H. Ishibuchi, T. Yamamoto, and T. Nakashima, “Hybridization of fuzzy GBML approaches for pattern classification problems,” *IEEE Trans. on Systems, Man, and Cybernetics- Part B: Cybernetics*, vol. 35, no. 2, pp. 359- 365, April 2005.

Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems

Hisao Ishibuchi, *Member, IEEE,* Takashi Yamamoto, *Member, IEEE,*
and Tomoharu Nakashima, *Member, IEEE*

Department of Computer Science and Intelligent Systems, Osaka Prefecture University

Corresponding Author: Prof. Hisao Ishibuchi
Department of Computer Science and Intelligent Systems,
Osaka Prefecture University
1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan
E-mail: hisaoi@cs.osakafu-u.ac.jp

Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems

Hisao Ishibuchi, *Member, IEEE,* Takashi Yamamoto, *Member, IEEE,*
and Tomoharu Nakashima, *Member, IEEE*
Department of Industrial Engineering, Osaka Prefecture University

Abstract - We propose a hybrid algorithm of two fuzzy genetics-based machine learning approaches (i.e., Michigan and Pittsburgh) for designing fuzzy rule-based classification systems. First, we examine the search ability of each approach to efficiently find fuzzy rule-based systems with high classification accuracy. It is clearly demonstrated that each approach has its own advantages and disadvantages. Next, we combine these two approaches into a single hybrid algorithm. Our hybrid algorithm is based on the Pittsburgh approach where a set of fuzzy rules is handled as an individual. Genetic operations for generating new fuzzy rules in the Michigan approach are utilized as a kind of heuristic mutation for partially modifying each rule set. Then, we compare our hybrid algorithm with the Michigan and Pittsburgh approaches. Experimental results show that our hybrid algorithm has higher search ability. The necessity of a heuristic specification method of antecedent fuzzy sets is also demonstrated by computational experiments on high-dimensional problems. Finally we examine the generalization ability of fuzzy rule-based classification systems designed by our hybrid algorithm.

Index Terms - Pattern classification, fuzzy rules, genetic algorithms, machine learning.

I. Introduction

Genetic algorithms have been successfully applied to various problems from combinatorial optimization to machine learning [1], [2]. Genetic algorithms can be viewed as a general-purpose optimization technique in a discrete search space. Some fuzzy genetics-based machine learning (GBML) algorithms have been proposed for designing fuzzy rule-based systems without linguistic knowledge from domain experts in the literature (see [3] for a survey of fuzzy GBML algorithms). For example, genetic algorithms were used for designing fuzzy rule tables [4] and determining an appropriate combination of antecedent and consequent linguistic values of each fuzzy rule [5]. In those studies, fixed membership functions of linguistic values were used when fuzzy rules were generated. Genetic algorithms were also used for tuning membership functions in [6] where it was assumed that a set of fuzzy rules had already been given. In [7], genetic algorithms simultaneously performed the determination of the number of fuzzy rules, the generation of fuzzy rules, and the tuning of membership functions. As in the case of non-fuzzy GBML algorithms, fuzzy GBML algorithms can be also classified into two categories: Pittsburgh approach [8] and Michigan approach [9]. The above-mentioned studies in

[4]-[7] can be viewed as fuzzy versions of the Pittsburgh approach where a set of fuzzy rules is handled as an individual. On the other hand, some fuzzy GBML algorithms (e.g., [10]-[13]) were proposed using the framework of the Michigan approach where a single fuzzy rule is handled as an individual.

In this paper, we propose a hybrid algorithm of these two approaches for designing fuzzy rule-based systems for pattern classification problems. First we examine the search ability of Pittsburgh-style and Michigan-style fuzzy GBML algorithms through computational experiments on commonly used data sets in the literature. We demonstrate advantages and disadvantages of each approach. Then, we combine the two approaches into a single hybrid algorithm. Our hybrid algorithm is basically the Pittsburgh approach where each rule set is handled as an individual. The Michigan approach, which has high search ability to efficiently find good fuzzy rules, is used as a kind of heuristic mutation for partially modifying each rule set. In this manner, our hybrid algorithm utilizes the high search ability of the Michigan approach together with the direct optimization ability of the Pittsburgh approach. While the basic idea of our hybrid algorithm has already been suggested in our former study [14], this paper includes the following new contributions in comparison with [14]:

1. Advantages and disadvantages of the two approaches of fuzzy GBML algorithms are demonstrated.
2. The search ability of our hybrid fuzzy GBML algorithm is compared with that of its non-hybrid versions.
3. It is demonstrated that a heuristic specification procedure of antecedent fuzzy sets is necessary for handling high-dimensional data sets by fuzzy GBML algorithms.
4. The generalization ability of fuzzy rule-based systems obtained by our hybrid algorithm is examined.
5. Various high-dimensional data sets involving up to 60 attributes are used in computational experiments for examining the above-mentioned issues.
6. A simple idea is incorporated into fuzzy GBML algorithms for handling the situation where we do not know an appropriate granularity of the fuzzy partition for each attribute. Our idea is to simultaneously use multiple fuzzy partitions with different granularities for generating fuzzy rules by fuzzy GBML algorithms.

II. Fuzzy Rule-Based Classification Systems

A. Fuzzy Rules for Pattern Classification

Let us assume that we have m labeled patterns $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1, 2, ..., m$ from M classes as training data. That is, we have an n-dimensional M-class pattern classification problem with m training patterns. We also assume that a set of linguistic values (and their membership functions) is given for describing each attribute. We use fuzzy rules of the following type for our *n*-dimensional pattern classification problem:

$$\text{Rule } R_q : \text{If } x_1 \text{ is } A_{q1} \text{ and ... and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q,\ q = 1, 2, ..., N_{\text{rule}}, \tag{1}$$

where R_q is the label of the *q*-th fuzzy rule, $\mathbf{x} = (x_1, ..., x_n)$ is an *n*-dimensional pattern vector, A_{qi} is an antecedent fuzzy set with a linguistic label (i.e., A_{qi} is a linguistic value such as "*small*" and "*large*"), C_q is a consequent class, CF_q is a rule weight, and N_{rule} is the number of fuzzy rules. The rule weight CF_q, which can be viewed as the certainty grade of the fuzzy rule R_q, is used as the strength of R_q in fuzzy reasoning. Classification boundaries by a set of fuzzy rules can be adjusted by changing their rule weights without modifying the membership functions of antecedent fuzzy sets [15].

