Soft Computing (Special issue on Genetic Fuzzy Systems) manuscript No.
(will be inserted by the editor)

Parallel Distributed Genetic Fuzzy Rule Selection

Yusuke Nojima, Hisao Ishibuchi, Isao Kuwajima

Graduate School of Engineering, Osaka Prefecture University

Received: date / Revised version: date

Abstract Genetic fuzzy rule selection has been successfully used to design accurate and compact fuzzy rule-based classifiers. It is, however, very difficult to handle large data sets due to the increase in computational costs. This paper proposes a simple but effective idea to improve the scalability of genetic fuzzy rule selection to large data sets. Our idea is based on its parallel distributed implementation. Both a training data set and a population are divided into subgroups (i.e., into training data subsets and sub-populations, respectively) for the use of multiple processors. We compare seven variants of the parallel distributed implementation with the original non-parallel algorithm through computational experiments on some benchmark data sets.

Key words Genetic Fuzzy Rule Selection, Parallel Distributed Implementation, Data Subdivision, Fuzzy Rule-based Classifier

1 Introduction

Recently genetic algorithms (GAs) have frequently been used in the field of data mining and knowledge extraction [1]. Their scalability to large data sets, however, is not high. This is because computational costs become expensive when GAs are applied to large data sets. There are two well-known approaches to the decrease in computational costs for the handling of large data sets. One is data reduction, which includes feature selection and instance selection [2–5]. The other is parallel implementation of genetic algorithms, which is usually based on spatial structures such as island and cellular models [6–10]. In addition to the reduction in computational costs, each approach has other benefits. For example, parallel implementation often improves the global search ability of GAs by maintaining the diversity of individuals (i.e., by avoiding premature convergence). On the other hand, data reduction in some cases improves the generalization ability of extracted knowledge by avoiding the overfitting to training data.

Genetic fuzzy rule selection is an effective approach to the design of accurate and compact fuzzy rule-based classifiers [11–13]. It is a two-step approach. In the first phase, a number of promising fuzzy rules are extracted as candidate rules by a data mining technique from training data. In this phase, rule evaluation criteria such as support and confidence are used to prescreening candidate fuzzy rules. In the second phase, only a small number of candidate rules are selected by a GA to maximize the classification accuracy of selected candidate rules and minimize their complexity.

One advantage of genetic fuzzy rule selection over other fuzzy genetics-based machine learning (GBML) algorithms [14–17] is its algorithmic simplicity. Each rule set (i.e., fuzzy rule-based classifier) is represented by a binary string in genetic fuzzy rule selection. This leads to much less implementation costs of genetic fuzzy rule selection than other fuzzy GBML algorithms. When the number of candidate fuzzy rules is small, its computational costs are also usually much less than other fuzzy GBML algorithms [18].

Another advantage is that fuzzy rules in the designed classifier are always meaningful in terms of their support and confidence. This is because these rule evaluation criteria are used for candidate rule prescreening in the first phase. We can use these criteria in various manners for candidate rule prescreening. For example, Pareto-optimality with respect to these two criteria was used to extract candidate fuzzy rules in [19, 20].

Genetic fuzzy rule selection can be viewed as a post-processing procedure in fuzzy data mining for choosing only a small number of fuzzy rules [21]. It is usually very difficult for human users to understand a large number of extracted fuzzy rules by a data mining technique. Thus the selection of only a small number of fuzzy rules helps human users to easily understand the extracted knowledge.

Whereas genetic fuzzy rule selection usually needs much less computational costs than other fuzzy GBML algorithms, its computational costs become unmanageably expensive when it is applied to large data sets. This is because the computational time for fitness evaluation of each individual linearly increases with the number of training patterns.

