Y. Nojima and H. Ishibuchi, "Effects of parallel distributed implementation on the search performance of Pittsburgh-style genetics-based machine learning algorithms," *Proc. of 2016 IEEE Congress on Evolutionary Computation (Accepted)*.

Effects of Parallel Distributed Implementation on the Search Performance of Pittsburgh-style Genetics-based Machine Learning Algorithms

Yusuke Nojima and Hisao Ishibuchi
Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University
Sakai, Osaka 599-8531, Japan
{nojima@, hisaoi@}cs.osakafu-u.ac.jp

***Abstract*—Pittsburgh-style genetics-based machine learning (GBML) algorithms have strong search ability for obtaining rule-based classifiers. However, when we apply them to data mining from large data, we need huge computation time for fitness evaluation. In our previous studies, we have proposed parallel distributed implementation of fuzzy GBML for fuzzy classifier design from large data. The basic idea of our parallel distributed implementation is to divide not only a population but also a training data set into *N* sub-populations and *N* training data subsets, respectively. A pair of a sub-population and a training data subset is assigned to each of *N* CPU cores in a workstation or a cluster. This dual division strategy achieved a quadratic speedup (i.e., N^2 times faster than the use of a single CPU core) while maintaining the generalization ability on test data. In this paper, we apply our parallel distributed implementation to GAssist which is a non-fuzzy Pittsburgh-style GBML algorithm. We examine the effects of the number of divisions on the search ability comparing with the parallel distributed fuzzy GBML.**

Keywords*—Pittsburgh-style genetics-based machine learning; parallel distributed implementation, classifier design.

I. Introduction

Genetics-based machine leaning (GBML) and learning classifier systems (LCS) are the representative evolutionary rule-based systems [1]-[4]. There are mainly three categories of GBML. One is the Pittsburgh approach where a set of if-then rules (i.e., a rule-based classifier) is coded as an individual. The population includes a number of individuals (i.e., classifiers) to be optimized. Another approach is the Michigan approach where an if-then rule is coded as an individual. The population represents a classifier. The other is the iterative rule learning approach where an if-then rule is coded as an individual. A classifier is generated by a number of algorithm executions. Among three categories, only the Pittsburgh approach can directly optimize the classifier and may find the global optima. In compensation for the high search ability, the Pittsburgh approach needs longer computation time than the other two approaches. This disadvantage becomes very serious when we apply the Pittsburgh-style GBML algorithms to data mining from large data. Each individual needs long computation time in order to classify a large number of patterns. As a result, the total computation time becomes very long.

This work was supported by JSPS KAKENHI Grant Numbers 25330292 and 16K00336.

In our previous studies [5]-[6], we have proposed parallel distributed implementation of evolutionary fuzzy systems (EFS). In the field of EFS [7]-[8], any part of a fuzzy system (e.g., the combinations of antecedent conditions and candidate rules, and the shapes of fuzzy membership functions) can be optimized by evolutionary computation. The Pittsburgh-style EFS methods also have the same problem for the application of data mining to large data. To solve this problem, we first proposed our parallel distributed implementation to genetic fuzzy rule selection [5]. Later, we applied it to our hybrid fuzzy GBML algorithm [6]. In these studies, we showed that our parallel distributed implementation can drastically reduce the computation time while maintaining the generalization ability on the test data. To examine its applicability to non-fuzzy methods, we applied our parallel distributed implementation to GAssist [9] which is one of the representative Pittsburgh-style GBML algorithms [10]. In this paper, we extend [9] including the results of more data sets and the comparison with our hybrid fuzzy GBML.

This paper is organized as follows. Section II explains our parallel distributed implementation of GBML. Section III briefly explains GAssist and the hybrid fuzzy GBML. Section IV shows the experimental results using ten data sets. Finally, Section V concludes this paper.

