= \frac{1}{2}H_2(\theta(t), \theta(t), force) + \theta(t) \\
\theta(t + h) &= \frac{h}{2}\theta(t + \frac{h}{2}) + \theta(t + \frac{h}{2}) \\
\theta(t) &= \frac{1}{2}H_2(\theta(t + \frac{h}{2}), \theta(t + \frac{h}{2}), force) + \theta(t + \frac{h}{2})
\end{aligned}$$

where h is the time step.

3.3 *Evaluating fuzzy system performance*

Unlike the fuzzy system representation, the evaluation function relies directly on the application. In this experiment, the function must be capable of ranking the fuzzy systems in the context of the inverted pendulum task. The objective of controlling an inverted pendulum is to balance it in the shortest amount of time for a wide range of initial conditions. To evaluate our fuzzy systems, we tried the fuzzy system on the inverted pendulum starting with eight different initial conditions (positions of Table 1 and their symmetric positions). Each trial terminated

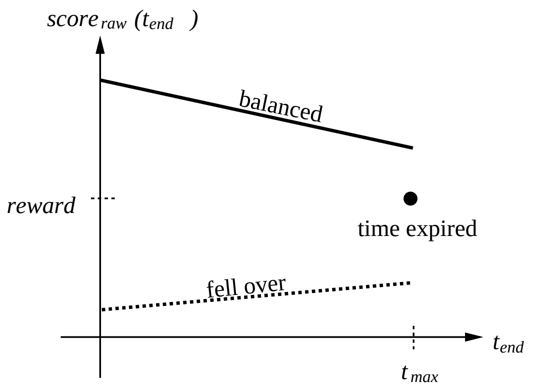

Figure 5: Raw scoring functions

under one of the following three conditions: either the pole fell over, time expired, or the system balanced the pole (using some epsilon criteria). Depending on the termination condition, we scored a trial in the following manner (see Figure 5):

$$score(t_{end}) = \begin{cases} a_1\,(t_{max} - t_{end}) + a_2 reward & (1) \\ reward & (2) \\ b \cdot t_{end} & (3) \end{cases}$$

where a_1, a_2, b and $reward$ are constants, and (1) pole balanced, (2) $t_{max} = t_{end}$, and (3) pole fell over ($|\theta| \geq 90°$).

The general idea is that if the system balances the pole, a shorter time is better than a longer time. If the pole falls over, we recognize potential success and credit the system according to the time it kept it from falling. We then added additional terms to consider steady state error and to penalize systems according to the number of rules in the system. The resulting fitness score for one trial was computed as follows:

$$score(t_{end}) = \frac{\left(score_{raw}\,(t_{end}) + c\sum_{0}^{t_{end}} |\theta_t|\right)}{\text{number of rules} + \text{offset}_{\text{rules}}}$$

The steady state error was a simple summation of the pole angle displacement and weighted with constant c. The $\text{offset}_{\text{rules}}$ parameter controls the degree of penalty for number of rules. The scores from the eight trials were accumulated to form a composite score, which was used as a controller's overall fitness. We varied the $\text{offset}_{\text{rules}}$ parameter in our experiments and results are discussed in section 6.

4 Incorporating Genetic Algorithms into Fuzzy System Design

To incorporate genetic algorithms into fuzzy system design, we must find a suitable genetic coding and determine a method to evaluate its fitness. An evaluation method was discussed in section 3.3 and can be used directly to determine fitness values. To address the coding problem, we first define a chromosome

center	left base	right base
10100110	10011000	01011000

membership function chromosome (MFC)

w_1	w_2	w_3
10100110	10011000	01011000

rule-consequent parameters chromosome (RPC)

Figure 6: Composite chromosomes

as a set of parameters that represent a higher level entity, such as a membership function or rule-consequent parameters set (see Figure 6). By linking these chromosomes together, we can form the entire fuzzy system representation (see Figure 7). In our experiments, all parameter values are encoded as 8 bit numbers.

