Data Set Subdivision for Parallel Distributed Implementation of Genetic Fuzzy Rule Selection

Yusuke Nojima, *Member, IEEE*, Isao Kuwajima, *Student member, IEEE*, and Hisao Ishibuchi, *Member, IEEE*

***Abstract*— Genetic fuzzy rule selection has been successfully used to design accurate and interpretable fuzzy classifiers. However there exists a computational complexity problem for large data sets. This paper proposes a simple but effective idea to improve the applicability of genetic fuzzy rule selection to large data sets. Our idea is based on the parallel distributed implementation of genetic fuzzy rule selection. We examine the advantage of the proposed approach through computational experiments on some benchmark data sets.**

I. Introduction

Genetic fuzzy rule selection is an effective approach to the design of accurate and interpretable fuzzy rule-based classifiers [1]-[3], [21]. It is the following two-step approach. In the first phase, a large number of promising fuzzy rules are generated by a data mining technique. In the second phase, a genetic algorithm is used to select a small number of fuzzy rules from the generated ones in the first phase. Each string is evaluated with respect to the accuracy on training patterns and the complexity of a fuzzy classifier represented by the string.

The main advantage of genetic fuzzy rule selection is less computational complexity than alternative algorithms such as genetics-based machine learning [4]-[7]. But there exists a computational complexity problem when we apply it to large data sets. The evaluation time of each string linearly increases as the number of patterns in a data set increases.

In this paper, we propose a simple but effective idea to improve the applicability of genetic fuzzy rule selection to large data sets. Our idea is based on the subdivision of a large data set into small subsets. A similar idea was used in Cano et al. [8], [9] for instance selection where a genetic instance selection algorithm was independently applied to each subset. That is, a large instance selection problem was subdivided into small sub-problems. Selected instances in each sub-problem were merged to form the final solution of the original instance selection problem. We use the subdivision of a large data set to efficiently evaluate each rule set (i.e., each string: fuzzy rule-based classifier).

This paper is organized as follows. First we explain genetic fuzzy rule selection for designing fuzzy classifiers in Section II. Next we explain its parallel distributed implementation in Section III. Then we examine the effect of the subdivision in Section IV. Finally we conclude this paper in Section V.

II. Classifier Design By Genetic Rule Selection

In this section, we briefly explain classification rules, classification methods and genetic rule selection.

A. Pattern Classification Problem

Let us assume that we have m training (i.e., labeled) patterns $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1, 2, ..., m$ from M classes in the n-dimensional continuous pattern space where x_{pi} is the attribute value of the p-th training pattern for the i-th attribute ($i = 1, 2, ..., n$). For the simplicity of explanation, we assume that all the attribute values have already been normalized into real numbers in the unit interval [0, 1]. That is, $x_{pi} \in [0, 1]$ for $p = 1, 2, ..., m$ and $i = 1, 2, ..., n$. Thus the pattern space of our pattern classification problem is an n-dimensional unit-hypercube $[0, 1]^n$.

B. Fuzzy Rules for Pattern Classification Problem

For our n-dimensional pattern classification problem, we use fuzzy rules of the following type:

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

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 ($i = 1, 2, ..., n$), C_q is a class label, and CF_q is a rule weight (i.e., certainty grade).

Since we usually have no *a priori* information about an appropriate granularity of the fuzzy discretization for each attribute, we simultaneously use multiple fuzzy partitions with different granularities in fuzzy rule extraction. In our computational experiments, we use four homogeneous fuzzy partitions with triangular fuzzy sets in Fig. 1. In addition to the 14 fuzzy sets in Fig. 1, we also use the domain interval [0, 1] as an antecedent fuzzy set in order to represent a *don't care* condition. That is, we use the 15 antecedent fuzzy sets for each attribute in our computational experiments. Even when we use these simple partitions, there still exist interpretability issues

This work was partially supported by Grand-in-Aid for Young Scientist (B): KAKENHI (18700228), and Grant-in-Aid for Scientific Research on Priority Areas: KAKENHI (18049065).