II. Parallel Distributed Implementation

The basic idea of our parallel distributed implementation is based on the parallelism of evolutionary computation like island models and data reduction [5]-[6]. Let us assume that *N* CPU cores in a workstation are available. Fig. 1 illustrates our parallel distributed implementation. A population is divided into *N* sub-populations. A training data set is also divided into *N* training data subsets. A pair of a sub-population and a training data subset is assigned to one of CPU cores. In each CPU core, GBML is performed in parallel. That is, each sub-population is optimized using the corresponding training data subset. Besides, the rotation of training data subsets is periodically performed in order to avoid the overfitting of the sub-population to the single training data subset. This rotation process gives perturbation to each sub-population. As island models, individual migration is also periodically applied. The

worst individual in each sub-population is replaced with the best one in the adjacent sub-population.

Through computational experiments, we showed that our parallel distributed implementation can drastically reduce the computation time [5], [9]. Since $1/N$ of the population is optimized using $1/N$ of the training data, the computation time can be reduced to $1/N^2$ of that by using a single CPU core. That is, we can achieve a quadratic speedup by using our parallel distributed implementation (i.e., N^2 times faster than non-parallel and non-distributed models).

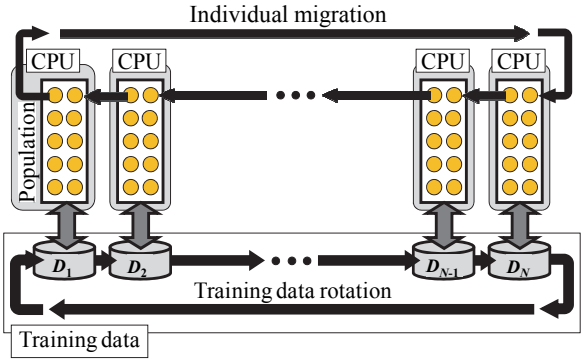

Fig. 1. Parallel distributed implementation.

We extended the framework of our parallel distributed implementation to imbalanced data mining [11], ensemble classifier design [12], and privacy preserving data mining [13]. The application of our parallel distributed implementation to multiobjective EFSs is also studied in [14]-[15].

In our model in Fig. 1, training data are rotated. However, this is not a good strategy when the size of the training data is huge. In this case, the same effect of the training data rotation can be implemented by the the rotation of sub-populations. Once a training data subset is assigned to each CPU in this implementation, no training data rotation is performed.

III. Pittsburgh-style GBML Algorithms

A. GAssist

GAssist is a Pittsburgh-style GBML algorithm proposed by J. Bacardit in 2004 [10]. We modified the source code of GAssist in Keel [16] to adapt it to our parallel distributed implementation.

For data sets with continuous values, adaptive discretization intervals [17] is employed to discretize continuous attributes into crisp intervals. In this paper, we use uniform widths with very coarse to very fine granularities (4, 5, 6, 7, 8, 10, 15, 20, 25 partitions). This is the default setting of GAssist in Keel. Which crisp intervals to use for each attribute are dynamically determined by joining and splitting sets of micro-intervals.

Each rule has the interval vector $\mathbf{I}$ consisting of sets of micro-intervals for classifying an input pattern $\mathbf{x}$. If all attribute values of $\mathbf{x}$ are within the intervals defined by a rule, the pattern is classified as the class of this rule. An individual (i.e., a classifier) consists of an ordered rule set as follows:

$$\begin{aligned}&\text{If } \mathbf{x} \text{ is } \mathbf{I}_1 \text{ then Class } C_1,\\ &\text{else if } \mathbf{x} \text{ is } \mathbf{I}_2 \text{ then Class } C_2,\\ &\quad\vdots\\ &\text{else if } \mathbf{x} \text{ is } \mathbf{I}_m \text{ then Class } C_m,\\ &\text{else Class } C_{default},\end{aligned} \tag{1}$$

where $\mathbf{I}_i$ and C_i are an interval vector and a class label of the ith rule, respectively. The upper rule has the highest priority. When a pattern is not classified by any rules, it is classified as the default class $C_{default}$ [18]. For comparison, we regard the number of rules in the above rule set as $(m + 1)$.

