Fuzzy Rule Selection by Multi-Objective Genetic Local Search Algorithms and Rule Evaluation Measures in Data Mining

Hisao Ishibuchi* **and Takashi Yamamoto**

Department of Industrial Engineering, Osaka Prefecture University

1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan

{hisaoi, yama}@ie.osakafu-u.ac.jp

* Corresponding Author. Phone: +81-72-254-9350; Fax: +81-72-254-9915; hisaoi@ie.osakafu-u.ac.jp

Abstract

This paper shows how a small number of simple fuzzy if-then rules can be selected for pattern classification problems with many continuous attributes. Our approach consists of two phases: Candidate rule generation by rule evaluation measures in data mining and rule selection by multi-objective evolutionary algorithms. In our approach, first candidate fuzzy if-then rules are generated from numerical data and prescreened using two rule evaluation measures (i.e., *confidence* and *support*) in data mining. Then a small number of fuzzy if-then rules are selected from the prescreened candidate rules using multi-objective evolutionary algorithms. In rule selection, we use three objectives: maximization of the classification accuracy, minimization of the number of selected rules, and minimization of the total rule length. Thus the task of multi-objective evolutionary algorithms is to find a number of non-dominated rule sets with respect to these three objectives. The main contribution of this paper is to propose an idea of utilizing the two rule evaluation measures as prescreening criteria of candidate rules for fuzzy rule selection. An arbitrarily specified number of candidate rules can be generated from numerical data for high-dimensional pattern classification problems. Through computer simulations, we demonstrate that such a prescreening procedure improves the efficiency of our approach to fuzzy rule selection. We also extend a multi-objective genetic algorithm (MOGA) in our former studies to a multi-objective genetic local search (MOGLS) algorithm where a local search procedure adjusts the selection (i.e., inclusion or exclusion) of each candidate rule. Furthermore, a learning algorithm of rule weights (i.e., certainty factors) is combined with our MOGLS algorithm. Such extensions to our MOGA for fuzzy rule selection are another contribution of this paper.

Keywords: Data mining, pattern classification, fuzzy rule selection, evolutionary multi-criterion optimization, hybrid genetic algorithms.

1. Introduction

Fuzzy rule-based systems have been successfully applied to various application areas such as control and classification [20,21]. While the main objective in the design of fuzzy rule-based systems has been the performance maximization, their comprehensibility has also been taken into account in some recent studies [3,4,19,24,25,27,28]. The comprehensibility of fuzzy rule-based systems is related to various factors:

(i) Comprehensibility of fuzzy partitions (e.g., linguistic interpretability of each fuzzy set, separation of neighboring fuzzy sets, the number of fuzzy sets for each variable).

(ii) Simplicity of fuzzy rule-based systems (e.g., the number of input variables, the number of fuzzy if-then rules).

(iii) Simplicity of fuzzy if-then rules (e.g., type of fuzzy if-then rules, the number of antecedent conditions in each fuzzy if-then rule).

(iv) Simplicity of fuzzy reasoning (e.g., selection of a single winner rule, voting by multiple rules).

In this paper, we show how a small number of simple fuzzy if-then rules can be selected for designing a comprehensible fuzzy rule-based system for a pattern classification problem with many continuous attributes. Among the above four issues, the second and third ones are mainly discussed in this paper. The first issue (i.e., comprehensibility of fuzzy partitions) is considered in this paper as a part of a preprocessing procedure for fuzzy rule generation. That is, we assume that the domain interval of each continuous attribute has already been discretized into several fuzzy sets. In computer simulations, we use simple homogeneous fuzzy partitions. See [19,24,25,27,28] for the determination of comprehensible fuzzy partitions from numerical data. Partition methods into non-fuzzy intervals have been studied in the area of machine learning (e.g., [6,7,26]).

A straightforward approach to the design of simple fuzzy rule-based systems is rule selection. In our former studies [14,15], we proposed a genetic algorithm-based approach for selecting a small number of fuzzy if-then rules from a large number of candidate rules. The GA-based approach was extended to the case of two-objective rule selection for explicitly considering a tradeoff between the number of fuzzy if-then rules and the classification accuracy [10]. This approach was further extended in [12] to the case of three-objective rule selection by including the minimization of the total rule length (i.e., total number of antecedent conditions). When the GA-based rule selection approach is applied to high-dimensional pattern classification problems, a prescreening procedure of candidate rules is necessary because the number of possible fuzzy if-then rules exponentially increases with the dimensionality of pattern spaces. In [10,14,15], the GA-based approach was only applied to low-dimensional pattern classification problems where no prescreening procedure was necessary for decreasing the number of candidate rules. A simple prescreening procedure based on rule length was used for handling high-dimensional problems in

[12]. In this paper, we propose an idea of utilizing rule evaluation measures in data mining as prescreening criteria. An arbitrarily specified number of candidate rules can be generated from numerical data for high-dimensional pattern classification problems using rule evaluation measures. We also extend our multi-objective genetic algorithm (MOGA) in [10,12] to a multi-objective genetic local search (MOGLS) algorithm by combining local search and rule weight learning with our MOGA.

This paper is organized as follows. In the next section, we show how we can use two rule evaluation measures (i.e., *confidence* and *support* of association rules) in data mining for prescreening candidate fuzzy if-then rules. Three prescreening criteria (i.e., confidence, support, and their product) are compared with one another through computer simulations on a wine classification problem with 13 continuous attributes. We also examine some alternative heuristic definitions of rule weights through computer simulations. In Section 3, we describe our MOGA designed for finding non-dominated rule sets with respect to three objectives from candidate rules. Through computer simulations, we demonstrate that the efficiency of our MOGA can be improved by the use of a prescreening procedure. In Section 4, we implement an MOGLS algorithm by combining local search and rule weight learning with our MOGA. In Section 5, we examine the generalization ability of rule sets obtained by our approach through computer simulations on several pattern classification problems. Section 6 concludes this paper.

2. Candidate Rule Generation from Numerical Data

2.1 Fuzzy If-Then Rules for Pattern Classification Problems

In our approach, first fuzzy if-then rules are generated from numerical data. Then the generated rules are used as candidate rules from which a small number of fuzzy if-then rules are selected by multi-objective genetic algorithms. Let us assume that we have m labeled patterns $\boldsymbol{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1, 2..., m$ from M classes in an n-dimensional continuous pattern space. We also assume that the domain interval of each attribute x_i is discretized into K_i linguistic values (i.e., K_i fuzzy sets with linguistic labels). Some typical examples of fuzzy discretization are shown in Fig. 1.

We use fuzzy if-then rules of the following form for our n-dimensional pattern classification problem:

$$\text{Rule } R_q\text{: If } x_1 \text{ is } A_{q1} \text{ and } ... \text{ and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \qquad (1)$$

where R_q is the label of the q-th fuzzy if-then rule, $\boldsymbol{x} = (x_1, ..., x_n)$ is an n-dimensional pattern vector, A_{qi} is an antecedent fuzzy set, C_q is a consequent class (i.e., one of the M classes), and CF_q is a rule

weight (i.e., certainty factor). The rule weight CF_q is a real number in the unit interval [0, 1]. While some studies (e.g., Castillo et al.[3] and Castro et al.[4]) used an arbitrary disjunction of multiple linguistic values as an antecedent fuzzy set (e.g., A_{qi} = "*small* or *large*"), we use only a single linguistic value as A_{qi} for keeping each fuzzy if-then rule simple. It should be noted that some antecedent conditions can be "*don't care*". Since *don't care* conditions are usually omitted, the number of antecedent conditions is not always *n* in our fuzzy if-then rules. Some fuzzy if-then rules may have *n* antecedent conditions (i.e., have no *don't care* conditions), and others may have only a few antecedent conditions (i.e., have many *don't care* conditions). It is easy for human users to understand short fuzzy if-then rules with only a few antecedent conditions. Thus the number of antecedent conditions in each fuzzy if-then rule (i.e., the rule length) is minimized in our fuzzy rule selection method.

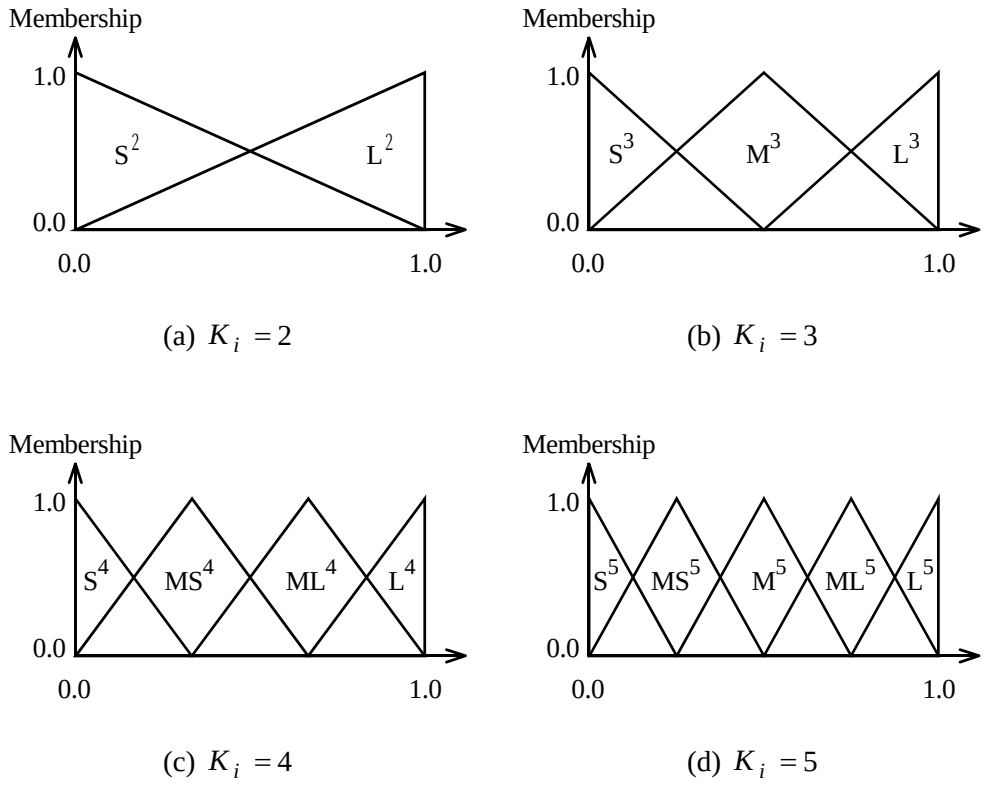

Fig. 1 Some typical examples of fuzzy partitions of the domain interval [0, 1]. The meaning of each label is as follows: S: *small*, MS: *medium small*, M: *medium*, ML: *medium large*, and L: *large*. The superscript of each label denotes the granularity of the corresponding fuzzy partition (i.e., the value of K_i).

