GA-Based Learning Algorithms to Identify Fuzzy Rules for Fuzzy Neural Networks

K Almejalli, K Dahal, Member IEEE, and A Hossain, Member IEEE
School of Informatics, University of Bradford, Bradford BD7 1DP, UK
{K.A.Al-mejalli; K.P.dahal; M.A.Hossain1}@Bradford.ac.uk

Abstract

Identification of fuzzy rules is an important issue in designing of a fuzzy neural network (FNN). However, there is no systematic design procedure at present. In this paper we present a genetic algorithm (GA) based learning algorithm to make use of the known membership function to identify the fuzzy rules form a large set of all possible rules. The proposed learning algorithm initially considers all possible rules then uses the training data and the fitness function to perform rule-selection. The proposed GA based learning algorithm has been tested with two different sets of training data. The results obtained from the experiments are promising and demonstrate that the proposed GA based learning algorithm can provide a reliable mechanism for fuzzy rule selection.

1. Introduction

Fuzzy neural networks are hybrid intelligent systems, which combine the advantages of both neural networks and fuzzy logic. The neural fuzzy system is a fuzzy system that uses the learning ability of the neural networks to determine its parameters (fuzzy sets, fuzzy memberships and fuzzy rules) by processing data [1]. An important topic in designing of a fuzzy neural network is the identification of the fuzzy rules. However, there is no systematic design procedure at present [2]. The recent research direction in the identification of the fuzzy rules in fuzzy neural networks is to learn and modify the rules from past experience. Currently, different approaches are used to identify the fuzzy rules in the fuzzy neural networks. Quek and Zhou [3] have classified these rule identification approaches into three categories. First category of approaches uses linguistic information from experts to identify fuzzy rules [4]. Although this approach converges faster during training and performs better, it is rather subjective since linguistic information from experts may vary from person to person, and from time to time. The second category of approaches uses unsupervised learning algorithms to identify fuzzy rules in the fuzzy neural networks prior to the application of neural network techniques to adjust the rules [5, 6]. In this approach the training data is the only source of information so it must be representative, otherwise, the derived fuzzy rules will be ill defined. The third group of approaches is using supervised leaning algorithm (particularly the Backpropagation technique) to identify the fuzzy rules in the fuzzy neural networks[7]. In this approach the fuzzy neural network appears as black box at the end of the training process.

This paper presents a genetic learning algorithm to make use of the known membership function to identify the fuzzy rules. The proposed learning algorithm belongs to the second approach for rule identification in fuzzy neural network. The preliminary work on the GA-based learning algorithm was presented in our previous work [8]. This paper extends this idea further by incorporating a weight parameter to trade-off between the number of fuzzy rules and overall error. In this paper we have used GA to make use of the known membership function to identify the fuzzy rules using similar fuzzy neural network structure discussed in [2]. The proposed genetic algorithm differs form existing algorithms such as [5, 6], in terms of being simple, fast and flexible to control the process of identification of the fuzzy rules based on the error level.

Several authors have proposed a genetic algorithm for fuzzy neural parameters optimization to adjust the control points of membership functions or to tune the weightings [9-14]. The pioneer was Karr[9] , who used GAs to adjust membership functions. Ishibuchi et al.[10] proposed a genetic- based method for selecting a small number of significant fuzzy rules to construct a compact fuzzy classification system with high classification power. Ishibuchi and Yamamoto farther developed this idea by using mult-objective genetic local search algorithms in [13]. Wang et al. [11] have pro-