B. Rule Generation and Fuzzy Reasoning

First we explain how the consequent class C_q is specified. We define the compatibility grade of each training pattern $\mathbf{x}_p$ with the antecedent part $\mathbf{A}_q = (A_{q1}, ..., A_{qn})$ using the product operator as

$$\mu_{\mathbf{A}_q}(\mathbf{x}_p) = \mu_{A_{q1}}(x_{p1}) \cdot \mu_{A_{q2}}(x_{p2}) \cdot\ \dots\ \cdot \mu_{A_{qn}}(x_{pn}), \tag{2}$$

where $\mu_{A_{qi}}(\,\cdot\,)$ is the membership function of the antecedent fuzzy set A_{qi} . The fuzzy conditional probability $\Pr(\text{Class } h \,|\, \mathbf{A}_q)$ of Class *h* ($h = 1, 2, ..., M$) for the antecedent part $\mathbf{A}_q$ is numerically approximated as follows [16]:

$$\Pr(\text{Class } h \,|\, \mathbf{A}_q) = \sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p) \bigg/ \sum_{p=1}^{m} \mu_{\mathbf{A}_q}(\mathbf{x}_p). \tag{3}$$

The consequent class C_q of the fuzzy rule R_q is specified by identifying the class with the maximum fuzzy conditional probability as

$$\Pr(\text{Class } C_q \,|\, \mathbf{A}_q) = \max_{h=1,2,...,M} \{\Pr(\text{Class } h \,|\, \mathbf{A}_q)\}. \tag{4}$$

Before explaining the specification of the rule weight CF_q, we describe fuzzy reasoning for classifying new patterns by fuzzy rules of the form in (1). Let S be a set of fuzzy rules. A new pattern $\mathbf{x}_p$ is classified by a single winner rule R_w, which is chosen from the rule set *S* as

$$\mu_{\mathbf{A}_w}(\mathbf{x}_p) \cdot CF_w = \max\{\mu_{\mathbf{A}_q}(\mathbf{x}_p) \cdot CF_q \ \,|\, R_q \in S\}. \tag{5}$$

The winner rule R_w has the maximum product of the compatibility grade $\mu_{\mathbf{A}_q}(\mathbf{x}_p)$ and the rule weight CF_q in *S*. If multiple fuzzy rules have the same maximum product but different consequent classes for the new pattern $\mathbf{x}_p$, the classification of $\mathbf{x}_p$ is rejected. The classification is also rejected if no fuzzy rule is compatible with the new pattern $\mathbf{x}_p$ (i.e.,

$\mu_{\mathbf{A}_q}(\mathbf{x}_p) = 0$ for $^{\forall} R_q \in S$). We use the fuzzy reasoning method based on a single winner rule because it makes the credit assignment in Michigan-style fuzzy GBML algorithms very simple. Since a single winner rule is responsible for the classification of the new pattern $\mathbf{x}_p$, we can easily assign a reward/punishment to the winner rule based on the classification result (i.e., correct classification or misclassification of $\mathbf{x}_p$).

It was shown in [16] that the following specification of the rule weight CF_q is appropriate for two-class problems when we use the single winner-based method:

$$CF_q = \Pr(\text{Class } C_q \mid \mathbf{A}_q) - \Pr(\text{Class } \overline{C}_q \mid \mathbf{A}_q)\,, \tag{6}$$

where $\overline{C}_q$ is the complementary class of C_q in two-class problems (e.g., $\overline{C}_q = 2$ when $C_q = 1$).

There are several alternative definitions of rule weights as extensions of (6) to the case of multi-class problems [17]. In this paper, we use the following specification of the second term in the right-hand side of (6) in the case of multi-class problems (i.e., $M > 2$).

$$\Pr(\text{Class } \overline{C}_q \mid \mathbf{A}_q) = \sum_{\substack{h=1 \\ h \neq C_q}}^{M} \Pr(\text{Class } h \mid \mathbf{A}_q)\,. \tag{7}$$

When CF_q becomes negative in (6), we do not generate any fuzzy rule with the antecedent part $\mathbf{A}_q$. That is, the rule set *S* includes no fuzzy rules with negative weights. We use the definition of rule weights in (6)-(7) because good results were reported using this definition in [17].

III. FUZZY GBML APPROACHES

In this section, we explain Michigan-style and Pittsburgh-style fuzzy GBML algorithms. The search ability of these algorithms is examined through computational experiments on three well-known data sets from the UCI machine learning repository: iris data with four attributes, wine data with 13 attributes, and sonar data with 60 attributes. We choose these three data sets because their dimensionality is totally different from each other (i.e., we choose the iris data as a low-dimensional problem, the wine data as a medium-dimensional problem, and the sonar data as a high-dimensional problem). Since we do not know an appropriate fuzzy partition for each attribute in those data sets, we simultaneously use four different fuzzy partitions in Fig. 1 for each attribute. This means that we have 15 antecedent fuzzy sets including *don't care*. Thus the total number of possible combinations of the antecedent fuzzy sets for an *n*-dimensional problem is 15^n (i.e., the number of possible fuzzy rules is 15^n). Our task in this section is to find a small number of good fuzzy rules with high classification ability from such a huge number of possible fuzzy rules. While we use four fuzzy partitions evenly divided with symmetric triangular fuzzy sets in Fig. 1, arbitrarily given fuzzy partitions (e.g., inhomogeneous fuzzy partitions with different fuzzy sets) can be used in fuzzy GBML algorithms in this paper.