Since each antecedent fuzzy set A_{qi} in (1) can be one of the K_i fuzzy sets or *don't care*, the total number of possible combinations of antecedent fuzzy sets is $(K_1+1)\times\cdots\times(K_n+1)$. We can examine all combinations for rule generation and use the generated fuzzy if-then rules as candidate rules for rule selection in the case of low-dimensional pattern classification problems (i.e., when *n* is small). Thus no prescreening of candidate rules is necessary. On the other hand, we need some prescreening procedure in the application of our approach to high-dimensional pattern classification problems since the total number of possible combinations of antecedent fuzzy sets exponentially increases with the dimensionality of pattern spaces. Our idea is to use rule evaluation measures in data mining for decreasing the number of candidate fuzzy if-then rules.

2.2 Confidence and Support of Fuzzy If-Then Rules

In the area of data mining, two measures called *confidence* and *support* have often been used for evaluating association rules [2]. Our fuzzy if-then rule in (1) can be viewed as a kind of association rule of the form $\boldsymbol{A}_q \Rightarrow C_q$ where $\boldsymbol{A}_q = (A_{q1}, ..., A_{qn})$. We use these two measures for prescreening candidate fuzzy if-then rules. In this subsection, we show how the definitions of these two measures can be extended to the case of the fuzzy if-then rule $\boldsymbol{A}_q \Rightarrow C_q$ [16].

Let D be the set of the m training patterns $\boldsymbol{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1, 2..., m$ in our pattern classification problem. Thus the cardinality of D is m (i.e., $|D| = m$). The confidence of $\boldsymbol{A}_q \Rightarrow C_q$ is defined as follows [2]:

$$c(\boldsymbol{A}_q \Rightarrow C_q) = \frac{|D(\boldsymbol{A}_q) \cap D(C_q)|}{|D(\boldsymbol{A}_q)|}, \tag{2}$$

where $|D(\boldsymbol{A}_q)|$ is the number of training patterns that are compatible with the antecedent $\boldsymbol{A}_q$, and $|D(\boldsymbol{A}_q) \cap D(C_q)|$ is the number of training patterns that are compatible with both the antecedent $\boldsymbol{A}_q$ and the consequent C_q. The confidence *c* indicates the grade of the validity of $\boldsymbol{A}_q \Rightarrow C_q$. That is, *c* ($\times 100\%$) of training patterns that are compatible with the antecedent $\boldsymbol{A}_q$ are also compatible with the consequent C_q. In the case of standard association rules in data mining, both the antecedent $\boldsymbol{A}_q$ and the consequent C_q are not fuzzy. Thus the calculations of $|D(\boldsymbol{A}_q)|$ and $|D(\boldsymbol{A}_q) \cap D(C_q)|$ can be performed by simply counting compatible training patterns. On the other hand, each training pattern has a different compatibility grade $\mu_{\boldsymbol{A}_q}(\boldsymbol{x}_p)$ with the antecedent $\boldsymbol{A}_q$ when $\boldsymbol{A}_q \Rightarrow C_q$ is a fuzzy if-then

rule. Thus such a compatibility grade should be taken into account. Since the consequent C_q is not fuzzy (i.e., C_q is a class label), the confidence in (2) can be rewritten as follows [16]:

$$c(\boldsymbol{A}_q \Rightarrow C_q) = \frac{|D(\boldsymbol{A}_q) \cap D(C_q)|}{|D(\boldsymbol{A}_q)|} = \frac{\sum_{p \in \text{Class } C_q} \mu_{\boldsymbol{A}_q}(\boldsymbol{x}_p)}{\sum_{p=1}^{m} \mu_{\boldsymbol{A}_q}(\boldsymbol{x}_p)} . \tag{3}$$

The compatibility grade $\mu_{\boldsymbol{A}_q}(\boldsymbol{x}_p)$ is usually defined by the minimum operator or the product operator. In this paper, we use the product operator as

$$\mu_{\boldsymbol{A}_q}(\boldsymbol{x}_p) = \mu_{A_{q1}}(x_{p1}) \times \cdots \times \mu_{A_{qn}}(x_{pn}), \tag{4}$$

where $\mu_{A_{qi}}(\cdot)$ is the membership function of the antecedent fuzzy set A_{qi}.

On the other hand, the support of $\boldsymbol{A}_q \Rightarrow C_q$ is defined as follows [2]:

$$s(\boldsymbol{A}_q \Rightarrow C_q) = \frac{|D(\boldsymbol{A}_q) \cap D(C_q)|}{|D|} . \tag{5}$$

The support s indicates the grade of the coverage by $\boldsymbol{A}_q \Rightarrow C_q$. That is, s ($\times 100\%$) of all the training patterns are compatible with the association rule $\boldsymbol{A}_q \Rightarrow C_q$ (i.e., compatible with both the antecedent $\boldsymbol{A}_q$ and the consequent C_q). In the same manner as the confidence in (3), the support in (5) can be rewritten as follows [16]:

$$s(\boldsymbol{A}_q \Rightarrow C_q) = \frac{|D(\boldsymbol{A}_q) \cap D(C_q)|}{|D|} = \frac{\sum_{p \in \text{Class } C_q} \mu_{\boldsymbol{A}_q}(\boldsymbol{x}_p)}{m} . \tag{6}$$

2.3 Prescreening of Candidate Rules

The confidence and the support can be used as prescreening criteria for finding a tractable number of candidate fuzzy if-then rules. We also use the product of these two measures as another prescreening criterion. For generating candidate rules, we first determine the consequent class C_q for each combination $\boldsymbol{A}_q = (A_{q1}, ..., A_{qn})$ of antecedent fuzzy sets using the confidence measure as

$$c(\boldsymbol{A}_q \Rightarrow C_q) = \max\{c(\boldsymbol{A}_q \Rightarrow \text{Class } 1), ..., c(\boldsymbol{A}_q \Rightarrow \text{Class } M)\}. \quad (7)$$

It should be noted that the same class C_q is obtained for $\boldsymbol{A}_q$ when we use the support instead of the confidence in (7):

$$s(\boldsymbol{A}_q \Rightarrow C_q) = \max\{s(\boldsymbol{A}_q \Rightarrow \text{Class } 1), ..., s(\boldsymbol{A}_q \Rightarrow \text{Class } M)\}. \quad (8)$$

This is because the following relation holds between the confidence and the support from their definitions:

$$s(\boldsymbol{A}_q \Rightarrow \text{Class } t) = c(\boldsymbol{A}_q \Rightarrow \text{Class } t) \times \frac{|D(\boldsymbol{A}_q)|}{|D|}, \quad t = 1, 2, ..., M. \quad (9)$$

Since the second term (i.e., $|D(\boldsymbol{A}_q)|/|D|$) of the right-hand side is independent of the consequent class, the class with the maximum confidence in (7) is the same as the class with the maximum support in (8). The same class also has the maximum product of these two measures. Thus usually we can uniquely specify the consequent class C_q for each combination $\boldsymbol{A}_q$ of antecedent fuzzy sets independent of the choice of a prescreening criterion among the three measures (i.e., confidence, support, and their product). Only when multiple classes have the same maximum value (including the case of no compatible training pattern with the antecedent part $\boldsymbol{A}_q$: $s(\boldsymbol{A}_q \Rightarrow \text{Class } t) = 0$ for all classes), we cannot specify the consequent class C_q for $\boldsymbol{A}_q$. In this case, we do not generate the corresponding fuzzy if-then rule R_q.

The generated fuzzy if-then rules are divided into *M* groups according to their consequent classes. Fuzzy if-then rules in each group are sorted in descending order of a prescreening criterion (i.e., confidence, support, or their product). For selecting *N* candidate rules, the first N/M rules are chosen from each of the *M* groups. In this manner, we can choose an arbitrarily specified number of candidate fuzzy if-then rules (i.e., *N* candidate rules). It should be noted that the aim of the candidate rule prescreening is not to construct a fuzzy rule-based system but to find candidate rules, from which a small number of fuzzy if-then rules are selected. For using a variety of candidate rules in rule selection, we choose the same number of fuzzy if-then rules (i.e., N/M candidate rules) for each class. While the same number of candidate rules are chosen, a different number of fuzzy if-then rules may be finally selected for each class by genetic algorithm-based rule selection.

As we have already mentioned, the total number of possible combinations of antecedent fuzzy sets is $(K_1 + 1) \times \cdots \times (K_n + 1)$ for an *n*-dimensional pattern classification problem. Thus it is impractical to

examine all combinations when n is large. In this case, we examine only short fuzzy if-then rules with a small number of antecedent conditions (i.e., with a large number of *don't care* conditions). When each attribute has K fuzzy sets (i.e., $K_i = K$ for all i's), the number of fuzzy if-then rules of the length L is calculated as ${}_n C_L \times K^L$. Even when n is large, ${}_n C_L \times K^L$ is not so large for a small L. This means that the number of short fuzzy if-then rules is not so large even when the total number of possible rules is huge.

2.4 Computer Simulations

We illustrate our prescreening procedure through computer simulations on wine data available from the UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/MLSummary.html). The wine data include 178 patterns with 13 continuous attributes from three classes. First we normalized each attribute value into a real number in the unit interval [0, 1]. Thus the wine data set was handled as a three-class pattern classification problem in the 13-dimensional unit hypercube $[0, 1]^{13}$. We used the five linguistic values in Fig. 1 (d) for each attribute.

The total number of possible combinations of antecedent fuzzy sets is $(5+1)^{13} = 1.31 \times 10^{10}$ since we used the five linguistic values and *don't care* for each of the 13 attributes. The examination of all combinations of antecedent fuzzy sets is time-consuming. Thus we generated only fuzzy if-then rules of the length three or less. The number of generated fuzzy if-then rules of each length is summarized in Table 1. As we have already mentioned, ${}_{13} C_L \times 5^L$ combinations of antecedent fuzzy sets were examined for generating fuzzy if-then rules of the length L. Fuzzy if-then rules were not generated when their consequent classes cannot be uniquely specified by (7).