The aim of this work is to decrease the computational cost for genetic fuzzy rule selection without severe deterioration in the accuracy of designed classifiers. To achieve this aim, we propose a simple but effective idea to improve the scalability of genetic fuzzy rule selection to large data sets. Our idea is based on the subdivision of both a training data set and a population. They are subdivided into training data subsets and sub-populations, respectively. Training data subsets and sub-populations are assigned to different CPUs. Each individual in a sub-population is evaluated by the corresponding training data subset assigned to the same CPU. Thus the computational time for the fitness evaluation of a single individual decreases by the magnitude of the number of training data subsets (i.e., the number of CPUs). Since the fitness evaluation is performed in parallel in each sub-population, the computational time for a single generation decreases by the magnitude of the number of sub-populations (i.e., the number of CPUs). That is, the computational time of genetic fuzzy rule selection decreases by the square of the number of CPUs in our parallel distributed implementation.

This paper is organized as follows. First we briefly review related studies in Section 2. Next we explain the original non-parallel algorithm of genetic fuzzy rule selection in Section 3. Then we explain our idea (i.e., parallel distributed implementation) in Section 4. In Section 5, we examine several variants of the parallel distributed implementation in comparison with the original non-parallel algorithm through computational experiments on some benchmark data sets. Finally we conclude this paper in Section 6.

2 Related Studies

In this section, we briefly review some related studies on parallel distributed GA-based data mining for large data sets.

Araujo et al. [8] proposed GA-PVMINER for parallelizing the fitness calculation of an individual. Both a population and a training data set are divided into subgroups as in our idea. A sub-population and a training data subset are assigned to one processor. In one generation, each sub-population passes through all the processors to calculate the fitness of each individual. This means that the fitness evaluation of each individual is based on the entire training data set. Thus, the effect of parallelization is not so high in comparison with our idea where a different training data subset is used for evaluating each individual.

Llora et al. [9,10] proposed GALE, which is a cellular-based approach. GALE uses a 2D grid for spreading individuals spatially. Each cell contains one or zero individual. Genetic operations are performed in a small neighborhood of each individual. If we can assign a different CPU to each cell, the computational time does not depend on the population size. As in [8], computational time decreases by the magnitude of the number of CPUs in [9,10] whereas it decreases by the square of the number of CPUs in our idea.

Cano et al. [4,5] proposed stratified strategies for instance selection. At its first stage, training patterns are divided into subgroups. Then instance selection is performed on each subgroup in the first stage. The selected patterns are combined and used as candidate patterns in the second stage. The final solution is obtained in the second stage by instance selection from the selected patterns. Any data mining techniques can be used in the second stage of this framework. This approach includes data reduction and parallelization. Whereas all training patterns are always used in each generation (i.e., their subsets are used on different CPUs) in the first stage, many patterns are disregarded in the second stage of [4, 5].

3 Classifier Design by Genetic Rule Selection

In this section, we explain fuzzy rules, fuzzy rule extraction and genetic fuzzy rule selection for classification problems.

3.1 Pattern Classification Problems

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]. This means that the pattern space of our pattern classification problem is an n-dimensional unit-hypercube $[0, 1]^n$.

3.2 Fuzzy Rules for Pattern Classification Problems

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

$$\begin{aligned} \text{Rule } R_q: \text{ If } x_1 \text{ is } A_{q1} \text{ and } ... \text{ and } x_n \text{ is } A_{qn} \\ \text{then Class } C_q \text{ with } CF_q, \end{aligned} \tag{1}$$

where R_q is the label of the q-th fuzzy rule, $\mathbf{x} = (x_1, ..., x_n)$ is an n-dimensional pattern vector, A_{qi} is an antecedent fuzzy set ($i = 1, 2, ..., n$), C_q is a class label,

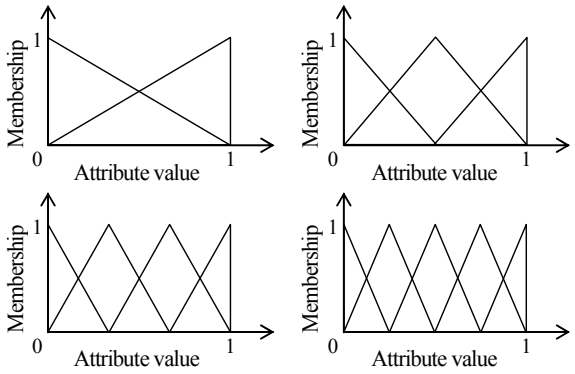

Fig. 1 Four fuzzy partitions used in our computational experiments.