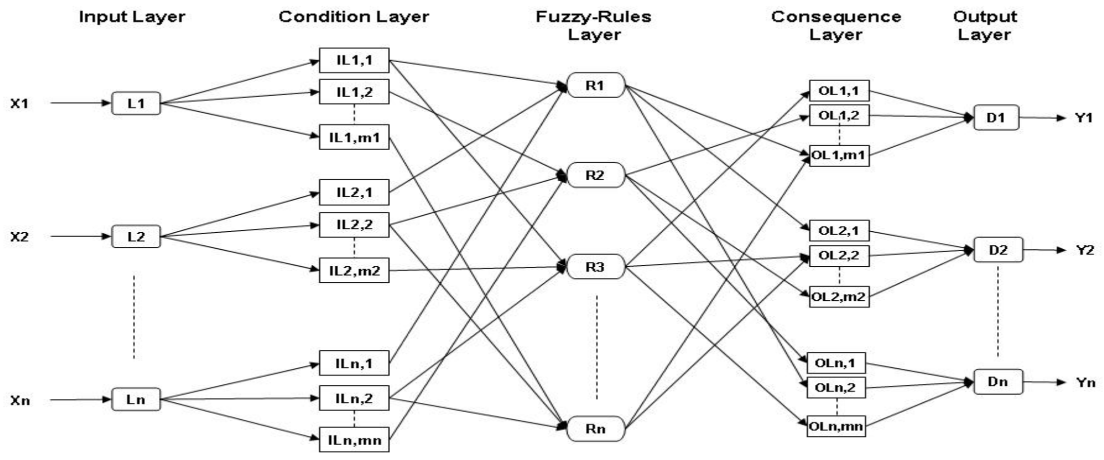

Figure 1. Structure of the fuzzy neural tool (FNN_Tool)

posed a simplified genetic algorithm to adjust both control points of B-spline membership functions and weights of fuzzy-neural networks. The proposed algorithm uses sequential-search-based crossover point method in which a better crossover point is determined and only the gene at the specified crossover point is crossed as a single point crossover operation. Lin [14] has proposed a hybrid learning algorithm for parameters learning. He has used GA to tune membership functions at the precondition part of fuzzy rules, while the least-squares estimate method has been used to tune parameters at the consequent part. Wang et al. [12] have proposed GA-based approach for a feedback direct adaptive fuzzy-neural controller to tune the online weighting factors. Specifically, they have used a reduced-form genetic algorithm (RGA) to adjust the weightings of the fuzzy-neural network.

The paper is organized as follows. The next section describes the structure of the GA-FNN. Section 3 discusses the proposed GA-based Learning Algorithm. The performance of the proposed GA-based learning algorithm is tested in Section 4. The conclusions and future works of this paper are given in Section 5.

2. The GA-FNN Structure

In this section, we describe the structure and function of the proposed GA-FNN model. The structure of the fuzzy neural network model used in this paper is similar to the structure proposed in [2]. It is a five-layer structure, as shown in Figure 1, where each layer performs an operation for building the fuzzy system. The process of each layer is described below (see Figure 1):

Layer 1: is the input layer. Nodes at this layer represent input linguistic variable and directly transmit non-fuzzy input values to the next layer. The input and the output of this layer are given as follows:

$$o_i^{(1)} = i_i^{(1)} \qquad \textbf{(1)}$$

where $i_i^{(1)}$ is the input and $o_i^{(1)}$ is the output of input neuron i in layer 1.

Layer 2: is the fuzzification layer, which defines the fuzzy sets and membership for each of the input factors. Nodes in this layer acts as a membership function and represents the terms of the respective linguistic variable. In our model the neurons of this layer are modeled as a common bell-shaped membership function [15], so the input $i_{i,j}^{(2)}$ and the output $o_{i,j}^{(2)}$ of fuzzification node i at the layer 2 are given as follows:

$$o_{i,j}^{(2)} = e^{-\frac{(i_{i,j}^{(2)}-m_{i,j})^2}{\sigma_{i,j}}} \qquad \textbf{(2)}$$

where $m_{i,j}$ and $\sigma_{i,j}$ are the centres and the widths of the membership function for the input-label neuron $LI_{i,j}$ respectively.

Layer 3: is the fuzzy rule layer, which defines all possible fuzzy rules to specify qualitatively how the output parameter is determined for various instances of the input parameters. Each node in this layer represents a fuzzy rule. The input and the output of a rule node at the layer 3 are given as follows:

$$y_i^{(3)} = \min(x_{ki}^{(3)}) \qquad \textbf{(3)}$$

where $x_{ki}^{(3)}$ are the inputs, and $y_i^{(3)}$ is the output of fuzzy rule i in the layer 3

Layer 4: is the consequence layer (or the output membership layer). Neurons in the consequence layer represent fuzzy sets used in the consequent part of the fuzzy rules. The input and the output of a consequence node in the layer 4 are given as follows:

$$y_i^{(4)} = \min\ (1, \sum_l x_{l,i}^{(4)}) \qquad \textbf{(4)}$$

where $x_{l,i}^{(4)}$ is the input (the output of neuron l in the fuzzy rule layer), and $y_i^{(4)}$ is the output of membership neuron i in the layer 4.