Table 1. The number of generated fuzzy if-then rules of each length for the wine data.

Length of rules	0	1	2	3
Number of rules	1	65	1,768	25,589

The generated 27423 fuzzy if-then rules were divided into three groups according to their consequent classes. The number of fuzzy if-then rules of each group was as follows:

Class 1: 7,554 rules,
Class 2: 12,464 rules,
Class 3: 7,405 rules.

Fuzzy if-then rules in each group were sorted in descending order of a prescreening criterion. When

multiple rules had the same value of the prescreening criterion, those rules were randomly ordered. That is, such a tie situation was randomly broken (i.e., random tiebreak). Then the first $N/3$ rules were chosen from each of the three groups for finding *N* candidate rules.

For examining the classification performance of the selected *N* candidate rules, we used a single winner rule method in the classification phase as in our previous studies on fuzzy rule-based classification systems [10-16]. Let *S* be the set of the selected *N* fuzzy if-then rules. Thus *S* can be viewed as a fuzzy rule-based classification system. For classifying an input pattern $\boldsymbol{x}_p = (x_{p1}, ..., x_{pn})$ by this classification system, a singe winner rule R_{q^*} is selected from the rule set *S* as

$$\mu_{\boldsymbol{A}_{q^*}}(\boldsymbol{x}_p) \cdot CF_{q^*} = \max\{\mu_{\boldsymbol{A}_q}(\boldsymbol{x}_p) \cdot CF_q \mid R_q \in S\}. \tag{10}$$

That is, the winner rule has the maximum product of the compatibility grade $\mu_{\boldsymbol{A}_q}(\boldsymbol{x}_p)$ and the rule weight CF_q. When multiple fuzzy rules with different consequent classes have the same maximum product in (10), the classification of $\boldsymbol{x}_p$ is rejected.

There are several alternative methods for determining the rule weight of each fuzzy if-then rule. One method is to directly use the confidence as the rule weight:

$$CF_q = c(\boldsymbol{A}_q \Rightarrow C_q). \tag{11}$$

This definition was used in Cordon et al.[5]. In our former studies [10-16], we used a different definition. Our previous definition can be rewritten using the confidence measure as

$$CF_q = c(\boldsymbol{A}_q \Rightarrow C_q) - \bar{c}, \tag{12}$$

where $\bar{c}$ is the average confidence over the $(M-1)$ classes except for the consequent class C_q:

$$\bar{c} = \frac{1}{M-1} \sum_{\substack{t=1 \\ t \neq C_q}}^{M} c(\boldsymbol{A}_q \Rightarrow \text{Class } t). \tag{13}$$

In the definition in (12), the rule weight CF_q is discounted by the average confidence $\bar{c}$ of fuzzy if-then rules with the same antecedent $\boldsymbol{A}_q$ and different consequent classes. In this paper, we propose the following definition:

$$CF_q = c(\boldsymbol{A}_q \Rightarrow C_q) - c(\boldsymbol{A}_q \Rightarrow C_{q^{**}}), \tag{14}$$

where $C_{q^{**}}$ is the class with the second largest confidence for the antecedent $\boldsymbol{A}_q$:

$$c(\boldsymbol{A}_q \Rightarrow C_{q^{**}}) = \max\{c(\boldsymbol{A}_q \Rightarrow \text{Class } t) \mid t = 1, 2, ..., M;\ t \neq C_q\}. \tag{15}$$

This definition of the rule weight CF_q is the same as (12)-(13) when pattern classification problems involve only two classes (i.e., when $M = 2$). In addition to these three definitions, we also examine the case of no rule weights. This case is examined by assigning the same rule weight to all fuzzy if-then rules (i.e., $CF_q = 1.0$ for all rules).

Using the four definitions of rule weights and the single winner method, we examined the classification performance of selected fuzzy if-then rules by each of the three prescreening criteria. When we examined the classification performance on training data, all the 178 patterns in the wine data were used for rule generation and performance evaluation. The average classification rate was calculated over 1000 runs for each combination of a rule weight definition and a prescreening criterion. Such a large number of runs were performed for decreasing the effect of the random tiebreak in the sorting of fuzzy if-then rules on simulation results. On the other hand, we used the leaving-one-out (LV1) procedure [29] for examining the classification performance on test data. In the LV1 procedure, 177 patterns were used as training data for rule generation. The remaining single pattern was used as test data for performance evaluation. This train-and-test procedure was iterated 178 times so that each of the 178 patterns was used as test data once. The whole LV1 procedure (i.e., 178 runs) was iterated 20 times for decreasing the effect of the random tiebreak on simulation results. Average classification rates by selected fuzzy if-then rules on training data and test data are summarized in Table 2 and Table 3, respectively. From those tables, we can see that the best performance was obtained from the combination of the product criterion ($c \cdot s$) and the definition in (14)-(15) when the number of selected fuzzy if-then rules are not too large (e.g., $N \leq 300$). Thus we use this combination in this paper hereafter. Good results were also obtained from the combination of the product criterion ($c \cdot s$) and the definition in (14)-(15) in computer simulations on other data sets (e.g., iris data and credit data) while we do not report them here. The iris data and the credit data will be used in computer simulations for examining the generalization ability of fuzzy rule-based systems in Section 5.

Table 2 Simulation results on training data. The best result in each column is indicated by "∗".

Prescreening criterion	Weight definition	Number of selected fuzzy rules									
		3	6	9	30	60	90	300	600	900	27423
c	(11)	8.5	15.8	22.0	49.9	68.1	77.2	93.6	97.8*	99.0*	94.9
c	(12)-(13)	8.5	15.8	22.0	49.9	68.1	77.2	93.6	97.8*	99.0*	96.6
c	(14)-(15)	8.5	15.8	22.0	49.9	68.1	77.2	93.6	97.8*	99.0*	99.4*
c	No weights	8.5	15.8	22.0	49.9	68.1	77.2	93.6	97.8*	99.0*	37.1
s	(11)	49.4	52.2	78.1	84.3	89.3	89.9	91.6	92.7	92.1	94.9
s	(12)-(13)	60.7	57.3	88.8	89.9	92.7	92.7	93.3	92.7	93.3	96.6
s	(14)-(15)	54.5	48.9	88.8	91.0	94.4	94.9*	96.1*	94.4	96.1	99.4*
s	No weights	39.9	39.9	39.9	39.9	39.9	39.9	39.9	39.3	39.3	37.1
$c \cdot s$	(11)	87.6	82.0	91.0	93.8	91.0	91.6	92.1	92.7	92.1	94.9
$c \cdot s$	(12)-(13)	89.3	88.8	93.8	94.9	92.7	93.8	93.3	93.3	93.8	96.6
$c \cdot s$	(14)-(15)	91.0*	91.0*	94.9*	96.1*	95.5*	94.9*	95.5	96.1	96.1	99.4*
$c \cdot s$	No weights	81.5	39.9	39.9	39.9	39.9	39.9	39.9	39.9	39.9	37.1

Table 3 Simulation results on test data. The best result in each column is indicated by "∗".

Prescreening criterion	Weight definition	Number of selected fuzzy rules									
		3	6	9	30	60	90	300	600	900	27423
c	(11)	7.1	14.5	19.9	47.1	65.1	73.8	89.5	92.8*	93.8*	90.4
c	(12)-(13)	7.1	14.5	19.9	47.1	65.1	73.8	89.5	92.8*	93.8*	91.6
c	(14)-(15)	7.1	14.5	19.9	47.1	65.1	73.8	89.5	92.8*	93.8*	93.3*
c	No weights	7.1	14.5	19.9	47.1	65.1	73.8	89.5	92.8*	93.8*	36.0
s	(11)	36.0	45.5	71.3	82.0	88.2	88.2	89.3	90.4	90.4	90.4
s	(12)-(13)	47.2	57.3	76.4	89.3	92.1	92.1	92.1	92.1	91.6	91.6
s	(14)-(15)	21.3	36.5	77.0	89.3	92.1	93.3*	93.3*	92.7	92.1	93.3*
s	No weights	39.9	39.9	39.9	39.9	39.9	39.9	39.9	39.3	39.3	36.0
$c \cdot s$	(11)	87.1	79.8	86.5	89.9	89.3	88.8	89.9	90.4	90.4	90.4
$c \cdot s$	(12)-(13)	88.8	89.3	93.3*	94.9	92.1	91.6	92.1	92.7	91.6	91.6
$c \cdot s$	(14)-(15)	90.4*	90.4*	93.3*	95.5*	93.8*	92.7	93.3*	92.7	92.1	93.3*
$c \cdot s$	No weights	81.5	28.7	39.9	39.9	39.9	39.9	39.9	39.9	39.3	36.0

From Table 2 and Table 3, we can see that rule weights have a significant effect on the classification performance of selected fuzzy if-then rules. When we did not use rule weights, classification rates were low. See [11] for further discussions on the effect of rule weights on the performance of fuzzy rule-based classification systems. Among the three definitions of rule weights, the direct use of the confidence in (11) is inferior to the other two definitions. Let us consider three fuzzy if-then rules $\boldsymbol{A}_q \Rightarrow \text{Class } 1$,

$\boldsymbol{A}_q \Rightarrow$ Class 2 and $\boldsymbol{A}_q \Rightarrow$ Class 3 with the same antecedent $\boldsymbol{A}_q$ and different consequent classes. Among these three rules, only a single fuzzy if-then rule R_q (i.e., $\boldsymbol{A}_q \Rightarrow C_q$) with the maximum confidence is generated by our rule generation method before the prescreening procedure. When these three rules have almost the same (but not exactly the same) confidence values, the rule weight CF_q of R_q is about $1/3$ in the direct use of the confidence in (11) while CF_q is almost zero in the other two definitions. In such a situation, we are not sure that an input pattern $\boldsymbol{x}_p$ is from C_q even when the compatibility of $\boldsymbol{x}_p$ with $\boldsymbol{A}_q$ is high. This means that the reliability of R_q is very low. Thus the rule weight is decreased to almost zero by the confidence values of the other rules in the definitions in (12)-(13) and (14)-(15) while it is about 1/3 in (11). This difference between the direct use of the confidence as the rule weight and the other two definitions leads to the difference in average classification rates. When we use the confidence criterion (*c*) for candidate rule prescreening, all the selected rules have high confidence. For example, the confidence of all the selected 900 fuzzy if-then rules in Table 2 is 1.00. This means that their rule weights are also 1.00 independent of the choice of a rule weight definition. Thus the same results were obtained from the four different definitions of rule weights in Table 2 and Table 3.