and CF_q is a rule weight (i.e., certainty grade). We denote the antecedent part of the fuzzy rule R_q by the fuzzy vector $\mathbf{A}_q = (A_{q1}, A_{q2}, ..., A_{qn})$. By using $\mathbf{A}_q$, the fuzzy rule R_q is denoted as "$\mathbf{A}_q \Rightarrow C_q$".

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 for fuzzy rule extraction. In our computational experiments, we use four homogeneous fuzzy partitions with triangular 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. Whereas we use only the simple fuzzy partitions in Fig. 1, the use of multiple fuzzy partitions may degrade more or less the interpretability of designed fuzzy rule-based classifiers. This is because some antecedent fuzzy sets are similar to each other. Although an interpretability-accuracy issue of fuzzy rule-based classifiers is not negligible [28,29], we skip discussions on this issue in order to focus our attention on parallel distributed implementation of genetic fuzzy rule selection in this paper.

3.3 Fuzzy Rule Extraction

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 can be used as 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 can be heuristically specified by the compatible training patterns with its antecedent part $\mathbf{A}_q$ in the following manner.

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

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

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 [22]:

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

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

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

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. When multiple classes have the same maximum value in (4), we do not generate any fuzzy rules with $\mathbf{A}_q$ in the antecedent part, either. This specification method of the consequent class of fuzzy rules has been used in many studies since [23].

The rule weight CF_q of each fuzzy rule R_q has a large effect on the performance of fuzzy rule-based classifiers [24]. 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 [25,26]:

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

3.4 Fuzzy Rule Evaluation

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 for human users 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 antecedent interval $[0, 1]$ can be omitted from fuzzy rules. Thus the rule length means the number of antecedent conditions excluding *don't care* conditions. 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.

Among short fuzzy rules, we generate only promising rules as candidate rules in genetic fuzzy rule selection using a heuristic rule evaluation criterion. In the field of

data mining, two rule evaluation criteria (i.e., confidence and support) have often been 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 [22]:

$$s(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{m}. \quad (6)$$

In our computational experiments, we extracted fuzzy rules satisfying pre-specified threshold values of support and confidence (i.e., minimum support and minimum confidence).

3.5 Classification in Fuzzy Rule-Based Classifiers

A subset of candidate fuzzy rules can be viewed as a fuzzy rule-based classifier. Let S be a subset of candidate 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:

$$R_{\mathrm{W}} = \text{argmax}\{\mu_{\mathbf{A}_q}(\mathbf{x}_p) \cdot CF_q \mid R_q \in S\}. \quad (7)$$

As shown in (7), the winner rule R_{W} has the maximum product of the compatibility grade and the rule weight in S. The classification of $\mathbf{x}_p$ is rejected when no rules are compatible with $\mathbf{x}_p$ (which was 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 fuzzy rules. Thus, the classification of $\mathbf{x}_p$ is also rejected when multiple fuzzy rules with different consequent classes have the same maximum value in (7).

For other fuzzy reasoning methods for pattern classification problems, see Cordon et al. [27] and Ishibuchi et al. [23, 25].

3.6 Genetic Fuzzy Rule Selection

Let us assume that N candidate fuzzy rules have already been extracted. The task of genetic fuzzy rule selection is to design an accurate and compact fuzzy rule-based classifier from the N candidate fuzzy rules.

Any subset S of the N candidate fuzzy rules can be denoted by a binary string of length N as $S = s_1 s_2 \cdots s_N$ where $s_i = 1$ and $s_i = 0$ mean that the i-th candidate fuzzy 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 each training pattern $\mathbf{x}_p$ by the rule set S. The classification is based on the single winner-based method explained in the previous subsection.