Yusuke Nojima is with the Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan (nojima@cs.osakafu-u.ac.jp).

1-4244-1210-2/07/$25.00 ©2007 IEEE.

[21]. But we skip this discussion and focus on parallel distributed implementation of genetic fuzzy rule selection in this paper.

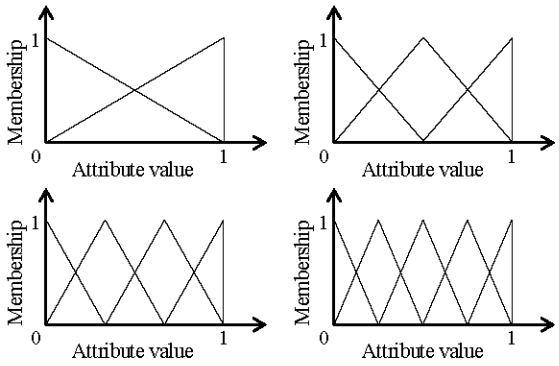

Fig. 1. Four fuzzy partitions used in our computational experiments.

C. Fuzzy Rule Generation

Since we use the 15 antecedent fuzzy sets for each attribute of our n-dimensional pattern classification problem, the total number of combinations of the antecedent fuzzy sets is 15^n. Each combination is used in the antecedent part of the fuzzy rule in (1). Thus the total number of possible fuzzy rules is also 15^n. The consequent class C_q and the rule weight CF_q of each fuzzy rule R_q are specified from the given training patterns in the following heuristic manner.

First we calculate the compatibility grade of each pattern $\mathbf{x}_p$ with the antecedent part $\mathbf{A}_q$ of the fuzzy rule R_q using the product operation as

$$\mu_{\mathbf{A}_q}(\mathbf{x}_p) = \mu_{A_{q1}}(x_{p1}) \cdot \ldots \cdot \mu_{A_{qn}}(x_{pn}), \quad (2)$$

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

Next we calculate the confidence of the fuzzy rule "$\mathbf{A}_q \Rightarrow$ Class h" for each class ($h = 1, 2, ..., M$) as follows [16]:

$$c(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{\sum_{p=1}^{m} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}. \quad (3)$$

The consequent class C_q is specified by identifying the class with the maximum confidence:

$$c(\mathbf{A}_q \Rightarrow \text{Class } C_q) = \max_{h=1,2,...,M} \{c(\mathbf{A}_q \Rightarrow \text{Class } h)\}. \quad (4)$$

The consequent class C_q can be viewed as the dominant class in the fuzzy subspace defined by the antecedent part $\mathbf{A}_q$. When there is no pattern in the fuzzy subspace defined by $\mathbf{A}_q$, we do not generate any fuzzy rules with $\mathbf{A}_q$ in the antecedent part. This specification method of the consequent class of fuzzy rules has been used in many studies since [10].

The rule weight CF_q of each fuzzy rule R_q has a large effect on the performance of fuzzy rule-based classification systems [11]. Different specifications of the rule weight have been proposed and examined in the literature. We use the following specification because good results were reported by this specification in the literature [12], [13]:

$$CF_q = c(\mathbf{A}_q \Rightarrow \text{Class } C_q) - \sum_{\substack{h=1 \\ h \neq C_q}}^{M} c(\mathbf{A}_q \Rightarrow \text{Class } h). \quad (5)$$

Let S be a set of fuzzy rules of the form in (1). A new pattern $\mathbf{x}_p$ is classified by a single winner rule R_w, which is chosen from the rule set S as follows:

$$R_w = \arg\max\{\mu_{\mathbf{A}_q}(\mathbf{x}_p) \cdot CF_q \mid R_q \in S\}. \quad (6)$$

As shown in (6), the winner rule R_w has the maximum product of the compatibility grade and the rule weight in S. For other fuzzy reasoning methods for pattern classification problems, see Cordon et al. [15] and Ishibuchi et al. [10], [12]. It should be noted that the choice of an appropriate rule weight specification depends on the type of fuzzy reasoning (i.e., single winner rule-based fuzzy reasoning) used in fuzzy rule-based classification systems [12], [13].