Layer 5: is the output layer (or the defuzzification layer). Each node at the output layer represents a single output variable. The input and the output of an output neuron in layer 5 are given as follows:

$$x_i^{(5)} = \sum (a_{ci} / b_{ci}) \times y_i^{(4)} \qquad \textbf{(5)}$$

$$y_i^{(5)} = \frac{x_i^{(5)}}{\sum_{i=1} (y_i^{(4)} / b_{ci})} \qquad \textbf{(6)}$$

where $x_i^{(5)}$ is the input and $y_i^{(5)}$ is the output neuron i in layer 5, a_{ci} and b_{ci} are the centre and the width of the fuzzy set respectively.

Due to the fact that the training data set is the only source of information in most cases, the learning process of the proposed GA-FNN completely relies on training data. We proposed the learning process consisting of three stages. First stage is initializing the membership functions of both input and output variables by determining their centres and widths. To perform this stage, we have employed a self-organizing algorithm [6] as in other works [2, 5, 16]. A proposed GA based learning algorithm is performed in the second stage to identify the fuzzy rules that are supported by the set of training data. In the last stage, the derived structure and parameters are fine tuned by using the back-propagation learning algorithm [15]. The remainder of this paper focuses on only the second stage, which is using GA to identify the fuzzy rules needed for constructing the fuzzy neural network.

3. The proposed GA-based Learning Algorithm

Most existing fuzzy neural networks use self-organizing algorithm to identify fuzzy rules, like [6]. However, in most integrated fuzzy neural networks, where the membership functions are determined prior to the identification of fuzzy rules, self-organizing or competitive learning becomes redundant, because the boundary of the clusters in the input and output spaces have already been predefined [2]. To avoid this redundancy of using self-organizing algorithms to determine the fuzzy parameters, some authors such as Lin and Lee [5] have used the known membership functions to find the fuzzy rules, where competitive learning is employed to identify the fuzzy rules. In Lin and Lee work, all the possible fuzzy rules must be listed prior to the initiation of competitive learning. After competitive learning, the link with the largest weight is selected and the consequence it connects to is subsequently held as the real consequence of the rule. Although this method shows satisfactory result with limited number of rules, it involves iterative training before the system comes to a stable state, because at least 50% of all possible fuzzy rules are useless. In this paper we have proposed a genetic learning algorithm to make use of the known membership function to identify only the relevant fuzzy rules.

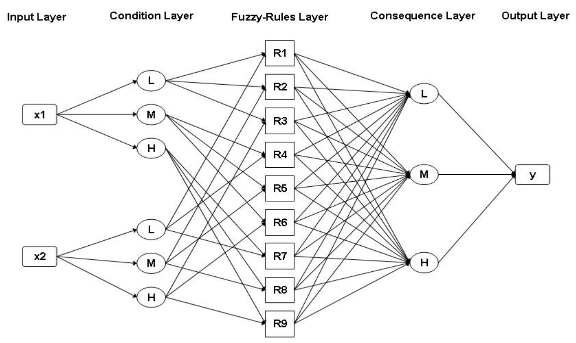

Figure 2. FNN of two inputs variable and one output variable for all possible fuzzy rules

To explain how we have designed a GA for identifying fuzzy rules, consider a simple example of FNN with two input linguistic variables x_1 and x_2, and one output linguistic variable y as shown in Figure 2. After performing the self-organization learning algorithm (first stage of learning), each linguistic variable has a number of fuzzy sets, say we have three fuzzy sets. Then the proposed genetic learning algorithm considers all possible rules. In our simple example there are a total of twenty seven possible rules. In fact these rules are made of nine possible antecedents (preconditions). These antecedents of fuzzy rules are represented by neurons $R_1 \ldots R_9$ of the Fuzzy-Rules Layer in Figure 2. Each antecedent has links with three possible decision fuzzy sets (neurons in Consequence Layer: Low(L), Medium(M) and High(H)). For example, the three possible fuzzy rules associated with neuron R_1 are:

If x_1 is L and x_2 is L, then y is L.

If x_1 is L and x_2 is L, then y is M.
If x_1 is L and x_2 is L, then y is H.

A number of decisions must be made in order to implement the GA for identifying appropriate fuzzy rules. There are problem specific decisions which are concerned with the search space (and thus the representation) and the form of the fitness function. The following steps are employed to identify appropriate fuzzy rules by the GA.