The second objective is calculated by just counting the number of 1's (i.e., the number of selected fuzzy 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 fuzzy rules may be used for the classification of no training patterns. Whereas the existence of such an unnecessary fuzzy rule in the rule set S has no effect on the first objective, it deteriorates the second and third objectives. Thus we remove from the rule set S all the unnecessary rules responsible for the classification of no training patterns before the second and third objectives are calculated. The third objective is the total number of antecedent conditions excluding *don't care* conditions of the selected fuzzy rules in S.

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

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

where w_1, w_2, and w_3 are pre-specified 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 fuzzy 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 Fuzzy Rule Selection
Phase I: Candidate Rule Extraction
Step 1: Extract candidate fuzzy rules from the training patterns using pre-specified values of the minimum support, the minimum confidence, and the maximum rule length. Let the number of extracted candidate fuzzy rules be N.
Phase II: Genetic Optimization of Rule Sets
Step 2: Randomly generate N_{pop} binary strings of length N as an initial population where N_{pop} is the population size. Calculate the fitness value of each string using the fitness function in (8) after removing unnecessary rules.
Step 3: Iterate the following operations N_{pop} times to generate an offspring population of N_{pop} strings.
3.1: Select a pair of parent strings from the current population by binary tournament selection with replacement.
3.2: Recombine the selected pair of parent strings to generate new strings 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 parent strings, 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., the number of selected fuzzy 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: Calculate the fitness value of the offspring string by the fitness function in (8) after removing unnecessary rules.

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

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

We use the total number of evaluated strings 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.

4 Parallel Distributed Implementation

In this section, we propose a simple but effective idea to improve the scalability of genetic fuzzy rule selection to large data sets.

Figure 2 explains a computer system used in our computational experiments in this paper. We use a cluster computer system with a single server CPU and a number of client CPUs (three client CPUs in our computational experiments). We can easily set up this system using multiple independent desktop computers and/or a single computer with multi-core CPUs. Currently we are developing a cluster computer system with 12 client CPUs [36].

Our idea to improve the scalability of genetic fuzzy rule selection to large data sets is to divide not only a population but also a training data set. They are divided into the same number of sub-populations and training data subsets, which is also the same as the number of client CPUs. Let us assume that the number of client CPUs is three as in Fig. 2. In this case, the training data set and the population are divided into three training data subsets and three sub-populations, respectively. Then each client CPU performs genetic fuzzy rule selection using a single training data subset and a single sub-population given by the server CPU.

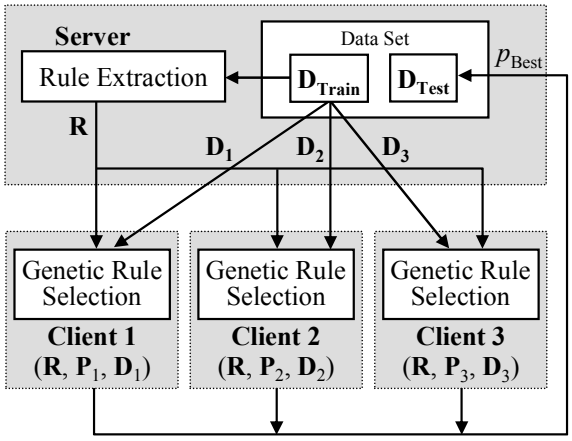

Fig. 2 Cluster computer system used for parallel distributed implementation of genetic fuzzy rule selection.

It seems that each sub-population is likely to overfit to the corresponding training data subset. Thus, we change the assignment of the training data subsets to the client CPUs after a pre-specified number of generations (i.e., every ten generations).