The difference between (12)-(13) and (14)-(15) is subtle. In the above-mentioned situation, almost the same rule weights are obtained by these two definitions (i.e., almost zero). For discussing the difference between (12)-(13) and (14)-(15), let us consider another situation. Now we assume that the confidence of $\boldsymbol{A}_q \Rightarrow$ Class 3 is zero and the other two rules have almost the same (but not exactly the same) confidence values. In this case, the rule weight CF_q of R_q is almost zero in (14)-(15) because the largest two confidence values are almost the same. On the other hand, the rule weight is about 0.25 in (12)-(13). When the largest two confidence values are almost the same, we are not sure that an input pattern $\boldsymbol{x}_p$ is from C_q even when the compatibility of $\boldsymbol{x}_p$ with $\boldsymbol{A}_q$ is high. Thus the rule weight is decreased to almost zero by the second largest confidence value in (14)-(15) while it is decreased to about 0.25 by the average confidence in (12)-(13). This difference between the definitions in (12)-(13) and (14)-(15) leads to the difference in average classification rates.

It is also shown in Table 2 and Table 3 that the product criterion outperforms the other prescreening criteria independent of the choice of a rule weight definition. For examining this observation further, we calculated the average length of selected 30 fuzzy if-then rules by each criterion in Table 2:

Confidence criterion: 2.93,
Support criterion: 1.13,
Product criterion: 1.47.

The confidence criterion tends to select long fuzzy if-then rules that have high confidence but low support. Such a fuzzy if-then rule can cover only a small number of patterns while its classification accuracy is high. Thus a small number of fuzzy if-then rules cannot classify many training patterns. This leads to low classification rates in Table 2 and Table 3 when a small number of fuzzy if-then rules were selected by the confidence criterion (*c*). While the best results were obtained from the confidence criterion (*c*) in Table 3 when a large number of candidate rules were selected (i.e., $N = 900$), we do not use this criterion for the candidate rule prescreening. This is because our final aim is to construct a fuzzy rule-based system by selecting a small number of fuzzy if-then rules from candidate rules. Since candidate rules selected by the confidence criterion (*c*) are very specific, high classification rates are not likely to be obtained by a small number of fuzzy if-then rules in the case of the confidence criterion (see Table 2 and Table 3).

On the other hand, the support criterion (*s*) tends to select short fuzzy if-then rules that have high support but low confidence. Such a fuzzy if-then rule may misclassify some patterns while it can cover many patterns. Good rule selection criteria may be obtained from the combination of the confidence and the support by finding a good tradeoff between the specificity and the generality. The product criterion ($c \cdot s$) is an attempt to find such a good tradeoff. Actually better results were obtained by the product criterion ($c \cdot s$) in Table 2 and Table 3 than the confidence criterion (*c*) and the support criterion (*s*) when the number of selected candidate rules was not too large.

3. Rule Selection

We have already explained how an arbitrarily specified number of fuzzy if-then rules can be generated from numerical data as candidate rules for rule selection. In this section, we describe a multi-objective genetic algorithm (MOGA) that is designed for finding non-dominated rules sets with respect to three objectives of our rule selection problem. Our MOGA will be extended to a multi-objective genetic local search (MOGLS) algorithm in the next section.

3.1 Three-Objective Optimization Problem

Let us assume that we have *N* candidate fuzzy if-then rules. Our task is to select a small number of simple fuzzy if-then rules with high classification performance. This task is performed by maximizing the classification accuracy, minimizing the number of selected rules, and minimizing the total rule length. That is, we formulate our rule selection problem as follows [12]:

$$\text{Maximize } f_1(S), \text{ minimize } f_2(S), \text{ and minimize } f_3(S), \tag{16}$$

where $f_1(S)$ is the number of correctly classified training patterns by a rule set *S*, $f_2(S)$ is the

number of fuzzy if-then rules in *S*, and $f_3(S)$ is the total rule length of fuzzy if-then rules in *S*.

Usually there is no optimal rule set with respect to all the above three objectives. Thus our task is to find multiple rule sets that are not dominated by any other rule sets. A rule set S_B is said to dominate another rule set S_A (i.e., S_B is better than S_A: $S_A \prec S_B$) if all the following inequalities hold:

$$f_1(S_A) \le f_1(S_B), \quad f_2(S_A) \ge f_2(S_B), \quad f_3(S_A) \ge f_3(S_B), \tag{17}$$

and at least one of the following inequalities holds:

$$f_1(S_A) < f_1(S_B), \quad f_2(S_A) > f_2(S_B), \quad f_3(S_A) > f_3(S_B). \tag{18}$$

The first condition (i.e., all the three inequalities in (17)) means that no objective of S_B is worse than S_A (i.e., S_B is not worse than S_A). The second condition (i.e., one of the three equalities in (18)) means that at least one objective of S_B is better than S_A. When a rule set S is not dominated by any other rule sets (i.e., any other subsets of the *N* candidate rules), *S* is said to be a Pareto-optimal solution of our three-objective rule selection problem in (15). In many cases, it is impractical to try to find true Pareto-optimal solutions. Thus, multi-objective genetic algorithms usually show non-dominated rule sets among examined ones as approximate Pareto-optimal solutions.

3.2 Implementation of MOGA

Many evolutionary multi-criterion optimization (EMO) algorithms have been proposed (see [30,31]). Since each rule set can be represented by a binary string, we can apply those algorithms to our rule selection problem. In this paper, we use a slightly modified version of a multi-objective genetic algorithm (MOGA) in our former studies [10,12] because our MOGA is easy to implement. It is also easy to extend our MOGA to hybrid algorithms with local search and rule weight learning as shown in the next section. Our MOGA has two characteristic features. One is to use a scalar fitness function with random weights for evaluating each solution (i.e., each rule set). Random weights are updated whenever a pair of parent solutions is selected for crossover. That is, each selection is governed by different weights. Genetic search in various directions in the three-dimensional objective space is realized by this random weighting scheme. The other characteristic feature is to store all non-dominated solutions as a secondary population separately from a current population. This secondary population is updated at every generation. A small number of non-dominated solutions are randomly chosen from the secondary population and their copies are added to the current population as elite solutions. The convergence speed of the current population to Pareto-optimal solutions is improved by this elitist strategy. Other parts of our MOGA are the same as a

standard single-objective genetic algorithm.

An arbitrary subset *S* of the *N* candidate rules can be represented by a binary string of the length *N* as

$$S = s_1 s_2 \cdots s_N, \tag{19}$$

where $s_q = 0$ means that the *q*-th candidate rule R_q is not included in the rule set S while $s_q = 1$ means that R_q is included in S. An initial population is constructed by randomly generating a pre-specified number of binary strings of the length *N*.

The first objective $f_1(S)$ of each string S is calculated by classifying all the given training patterns by *S*. Since we use the single winner rule method, the classification is performed by finding a single winner rule for each pattern. Thus it is possible that some rules are not chosen as winner rules for any patterns. We can remove those rules without degrading the classification accuracy of the rule set *S*. At the same time, the second and third objectives are improved by removing unnecessary rules. Thus we remove all fuzzy if-then rules that are not selected as winner rules of any patterns from the rule set *S*. The removal of unnecessary rules is performed for each string of the current population by changing the corresponding 1's to 0's. From the combinatorial nature of our rule selection problem, some rules with no contribution in one rule set may have large contribution in another rule set. Thus we cannot remove any rules from all the strings in the current population without examining each string. The second and third objectives are calculated for each string after unnecessary rules are removed.

When the three objectives of each string in the current population are calculated, the secondary population is updated so that it includes all the non-dominated strings among examined ones during the execution of our MOGA. That is, each string in the current population is examined whether it is dominated by other strings in the current and secondary populations. If it is not dominated by any other strings, its copy is added to the secondary population. Then all strings dominated by the newly added one are removed from the secondary population. In this manner, the secondary population is updated at every generation.

The fitness value of each string S (i.e., each rule set *S*) in the current population is defined by the three objectives as

$$fitness(S) = w_1 \cdot f_1(S) - w_2 \cdot f_2(S) - w_3 \cdot f_3(S), \tag{20}$$

where w_1, w_2 and w_3 are weights satisfying the following conditions:

$$w_1, w_2, w_3 \geq 0, \tag{21}$$

$$w_1 + w_2 + w_3 = 1. \tag{22}$$

As we have already mentioned, one characteristic feature of our MOGA is to randomly specify these weights whenever a pair of parent strings is selected from the current population. In our former studies [10,12], we used a roulette wheel for selecting parent strings. Thus we had to calculate (20) for all strings in the current population to select a pair of parent strings. For selecting another pair of parent strings, we had to calculate (20) for all strings in the current population again using different weights. This is time-consuming especially when the population size is large. In this paper, we use binary tournament selection with replacement instead of roulette wheel selection to avoid such a time-consuming calculation. We have to calculate (20) for only four strings when a pair of parent strings is selected using the binary tournament selection (i.e., two strings for each parent). A pre-specified number of pairs are selected from the current population using the binary tournament selection.

Uniform crossover is applied to each pair of parent strings to generate a new string. Biased mutation is applied to the generated string for efficiently decreasing the number of fuzzy if-then rules included in each string. That is, different mutation probabilities are used for the mutation from 1 to 0 and that from 0 to 1. For example, the mutation from 1 to 0 may have a probability 0.1 even when the mutation probability from 0 to 1 is 0.001. A larger probability is assigned to the mutation from 1 to 0 than that from 0 to 1 for efficiently decreasing the number of fuzzy if-then rules (i.e., the number of 1's) included in each string. Both the biased mutation and the above-mentioned removal of unnecessary rules are used for efficiently decreasing the number of fuzzy if-then rules in each string. As we will show in the next subsection, the reduction in computation time by these two heuristics is significant. Without them, we cannot apply our MOGA to a large number of candidate rules. This is because the computation time required for the evaluation of each string strongly depends on the number of fuzzy if-then rules (i.e., because the computation time required for the classification of each training pattern strongly depends on the number of fuzzy if-then rules). At the same time, these two heuristics may have a negative effect on the evolutionary search for rule sets. That is, they may provoke the premature convergence of the population to strings with a small number of fuzzy if-then rules but low accuracy. Such a negative effect, however, is not clear in our computer simulations reported in the next subsection. This is because the storage of non-dominated strings in the secondary population and the use of them as elite strings prevent the genetic search from losing the variety of the population. Of course, the two heuristics are not always necessary especially when the number of candidate rules is small as shown in the next subsection. We use these two heuristics for handling a large number of candidate rules.