D. Rule Evaluation Criteria

Using the above-mentioned procedure, we can generate a large number of fuzzy rules by specifying the consequent class and the rule weight for each of the 15^n combinations of the antecedent fuzzy sets. It is, however, very difficult for human users to handle such a large number of generated fuzzy rules. It is also very difficult to intuitively understand long fuzzy rules with many antecedent conditions. Thus we only generate short fuzzy rules with only a small number of antecedent conditions. It should be noted that *don't care* conditions with the special antecedent fuzzy set [0, 1] can be omitted from fuzzy rules. The rule length means the number of antecedent conditions excluding *don't care* condition. We examine only short fuzzy rules of length $L_{\max}$ or less (e.g., $L_{\max} = 3$). This restriction is to find a small number of short (i.e., simple) fuzzy rules with high interpretability.

Among short fuzzy rules, we choose promising rules by a heuristic rule evaluation criterion as candidate rules in genetic fuzzy rule selection. In the field of data mining, two rule evaluation criteria (i.e., confidence and support) have been often used. We have already shown the fuzzy version of the confidence criterion in (3). In the same manner, the support of the fuzzy rule "$\mathbf{A}_q \Rightarrow$ Class h" is calculated as follows [16]:

$$s(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{m}. \quad (7)$$

In our computational experiments, we extracted candidate rules using prespecified values of the thresholds on support and confidence.

E. Genetic Fuzzy Rule Selection

Let us assume that N candidate rules have already extracted. The task of genetic fuzzy rule selection is to find an accurate and compact rule set from the N candidate rules.

Any subset S of the N candidate rules can be denoted by a binary string of length N as $S = s_1 s_2 s_3 \cdots s_N$ where $s_i = 1$ and $s_i = 0$ mean that the ith candidate rule is included in and excluded from the rule set S, respectively. Such a binary string is used as an individual in genetic fuzzy rule selection.

In this paper, we use the following three objectives to find an accurate and compact rule set S:

$f_1(S)$: The number of correctly classified training patterns by S,

$f_2(S)$: The number of fuzzy rules in S,

$f_3(S)$: The total number of antecedent conditions in S.

The first objective is maximized while the second and third objectives are minimized.

The first objective is calculated by classifying training patterns $\mathbf{x}_p$. We use a single winner-based (i.e., winner-take-all) classification method. That is, $\mathbf{x}_p$ is classified by the winner rule that has the maximum rule weight among the compatible rules with $\mathbf{x}_p$ in S. The classification of $\mathbf{x}_p$ is rejected when no rules are compatible with $\mathbf{x}_p$ (which is counted as an error in our computational experiments). In our genetic fuzzy rule selection, random tiebreak is not used to efficiently search for a small number of necessary rules. That is, the classification of $\mathbf{x}_p$ is rejected when multiple compatible rules with different consequent classes have the same maximum rule weight.

The second objective is calculated by just counting the number of 1's (i.e., selected rules) in S. Since we use the single winner-based method without random tiebreak to evaluate the accuracy of the rule set S, only a single rule is responsible for the classification of each training pattern. As a result, some rules may be used for the classification of no training patterns. Whereas the existence of such an unnecessary rule in the rule set S has no effect on the first objective of the weighted sum fitness function, it deteriorates the second and third objectives. Thus we remove from the offspring all the unnecessary rules responsible for the classification of no training patterns.

The third objective is the total number of antecedent conditions except for *don't care* conditions in the selected rules in S.