Our parallel distributed implementation of genetic fuzzy rule selection is written as follows:

Parallel Distributed Implementation
Phase I: Candidate Rule Extraction
Step 1: Extract candidate fuzzy rules in the same manner as in Section 3. This phase is executed on the server CPU. Let the number of extracted fuzzy rules be N.
Phase II: Genetic Optimization of Rule Sets
Step 2: Randomly generate N_{pop} binary strings of length N as an initial population on the server CPU.
Step 3: Randomly divide the current population and the training data set into sub-populations and training data subsets, respectively, on the server CPU.
Step 4: Distribute the sub-populations and the training data subsets from the server CPU to the client CPUs.
Step 5: Evaluate each string in the sub-population using the assigned training data subset on each client CPU.
Step 6: Execute genetic fuzzy rule selection for a pre-specified computation load (which is specified by the total number of evaluated strings in this paper) on each client CPU using the assigned training data subset and the assigned sub-population.
Step 7: Systematically change the assignment of the training data subsets to the client CPUs (e.g., from the first client CPU to the second one, from the second one to the third one, and from the third one to the first one in the case of three client CPUs).
Step 8: If a pre-specified termination condition (the total number of evaluated strings in this paper) is not satisfied, return to Step 5. Otherwise go to Step 9.
Step 9: Calculate the fitness value of each string in each sub-population using the whole training data set on the server CPU. Choose the best string as the final solution (i.e., as the finally obtained fuzzy rule-based classifier).

Our parallel distributed implementation decreases the computational time by the magnitude of the square of the number of client CPUs. For example, it is nine times faster than the original non-parallel algorithm in Section 2 when we have three client CPUs. This is because both the population size and the number of training patterns at each client CPU are 1/3 of those in the original non-parallel algorithm.

5 Computational Experiments

Through computational experiments on some benchmark data sets in the UCI machine learning repository, we examined several variants of the proposed parallel distributed implementation in comparison with the original non-parallel algorithm.

Table 1 shows the benchmark data sets used in our computational experiments. Whereas these data sets in Table 1 are not actually very large, they can be used to demonstrate the effectiveness of the proposed idea. We evaluated the generalization ability of obtained fuzzy rule-based classifiers by iterating the ten-fold cross validation procedure two times (i.e., 2×10CV).

We first extracted candidate fuzzy rules using the minimum confidence, the minimum support, and the maximum rule length. The maximum rule length was specified as three for all the data sets. Table 2 shows the minimum support and the minimum confidence used for each data set. We also show the average number of extracted candidate rules and the average CPU time for rule extraction (hour: minute: second) in Table 2. Since the candidate rule extraction phase was performed using the entire training data set, the same candidate rules were extracted in all variants examined in our computational experiments. Then genetic fuzzy rule selection was performed. The weight vector in the weighted sum fitness function in (8) was specified as $\mathbf{w} = (100, 1, 1)$. We used three client CPUs. The population size N_{pop} was specified as 300 (i.e., the size of each sub-population was 100). The total number of evaluated strings for each variant was specified as 300300. This is equal to an initial population with 300 strings plus 1000 generation updates in the case of non-parallel genetic fuzzy rule selection.

We examined the following eight variants of genetic fuzzy rule selection (one original non-parallel algorithm and seven parallel distributed ones).

Type 0: The original non-parallel algorithm, which was executed at a single server CPU.

Type 1: A parallel distributed algorithm, which was executed at a cluster system with a single sever CPU and three client CPUs. The assignment of training data subsets to the client CPUs was not changed.

Type 2: The same algorithm as Type 1 except that the assignment of training data subsets was changed every 100 generations.

Table 1 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
Pendig	16	10992	10

∗ Incomplete patterns with missing values are not included.

Table 2 Minimum confidence and support levels, the average number of generated candidate rules, and the average CPU time for candidate rule extraction for each data set.

Data set	Confidence	Support	Rules	Time
Wine	0.8	0.1	2137.7	0:00:08
Breast W	0.9	0.2	6882.6	0:00:09
Yeast	0.5	0.002	12338.8	0:00:15
Pendig	0.5	0.04	18297.9	0:19:12

Type 3: The same algorithm as Type 1 except that the assignment of training data subsets was changed every 10 generations.

Type 4: The same algorithm as Type 1 except that the assignment of training data subsets was changed every generation.