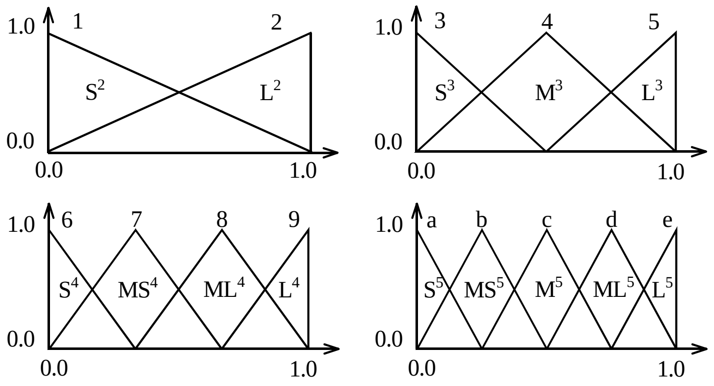

Fig. 1. Four fuzzy partitions used in our computational experiments. The superscript of each fuzzy set means the granularity of the fuzzy partition. Each of the 14 fuzzy sets is represented by one of the 14 symbols (i.e., 1, 2, ..., 9, a, b, c, d, e) as shown in this figure.

A. Michigan-Style Algorithm

In our Michigan-style algorithm [12], [13], the search for good fuzzy rules corresponds to the evolution of a population of fuzzy rules. A single fuzzy rule is handled as an individual. A population consists of a prespecified number of fuzzy rules. As we have already explained in Section II, the consequent class and the certainty grade of each fuzzy rule can be easily specified from compatible training patterns with its antecedent part. Thus we code each fuzzy rule as a string using its antecedent fuzzy sets. We use 15 symbols (e.g., 0, 1, ..., 9, a, b, c, d, e) for representing *don't care* and the 14 antecedent fuzzy sets in Fig. 1. For example, "0102d0" denotes the fuzzy rule "If x_2 is S^2 and x_4 is L^2 and x_5 is ML^5 then Class C_q with CF_q" where *don't care* conditions on x_1, x_3 and x_6 represented by 0's in the string are omitted.

First, our Michigan-style algorithm randomly generates a number of fuzzy rules (say, N_{rule} fuzzy rules) as an initial population. That is, N_{rule} strings of the length *n* are generated for our *n*-dimensional problem by randomly selecting each of the 15 symbols with the probability 1/15. Next, the fitness value of each fuzzy rule is evaluated. Let *S* be the set of the fuzzy rules in the current population. The evaluation of each fuzzy rule is performed by classifying all the given training patterns by the rule set *S* using the single winner-based method in (5). After all the given training patterns are classified by *S*, the fitness value $fitness(R_q)$ of each fuzzy rule R_q in *S* is calculated as the number of correctly classified training patterns by R_q. Then, new fuzzy rules are generated from the existing rules in the current population by genetic operations. As parent strings, two fuzzy rules are selected from the current population using the binary tournament selection. From the selected two strings, two new strings are generated by the uniform crossover with a prespecified crossover probability. Each symbol of the generated strings by the crossover operation is randomly replaced with a different symbol using a prespecified mutation probability. The selection, crossover and mutation are iterated until a prespecified number of new strings (say, N_{replace} strings) are generated. Finally, the worst

N_{replace} strings with the smallest fitness values in the current population are removed, and the newly generated N_{replace} strings are added to the remaining strings to form a new population.

The above procedures are applied to the new population again. The generation update is iterated until a stopping condition is satisfied. In most of our computational experiments, we use the total number of generations as the stopping condition to compare different algorithms under the same condition. In this case, the specification of the stopping condition is based on the available computation time. Our Michigan-style algorithm can be written as follows:

[Michigan-Style Fuzzy GBML Algorithm]

Step 1: Generate N_{rule} fuzzy rules.
Step 2: Calculate the fitness value of each fuzzy rule in the current population.
Step 3: Generate N_{replace} fuzzy rules using the genetic operations.
Step 4: Replace the worst N_{replace} fuzzy rules in the current population with the newly generated N_{replace} rules.
Step 5: Return to Step 2 if the prespecified stopping condition is not satisfied.

During the execution of this algorithm, we monitor the classification rate of the current population on the given training patterns. The rule set (i.e., population) with the highest classification rate is chosen as the final solution.

This algorithm is the simplest version of our Michigan-style algorithm [12], [13]. Its search ability can be improved by various heuristic tricks (e.g., by adding a misclassification penalty term to the fitness function, using a tailored initial population, and generating new fuzzy rules directly from misclassified and rejected training patterns [13]). In our computational experiments, we first use this simplest version for comparing the Michigan approach with the Pittsburgh approach under the condition of no heuristic tricks. We also examine the search ability of each approach with some heuristic tricks. In the following, we briefly explain the heuristic tricks used in our computational experiments.

It was shown in [13] that the search ability of our fuzzy classifier system was drastically improved by directly generating initial fuzzy rules from training patterns in the following heuristic manner. First we randomly select N_{rule} training patterns. Next we generate a fuzzy rule from each of the selected training patterns by choosing the most compatible antecedent fuzzy set (excluding *don't care*) with each attribute value. That is, the antecedent part $\mathbf{A}_q = (A_{q1}, ..., A_{qn})$ is specified so that A_{qi} has the maximum compatibility grade with x_{pi} when the fuzzy rule R_q is generated from the training pattern $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$. Then each antecedent fuzzy set of the generated fuzzy rule is replaced with *don't care* using a prespecified probability $P_{don't\,care}$. In this manner, N_{rule} initial fuzzy rules are generated in our former study [13]. In this paper, we modify this heuristic procedure because some antecedent fuzzy sets from fine fuzzy partitions are never selected. Instead of choosing the antecedent fuzzy set with the maximum compatibility grade, we probabilistically choose an antecedent fuzzy set from the 14 candidates B_k ($k = 1, 2, ..., 14$) in Fig. 1 where each candidate B_k has the following selection probability for the attribute value x_{pi}:

$$P(B_k) = \frac{\mu_{B_k}(x_{pi})}{\sum_{j=1}^{14} \mu_{B_j}(x_{pi})}. \tag{8}$$

Then each antecedent fuzzy set of the generated fuzzy rule is replaced with *don't care* using a prespecified probability $P_{don't\,care}$.