The above-mentioned three objectives are combined into the following weighted sum fitness function:

$$fitness(S) = w_1 \cdot f_1(S) - w_2 \cdot f_2(S) - w_3 \cdot f_3(S), \quad (8)$$

where w_1, w_2 and w_3 are non-negative weights. This fitness function is maximized in genetic fuzzy rule selection. As a result, the accuracy is maximized while the complexity is minimized. Of course, the final solution (i.e., the rule set S) strongly depends on the specification of the weight vector $\mathbf{w} = (w_1, w_2, w_3)$.

Genetic rule selection is implemented in the following manner to find the optimal rule set S with respect to the weighted sum fitness function in (8).

Genetic Rule Selection

Step 1: Generate a number of promising rules (i.e., N rules) from the training patterns.

Step 2: Randomly generate N_{pop} binary strings of length N as an initial population where N_{pop} is a user-definable parameter called the population size.

Step 3: Iterate the following operations N_{pop} times to generate an offspring population with N_{pop} strings.

3.1: Select a pair of parent strings from the current population by binary tournament selection.

3.2: Recombine the selected parent strings to generate an offspring by the uniform crossover operation. One of the generated strings is randomly chosen as an offspring. This operation is applied with a pre-specified probability. The crossover probability is specified as 0.9 in this paper. When the crossover operation is not applied to the selected pair of parents, one of the two parents is randomly chosen and used as an offspring in the following steps.

3.3: Apply a biased mutation operation to the offspring. This operation changes 0 to 1 with a small probability and 1 to 0 with a large probability to decrease the number of 1's (i.e., selected rules) in the offspring. The mutation probabilities from 0 to 1 and from 1 to 0 are specified as $1/N$ and 0.05, respectively, where N is the number of candidate rules. In our computational experiments, $N >> 100$.

3.4: Remove unnecessary rules from the offspring in the heuristic manner.

Step 4: Select the best N_{pop} strings with respect to the weighted sum fitness function in (8) from the current and offspring populations.

Step 5: If a prespecified termination condition is not satisfied, return to Step 3 with the best N_{pop} strings selected in Step 4 as the population in the next generation. Otherwise, terminate the execution of the algorithm.

We use the total number of iterations of the algorithm (i.e., the total number of generations) as the termination condition in this paper. The best rule set among examined ones during the execution of our genetic rule selection algorithm is returned to human users as the final result.

III. Parallel Distributed Genetic Fuzzy Rule Selection

Whereas genetic algorithms have been frequently used in the field of data mining and knowledge extraction [18], their applicability to large data sets is not high. This is because a large number of rule sets are to be generated and evaluated in evolutionary approaches. In this section, we propose a simple but effective idea to improve the applicability of our genetic rule selection algorithm to large data sets.

Figure 2 shows the overview of our approach. To handle a large data set, we use a cluster computer system composed of a server CPU and a number of client CPUs. We can easily set up this system by using four independent desktop computers or a single computer with multi-core CPUs.

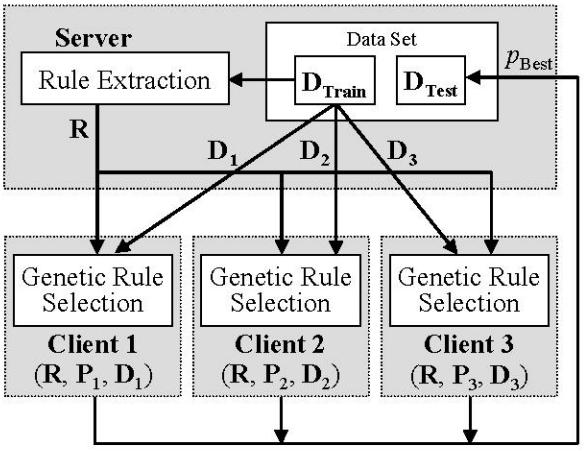

Fig. 2. Parallel distributed genetic fuzzy rule selection.