Type 5: The same algorithm as Type 3 (i.e., the assignment of training data subsets was changed every 10 generations) except that the population subdivision was randomly performed every 200 generations. This means that 300 strings in the current population (i.e., three sub-populations) were randomly reassigned to the three client CPUs every 200 generations.

Type 6: The same algorithm as Type 5 except that the population subdivision was randomly performed every 100 generations.

Type 7: The same algorithm as Type 5 except that the population subdivision was randomly performed every 10 generations.

The last three types (i.e., Types 5-7) can be viewed as the incorporation of a very simple migration procedure into our parallel distributed implementation.

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

Tables 3-6 show the average training data accuracy, the average test data accuracy, the average number of selected fuzzy rules, the average total rule length, and the average CPU time (hour: minute: second) over two iterations of the ten-fold cross validation procedure (i.e., over 20 runs). We performed statistical tests [30] for examining the statistical significance of the difference between the original non-parallel algorithm (i.e., Type 0) and our parallel distributed implementation (i.e., Type 1 - Type 7) in the training data accuracy and the test

data accuracy. We used a paired student's t-test when the distribution of experimental results can be regarded as a normal distribution. Otherwise, we used a Wilcoxon signed-ranks test. Average classification rates which are significantly different with the significance level $\alpha = 0.05$ from the results by Type 0 are underlined in each table. In the same way, the rates which are strongly significantly different with the significance level $\alpha = 0.01$ are highlighted by bold face.

As shown in Tables 3-6, our parallel distributed implementation without too frequent assignment changes for a small data set (i.e., except for Type 4 on the Wine data set) decreased the average CPU time of Type 0 (i.e., the original non-parallel algorithm). The decrease in the CPU time was more significant in the case of larger data sets (i.e., Yeast and Pendig data sets). This observation shows that our parallel distributed implementation can improve the scalability of genetic fuzzy rule selection to large data sets.

As we can see from many bold-face fonts and underlines in the second column labeled as "training" in Tables 3-6, the training data accuracy was significantly degraded by the parallel distributed implementation in many cases. This is because genetic fuzzy rule selection was performed at each client CPU by using only a part of training data. We can also see that the periodical reassignment of training data subsets to the client CPUs (i.e., Type 2 and Type 3) and the population re-subdivision (i.e., Type 5 and Type 6) somewhat improved the training data accuracy of Type 1 with no reassignment. This is because these procedures can help genetic fuzzy rule selection to adapt the entire training data. On the contrary, the reassignment and re-subdivision at every generation (i.e., Type 4 and Type 7) did not work well. Too frequent reassignment and re-subdivision may disturb the genetic search for good rule sets.

Whereas the training data accuracy was significantly degraded by the parallel distributed implementation in many cases in Tables 3-6, the test data accuracy was not significantly degraded with only a few exceptions. Almost the same test data accuracy was obtained from the original non-parallel algorithm and the parallel distributed implementation in many cases. This observation clearly shows the usefulness of our parallel distributed implementation since the average CPU time was drastically decreased by our parallel distributed implementation.

We can further observe that the complexity (i.e., the number of fuzzy rules and the total rule length) was also decreased by our parallel distributed implementation, especially for large data sets. This is a by-product of the training data subdivision.

Table 3 Results on the Wine data set.

	Training	Test	Rules	Length	Time
Type 0	100.00	93.82	5.80	11.05	0:02:52
Type 1	**98.06**	91.80	5.65	11.05	0:00:26
Type 2	**98.84**	92.43	5.35	12.10	0:00:27
Type 3	99.97	95.18	5.80	12.15	0:00:33
Type 4	100.00	93.25	6.65	12.85	0:03:15
Type 5	100.00	94.33	5.55	11.50	0:00:33
Type 6	99.84	93.76	5.20	10.40	0:00:33
Type 7	**98.53**	94.95	6.35	11.85	0:00:35

Table 4 Results on the Breast W data set.