Step 1: Initialization: the first and most important step in the implementation of the GA technique is encoding of the problem using an appropriate representation. The encoding used to represent chromosomes (solutions) defines the size and the structure of the search space. Here we propose integer strings as chromosomes to represent candidate solutions of the problem. The string is given by $t_1, t_2, ..., t_i, ..., t_N$, where t_i is an integer $0 \le t_i \le M$ which indicates the link of neuron R_i (i.e. neurons in Fuzzy-Rules Layer) with output neurons (i.e., neurons in Consequence Layer). N is the number of neurons in the Fuzzy-Rules Layer and M is the number of neurons in the Consequence Layer. For our example, the chromosome has nine integers, and $0 \le t_i \le 3$. $t_i = 0$ indicates there is no link of R_i with output neuron; $t_i = 1$ indicates that there is a link with 'L' neuron in consequence Layer and so on. When $t_i = 0$, this means that R_i has no relation to that particular output variables. Hence, that rule and all the corresponding links in this case can be deleted without affecting the outputs.

Step 2. Fitness function: in this step the goodness of every chromosome is evaluated by using a fitness function. The fitness function can be any nonlinear, nondifferentiable, or discontinuous positive function, because the GA only needs a fitness value assigned to each chromosome. In this paper, we use a set of training data to calculate the fitness of each chromosome based on the following fitness function:

$$\text{FIT}(i) = \frac{1}{RMS\_ERROR(i)} \tag{7}$$

where *RMS_ERROR(i)* represents the root-mean-square error between the fuzzy-neural network outputs and the desired outputs for the *i*th string. The GA aims to maximize the fitness function (7) to minimize the error value (*e*). This error value is depended on the selected fuzzy rules (by the GA chromosome) and rule weightings. The proposed GA chromosome represents a set of fuzzy rules with $t_i \neq 0$ for inclusion and a set of fuzzy rules with $t_i = 0$ for ignoring. The weight for all rules assumed to be 1 at this stage. However, our experiment showed that the inclusions of some these rules ($t_i = 0$) with low weightings can still improve the error value.

In order to correctly identify the minimum number of the appropriate fuzzy rules without ignoring any relevant rule that might improve the error value, the fitness value of a chromosome is calculated in two stages: Firstly, a chromosome is evaluated as given by GA (*fit_1*). I.e. the fitness of the chromosome is calculated with considering all rules represented by $t_i \neq 0$ (taking weight 1), and rules represented by are ignored. Secondly, for each rule (R*i*) represented by $t_i = 0$, the fitness of the chromosome is calculated again with considering a low weight *LW* (e.g. 0.01) for each possible rule associated with that rule (R*i*) (in our example there are three possible rules) and then the best one is selected (*fit_2*). Then these two fitness values (*fit_1, fit_2*) are compared and the best fitness value is taken. The chromosome is adjusted if the second fitness (*fit_2*) appeared to be the better one.

Step 3. GA operators: Based on our previous experience with GA and a number of experiments we have selected GA operators and their parameters to be used for this application. The GA operators used are steady state replacement approach[17] , tournament selection [18], standard two-point crossover and a higher mutation probability. The steady state approach directly inserts a new solution into the population pool replacing a less fit solution. The tournament selection method picks a subset of solutions at random from the population to form a tournament selection pool, from which two solutions are selected with probability based upon the fitness values of the solutions. The two-point crossover operator splits the selected solutions at two randomly chosen positions and exchanges the centre sections with probability a crossover probability. The mutation operator changes the integer at each position in the solution within the allowed range with a defined mutation probability. We use a higher mutation priority in our case because the diversity in the population is not driven by recombination. The elitist approach, which ensures that the best solution in the population pool is always retained, has been applied. The initial population of chromosomes is created randomly. The stopping criterion for a GA run is to achieve the pre-specified error level (*e*).

When the GA learning process has completed (i.e. when pre-specified error level is achieved) after running the GA over a large number of runs, we choose the best GA chromosome. This best chromosome is decoded to get the structure of the FNN by keeping only the rules that are indicated by the chromosome. Then the error level (*e*) can be improved by using the back-propagation learning algorithm to fine tune the rules weights. By doing so, we only train the FNN with the relevant fuzzy rules only.