Our main trick to handle a large data set is to divide a population in GA and training patterns into the same number of subgroups as the number of client CPUs. Let the number of client CPUs be three like Fig. 2. The training patterns are divided into three subgroups. In the same way, the population is also divided into three subpopulations. Then each client CPU performs genetic fuzzy rule selection with a subgroup of training patterns and a subpopulation given by the server CPU.

Of course, each subpopulation easily overfits to the corresponding subgroup of training patterns if the combination is fixed. Thus, we change the subgroups of training patterns after the prespecified generations. Our parallel distributed genetic fuzzy rule selection is implemented in the following manner.

Parallel Distributed Genetic Fuzzy Rule Selection

Step 1: Generate a number of promising rules (i.e., N rules) from the whole training patterns as in Section II.
Step 2: Randomly generate N_{pop} binary strings of length N as an initial population.
Step 3: Randomly divide the current population and the training patterns into subpopulations and subgroups, respectively.
Step 4: Send the subpopulations and the subgroups of the training patterns to client CPUs.
Step 5: Iterate genetic fuzzy rule selection for a prespecified number of generations in each client CPU.
Step 6: Systematically change the subgroups of the training patterns to optimize each subpopulation for a different subgroup of the training patterns.
Step 7: If a prespecified termination condition is not satisfied, return to Step 5. Otherwise go to Step 8.
Step 8: Choose the best fuzzy classifier with respect to (8) from the whole population and examine the generalization ability for test patterns.

Roughly speaking, in the case that the number of CPUs is three, the proposed algorithm is nine times as fast as the original one in Section II. This is because the population size and the number of training patterns for a single CPU are decreased to 1/3, respectively.

IV. Computational Experiments

Through computational experiments on some test problems in the UCI database, we demonstrate advantages of our proposed parallel distributed genetic fuzzy rule selection algorithm in Section III over the original one in Section II.

Table I shows the test problems used in our computational experiments. Whereas these test problems are not actually very large, they can be used to demonstrate the effectiveness of the proposed idea. We use the whole ten-fold cross-validation procedure.

We first extracted candidate rules using the minimum confidence and support levels. Table II shows these values for each test problems. In this paper, we set these values to about ten times the number of training patterns. Table II also shows the average number of extracted rules for the each test problem. Since rule extraction is performed on the server, the number of extracted rules for each approach is the same among the following five implementations. Then genetic fuzzy rule selection was performed using the weight vector (w_1, w_2, w_3) = (100, 1,

1). The population size N_{pop} and the number of client CPUs are 300 and 3, respectively. Thus, the each client CPU has a subpopulation of size 100. The number of generations is 1000.

We examined the following five implementations.

Type 0: Only the server performs rule extraction and genetic fuzzy rule selection. No subdivision is used in this case. Thus, this type is the same as the original algorithm in Section II.

Type 1: Three clients perform genetic fuzzy rule selection. But the subgroups of the training patterns are never changed during the execution.

Type 2: Three clients perform genetic fuzzy rule selection. The subgroups are changed every 100 generations.

Type 3: Three clients perform genetic fuzzy rule selection. The subgroups are changed every 10 generations.

Type 4: Three clients perform genetic fuzzy rule selection. The subgroups are changed every generation.

The CPU time was measured on a workstation with two Xeon 3.0 GHz dual processors (i.e., four CPU cores). We regarded one of them as a server CPU. The others are client CPUs.

TABLE I
DATA SETS USED IN OUR COMPUTATIONAL EXPERIMENTS

Data set	Attributes	Patterns	Classes
Wine	13	178	3
Breast W	9	683*	2
Yeast	8	1484	10

* Incomplete patterns with missing values are not included.