	Training	Test	Rules	Length	Time
Type 0	98.55	96.12	5.75	12.50	0:26:53
Type 1	**97.75**	96.93	5.45	11.10	0:03:20
Type 2	**97.84**	96.41	5.25	10.95	0:03:19
Type 3	**98.42**	95.90	5.25	11.45	0:03:33
Type 4	**98.27**	96.34	5.45	12.00	0:06:10
Type 5	**98.41**	96.27	5.55	12.40	0:03:49
Type 6	**98.34**	96.20	5.10	10.75	0:03:33
Type 7	**97.82**	96.27	5.40	11.35	0:03:35

Table 5 Results on the Yeast data set.

	Training	Test	Rules	Length	Time
Type 0	63.77	56.93	37.85	107.20	2:11:24
Type 1	**60.91**	56.31	25.65	72.00	0:15:06
Type 2	**61.51**	56.14	23.00	67.50	0:14:33
Type 3	**63.23**	57.42	22.45	65.45	0:14:56
Type 4	**61.31**	56.71	22.85	65.50	0:17:48
Type 5	63.41	57.25	23.15	66.95	0:15:03
Type 6	63.26	57.18	23.50	68.60	0:14:57
Type 7	**60.67**	56.27	24.15	69.25	0:15:24

Table 6 Results on the Pendig data set.

	Training	Test	Rules	Length	Time
Type 0	80.94	80.32	40.35	117.60	24:43:27
Type 1	**80.38**	**79.81**	30.80	89.35	2:42:14
Type 2	**80.79**	80.32	28.25	82.95	2:43:14
Type 3	**80.75**	80.26	29.70	86.20	2:55:12
Type 4	**80.12**	**79.64**	28.25	82.05	2:57:34
Type 5	**80.75**	80.13	30.00	87.20	2:54:11
Type 6	80.82	80.11	30.30	88.05	2:54:22
Type 7	**80.38**	**79.98**	29.40	85.75	2:48:37

6 Conclusions

In this paper, we proposed a parallel distributed implementation of genetic fuzzy rule selection to improve its scalability to large data sets. Through computational experiments, it was shown that the proposed parallel distributed implementation found fuzzy rule-based classifiers with almost the same test data accuracy as the original non-parallel algorithm while it drastically decreased the average CPU time. It was also shown that the reassignment of training data subsets helped our parallel distributed implementation to find good fuzzy rule-based classifiers with high generalization ability.

The extension of parallel distributed genetic fuzzy rule selection to evolutionary multiobjective optimization [31–35] is an interesting future research issue.

This work was partially supported by Foundation for C&C Promotion and Grant-in-Aid for Young Scientists (B): KAKENHI (18700228).