The next population consists of the newly generated strings by the selection, crossover, and mutation. Some non-dominated strings in the secondary population are randomly selected as elite solutions and their

copies are added to the new population.

Our MOGA is summarized as follows:

Step 0: Parameter Specification.

Specify the population size N_{pop}, the number of elite solutions N_{elite} that are randomly selected from the secondary population and added to the current population, the crossover probability p_c, two mutation probabilities $p_m\ (1 \to 0)$ and $p_m\ (0 \to 1)$, and the stopping condition.

Step 1: Initialization.

Randomly generate N_{pop} binary strings of the length N as an initial population. For generating each initial string, 0 and 1 are randomly chosen with the equal probability (i.e., 0.5). Calculate the three objectives of each string. In this calculation, unnecessary rules are removed from each string. Find non-dominated strings in the initial population. A secondary population consists of copies of those non-dominated strings.

Step 2: Genetic Operations.

Generate ($N_{pop} - N_{elite}$) strings by genetic operations (i.e., binary tournament selection, uniform crossover, biased mutation) from the current population.

Step 3: Evaluation.

Calculate the three objectives of each of the newly generated ($N_{pop} - N_{elite}$) strings. In this calculation, unnecessary rules are removed from each string. The current population consists of the modified strings.

Step 4: Secondary Population Update.

Update the secondary population by examining each string in the current population as mentioned above.

Step 5: Elitist Strategy.

Randomly select N_{elite} strings from the secondary population and add their copies to the current population.

Step 6. Termination Test.

If the stopping condition is not satisfied, return to Step 2. Otherwise terminate the execution of the algorithm. All the non-dominated strings among examined ones in the execution of the algorithm are stored in the secondary population.

3.3 Effect of Prescreening of Candidate Rules

We applied our MOGA to the three-objective rule selection problem. All the 178 patterns in the wine data set were used as training data in computer simulations of this subsection. As candidate rules, we used

900 fuzzy if-then rules generated in Subsection 2.4 using the product criterion (see Table 2). Parameter values were specified in Step 0 of our MOGA as

Population size: $N_{pop} = 50$,

Number of elite solutions: $N_{elite} = 5$,

Crossover probability: $p_c = 0.9$,

Mutation probability: $p_m(1 \to 0) = 0.1$ and $p_m(0 \to 1) = 1/N$,

Stopping condition: 10,000 generations.

These parameter specifications are almost the same as our former study [12] except for the stopping condition. We iterated our MOGA much longer than [12] where the stopping condition was 1000 generations. This is because we also applied our MOGA to the rule selection problem with much more candidate rules as shown in this subsection later. Simulation results are summarized in Table 4. This table shows non-dominated solutions with high classification rates (i.e., higher than 90%) obtained by a single trial of our MOGA. We can obtain multiple non-dominated rule sets by a single run of our MOGA. This is one advantage of our multi-objective approach to rule selection over single-objective ones. From the comparison between Table 2 (i.e., rule selection by the prescreening criteria) and Table 4 (by our MOGA), we can see that our MOGA could find rule sets with much higher classification rates than the rule prescreening criteria. As shown in Table 2, the 900 candidate rules selected by the product criterion ($c \cdot s$) have a 96.1% classification rate on the training data. Our MOGA found a rule set of seven fuzzy if-then rules with a 100% classification rate from those candidate rules in Table 4. While the maximum classification rate by six fuzzy if-then rules was 91.0% in Table 2, it is 99.4% in Table 4. These observations suggest the possibility that the GA-based rule selection can improve the quality of extracted rule sets by data mining techniques.

Table 4 Non-dominated rule sets obtained from 900 candidate rules.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100\times f_1(S)/m$ (%)
3	1.00	91.6
3	1.33	93.3
4	1.00	95.5
4	1.25	96.1
4	1.50	97.2
5	1.40	97.8
5	1.60	98.3
6	1.33	98.9
6	2.00	99.4
7	1.57	99.4
7	2.00	100.0

For examining the effect of the prescreening procedure on the search ability of our MOGA, we also performed the same computer simulation using various specifications of the number of candidate rules: $N = 9$, 90, and 27423 where the product criterion was used for prescreening. Since candidate rules were selected from 27423 fuzzy if-then rules (see Subsection 2.4), the specification of N as $N = 27423$ means that no prescreening was used. Simulation results are summarized in Table 5 ~ Table 7. In Table 5, seven fuzzy if-then rules have a 96.1% classification rate while the classification rate of the nine candidate rules was 94.9% in Table 2. This means that the classification rate was improved by removing two candidate rules by the MOGA. One the other hand, seven fuzzy if-then rules in Table 6 have a 99.4% classification rate while the classification rate of the 90 candidate rules was 94.9% in Table 2. In this case, the 4.5% increase in the classification rate was obtained by removing 83 candidate rules. These observations suggest that better results can be obtained from the combination of the heuristic rule prescreening and the genetic algorithm-based rule selection than the heuristic rule prescreening only.

Table 5 Non-dominated rule sets obtained from 9 candidate rules.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100\times f_1(S)/m$ (%)
3	1.00	91.6
4	1.00	94.9
5	1.00	95.5
7	0.86	96.1

Table 6 Non-dominated rule sets obtained from 90 candidate rules.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100\times f_1(S)/m$ (%)
3	1.00	91.6
3	1.33	93.3
4	1.00	94.9
4	1.25	96.1
4	1.50	96.6
5	1.00	97.2
5	1.40	98.3
6	1.00	97.8
7	1.14	98.9
7	1.29	99.4

Table 7 Non-dominated rule sets obtained from 27423 candidate rules with no prescreening.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100\times f_1(S)/m$ (%)
3	1.00	91.6
3	1.33	93.3
4	1.00	95.5
4	1.25	96.1
4	1.50	96.6
5	1.20	97.2
5	1.40	97.8
5	1.80	98.3
6	1.33	98.3
6	1.67	98.9
6	2.00	99.4
7	1.43	99.4
7	2.14	100.0
8	1.63	100.0

From these tables, we can see that good results were obtained when N was 90 in Table 6. Simulation results in Table 5 with $N = 9$ are slightly inferior to those with $N = 90$ and $N = 900$. This means that the performance of our rule selection method was slightly deteriorated when the number of candidate rules was too small. From careful comparison between Table 4 and Table 7, we can see that three rule sets in Table 7 are dominated by rule sets in Table 4. That is, the performance of our rule selection method was slightly deteriorated when the number of candidate rules was too large. This observation suggests the necessity of the prescreening of candidate rules in our rule selection method.

The necessity of the prescreening becomes clear by examining computation time of our genetic algorithm-based fuzzy rule selection method. In our MOGA, the size of the search space is 2^N because each solution is denoted by a binary string of the length *N*. When *N* is too large, it is difficult to find good solutions in the huge search space of the size 2^N. Thus we could not easily obtain good results when we did not use any prescreening procedure. CPU time required for each computer simulation is shown in Table 8 for the following four versions of our MOGA:

MOGA: Our MOGA.

MOGA-B: A version of our MOGA where the mutation is not biased.

MOGA-R: A version of our MOGA where the removal of unnecessary rules is not used.

MOGA-BR: A version of our MOGA where the mutation is not biased and the removal of unnecessary rules is not used.

From this table, we can see that the biased mutation and the removal of unnecessary rules have significant effects on the decrease in the computation time. When we used neither of these two heuristics (i.e., MOGA-BR), the computation time was about 19 hours in the case of 27423 candidate rules. Note that the computation time was about 19 minutes when both heuristics were used (i.e., MOGA) in Table 8. We also see from Table 8 that the prescreening of candidate rules decreased the CPU time of our MOGA. Let $S^*(N)$ be a Pareto-optimal rule set when the number of candidate rules is *N*. Theoretically, we can see that $S^*(N = 27423)$ is never dominated by any other rule set $S^*(N < 27423)$ because *N* candidate rules are chosen from the 27423 rules. This means that the best results would be obtained in the case of $N = 27423$ if there were no restrictions on computation time for executing our MOGA. Thus we are likely to obtain good rule sets from a large value of *N* when long computation time is available. This is not always the case in practical situations. Actually, some rule sets obtained in the case of $N = 27423$ in Table 7 are dominated by rule sets in the case of $N < 27423$ in Tables 4-6. For example, the seventh rule set in Table 7 with a 97.8% classification rate is dominated by the seventh rule set in Table 6 with a 98.3% classification rate.

Table 8 CPU time of our MOGA with various specification of the number of candidate rules.

The number of candidate rules	CPU time (minutes)			
	MOGA	MOGA-B	MOGA-R	MOGA-BR
9	2.38	2.30	1.10	1.08
90	3.29	4.00	1.73	2.22
900	4.03	4.73	2.42	4.12
27,423	19.12	20.18	19.16	1112.12

For further examining the relation between the efficiency of our MOGA and the number of candidate rules *N*, we monitored the highest classification rate of non-dominated rule sets with three fuzzy if-then rules at each generation during the execution of our MOGA for various specifications of the number of candidate rules *N*. Average simulation results over 20 trials with different initial populations are summarized in Fig. 2. From this figure, we can see that our MOGA could quickly find good rule sets in early generations when the number of candidate rules was small.