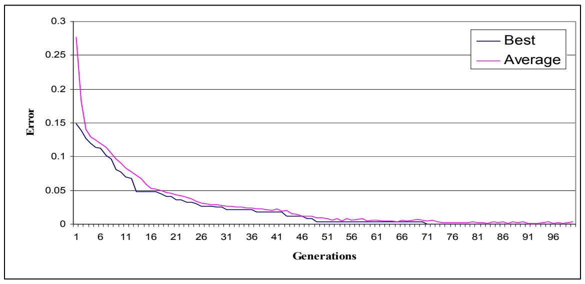

Figure 3. Performance graph for 100 generation of 20 chromosomes.

4. Experiments and Comparison

In this section, the performance of the proposed GA-based learning algorithm is tested with two different sets of training data.

Experiment 1:

In this Experiment, a set of training data consists of more than thousand data samples which are generated from 27 rules, some of them listed below:

If x1 is low and x2 is low and x3 is low then y is very low;
If x1 is low and x2 is medium and x3 is medium then y is low;
If x1 is medium and x2 is high and x3 is high then y is high;
If x1 is high and x2 is high and x3 is high then y is very high;
The weight of all rules set to unity (i.e. wi = 1).

From the rules above, it can be observed that there are three inputs and one output linguistic variables. For each input linguistic variable, its term set is defined as {low, medium, high}, while the output linguistic variable term set is defined as { very low, low, medium, high, very high}. Therefore, there are a total of 135 possible rules, and the resulting FNN consists of three linguistic nodes, nine input-label nodes (three for each linguistic node), twenty seven initial rule nodes, five output-label nodes, and one output node. After the self-organized learning to find the membership function for each input-label (output-label) node, the proposed genetic learning algorithm is employed to identify the fuzzy rules. Because of the fact that the weight of all rules used to generate the training data was unity, the weight for rules to be ignored used in the calculation of the fitness in the second stage with low weight (*LW*), as discussed in perfuse section, does not affect the result. In other words, we did not have to use thesecond stage of fitness calculation. After a number of runs, the result shows that the proposed learning algorithm is able to correctly identify all the relevant fuzzy rules with error level equals zero. Figure 3 presents an average performance graph of 20 experiments created by 100 generations of 20 chromosomes.

Experiment 2:

In the second experiment, we use a similar set of training data which consists of more than thousand data samples, but in this experiment the rules have different weights (i.e. $w_i = n$, where $0 \leq n \leq 1$).

After the self-organized learning to find the membership function for each input-label (output-label) node, the proposed genetic learning algorithm is employed to identify the fuzzy rules. In this case, due to the fact that some rules used in generating the training data may have a low weight *(LW)* (e.g. 0.1), the weight for rules to be ignored should be set for two stage fitness calculations. After a number of runs, the result shows that the number of rules identified by the algorithm and the error level (e) is affected by the value of *LW*. Table I shows the relationship between *LW* and the number of generated rules and the error level (*e)*, created by 100 generations of 20 chromosomes.

From Table I, it is clear that, any increase of the value of *LW* results a decrease of the number of relevant rules identified by the algorithm and also results an increase of the value of the error level (*e*). The explanation of these observations is that when the value of *LW* is large the possibility of ignoring some relevant rules with low weight is high, which affects negatively on the level error. For example, when the value of *LW* was 0.9 the number of relevant rules was 18 (9 ignored rules) and the error level was 0.2375, while the value of *LW* was 0.1, the number of relevant rules identified is 27 (i.e. all rules considered) and the error level decreased to 0.0760.

Table 1. The effect of LW on the number of rules and the error level

LW	**Number of Rules**	***e***
0.9	15	*0.2105*
0.7	19	*0.1975*
0.5	21	*0.1358*
0.3	25	*0.0902*
0.1	27	*0.0760*

It is usually the case that a low value of *LW* produces a better error value as this allows to consider a large number of rules. In this example, where we have a limited number of rules (i.e. 27), we can consider all rules by setting a low value of *LW* to get a minimum error level. However, for a case with a very large number of possible rules (which are impossible to consider), it is important to limit the number of rules for consideration by setting an appropriate value of *LW* to obtain an acceptable error level. For example, if we have a problem with a large number of rules (large number of inputs and outputs), and there are two values of *LW* *lw*1,*lw*2 (*lw*1<*lw*2). If *lw*1 results a number of rules *N* and a level error *e,* and *lw*2 results a decrease of N by 30% and an increase of e by only 0.1%. In this case possibly we set LW to lw2.