References

1. A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, (Springer, Berlin, 2002).
2. H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, (Kluwer, 1998).
3. H. Liu and H. Motoda, *Instance Selection and Construction for Data Mining*, (Kluwer, 1998).
4. J. R. Cano, F. Herrera, and M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognition Letters, **26** 7 (2005) 953-963.
5. J. R. Cano, F. Herrera, and M. Lozano, On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining, Applied Soft Computing, **6** 3 (2006) 323-332.
6. E. Cantu-Paz, A survey of parallel genetic algorithms, IlliGAL Report No. 95003 (1997).
7. E. Alba and M. Tomassini, Parallelism and Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, **6** 5 (2002) 443-462.
8. D. L. A. Araujo, H. S. Lopes, and A. A. Freitas, Rule discovery with a parallel genetic algorithm, Proc. of GECCO Workshop on Data Mining with Evolutionary Computation (2000) 89-92.
9. X. Llora and J. M. Garrell, Knowledge-independent data mining with fine-grained parallel evolutionary algorithms, Proc. of the Genetic and Evolutionary Computation Conference (2001) 461-468.
10. X. Llora and J. M. Garrell, Coevolving different knowledge representations with fine-grained parallel learning classifier systems, Proc. of the Genetic and Evolutionary Computation Conference (2002) 934-941.
11. H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, Selecting fuzzy if-then rules for classification problems using genetic algorithms, IEEE Transactions on Fuzzy Systems, **3**, 3 (1995) 260-270.
12. 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, **89**, 2 (1997) 135-150.
13. H. Ishibuchi and T. Yamamoto, Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining, Fuzzy Sets and Systems **141** (1) (2004) 59-88.
14. O. Cordon, F. Herrera, F. Hoffman, and L. Magdalena, *Genetic Fuzzy Systems* (World Scientific, 2001).
15. O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, Ten years of genetic fuzzy systems: Current framework and new trends, Fuzzy Sets and Systems, **141** 1 (2004) 5-31.
16. F. Herrera, Genetic fuzzy systems: Status, critical considerations and future directions, International Journal of Computational Intelligence Research **1** 1 (2005) 59-67.
17. H. Ishibuchi, T. Nakashima, and T. Murata, Three-objective genetics-based machine learning for linguistic rule extraction, Information Sciences, **136** 1-4 (2001) 109-133.
18. Y. Nojima, H. Ishibuchi, and I. Kuwajima, Comparison of Search Ability between Genetic Fuzzy Rule Selection and Fuzzy Genetics-Based Machine Learning, Proc. of 2006 International Symposium on Evolving Fuzzy Systems (2006) 125-130.
19. H. Ishibuchi, I. Kuwajima, and Y. Nojima, Use of Pareto-optimal and near Pareto-optimal rules as candidate rules in genetic fuzzy rule selection, In P. Melin et al (eds.): Analysis and Design of Intelligent Systems using Soft Computing Techniques (Advances in Soft Computing 41), Springer, Berlin (2007) 387-396.
20. H. Ishibuchi, Evolutionary multiobjective design of fuzzy rule-based systems, Proc. of First IEEE Symposium on Foundations of Computational Intelligence (2007) 9-16.
21. H. Ishibuchi, Y. Nojima, and I. Kuwajima, Genetic Rule Selection as a Postprocessing Procedure in Fuzzy Data Mining, Proc. of 2006 International Symposium on Evolving Fuzzy Systems (2006) 286-291.
22. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, Fast discovery of association rules, in U. M. Fayyad et al (eds.), Advances in Knowledge Discovery and Data Mining, AAAI Press, Menlo Park (1996) 307-328.
23. H. Ishibuchi, K. Nozaki, and H. Tanaka, Distributed representation of fuzzy rules and its application to pattern classification, Fuzzy Sets and Systems, **52** 1 (1992) 21-32.
24. H. Ishibuchi, and T. Nakashima, Effect of rule weights in fuzzy rule-based classification systems, IEEE Transactions on Fuzzy Systems, **9** 4 (2001) 506-515.
25. H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*, (Springer, Berlin, 2004).
26. H. Ishibuchi, T. Nakashima, and T. Morisawa, Voting in fuzzy rule-based systems for pattern classification problems, Fuzzy Sets and Systems, **103** 2 (1999) 223-238.
27. 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, **20** 1 (1999) 21-45.
28. J. Casillas, O. Cordon, F. Herrera, and L. Magdalena (eds.), Interpretability Issues in Fuzzy Modeling, Springer, Berlin (2003).
29. J. Casillas, O. Cordon, F. Herrera, and L. Magdalena (eds.), Accuracy Improvements in Linguistic Fuzzy Modeling, Springer, Berlin (2003).
30. D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & Hall, 4th edition (2007).
31. A. Abraham, L. Jain, and R. Goldberg (eds.), Evolutionary Multiobjective Optimization. Springer, London (2005).
32. C. A. C. Coello, A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowledge and Information Systems, **1** 3 (1999) 269-308.
33. C. A. C. Coello, D. A. van Veldhuizen, and G. B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, Boston (2002).

34. K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Chichester (2001).
35. Y. Jin (ed.), Multi-Objective Machine Learning, Springer, Berlin (2006).
36. Y. Nojima and H. Ishibuchi, Computational efficiency of parallel distributed genetic fuzzy rule selection for large data sets, Proc. of Information Processing and Management of Uncertainty in Knowledge-Based Systems (2008) (under review).