TABLE II
MINIMUM CONFIDENCE AND SUPPORT LEVELS FOR EACH DATA SET, AND THE NUMBER OF GENERATED CANDIDATE RULES

Data set	Confidence	Support	Rules
Wine	0.8	0.1	2137.7
Breast W	0.9	0.2	6882.6
Yeast	0.02	0.002	12338.8

Table III, IV, and V show the average training data accuracy, test data accuracy, number of fuzzy rules, rule length, and CPU time of the best fuzzy classifier on the training patterns over the whole ten-fold cross validation procedure (i.e., ten runs). The proposed approach (i.e., Type 1 to Type 4) could not outperform the conventional approach with respect to the training data accuracy. But some types of them are comparable to the original algorithm. For example, the average training data accuracy of Type 3 is very close to that of Type 0. In Type 3, the subgroups of the training patterns are frequently changed. Thus, each subpopulation was not overfitting to a particular subgroup of the training patterns.

The main advantage of the proposed approach is the decrease in CPU time. In Type 1 to Type 3, CPU time was very short compared with Type 0. But, in Type 4, we needed more CPU time. This is because, whenever applying genetic fuzzy rule selection, we calculated the compatibility grade of each pattern in the given subgroup to avoid the repetition of this calculation of the same pattern with the same rule. We need the modification of this implementation.

Figure 3 shows the trajectories of the average number of fuzzy rules in the best fuzzy classifier on Wisconsin breast cancer data set. The number of CPUs (Core) and the generation interval between subgroup changes are denoted in parentheses. The result on Type 4 is not shown in Fig. 3 because the result was out of the range of the vertical axis. We can see that Type 3 keeps the diversity of the population. This must be the reason why Type 3 is better than the other alternatives.

TABLE III
RESULTS ON WINE DATA SET

	Train	Test	Rules	Length	Time
Type 0	100.00	94.38	5.8	11.2	0:02:24
Type 1	98.13	89.80	5.7	11.6	0:00:25
Type 2	99.13	92.71	5.3	10.9	0:00:25
Type 3	99.94	93.24	4.9	9.6	0:00:33
Type 4	99.69	96.60	18.7	37.9	0:05:58

TABLE IV
RESULTS ON WISCONSIN BREAST CANCER DISEASE DATA SET

	Train	Test	Rules	Length	Time
Type 0	98.44	96.05	5.5	12	0:23:53
Type 1	97.67	96.78	5.4	11	0:03:11
Type 2	97.79	96.93	5.2	11	0:03:15
Type 3	98.34	96.05	5.2	11	0:03:50
Type 4	94.75	94.14	18.0	45	0:13:28

TABLE V
RESULTS ON YEAST DATA SET

	Train	Test	Rules	Length	Time
Type 0	63.81	56.98	39.2	111.5	1:32:46
Type 1	61.03	56.63	25.5	71.9	0:11:34
Type 2	61.71	56.36	22.7	66.3	0:11:37
Type 3	63.05	58.11	23.7	68.6	0:13:52
Type 4	56.28	54.17	82.7	241.5	0:37:09

V. Conclusions

We proposed the parallel distributed genetic fuzzy rule selection algorithm to handle large data sets. Through computational experiments, we examined the effect of the parallel implementation. The main advantage of the proposed approach is low computational time with almost the same performance as the original approach.

Computational complexity strongly depends on the number of candidate rules in our genetic fuzzy rule selection algorithm. We have proposed a candidate rule reduction to deal with this problem in [19], [20]. The combination of parallel distributed implementation and the candidate rule reduction may have a larger effort on the applicability for huge data sets and computational time reduction. As another future work, we will further consider migration and its communication topologies to improve parallel search ability. The extension of our parallel implementation to an evolutionary multiobjective optimization will be an interesting research topic.

This work was partially supported by Grant-in-Aid for Scientific Research on Priority Areas: KAKENHI (18049065) and for Young Scientists (B): KAKENHI (18700228).