The maximum number of fuzzy sets per input variable was set to ten in our experiments. This limit was set through experience with the inverted pendulum application. Because we included a penalty strategy that involves the number of rules, setting this number is not so critical. There is no danger in setting this to an arbitrarily large number (with the exception of consuming excessive computing resources). The resulting genetic representation for fuzzy systems used in our experiments consisted of 360 parameters or 2880 bits (see Figure 6 and 7).

5 Experimental Results

Our method combines a genetic algorithm, a penalty strategy, and unconstrained membership function overlap to automatically design fuzzy systems. In this section, we present results of our method applied to the inverted pendulum problem.

In our experiments, we used a genetic algorithm with two-point crossover and mutation operators. Population size was set to 10 and crossover and mutation probabilities were 0.6 and 0.0333 respectively. We also used an elitist strategy in which the member with the highest fitness value automatically advanced to the next generation. All members were initialized with random values in most experiments. In some experiments we used apriori knowledge to initialize one member of the population with seven uniformly spaced fuzzy sets for both θ and $\delta\theta/\delta t$. Each of its 49 rules were initialized with $w_1 = w_2 = 0$ and w_3 equal to a value computed using the center points of the antecedent membership functions and the control law: force = $c_1 \sin(\theta) + c_2 \frac{\delta\theta}{\delta t}$.

The automatic design process was initiated by first setting the $\text{offset}_{\text{rules}}$ parameter and then letting the genetic algorithm generate 5000 generations of solutions. The best solution was kept and the rest were discarded. In one experiment with the $\text{offset}_{\text{rules}}$ parameter set to 100, the method produced a symmetric system with only four rules. Figure 8 shows the fitness level as a function of generation for this experiment comparing

fuzzy variable θ			fuzzy variable $\delta\theta/\delta\tau$			rule-consequent parameters		
MFC_1	$\cdots$	MFC_{10}	MFC_1	$\cdots$	MFC_{10}	RPC_1	$\cdots$	RCP_{100}

Figure 7: Gene map

Table 1: Initial conditions of pendulum

θ	5.22	5.11	-8.41	6.22
$\delta\theta/\delta t$	6.93	6.97	-1.37	-7.14

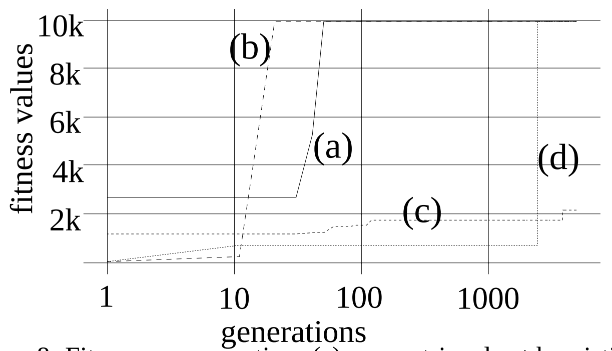

Figure 8: Fitness vs. generation: (a) symmetric rules + heuristic initialization, (b) symmetric rules + random initialization, (c) asymmetric rules + heuristic initialization, and (d) asymmetric rules + random initialization.

combinations of heuristic initialization and structural symmetry constraints. Figures 9 and 10 show trajectory plots for several initial conditions using this system. The symmetric rules are:

IF θ is A_i and $\frac{\delta\theta}{\delta t}$ is B_i THEN $y = w_{1i}\theta + w_{2i}\frac{\delta\theta}{\delta t} + w_{3i}$, where $i = 1 \sim 4$. The obtained parameters in four consequent parts were (w_{1i}, w_{2i}, w_{3i}) = (0.44, 1.02, -31.65), (1.54, -0.61, -30.14), (1.54, -0.61, 30.14), and (0.44, 1.02, 31.65). The obtained triangular membership functions, A_i and B_i were A_1 = A_3 = {-119.65, -62.12, 4.59}, A_2 = A_4 = {-4.59, 62.12, 119.65}, B_1 = B_3 = {-219, -1.99, 238.56}, and B_2 = B_4 = {-238.56, 1.99, 219.64}.