The specification of each antecedent fuzzy set by (8) and the replacement with *don't care* using the probability $P_{don't\,care}$ can be used not only for generating initial fuzzy rules but also for updating the current population. In the basic form of our Michigan-style algorithm, new fuzzy rules are generated by the genetic operations (i.e., selection, crossover and mutation) from the existing fuzzy rules in the current population. Fuzzy rules can be directly generated from misclassified or rejected training patterns to increase the search ability (see [13] for the effect of this procedure). We generate at least a half of new fuzzy rules by the genetic operations (i.e., at least $N_{\text{replace}}/2$ fuzzy rules). Let N_{MR} be the sum of the number of misclassified and rejected training patterns by the existing fuzzy rules in the current population. When N_{MR} is less than or equal to $N_{\text{replace}}/2$, N_{MR} fuzzy rules are directly generated from the N_{MR} training patterns in exactly the same manner as the above-mentioned procedure for generating initial fuzzy rules. Other fuzzy rules (i.e., $(N_{\text{replace}} - N_{\text{MR}})$ rules) are generated by the genetic operations. On the other hand, when N_{MR} is larger than $N_{\text{replace}}/2$, $N_{\text{replace}}/2$ training patterns are randomly chosen from the N_{MR} training patterns. Then $N_{\text{replace}}/2$ fuzzy rules are directly generated from the chosen patterns. Other fuzzy rules are generated by the genetic operations.

B. Pittsburgh-Style Algorithm

For the comparison with our Michigan-style algorithm in the previous subsection, we implement a fuzzy GBML algorithm based on the Pittsburgh approach where a set of fuzzy rules is handled as an individual. As in the previous subsection, each fuzzy rule is coded as a string of the length *n*. Thus a rule set with N_{rule} fuzzy rules is coded as a concatenated string of the length $n \times N_{\text{rule}}$ where each substring of the length *n* represents a fuzzy rule.

In our Pittsburgh-style algorithm, first a number of rule sets (say, N_{pop} rule sets) with N_{rule} fuzzy rules are randomly generated in the same manner as in the basic version of our Michigan-style algorithm in the previous subsection. The generated N_{pop} rule sets comprise an initial population. Next each rule set is evaluated by classifying the given training patterns. The fitness value of each rule set S_i in the current population is calculated as the number of correctly classified training patterns by S_i. Then, new rule sets are generated from the existing rule sets in the current population by genetic operations. As parent strings, two rule sets are selected from the current population using the binary tournament selection scheme. From the selected two strings, two new strings are generated by the uniform crossover with the prespecified crossover probability. The uniform crossover in the Pittsburgh-style algorithm exchanges substrings between the two parent strings. That is, we use the substring-wise (i.e., rule-wise) uniform crossover. We also examine the standard (i.e., bit-wise) uniform crossover. Each symbol of the generated strings by the crossover operation is randomly replaced with a different symbol using a prespecified mutation probability as in the Michigan-style algorithm. The selection, crossover and mutation are iterated until $(N_{\text{pop}} - 1)$ rule sets are generated. Finally, the best rule set in the current population is added to the newly generated rule sets as an elite rule set to form a new population including N_{pop} rule sets.

The generation update is iterated until a prespecified stopping condition is satisfied. As in the Michigan-style algorithm, we use the total number of generation updates as the stopping condition. The final solution is the best rule set in the final population because the best rule set in the current population is always inherited to the next population by the elitist strategy. Our Pittsburgh-style algorithm can be written as follows:

[Pittsburgh-Style Fuzzy GBML Algorithm]

Step 1: Generate N_{pop} rule sets with N_{rule} fuzzy rules.
Step 2: Calculate the fitness value of each rule set in the current population.
Step 3: Generate $(N_{\text{pop}} - 1)$ rule sets using the genetic operations.
Step 4: Add the best rule set in the current population to the newly generated $(N_{\text{pop}} - 1)$ rule sets to form the next population.
Step 5: Return to Step 2 if the prespecified stopping condition is not satisfied.

The heuristic procedure for directly generating initial fuzzy rules from training patterns can be used in our Pittsburgh-style algorithm in the same manner as in our Michigan-style algorithm.

C. Comparison of Two Approaches

For comparing the two fuzzy GBML algorithms with each other, we performed computational experiments on the following data sets available from the UC Irvine machine learning repository:

Iris data: 150 samples with 4 attributes from three classes.
Wine data: 178 samples with 13 attributes from three classes.
Sonar data: 208 samples with 60 attributes from two classes.

We applied the basic versions of our two fuzzy GBML algorithms to the three data sets. We examined the substring-wise (i.e., rule-wise) uniform crossover as well as the standard bit-wise uniform crossover in the Pittsburgh-style algorithm. Parameter values are summarized in Table 1 where the mutation probability is specified by the number of attributes in each data set (i.e., *n*). Table 1 shows that our task is to design a fuzzy rule-based classification system with 10 or 20 fuzzy rules by each algorithm for each data set. From Table 1, we can see that only 1,000 rule sets are examined in a single run of the Michigan-style algorithm. On the other hand, 200,000 rule sets are examined in the Pittsburgh-style algorithm since 200 rule sets are included in each population (i.e., population size is 200 in the Pittsburgh-style algorithm).

Average classification rates and average CPU time over 20 runs of each algorithm for each data set are summarized in Table 2 and Table 3, respectively. The CPU time was measured on a personal computer with a Pentium 4 (2.53 GHz) processor. Classification rates on each data set were measured on training patterns, which were the same as the whole data set in this subsection. Thus the experimental results in Table 2 should be used for evaluating the search ability of each algorithm. The generalization ability of extracted fuzzy rules (i.e., classification rates on test patterns) will be examined later in this paper.

Table 1. Parameter values in each algorithm.