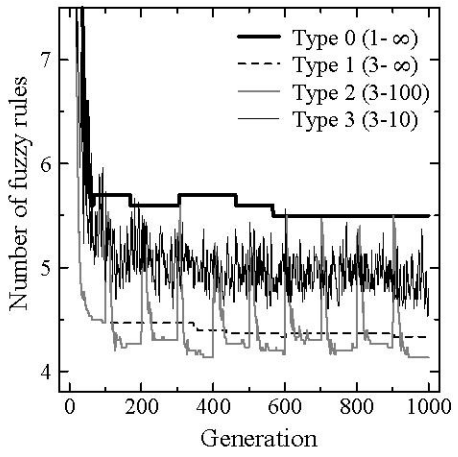

Fig. 3. The trajectories of the average number of fuzzy rules in the best classifier at each generation over the whole 10CV procedure on Wisconsin breast cancer data set.

References

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

[2] H. Ishibuchi, T. Murata, 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, July 1997, pp. 135-150.

[3] H. Ishibuchi, S. Namba, "Evolutionary multiobjective knowledge extraction for high-dimensional pattern classification problems," *Lecture Notes in Computer Science 3242: Parallel Problem Solving from Nature - PPSN VIII*, Springer, Berlin, September 2004, pp. 1123-1132.

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

[5] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, L. Magdalena, "Ten years of genetic fuzzy systems: Current framework and new trends," *Fuzzy Sets and Systems*, 141 (1) 5-31, 2004.

[6] F. Herrera, "Genetic fuzzy systems: Status, critical considerations and future directions," *International Journal of Computational Intelligence Research* 1 (1) 59-67, 2005.

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

[8] J. R. Cano, F. Herrera, M. Lozano, "Stratification for scaling up evolutionary prototype selection," *Pattern Recognition Letters*, vol. 26, no. 7, May 2005, pp. 953-963.

[9] J. R. Cano, F. Herrera, M. Lozano, "On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining," *Applied Soft Computing*, vol. 6, no. 3, March 2006, pp. 323-332.

[10] H. Ishibuchi, K. Nozaki, H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets and Systems*, vol. 52, no. 1, pp. 21-32, November 1992.

[11] H. Ishibuchi, 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.

[12] H. Ishibuchi, T. Nakashima, M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*, Springer, Berlin, November 2004.

[13] H. Ishibuchi, T. Nakashima, T. Morisawa, "Voting in fuzzy rule-based systems for pattern classification problems," *Fuzzy Sets and Systems*, vol. 103, no. 2, pp. 223-238, April 1999.

[14] H. Ishibuchi, T. Yamamoto, "Rule weight specification in fuzzy rule-based classification systems," *IEEE Trans. on Fuzzy Systems*, vol. 13, no. 4, pp. 428-435, August 2005.

[15] O. Cordon, M. J. del Jesus, 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, January 1999.

[16] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo, "Fast discovery of association rules," in U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, pp. 307-328, 1996.

[17] F. Coenen, P. Leng, L. Zhang, "Threshold tuning for improved classification association rule mining," *Lecture Notes in Computer Science 3518: Advances in Knowledge Discovery And Data Mining - PAKDD 2005*, pp. 216-225, Springer, Berlin, May 2005.

[18] A. A. Freitas, Data Mining and Knowledge Discovery with Evolutionary Algorithms, Springer, Berlin, 2002.

[19] H. Ishibuchi, I. Kuwajima, Y. Nojima, "Relation between Pareto-optimal fuzzy rules and Pareto-optimal fuzzy rule sets," *Proc. IEEE SSCI MCDM 2007*, pp. 42-49, April 2007.

[20] H. Ishibuchi, I. Kuwajima, Y. Nojima, "Use of Pareto-optimal and near Pareto-optimal rules as candidate rules in genetic fuzzy rule selection," *Proc. IFSA 2007* (in press).

[21] J. Espinosa, J. Vandewalle, "Constructing fuzzy models with linguistic integrity from numerical data-AFRELI algorithm," *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 5, October 2000.

[22] E. Cantu-Paz, "A survey of parallel genetic algorithms," IlliGAL Report No. 95003, May 1997.