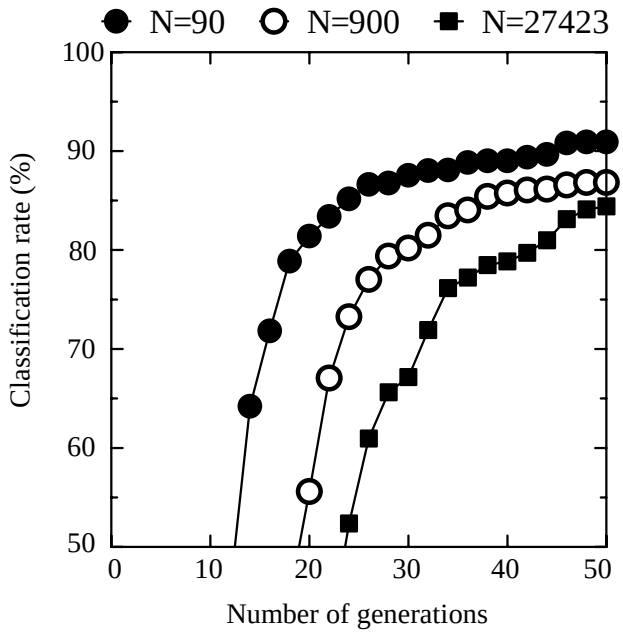

Fig. 2 The highest classification rate of rule sets with three rules at each generation.

For comparison, we performed the same computer simulation using randomly selected 90 candidate fuzzy if-then rules from the generated 27423 rules. The randomly selected 90 rules were used as candidate rules in our MOGA. Simulation results are summarized in Table 9. This table shows non-dominated rule sets with classification rates higher than 85%. From the comparison between Table 6 and Table 9, we can see that much better results were obtained in the case of prescreening using the product criterion than the case of random selection of candidate rules.

Table 9 Non-dominated rule sets obtained from randomly selected 90 candidate rules.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100\times f_1(S)/m$ (%)
9	3.00	85.4
10	3.00	86.5
12	2.83	87.1
12	3.00	87.6
13	2.85	88.2
13	3.00	89.3
19	2.95	89.9
20	2.95	90.4

3.4 Use of Various Partitions for Each Input

In the previous computer simulation, we used the same fuzzy partition with the five linguistic values in Fig. 1 (d) for all the 13 attributes of the wine data. In many cases, an appropriate fuzzy partition for each attribute is not the same. Moreover, we do not always know an appropriate fuzzy partition for each attribute. Such a situation can be handled by simultaneously using multiple fuzzy partitions for each attribute. For example, we can use the four fuzzy partitions in Fig. 1 for each attribute. In this case, each antecedent fuzzy set can be one of the 14 linguistic values in Fig. 1 or *don't care*. Thus the total number of possible combinations of antecedent fuzzy sets is $(14+1)^{13}$ for the wine data.

As in Section 2, we generated fuzzy if-then rules of the length 3 or less. The number of the generated fuzzy if-then rules was 711727. It is difficult to handle all the generated fuzzy if-then rules as candidate rules in our MOGA. Thus we chose 900 candidate rules using the product criterion. Our MOGA was applied to those candidate rules. Simulation results are summarized in Table 10. From the comparison between Table 4 and Table 10, we can see that better results were obtained in Table 10 from 900 candidate rules with various fuzzy partitions. For example, the highest classification rate of three rules in Table 10 is 98.3% while it was 93.3% in Table 4.

Table 10 Non-dominated rule sets obtained from a single trial with 900 candidate rules and various fuzzy partitions.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100 \times f_1(S)/m$ (%)
3	1.00	92.7
3	1.33	96.1
3	1.67	97.8
3	2.33	98.3
4	0.75	93.8
4	1.50	98.3
4	1.75	98.9
4	2.00	99.4
4	2.25	100.0

For further examining non-dominated rule sets of our fuzzy rule selection problem with various fuzzy partitions, we performed the above computer simulation 20 times using different initial populations. Obtained rule sets from the 20 trials were compared with one another. When a rule set was dominated by another rule set, the dominated one was discarded. In this manner, we found non-dominated rule sets from the 20 trials. Table 11 shows the non-dominated rule sets obtained from the 20 trials. Slightly better rules sets are included in Table 11 than Table 10. Three rules with the 94.9 classification rate in Table 11 are shown in Fig. 3. From this figure, we can see that our rule selection method found a very simple fuzzy rule-based system. The compactness of fuzzy rule-based systems and the simplicity of each rule are the main advantage of our fuzzy rule selection method because these two criteria are directly optimized by our three-objective genetic algorithm. It should be noted that only the three attributes in Fig. 3 have antecedent conditions (i.e., all the other attributes have *don't care* conditions in all the three rules). On the other hand, Fig. 4 shows three rules with the 100 % classification rate in Table 11. The three rules in Fig. 4 with higher classification ability are longer than those in Fig. 3. Another advantage of our approach is that multiple rule sets with different complexity can be obtained. That is, a tradeoff between accuracy and interpretability of fuzzy rule-based systems is clearly shown by obtained rule sets.

Table 11 Non-dominated rule sets obtained from 20 trials.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100\times f_1(S)/m$ (%)
3	1.00	94.9
3	1.33	96.1
3	1.67	98.3
3	2.00	99.4
3	2.33	100.0
4	0.75	96.1
4	1.00	97.2
4	1.25	98.9

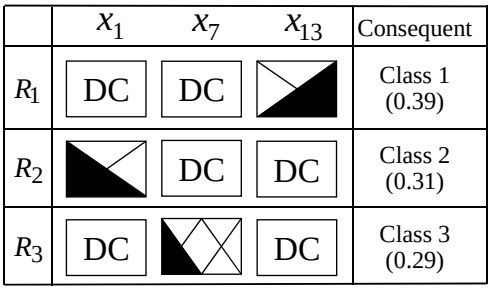

Fig. 3 Selected three fuzzy if-then rules with a 94.9% classification rate.

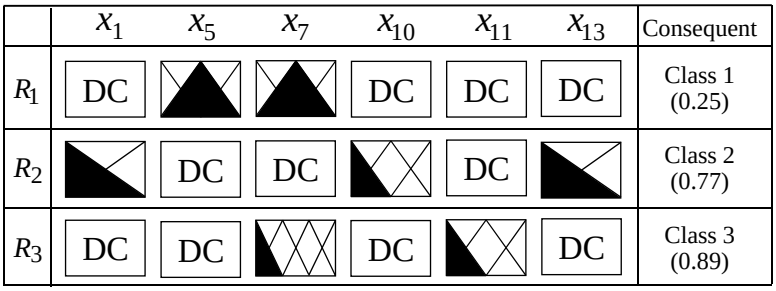

Fig. 4 Selected three fuzzy if-then rules with a 100% classification rate.

4. Extension to Hybrid Algorithms

4.1 Hybridization with Local Search

While genetic algorithms have high global search ability, their local search ability is not high. That is, their convergence speed to optimal solutions is not high while they can avoid being trapped in local solutions. One standard approach for improving the convergence speed is to try to find a good valance

between exploration and exploitation in the implementation of genetic algorithms. Another approach is to combine local search with genetic algorithms. Hybridization of MOGAs with local search was proposed in [8]. It was shown through computer simulation in several studies [9,17,18] that the hybridization improved the convergence speed of populations in MOGAs to Pareto-optimal solutions.

In multi-objective genetic local search (MOGLS) algorithms, local search is applied to newly generated strings by genetic operations. For implementing an MOGLS algorithm for our rule selection problem, first we have to specify an objective function to be optimized by local search. As in the previous studies [8,9,17,18] on MOGLS algorithms, we use the scalar fitness function in (20) with random weights. In those studies, the weights specified for the selection of a pair of parent stings were also used in local search for their offspring. That is, local search for each offspring was governed by the randomly specified weights for the selection of its parent strings. One drawback of this weight specification scheme is that the inherited weights from parents are not always appropriate for their offspring. Another drawback is that local search is applied to all offspring independent of their performance. It may be waste of CPU time to apply local search to poor offspring. For overcoming these drawbacks, we modify the above-mentioned weight specification scheme in the local search part of our MOGLS algorithm as follows. In our proposed scheme, we randomly specify the weights in the scalar fitness function in (20) for choosing an initial solution for local search and specifying the local search direction for the selected one. We choose an initial solution from the current population using tournament selection with replacement based on the scalar fitness function. The weights are randomly updated whenever a new initial solution is selected. This mean that local search from each initial solution is governed by the scalar fitness function with different weights. In our computer simulations, the tournament size was specified as four.

For implementing a MOGLS algorithm for our rule selection problem, we also have to specify a mechanism for generating a neighboring solution from the current solution in local search. We use the following three mechanisms:

(1) Generate a neighboring solution of the current solution S by removing a single rule from S. The number of neighboring solutions of S is $|S|$, i.e., the number of rules in S.

(2) Generate a neighboring solution of S by adding a single rule to S. The number of neighboring solutions of S is $N-|S|$ where N is the number of candidate rules.

(3) Generate a neighboring solution of S by removing a single rule from S and add another rule to S. The number of neighboring solutions of S is $|S|\times(N-|S|)$.

The total number of neighboring solutions of the current solution is $N+|S|\times(N-|S|)$. If we do not restrict the number of examined solutions, at least $N+|S|\times(N-|S|)$ neighboring solutions have to be examined in the execution of local search for each initial solution generated by genetic operations. Thus

almost all the available computation time is spent by local search. In this case, our MOGLS algorithm is almost the same as a multi-start local search algorithm because only a few generation updates are performed.

For decreasing the computation load of local search, we only examine three neighboring solutions of the current solution in our MOGLS algorithm in this paper. A single neighboring solution is randomly generated using each of the above three mechanisms. If none of the three neighboring solutions is better than the current solution, local search for the current solution is terminated. Otherwise, the current solution is replaced with the best one among the examined three solutions. In this case, local search continues to examine three neighboring solutions of the new current solution in the same manner.

The local search part of our MOGLS algorithm is executed by iterating the following three steps for $(N_{pop} - N_{elite})$ times:

1. Randomly specify the weights of the scalar fitness function.
2. Select an initial solution from the current population by the tournament selection of the tournament size four with replacement using the scalar fitness function.
3. Apply the above-mentioned local search algorithm to the selected initial solution.

This local search part is used after the genetic operations (i.e., Step 2 of our MOGA).