Algorithm	Michigan	Pittsburgh
Number of fuzzy rules	10 or 20	10 or 20
Number of rule sets	1	200
Crossover probability	0.9	0.9
Mutation probability	$1/n$	$1/n$
Number of replaced rules	2 or 4	N. A.
Total number of generations	1000	1000

Table 2. Average classification rates (%). P-rule and P-bit mean the Pittsburgh-style fuzzy GBML algorithm with the rule-wise and bit-wise uniform crossover operations, respectively.

# of rules	10 rules			20 rules		
Algorithms	Mich	P-rule	P-bit	Mich	P-rule	P-bit
Iris	96.77	99.53	99.33	96.83	99.70	99.53
Wine	94.24	99.89	100	97.11	100	100
Sonar	0	0	0	0	0	0

Table 3. Average CPU time (seconds).

# of rules	10 rules			20 rules		
Algorithms	Mich	P-rule	P-bit	Mich	P-rule	P-bit
Iris	0.4	73	69	0.65	139	133
Wine	0.75	159	144	1.35	276	239
Sonar	0.45	102	105	0.95	202	209

From Table 2, we can see that no meaningful fuzzy rules were found for the sonar data by the genetic search from randomly generated initial populations. This is because the search space for the sonar data with 60 attributes is too large to search for fuzzy rules using no heuristic procedure. We will examine the effect of the heuristic specification procedure of antecedent fuzzy sets on the performance of our two fuzzy GBML algorithms later in this section. We can also see from Table 2 that the Pittsburgh-style algorithm outperformed the Michigan-style algorithm on the iris data and the wine data. The performance of the two versions (i.e., rule-wise crossover and bit-wise crossover) of the Pittsburgh-style algorithm is almost the same in Table 2.

From Table 3, we can see that the Michigan-style algorithm spent much less CPU time than the Pittsburgh-style algorithm. So we compared the two approaches with each other under the same computation load. Experimental results on the iris data and the wine data are summarized in Fig. 2 and Fig. 3, respectively, for the case of 20 rules. It should be noted that we did not use 1000 generations as the stopping condition in Fig. 2 and Fig. 3. The horizontal axis of each figure is the number of examined rule sets, which is the same as the number of generations in the case of the Michigan-style algorithm. Those figures show the average classification rate over 20 trials of each algorithm for each data set. The average classification rate at each generation was calculated using the best rule set obtained until that generation in each trial. From those figures, we can see that the Michigan-style algorithm has much higher search ability to efficiently find good fuzzy rules than the Pittsburgh-style algorithm in the early stage of evolution. That is, the average classification rate was rapidly improved by the Michigan-style algorithm. Further improvement by the Michigan-style algorithm, however, was very slow after examining a certain number of rule sets (e.g., about 100 rule sets in Fig. 2). In the long run, the Pittsburgh-style algorithm outperformed the Michigan-style algorithm in Fig. 2 and Fig. 3.

These observations suggest that the Michigan-style algorithm has high search ability to efficiently find good fuzzy rules. The Michigan-style algorithm, however, cannot directly optimize fuzzy rule-based systems because it only measures the performance of each fuzzy rule. That is, the optimization of fuzzy rule-based systems is indirectly performed by finding good fuzzy rules. The lack of the direct optimization ability leads to inferior results by the Michigan-style algorithm in the long run.

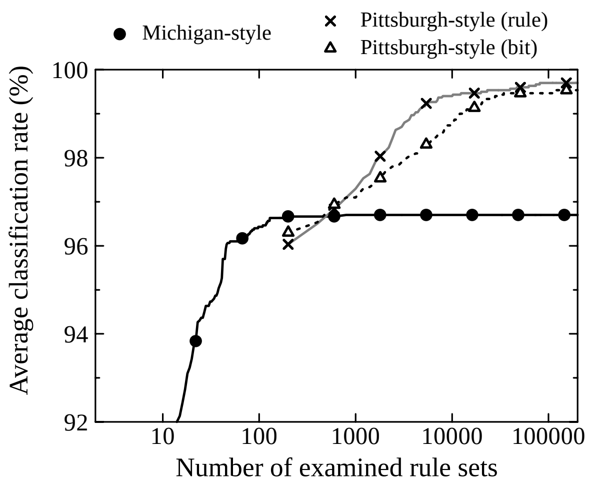

Fig. 2. Experimental results on the iris data set.

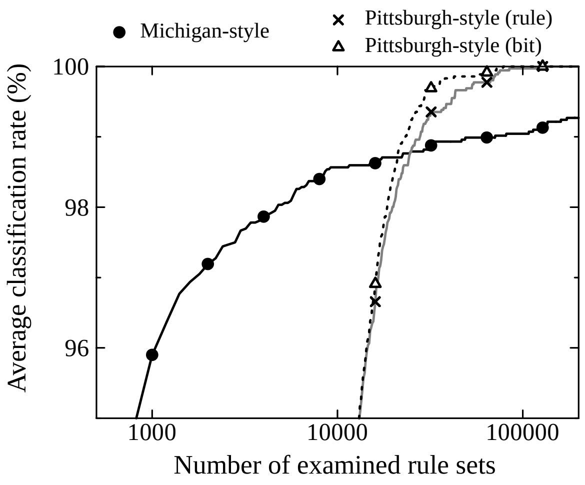

Fig. 3. Experimental results on the wine data set.

Next we examined the effect of the heuristic specification procedure of antecedent fuzzy sets directly from training patterns on the search ability of each algorithm. We applied each algorithm with the heuristic procedure to the three data sets in the same manner as in Tables 1-3. The two versions of the Pittsburgh-style algorithm used the heuristic procedure only for generating initial fuzzy rules. On the other hand, it was used for generating new fuzzy rules in each generation as well as for generating initial fuzzy rules in the Michigan-style algorithm. The probability of *don't care* was specified as follows: $P_{don't\ care} = 0.5$ for the iris data set, $P_{don't\ care} = 0.8$ for the wine data set, and $P_{don't\ care} = 0.95$ for the sonar data set so that each fuzzy rule had a few antecedent fuzzy sets (excluding *don't care*) on average.