6 Discussion

Our results show that our proposed fuzzy system design method can automatically determine fuzzy system membership functions, number of rules, and consequent parameters simultaneously. In one experiment, the resulting system balanced a pendulum for all initial angles in about 0.4 seconds. This performance compares favorably with a system produced by Jang [6, 7]. Both methods produce fuzzy systems that have four

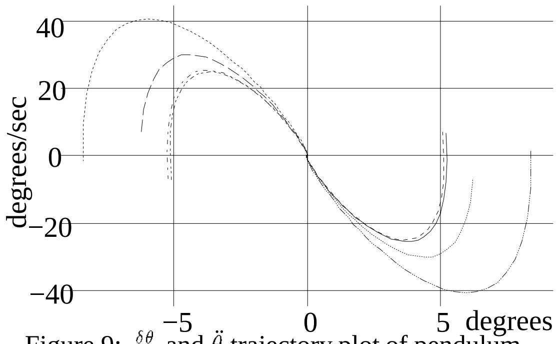

Figure 9: $\frac{\delta\theta}{\delta t}$ and $\ddot{\theta}$ trajectory plot of pendulum

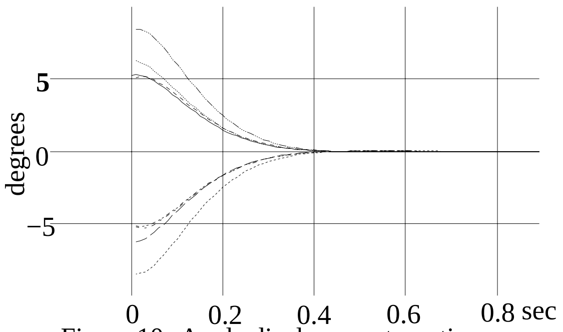

Figure 10: Angle displacement vs. time

rules, however our system automatically determines the number of rules while Jang sets the rule number by hand.

Although the inverted pendulum involved only two input and one output variable, our design method can be extended to handle problems of higher dimension. The mechanics of the genetic algorithm remains unchanged, only the fitness function evaluation changes.

In our experiments we also found that the time to find a solution that balances all initial conditions was not correlated with the rule penalty weight. In some cases, the experiments with higher rule penalty settings found solutions before experiments with lower penalties settings. This may be due to initial condition effects due to the small population or due to the nature of the application. The sensitivity of our method to penalty settings must be studied in more detail.

A large percentage of the computing resources is devoted to evaluating population members simulating the pendulum. The time to perform the genetic operations is small compared to time to simulate the pendulum under the eight different initial conditions. If the epsilon criteria for determining whether the pendulum was balanced were relaxed, some trials might require fewer time steps.

7 Conclusions and Further Research

We have proposed a method for automatically designing complete fuzzy systems. Our method uses a genetic algorithm and a penalty strategy to determine membership function shape and position, number of fuzzy rules, and consequent parameters simultaneously. Our experimental results demonstrate the practicality of our method, by producing systems that perform comparably to a system produced by another method.

Other extensions to this work that need to be explored include applying this method to more complex tasks, directly comparing results with a sequential method, applying this method to other types of fuzzy systems, and eliminating unnecessary rules by considering overlap.

Acknowledgments

This research is supported in part by NASA Grant NCC-2-275, MICRO State Program Award No.90-191, and EPRI Agreement RP8010-34. We also would like to thank Prof. David Wessel and the Center for New Music and Audio Technologies at UC Berkeley for use of computing resources.

References

[1] Araki, S., Nomura, H., Hayashi, I., and Wakami, N., "Self-generating method of fuzzy inference rules", Int'l Fuzzy Engineering Symposium (IFES'92), 1992, pp.1047-1058

[2] Davis, L., ed. Handbook of Genetic Algorithms, Van Nostrand, Reinhold, 1991

[3] Goldberg, D., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989

[4] Hirota, K. and Yoshinari, Y., "Identification of Fuzzy Control Rules Based on Fuzzy Clustering Method", 5th Fuzzy System Symposium, June, 1989, pp.253-258 (in Japanese)

[5] Ichikawa, R., Nishimura, K., Kunugi, M., Shimada, K., "Auto-Tuning Method of Fuzzy Membership Functions Using Neural Network Learning Algorithm," Proc. of the 2nd Int. Conf. on Fuzzy Logic and Neural Networks (IIZUKA'92), 1992, pp.345-348