We applied our MOGLS algorithm to our rule selection problem with 900 candidate rules generated from various fuzzy partitions in Subsection 3.4. For fair comparison, our MOGLS algorithm was terminated when 500000 rule sets were examined. Thus the computation load of our MOGLS algorithm in this subsection is the same as that of the MOGA in Subsection 3.4. Non-dominated solutions obtained from 20 trials are summarized in Table 12. We can see that the five rule sets with three rules in Table 12 are the same as those in Table 11. That is, the hybridization with local search did not clearly improve the search ability of the MOGA. This is mainly because good non-dominated rule sets had already been obtained by the MOGA in Table 11. Note that the elimination procedure of unnecessary rules in the MOGA works as a kind of local search. In Table 12, no rule sets with four rules were obtained while three rule sets were obtained in Table 11 by the MOGA. The decrease in the variety of obtained non-dominated solutions is a negative effect of the hybridization with local search. This negative effect, however, is not severe in our simulation results because rule sets with three rules have high classification rates in Table 12. One advantage of the MOGLS algorithm over the MOGA is less computation time. The average CPU time for a single trial was decreased from 4.08 minutes in Table 11 to 1.68 minutes in Table 12 by the hybridization of the MOGA with local search. This is because local search can be more efficiently executed than genetic search. In our computer simulations, the MOGA in Table 11 and the MOGLS in Table 12 were compared under the same number of examined rule sets. Of course, the MOGLS spends much more computation time than the MOGA if we compare them under the same number of generation

updates. In this case, much more rule sets are examined by the MOGLS than the MOGA.

Table 12 Simulation results by our MOGLS algorithm.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100\times f_1(S)/m$ (%)
3	1.00	94.9
3	1.33	96.1
3	1.67	98.3
3	2.00	99.4
3	2.33	100.0

4.2 Combination with Weight Learning

As shown in [11,23], the classification accuracy of fuzzy rule-based systems can be improved by adjusting the rule weight of each fuzzy if-then rule. We use a simple reward-and-punishment scheme [23]. When a training pattern $\boldsymbol{x}_p$ is correctly classified by the winner rule R_{q^*} in a rule set, its rule weight CF_{q^*} is increased as

$$CF_{q^*}^{New} = CF_{q^*}^{old} + \eta_1 (1 - CF_{q^*}^{Old}), \tag{23}$$

where η_1 is a learning rate for increasing rule weights. The rule weights of the other rules in the rule set are not changed. On the other hand, when the training pattern $\boldsymbol{x}_p$ is misclassified by the winner rule R_{q^*}, its rule weight CF_{q^*} is decreased as

$$CF_{q^*}^{New} = CF_{q^*}^{old} - \eta_2 CF_{q^*}^{Old}, \tag{24}$$

where η_2 is a learning rate for decreasing rule weights. The rule weights of the other rules are not changed. In our computer simulations, η_1 and η_2 were specified as $\eta_1 = 0.001$ and $\eta_2 = 0.1$ as in [23], respectively.

This learning procedure is applied to all strings in the current population after local search. All the given training patterns are examined in a single iteration of our learning procedure for each string. Thus our learning procedure may be time-consuming if it is iterated many times. In our computer simulations, we iterated our learning procedure just twice for each string in the current population. Note that the

adjusted rule weights in each generation are not inherited to the next generation.

We applied our MOGLS algorithm with rule weight learning to the wine data in the same manner as in the previous subsection. Simulation results of 20 trials are summarized in Table 13. We can see that obtained non-dominated rule sets in Table 13 are almost the same as those in Table 11 and Table 12. This is because good non-dominated rule sets had already been obtained by the MOGA in Table 11 using the best heuristic definition of rule weights. The average CPU time increased from 1.68 minutes in Table 12 to 2.52 minutes in Table 13 by combining our MOGLS algorithm with rule weight learning. The increase in the CPU time by the hybridization with rule weight learning was not significant because the number of iterations of the learning algorithm was just two for each rule set.

Table 13 Simulation results by our MOGLS algorithm with rule weight learning.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate $100 \times f_1(S)/m$ (%)
3	1.00	92.7
3	1.33	96.6
3	1.67	98.3
3	2.00	99.4
3	2.67	100.0
4	1.00	97.2

The effect of the hybridization with rule weight learning becomes clear if we use a different heuristic definition of rule weights. In our previous computer simulations, we used the best heuristic definition chosen in Section 2 (i.e., the proposed heuristic definition in (14)-(15)). We also performed computer simulations using the previous heuristic definition of rule weights in (12)-(13). The highest classification rate of rule sets with three fuzzy if-then rules was 99.4% by the MOGA, 99.4% by the MOGLS, and 100% by the MOGLS with rule weight learning. These simulation results show that rule weight learning improved the classification ability of obtained rule sets.

5. Performance Evaluation of Selected Rules

Through the previous computer simulations in this paper, we have already shown that the prescreening of candidate rules using the product criterion improved the efficiency of our MOGA for rule selection. We have also implemented and examined hybrid algorithms of our MOGA with local search and rule weight learning. In those computer simulations, we only calculated classification rates of selected rule sets on training patterns. In this section, we examined classification rates on unseen test patterns for

evaluating the generalization ability of selected rule sets.

5.1 Data Sets and Simulation Conditions

The iris data and the Australian credit approval data in the UCI Machine Learning Repository were used in our computer simulations in this section in addition to the wine data used in the previous sections. The iris data set is a three-class pattern classification problem involving 150 patterns with four continuous attributes. We used the iris data set in our computer simulations because it is one of the most frequently used test problems in the literature. Each attribute value was normalized in a real number in the closed interval $[0, 1]$. Thus the iris data set was handled as a three-class problem in the four-dimensional unit hypercube $[0,1]^4$. Since the iris data set is not a high-dimensional classification problem, we examined all the possible combinations of 14 linguistic values in Fig. 1 and *don't care* (i.e., $(14+1)^4$ combinations) for generating candidate rules. Using the product criterion, we chose 900 candidate rules from the generated fuzzy if-then rules.

The credit data set is a two-class pattern classification problem involving 690 patterns with 14 attributes. Among the 14 attributes, four are binary, and two are ternary. One attribute involves nine discrete values. Another one has 14 discrete values. The other six are continuous attributes. This data set is also well known because it has often been used in the literature such as Quinlan's C4.5 book [26]. All attribute values (including discrete ones) were normalized into real numbers in the unit interval [0, 1]. For example, we used {0, 1} and {0, 0.5, 1} for binary and ternary attributes, respectively. Thus the credit data set was handled as a two-class problem in the 14-dimensional unit hypercube $[0, 1]^{14}$. When we generated candidate fuzzy if-then rules, we used only two linguistic values S^2 and L^2 in Fig. 1 (a) and *don't care* for the binary attributes. For the ternary attributes, we used five linguistic values in Fig. 1 (a)-(b) and *don't care*. For the other attributes, we used all the 14 linguistic values in Fig. 1 and *don't care*. We generated 900 candidate rules from the credit data in the same manner as the computer simulations on the wine data in Subsection 3.4.

For calculating average classification rates on test data for the wine data and the iris data, we used the leaving-one-out (LV1) technique [29]. The whole LV1 procedure was iterated ten times for calculating average classification rates of selected rule sets for each of the iris and wine data sets. For the credit data set, we use the 10-fold cross-validation (10-CV) technique [29]. In the 10-CV technique, the credit data set was divided into ten subsets of the same size. Nine subsets were used as training data, and the other subset was used as test data. This train-and-test procedure was iterated ten times so that all the ten subsets were used as test data once. The whole 10-CV procedure was iterated 50 times using different partitions of the credit data into ten subsets for calculating average classification rates on test data. In all computer

simulations in this section, we used our MOGLS algorithm with rule weight learning.

5.2 Simulation Results on Wine Data

Simulation results on the wine data set with 178 patterns are summarized in Table 14. As we have already mentioned, the whole LV1 procedure was iterated ten times. This means that the rule selection was performed 1780 times. The last column of Table 14 shows the number of trials where the corresponding combination of the number of rules and the average rule length was obtained. In Table 14, we only show frequently obtained combinations (i.e., more than 500 trials) of the number of rules and the average rule length. Rule sets with less than three rules are not shown in Table 14 because the wine data with three classes need at least three rules (i.e., at least one rule for each class).

For the wine data set, Setnes & Roubos [28] reported a 98.3% classification rate on training data by three fuzzy if-then rules. Our results on training data in the previous sections (e.g., a 100% classification rate by three rules) were better than the reported result in [28]. Castillo et al. [3] reported a 96.76% average classification rate on test data (30% of the wine data) where the average number of fuzzy if-then rules was 5.2 over five independent trials. Since their SLAVE algorithm used a union (i.e., disjunction) of multiple linguistic values as a single antecedent fuzzy set, the number of fuzzy if-then rules can be decreased. For example, the fuzzy if-then rule "If x_1 is *small* or *medium* and x_2 is *medium* or *large* then Class 2" was handled as a single rule in [3] while it is handled as four rules in this paper. From Table 14, we can see that our approach found rule sets with fewer rules than the SLAVE algorithm. Classification rates on test data by our approach are comparable to the reported result by the SLAVE algorithm (i.e., 96.76%) when fuzzy rules are not too short in Table 14 (i.e., 96.1% and 97.2%). It should be noted that our fuzzy rules are much simpler than those in the SLAVE with arbitrary conjunctions of linguistic values in the antecedent part (see Fig. 3 and Fig. 4 in Section 3).

Table 14 Classification rates on test data for the wine data set.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate on test data (%)	Number of runs among 1,780 runs
3	1.00	86.3	1,660
3	1.33	90.1	1,743
3	1.67	92.8	1,688
3	2.00	93.9	1,618
3	2.33	96.1	1,419
3	2.67	97.2	503

5.3 Simulation Results on Iris Data

Simulation results on the iris data set are summarized in Table 15 in the same manner as Table 14 in

the previous subsection. For the iris data set, a 97.3% classification rate on test data (75 patterns) and a 100% classification rate on training data (75 patterns) were obtained from three fuzzy if-then rules with ellipsoidal regions, which were generated by a clustering technique and tuned by an analytical learning scheme in Abe & Thawonmas [1]. In Nauck & Kruse [22], a 96.0% classification rate on test data (75 patterns) and a 97.3% classification rate on training data (75 patterns) were obtained from three fuzzy if-then rules, which were generated and tuned by a neuro-fuzzy technique after heuristically selecting two attributes out of the given four attributes. The generalization ability of selected three rules with the average length two (i.e., a 96.4% classification rate) in Table 15 is comparable to the 96.0% classification rate in [22] and slightly inferior to the 97.3% classification rate in [1]. Note that our approach uses given linguistic values with no modification while antecedent fuzzy sets in [1] were generated from numerical data and tuned using neuro-fuzzy techniques. When we use many fuzzy if-then rules with no rule selection, high classification rates can be obtained without tuning antecedent fuzzy sets (e.g., 98.0% on test data in [23]).