Average experimental results over 20 runs are summarized in Table 4 and Table 5. From the comparison between Table 2 and Table 4, we can see that the average classification rates on the sonar data were drastically increased by the use of the heuristic procedure. The increase in the CPU time by the heuristic procedure, however, was not so large (compare Table 5 with Table 3). Since the performance of the two versions of the Pittsburgh-style algorithm is almost the same, hereafter we only use its substring-wise (i.e., rule-wise) uniform crossover version.

Table 4. Average classification rates by each algorithm with the heuristic specification procedure of antecedent fuzzy sets (%).

# of rules	10 rules			20 rules		
Algorithms	Mich	P-rule	P-bit	Mich	P-rule	P-bit
Iris	96.50	99.57	99.50	96.73	99.53	99.60
Wine	97.25	100	99.97	98.03	100	100
Sonar	80.72	94.76	92.21	84.01	97.36	96.18

Table 5. Average CPU time by each algorithm with the heuristic specification procedure of antecedent fuzzy sets (seconds).

# of rules	10 rules			20 rules		
Algorithms	Mich	P-rule	P-bit	Mich	P-rule	P-bit
Iris	0.3	70	69	0.65	136	134
Wine	0.6	130	130	1.15	232	226
Sonar	1.65	306	266	3.15	578	515

IV. HYBRID ALGORITHM

From the experimental results in the previous section, we can see that each algorithm has its own advantages and disadvantages. In this section, we combine the two fuzzy GBML algorithms into a single hybrid algorithm. Our aim is to implement a hybrid algorithm that has the advantages of the two fuzzy GBML algorithms. Our hybrid algorithm can be written as follows:

[Hybrid Fuzzy GBML Algorithm]

Step 1: Generate N_{pop} rule sets with N_{rule} fuzzy rules.

Step 2: Calculate the fitness value of each rule set in the current population.

Step 3: Generate $(N_{\text{pop}} - 1)$ rule sets by the selection, crossover and mutation in the same manner as the Pittsburgh-style algorithm. Apply a single iteration of the Michigan-style algorithm (i.e., the rule generation and the replacement) to each of the generated rule sets with a prespecified probability (0.5 in our computational experiments).

Step 4: Add the best rule set in the current population to the newly generated $(N_{\text{pop}} - 1)$ rule sets to form the next population.

Step 5: Return to Step 2 if the prespecified stopping condition is not satisfied.

Our hybrid algorithm is the same as the Pittsburgh-style algorithm except that the Michigan-style algorithm is applied to each rule set after the mutation operation in Step 3 for generating new fuzzy rules. Our hybrid algorithm has the high search ability of the Michigan-style algorithm as well as the direct optimization ability of the Pittsburgh-style algorithm.

We applied our hybrid algorithm to the three data sets using the parameter specifications in Table 1. Average experimental results over 20 runs are summarized in Table 6 and Table 7. From the comparison between Table 4 and Table 6, we can see that the average classification rates on the sonar data were improved by the hybridization. We can also see that a 100% classification rate was always obtained for the wine data in all the 20 runs for the four cases in Table 6. This observation shows high search ability of our hybrid algorithm. It should be noted that the heuristic specification procedure of antecedent fuzzy sets is necessary in the application of the hybrid algorithm to the sonar data set in Table 6 as the other algorithms in Table 2.

Table 6. Average classification rates by the hybrid algorithm (%).

# of rules	10 rules		20 rules	
Heuristics	Off	On	Off	On
Iris	99.53	99.57	99.63	99.23
Wine	100	100	100	100
Sonar	0	95.82	0	99.35

Table 7. Average CPU time by the hybrid algorithm (seconds).

# of rules	10 rules		20 rules	
Heuristics	Off	On	Off	On
Iris	110	105	225	210
Wine	287	205	501	395
Sonar	146	518	300	1000

The effect of the hybridization with the Michigan-style algorithm on the search ability of the Pittsburgh-style algorithm is demonstrated in Fig. 4 for the sonar data set in the case of 20 rules. This figure shows the average classification rate of the best rule set in each generation over 20 trials of each algorithm. Each algorithm used the heuristic specification procedure of antecedent fuzzy sets. The hybrid algorithm was implemented using the rule-wise uniform crossover operation. From Fig. 4, we can see that the hybridization improved the ability of the Pittsburgh-style algorithm to efficiently find good rule sets. The effect of the hybridization can be more clearly shown in computational experiments without the heuristic procedure. Fig. 5 shows experimental results on the wine data set in the case of 20 rules where the heuristic procedure was not used. From Fig. 5, we can see that much better rule sets were found by the hybrid algorithm than the Pittsburgh-style algorithm in the early stage of evolution. This is because the Michigan-style algorithm has high search ability to efficiently find good fuzzy rules.

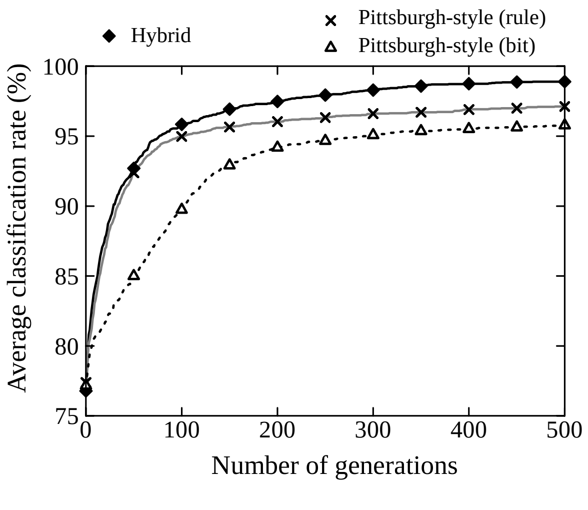

Fig. 4. Experimental results on the sonar data set.