Also, in order to evaluate the effectiveness of application of the proposed genetic learning algorithm in FNN, we have employed the proposed GA-learning algorithm in GA-FNN to a real world case study of road traffic management. The results obtained from the case study are promising and demonstrate that the proposed GA based learning algorithm can provide a reliable mechanism for fuzzy rule selection. The results of this case study have been presented and discussed in [8].

5. Conclusion and Future Work

This paper has described a GA based learning algorithm to identify the fuzzy rules for FNNs. The proposed algorithm makes use of the known membership functions to identify only the relevant fuzzy rules. Initially, it has considered all possible rules and then used the training data and the fitness function to select a limited number of the relevant fuzzy rules by introducing a weight parameter to trade-off between number of rules and error value. In order to test the capabilities of the proposed GA-based learning algorithm for correct identification of all the relevant fuzzy rules, it has been tested with two different sets of training data. The results of the proposed GA-based learning algorithm demonstrate its capabilities and merits in term of identification of the fuzzy rules fast and correctly. Further experiments on its application to a case study of road traffic management, which have been reported elsewhere, also show the effectiveness of the proposed algorithm in FNN.

Currently we have demonstrated the technical feasibility of the proposed GA-learning algorithm to identify fuzzy rules, in which all possible rules were used to select the relevant rules. In the next stage we will investigate the possibility of developing the proposed learning algorithm to start with a limited number of relevant rules and then the learning process will be increased (or decreased) to that number rules instead of considering all possible rules.

6. References

[1] T. Lin, *Neural Fuzzy Control Systems With Structure and Parameter Learning*. Singapore: World Scientific 1994.

[2] C. Quek, M. Pasquier, and B. B. S. Lim, "POP-TRAFFIC: a novel fuzzy neural approach to road traffic analysis and prediction," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, pp. 133, 2006.

[3] C. Quek and R. W. Zhou, "The POP learning algorithms: reducing work in identifying fuzzy rules," *Neural Netw*, vol. 14, pp. 1431-45, 2001.

[4] B. Krause, A. S. von, W , and K. Limper, "A neuro-fuzzy adaptive control strategy for refuse incineration plants," *Fuzzy Sets and Systems*, vol. 63, pp. 329-338, 1994.

[5] C. T. Lin and C. S. G. Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System," *IEEE Transactions on Computers*, vol. 40, pp. 1320-1336, 1991.

[6] R. R. Yager, "Modeling and formulating fuzzy knowledge bases using neural networks," *Neural Networks*, vol. 7, pp. 1273-1283, 1994.

[7] H. Ishibuchi, H. Tanaka, and H. Okada, "Interpolation of fuzzy if-then rules by neural networks," *International Journal of Approximate Reasoning*, vol. 10, pp. 3–27, 1994.

[8] K. Almejalli, K. Dahal, and A. Hossain, "Intelligent Traffic Control Decision Support System," in *EVOWorkshop2007*. Valencia, Spain: (to be published in Computer Science Lecture notes by Spinger), 2007.

[9] C. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert*, vol. 6, pp. 26-33, 1991.

[10] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems usinggenetic algorithms," *Fuzzy Systems, IEEE Transactions on*, vol. 3, pp. 260-270, 1995.

[11] W. Y. Wang, T. T. Lee, C. C. Hsu, and Y. H. Li, "GA-based learning of BMF fuzzy-neural network," *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*, vol. 2, 2002.

[12] W. Y. Wang, C. Y. Cheng, and Y. G. Leu, "An online GA-based output-feedback direct adaptive fuzzy-neural controller for uncertain nonlinear systems," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 34, pp. 334-345, 2004.

[13] H. Ishibuchi and T. Yamamoto, "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining," *Fuzzy Sets and Systems*, vol. 141, pp. 59-88, 2004.

[14] C. J. Lin, "A GA-based neural fuzzy system for temperature control," *Fuzzy Sets and Systems*, vol. 143, pp. 311-333, 2004.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation, Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations," MIT Press, Cambridge, MA, 1986.

[16] P. J. Werbos, "Neurocontrol and fuzzy logic: connections and designs," *International Journal of Approximate Reasoning*, vol. 6, pp. 185-219, 1992.

[17] G. Sywerda, "Uniform crossover in genetic algorithms," *Proceedings of the third international conference on Genetic algorithms table of contents*, pp. 2-9, 1989.

[18] D. E. Goldberg and K. Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," *Urbana*, vol. 51, pp. 61801-2996, 1991.