[6] Jang, R, "Fuzzy Controller Design without Domain Experts," Proc. IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'92), 1992, pp. 289-296

[7] Jang, R, "Self-Learning Fuzzy Controllers Based on Temporal Back Propagation," IEEE Trans. on Neural Networks, Vol.3, No.5, 1992, pp.714-723

[8] Karr, C., Freeman, L., Meredith, D., "Improved Fuzzy Process Control of Spacecraft Autonomous Rendezvous Using a Genetic Algorithm," Proc. of the SPIE Conf. on Intelligent Control and Adaptive Systems, Philadelphia, PA, Nov., 1989, pp.274-283

[9] Karr, C., "Applying Genetics to Fuzzy Logic," AI Expert, Vol.6, No.2, 1991, pp.26-33

[10] Karr, C., "Design of an Adaptive Fuzzy Logic Controller using a Genetic Algorithm," Proc. of the Int. Conf. of Genetic Algorithms (ICGA'91), 1991, pp. 450-457

[11] Karr, C., Gentry, E., "A Genetics-Based Adaptive pH Fuzzy Logic Controller," Proc. of the Int. Fuzzy Systems and Intelligent Control Conf. (IFSICC'92), Louisville, KY, 1992, pp.255-264

[12] Karr, C., Sharma, S., Hatcher, W., Harper, T., "Control of an Exothermic Chemical Reaction using Fuzzy Logic and Genetic Algorithms," Proc. of the Int. Fuzzy Systems and Intelligent Control Conf. (IFSICC'92), Louisville, KY, 1992, pp.246-254

[13] Katayama, R., Kajitani, Y., Nishida, Y., "A Self Generating and Tuning Method for Fuzzy Modeling using Interior Penalty Method," Proc. of the 2nd Int. Conf. on Fuzzy Logic and Neural Networks (IIZUKA'92), 1992, pp.349-352

[14] Nishiyama, T., Takagi, T., Yager, R., and Nakanishi, S., "Automatic Generation of Fuzzy Inference Rules by Genetic Algorithm", 8th Fuzzy System Symposium, 1992, pp.237-240 (in Japanese)

[15] Nomura, H., Hayashi, I., and Wakami, N., "A self-tuning method of fuzzy control by descent method", 4th IFSA Congress, Vol. Engineering, July, 1991, pp.155-158

[16] Nomura, H., Hayashi, I., Wakami, N., "A Self-Tuning Method of Fuzzy Reasoning By Genetic Algorithm," Proc. of the Int. Fuzzy Systems and Intelligent Control Conf. (IFSICC'92), Louisville, KY, 1992, pp.236-245

[17] Qian, Y., Tessier, P., Dumont, G., "Fuzzy Logic Based Modeling and Optimization," Proc. of the 2nd Int. Conf. on Fuzzy Logic and Neural Networks (IIZUKA'92), 1992, pp.349-352

[18] Takagi, H. and Hayashi, I., "NN-driven Fuzzy Reasoning", Int'l J. Approximate Reasoning (Special Issue of IIZUKA'88), Vol.5, No.3, 1991, pp.191-212

[19] Takagi, T. and Sugeno, M., "Fuzzy Identification of Systems and Its Applications to Modeling and Control", IEEE Trans. SMC-15-1, 1985, pp.116-132

[20] Takahama, T., Miyamoto, S., Ogura, H., and Nakamura, M., "Acquisition of Fuzzy Control Rules by Genetic Algorithm", 8th Fuzzy System Symposium, 1992, pp.241-244 (in Japanese)

[21] Thrift, P., "Fuzzy Logic Synthesis with Genetic Algorithms," Proc. of the Int. Conf. of Genetic Algorithms (ICGA'92), 1992, pp.509-513

[22] Tsuchiya, T., Matsubara, Y., and Nagamachi, M., "A Learning Fuzzy Rule Parameters Using Genetic Algorithm", 8th Fuzzy System Symposium, 1992, pp.245-248 (in Japanese)