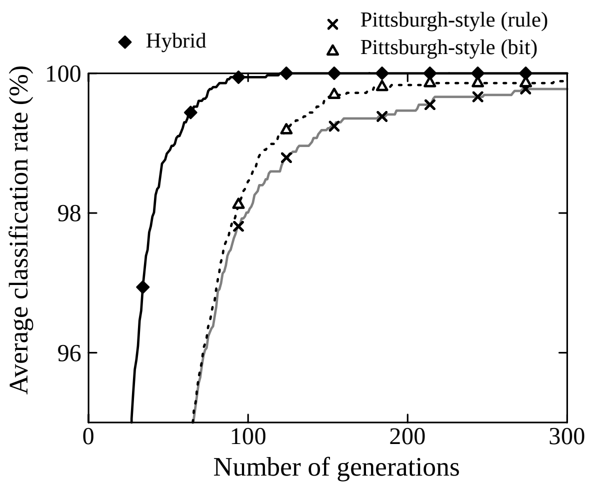

Fig. 5. Experimental results on the wine data set.

Finally we examine the generalization ability of fuzzy rule-based systems designed by our hybrid algorithm through computational experiments on seven data sets in Table 8 from the UC Irvine machine learning repository. Since the C4.5 algorithm [18] is one of the most well-known and frequently-used methods for designing non-fuzzy rule-based systems, it is compared with our hybrid algorithm. As benchmark results, we cite in Table 9 the best error rate for each data set on test patterns among six variants of the C4.5 algorithm examined in [19]. The performance of each variant was examined by ten iterations of the whole ten-fold cross-validation (10-CV) procedure (i.e., 10×10-CV). In our computational experiments, we also used ten iterations of the whole 10-CV procedure for examining the performance of our hybrid algorithm. In Table 9, we also cite some recently reported results of a decision tree-based fuzzy classifier in [20] and a GA-based non-fuzzy classifier in [21]. In these studies, the average classification rates on test patterns were evaluated using the 10-CV procedure for each data set.

Experimental results by our hybrid algorithm are summarized in Table 9 where two parameter specifications (i.e., 10 rules and 20 rules) were examined. From Table 9, we can see that the average error rates by our hybrid algorithm are better than or comparable to the best results by the C4.5 algorithm in [19] except for the glass data. For the glass data, our results are inferior to the best result in [19]. We can also see that our results are very similar to the results of the decision tree-based fuzzy classifier in [20] except for the wine data. For the wine data, much better results were obtained by our hybrid algorithm than the decision tree-based fuzzy classifier in [20]. Our hybrid algorithm outperformed the GA-based non-fuzzy classifier in [21] for the Wisconsin breast cancer data (Breast W) and the wine data in Table 9. From these observations, we can see that our hybrid algorithm has high generalization ability.

In [21], experimental results on training patterns were also reported. For example, the average classification rates on training patterns for the iris data and the wine data were 98.20% and 99.78% in [21], respectively. From the comparison between these reported results and our results in Table 6 (i.e., 99.68% for the iris data and 100% for the wine data), we can see that our hybrid algorithm has high search ability.

Table 8. Data sets used in our computer simulations.

Data set	Number of attributes	Number of patterns	Number of classes
Breast W	9	683*	2
Diabetes	8	768	2
Glass	9	214	6
Heart C	13	297*	5
Iris	4	150	3
Sonar	60	208	2
Wine	13	178	3

* Incomplete patterns with missing values are not included.

Table 9. Average error rates by our hybrid algorithm and reported results in [19]-[21]. The best result in each row is highlighted by boldface.

Data set	Our algorithm		C4.5	Fuzzy	GA
	10 rules	20 rules	[19]	[20]	[21]
Breast W	3.54	3.32	5.1	**3.18**	4.70
Diabetes	25.08	**24.17**	25.0	26.95	-
Glass	37.80	34.63	**27.3**	33.97	-
Heart C	46.50	**45.52**	46.3	-	-
Iris	5.33	5.60	5.7	4.89	**4.40**
Sonar	**23.70**	**23.70**	24.6	-	-
Wine	**4.94**	5.11	5.6	8.78	8.33

V. CONCLUDING REMARKS

In this paper, we first examined the search ability of two fuzzy GBML algorithms through computational experiments on commonly used data sets in the literature. These two algorithms were based on the Michigan approach and the Pittsburgh approach, respectively. From experimental results, we had the following observations. The Michigan-style fuzzy GBML algorithm had high search ability to efficiently find good fuzzy rules. Because the evolution of fuzzy rule-based systems in the Michigan-style algorithm was driven only by the performance of each fuzzy rule, it did not have high search ability to find a good combination of fuzzy rules. That is, the execution of the Michigan-style algorithm was not directly related to the optimization of fuzzy rule-based systems. On the other hand, the Pittsburgh-style algorithm could directly optimize fuzzy rule-based systems. Thus, it could find a good combination of fuzzy rules. The Pittsburgh-style algorithm, however, did not have high search ability to efficiently find good fuzzy rules because the performance of each fuzzy rule was not taken into account in the evolution of fuzzy rule-based systems. Next we combined the two fuzzy GBML algorithms into a single hybrid algorithm based on these observations. In our hybrid algorithm, the Michigan approach was used for generating good fuzzy rules while the Pittsburgh approach was used for finding good combinations of generated fuzzy rules. In this manner, advantages of these two approaches were utilized in our hybrid algorithm. It was shown by computational experiments that our hybrid algorithm outperformed its non-hybrid versions. We also demonstrated the importance of the heuristic specification procedure of antecedent fuzzy sets when the three fuzzy GBML algorithms were applied to high-dimensional data sets (e.g., sonar data with 60 attributes). Finally the generalization ability of fuzzy rule-based systems designed by our hybrid algorithm was examined through computational experiments on various data sets. It was shown that the performance of our hybrid algorithm was better than or comparable to the C4.5 algorithm for many data sets.

Our hybrid algorithm can be easily extended to the case of variable string length. In this case, the number of fuzzy rules in each rule set (i.e., each string) is not fixed but evolved during the execution of our hybrid algorithm. The extended version can handle not only the maximization of the classification accuracy but also the minimization of the number of fuzzy rules. This version can be further extended to the multiobjective design of fuzzy rule-based classification systems for discussing the tradeoff between the accuracy of fuzzy rule-based systems and their complexity (e.g., the number of fuzzy rules and the number of antecedent conditions).