Table 15 Classification rates on test data for the iris data set.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate on test data (%)	Number of runs among 1,500 runs
3	1	94.9	1,491
3	1.3333	94.5	1,429
3	1.6667	94.9	1,052
3	2	96.4	690

5.5 Simulation Results on Credit Data

Simulation results on the credit data are summarized in Table 16. This table shows combinations of the number of rules and the average rule length obtained in more than 300 among 500 trials (i.e., 50 iterations of the 10-CV procedures). In Quinlan [26], the C4.5 algorithm was applied to the credit data. The following results were reported in [26] as classification rates on test data by the C4.5 algorithm with various parameter specifications. The best result was 85.8%, the average result was 84.3%, and the worst result was 82.5% (see Table 9-1 of [26]). Our results in Table 16 are comparable to those results by the C4.5 algorithm. Note that very simple rule sets with only a few fuzzy rules were obtained in Table 16 by our approach.

Table 16 Classification rates on test data for the credit data.

Number of rules $f_2(S)$	Average rule length $f_3(S)/\|S\|$	Classification rate on test data (%)	Number of runs among 500 runs
2	0.50	85.5	468
2	1.00	85.2	495
2	1.50	85.0	356
2	2.00	84.6	305
3	1.67	85.2	362
3	2.00	84.8	357

6. Concluding Remarks

In this paper, we proposed an idea of using rule evaluation measures (i.e., confidence, support, and their product) as rule selection criteria for prescreening candidate fuzzy if-then rules used in rule selection. In our approach, first a number of candidate rules were selected using the product criterion. That is, fuzzy if-then rules with large values of this criterion were chosen as candidate rules. Then non-dominated subsets of the candidate rules were found by our multi-objective genetic algorithm (MOGA) with three objectives: to maximize the classification accuracy, to minimize the number of rules, and to minimize the total rule length. Through computer simulations, we demonstrated that the prescreening of candidate rules significantly improved the efficiency of our rule selection method. That is, better rule sets were obtained in shorter CPU time by our approach than the case with no candidate rule prescreening. Simulation results also showed that better results were obtained by our GA-based rule selection than heuristic rule selection using the three rule evaluation measures. This is because the performance of rule sets was not taken into account when rule selection was performed based on the rule evaluation measures. That is, only the performance of each individual rule was taken into account independently of other rules in the same rule set. Our approach, however, can choose a small number of fuzzy if-then rules by evaluating the classification performance of rule sets. For improving the classification ability of fuzzy rule-based systems, we also proposed a heuristic specification method of rule weights (i.e., certainty factors) of fuzzy if-then rules. Simulation results showed that the proposed heuristic method outperformed other existing methods.

Our MOGA in our former studies [10,12] was extended to a multi-objective genetic local search (MOGLS) algorithm by the hybridization with local search. The efficiency of our MOGA was improved by this hybridization with respect to CPU time. This is because local search can be executed more efficiently than genetic search. At the same time, the hybridization with local search has a negative effect on the search ability of our MOGA. Local search tends to decrease the diversity of populations. We

further combined our MOGLS algorithm with rule weight learning. This hybridization improved the classification performance of selected rule sets when heuristic specifications of rule weights were not appropriate. Finally, we examined the generalization ability of selected rule sets by our MOGLS algorithm with rule weight learning. Through computer simulations on the wine data, the iris data and the credit data, it was shown that the generalization ability of selected rule sets was comparable to reported results in the literature by other approaches while membership functions of antecedent fuzzy sets were not tuned in our approach. While we used the MOGLS with rule weight learning in our computer simulations, our MOGA may also work well because the number of candidate rules was not large (i.e., 900 candidate rules) and the rule weight of each fuzzy if-then rule was appropriately specified by the proposed definition.

One advantage of our approach is that compact rule sets with very simple fuzzy if-then rules can be obtained. That is, obtained rule sets have high interpretability. Another advantage is that multiple rule sets with different complexity can be obtained. That is, a tradeoff between interpretability and accuracy of fuzzy rule-based systems can be examined by our approach. Simulation results clearly show these advantages of our approach. Currently we are examining the application of our approach to real-world credit data with 28 attributes and more than 3000 cases. Our approach is applicable to such a large data set because the number of candidate rules is decreased by the prescreening procedure based on the rule evaluation measures in data mining. Our preliminary results on this data set are promising when we use inhomogeneous fuzzy partitions. We will report complete simulation results in another paper.

Our approach can be further improved in several aspects. One is the sophistication of the candidate rule prescreening procedure. In our computer simulations of this paper, we first generated fuzzy if-then rules of the length L or less where L is a user-definable parameter. Next the generated fuzzy if-then rules were divided into M groups according to their consequent classes where M is the number of classes. Then N / M candidate rules were chosen from each of the M groups where N is a user-definable parameter. In this candidate rule prescreening procedure, the generation of all the fuzzy if-then rules of the length L or less is not necessary. The sorting of all the fuzzy if-then rules in each group is not necessary, either. This is because the aim of the candidate rule prescreening is not to order all the fuzzy if-then rules of the length L or less but to find N candidate rules. That is, we do not have to generate and sort poor fuzzy if-then rules. The computation time for generating candidate rules may be significantly decreased by examining only promising fuzzy if-then rules. In such a sophisticated prescreening procedure, the specification of the value of L may be unnecessary because long fuzzy if-then rules (i.e., specific fuzzy if-then rules) are not likely to be selected as candidate rules by the product criterion of the confidence and the support. Another topic for future research is the improvement of the local search procedure. The effect of local search strongly depends on the choice of a neighborhood structure that is used for generating neighboring

solutions of the current one. In this paper, we generated three neighboring solutions. The procedure for generating neighboring solutions may be improved in future research. The search ability of our MOGA can be also improved by using state-of-the-art evolutionary algorithms for multi-objective optimization problems (e.g., see [30,31]) because our MOGA is very simple.

References

[1] S. Abe and R. Thawonmas, "A fuzzy classifier with ellipsoidal regions," *IEEE Trans. on Fuzzy Systems*, vol. 5, no. 3, pp. 358-368, 1997.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proc. of 20th International Conference on Very Large Data Bases*, pp. 487-499, 1994. Expanded version is available as IBM Research Report RJ9839, 1994.

[3] L. Castillo, A. Gonzalez, and P. Perez, "Including a simplicity criterion in the selection of the best rule in a genetic fuzzy learning algorithm," *Fuzzy Sets and Systems*, vol. 120, vol. 2, pp. 309-321, 2001.

[4] J. L. Castro, J. J. Castro-Schez, and J. M. Zurita, "Use of a fuzzy machine learning technique in the knowledge acquisition process," *Fuzzy Sets and Systems*, vol. 123, no. 3, pp. 307-320, 2001.

[5] O. Cordon, M. J. del Jesus, and F. Herrera, "A proposal on reasoning methods in fuzzy rule-based classification systems," *International Journal of Approximate Reasoning*, vol. 20, no. 1, pp. 21-45, 1999.

[6] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," *Proc. of 12th International Conference on Machine Learning*, pp. 194-202, 1995.

[7] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," *Proc. of 13th International Joint Conference on Artificial Intelligence*, pp. 1022-1027, 1993.

[8] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm," *Proc. of 3rd IEEE International Conference on Evolutionary Computation*, pp. 119-124, 1996.

[9] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 28, no. 3, pp. 392-403, 1998.

[10] H. Ishibuchi, T. Murata, and I. B. Turksen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets and Systems*, vol. 89, no. 2, pp. 135-149, 1997.

[11] 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, 2001.

[12] 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, 2001.

[13] H. Ishibuchi, K. Nozaki, and H. Tanaka, “Distributed representation of fuzzy rules and its application to pattern classification,” *Fuzzy Sets and Systems*, vol. 52, no. 1, 1992.

[14] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka: “Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms,” *Fuzzy Sets and Systems*, vol. 65, no. 2/3, pp. 237-253, 1994.

[15] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, “Selecting fuzzy if-then rules for classification problems using genetic algorithms,” *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 3, pp. 260-270, 1995.

[16] H. Ishibuchi, T. Yamamoto, and T. Nakashima, “Fuzzy data mining: Effect of fuzzy discretization,” *Proc. of 1st IEEE International Conference on Data Mining*, pp. 241-248, 2001.

[17] A. Jaszkiewicz, M. Hapke, and P. Kominek, “Performance of multiple objective evolutionary algorithms on a distributed system design problem - Computational experiment,” *Proc. of 1st International Conference on Evolutionary Multi-Criterion Optimization*, pp. 241-255, 2001.

[18] A. Jaszkiewicz, “Genetic local search for multi-objective combinatorial optimization,” *European Journal of Operational Research* (to appear).

[19] Y. Jin, “Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement,” *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 2, pp. 212-221, 2000.

[20] L. I. Kuncheva, *Fuzzy Classifier Design*, Physica-Verlag, Heidelberg, 2000.

[21] C. T. Leondes (Ed.), *Fuzzy Theory Systems: Techniques and Applications (Vols. 1-4)*, Academic Press, San Diego, 1999.

[22] D. Nauck and R. Kruse, “A neuro-fuzzy method to learn fuzzy classification rules from data,” Fuzzy Sets and Systems, vol. 89, no. 3, pp. 277-288, 1997.

[23] K. Nozaki, H. Ishibuchi, and H. Tanaka, “Adaptive fuzzy rule-based classification systems,” *IEEE Trans. on Fuzzy Systems*, vol. 4, no. 3, pp. 238-250, 1996.

[24] V. de Oliveira, “Semantic constraints for membership function optimization,” *IEEE Trans. on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 29, no. 1, pp. 128-138, 1999.

[25] W. Pedrycz and V. de Oliveira, “Optimization of fuzzy models,” *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 26, no. 4, pp. 627-637, 1996.

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

[27] M. Setnes, R. Babuska, and B. Verbruggen, “Rule-based modeling: Precision and transparency,”

IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews, vol. 28, no. 1, pp. 165-169, 1998.

[28] M. Setnes and H. Roubos, "GA-based modeling and classification: Complexity and performance," *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 5, pp. 509-522, 2000.

[29] S. M. Weiss and C. A. Kulikowski, *Computer Systems That Learn*, Morgan Kaufmann Publishers, San Mateo, 1991.

[30] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173-195, 2000.

[31] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, 1999.