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[3] O. Cordon, F. Herrera, F. Hoffman, and L. Magdalena, *Genetic Fuzzy Systems*, World Scientific, Singapore, 2001.

[4] P. Thrift, “Fuzzy logic synthesis with genetic algorithms,” *Proc. of 4th International Conf. on Genetic Algorithms*, pp. 509-513, University of California, San Diego, CA, July 13-16, 1991.

[5] D. S. Feldman, “Fuzzy network synthesis with genetic algorithms,” *Proc. of 5th International Conf. on Genetic Algorithms*, pp. 312-317, University of Illinois, Urbana-Champain, IL, July 17-21, 1993.

[6] C. L. Karr, “Design of an adaptive fuzzy logic controller using a genetic algorithm,” *Proc. of 4th International Conf. on Genetic Algorithms*, pp. 450-457, University of California, San Diego, CA, July 13-16, 1991.

[7] H. Nomura, I. Hayashi, and N. Wakami, “A self-tuning method of fuzzy reasoning by

genetic algorithm," *Proc. of 1992 International Fuzzy Systems and Intelligent Control Conf.*, pp. 236-245, Louisville, KY, March 16-18, 1992.

[8] S. F. Smith, "A learning system based on genetic algorithms," Ph.D. Dissertation, University of Pittsburgh, Pittsburgh, PA, 1980.

[9] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artificial Intelligence*, vol. 40, no. 1-3, pp. 235-282, September 1989.

[10] M. Valenzuela-Rendon, "The fuzzy classifier system: A classifier system for continuously varying variables," *Proc. of 4th International Conf. on Genetic Algorithms*, pp. 346-353, University of California, San Diego, CA, July 13-16, 1991.

[11] A. Parodi and P. Bonelli, "A new approach to fuzzy classifier systems," *Proc. of 5th International Conf. on Genetic Algorithms*, pp. 223-230, University of Illinois, Urbana-Champain, IL, July 17-21, 1993.

[12] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multi-dimensional pattern classification problems," *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 29, no. 5, pp. 601-618, October 1999.

[13] H. Ishibuchi and T. Nakashima, "Improving the performance of fuzzy classifier systems for pattern classification problems with continuous attributes," *IEEE Trans. on Industrial Electronics*, vol. 46, no. 6, pp. 1057-1068, December 1999.

[14] H. Ishibuchi, T. Nakashima, and T. Murata, "Three-objective genetics-based machine learning for linguistic rule extraction," *Information Sciences*, vol. 136, no. 1-4, pp. 109-133, August 2001.

[15] H. Ishibuchi and T. Nakashima, "Effect of rule weights in fuzzy rule-based classification systems," *IEEE Trans. on Fuzzy Systems*, vol. 9, no. 4, pp. 506-515, August 2001.

[16] J. van den Berg, U. Kaymak, and W. -M. van den Bergh, "Fuzzy classification using probability based rule weighting," *Proc. of 11th IEEE International Conference on Fuzzy Systems*, pp. 991-996, Honolulu, HI, May 12-17, 2002.

[17] H. Ishibuchi and T. Yamamoto, "Comparison of heuristic rule weight specification methods," *Proc. of 11th IEEE International Conference on Fuzzy Systems*, pp. 908-913, Honolulu, HI, May 12-17, 2002.

[18] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[19] T. Elomaa and J. Rousu, "General and efficient multisplitting of numerical attributes," *Machine Learning*, vol. 36, no. 3, pp. 201-244, September 1999.

[20] J. Abonyi, J. A. Roubos, and F. Szeifert, "Data-driven generation of compact, accurate, and linguistically-sound fuzzy classifiers based on a decision-tree initialization," *International Journal of Approximate Reasoning*, vol. 32, no. 1, pp. 1-21, January 2003.

[21] S. U. Guan and F. Zhu, "Class decomposition for GA-based classifier agents - A Pitt approach," *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 34, no. 1, pp. 381-392, February 2004.

Hisao Ishibuchi (M’93) received the B.S. and M.S. Degrees in precision mechanics from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively, and the Ph.D. degree from Osaka Prefecture University, Osaka, Japan, in 1992.

Since 1987, he has been with Department of Industrial Engineering, Osaka Prefecture University, where he is currently a Professor. He was a Visiting Research Associate at the University of Toronto, Toronto, ON, Canada, from August 1994 to March 1995 and from July 1997 to March 1998. His research interests include fuzzy rule-based classification systems, evolutionary multiobjective optimization, and data mining.

Dr. Ishibuchi received the best paper award at the GECCO 2004 conference. He is currently an Associate Editor for *IEEE Trans. on Systems, Man, and Cybernetics - Part B* and *IEEE Trans. on Fuzzy Systems*. He is also a fuzzy technical committee member of the IEEE Computational Intelligence Society.

Takashi Yamamoto (SM’01-M’04) received the B.S., M.S. and Ph.D. Degrees in industrial engineering from Osaka Prefecture University, Osaka, Japan, in 2001, 2003 and 2004, respectively. Since 2004, he has been with NTT DoCoMo, Inc., Tokyo, Japan.

Dr. Yamamoto received the best student paper award at the 4th International Conference on Intelligent Technologies (InTech 2004, Chang Mai, Thailand).

Tomoharu Nakashima (SM’99-M’01) received the B.S., M.S., and Ph.D. degrees in industrial engineering from Osaka Prefecture University, Osaka, Japan, in 1995, 1997, and 2000, respectively. He is currently an Assistant Professor, Department of Industrial Engineering, Osaka Prefecture University. From June 1998 to September 1998, he was a postgraduate scholar at the Knowledge-Based Intelligent Engineering Systems Centre, University of South Australia. From July 2004 to March 2005, he was a visiting fellow at the Nottingham Trent University, U.K. His research interests include fuzzy systems, data mining, evolutionary computing, multi-agent systems, and financial engineering.