The fitness function of GAssist is based on the minimum description length (MDL) principle [19]:

$$fitness(S) = f_1(S) + wf_2(S), \tag{2}$$

where S and w represent a classifier and an automatically adjusted weight, respectively. $f_1(S)$ is defined as follows:

$$f_1(S) = 105 - \text{Accuracy}(S), \tag{3}$$

where Accuracy(S) is the ratio of correctly classified training patterns in percentage. $f_2(S)$ is the total number of intervals of all rules in S. The default rule is not counted in $f_2(S)$. The weight w is reduced from an initial value if no superior individual is found.

As population updating, the newly generated offspring population is used as the next population, with a single elite individual being carried over from the previous population.

To apply our parallel distributed implementation to GAssist, we slightly adjust some parts as follows:

1. Frequency of training data rotation,
2. Weight updating in the fitness function,
3. Elite preservation.

While the best interval of the training data rotation was 100 for our parallel distributed fuzzy GBML in [6], we specify it for parallel distributed GAssist as 1 (i.e., every generation). This specification is based on the preliminary experiments.

GAssist uses an updating mechanism for the weight in the fitness function. For parallel distributed GAssist, the weight corresponding to each sub-population is updated independent of the other sub-populations. The problem is how to select the final individual. At the final generation, the weights of the sub-populations may be different from one another. Thus, we select the best individual with respect to the training data accuracy instead of the fitness function of (2).

GAssist normally utilizes the windowing scheme. The training data set is divided into a number of training data subsets (i.e., windows). Each individual is evaluated by using one of them. The rotation of training data windows is also performed in GAssist. The idea is the same as our parallel distributed implementation. The difference is that one elite individual is held per each training data window. The elite individual is compared within only the same training data window. For parallel distributed GAssist, each sub-population maintains an elite individual per each training data window. Thus, the whole population has N^2 elite individuals at the same time.

B. Hybrid Fuzzy GBML

The basic framework of our hybrid fuzzy GBML [20] is the Pittsburgh approach, while a single iteration of the Michigan approach is employed as local search with a prespecified probability.

In the hybrid fuzzy GBML, fuzzy membership functions are used for the antecedent conditions. In this paper, triangular membership functions with different granularities (i.e., from two to five partitions) are simultaneously used. "*don't care*" condition is also used for reducing the complexity of each rule.

Each rule has a set of membership functions **A**. An individual consists of a number of fuzzy if-then rules as:

$$\begin{aligned} &\text{If } \mathbf{x} \text{ is } \mathbf{A}_1 \text{ then Class } C_1 \text{ with } CF_1, \\ &\text{If } \mathbf{x} \text{ is } \mathbf{A}_2 \text{ then Class } C_2 \text{ with } CF_2, \\ &\quad \vdots \\ &\text{If } \mathbf{x} \text{ is } \mathbf{A}_m \text{ then Class } C_m \text{ with } CF_m, \end{aligned} \tag{4}$$

where $\mathbf{A}_i$ is a set of fuzzy membership functions in the antecedent part of the ith rule. C_i and CF_i are a class label, and a rule weight of the ith rule, respectively. The class label and the rule weight are determined by the compatibility grade of the training patterns in a heuristic manner.

The input pattern **x** is classified as the class of the rule which has the maximum product of the compatibility grade for the pattern **x** and the rule weight.

The following fitness function is used in our fuzzy GBML:

$$fitness(S) = w_1 f_1(S) + w_2 f_2(S) + w_3 f_3(S), \tag{5}$$

where w_1, w_2 and w_3 are non-negative weights. In this paper, we specify (w_1, w_2, w_3) as (100, 1, 1). $f_1(S)$ is the error rate by S on the training patterns in percentage. $f_2(S)$ is the number of fuzzy rules in S. $f_3(S)$ is the total number of fuzzy membership functions of all rules in S.

At initialization, a number of rule sets (i.e., classifiers) are generated to make a population. Each rule in a classifier is generated from a randomly selected training pattern. At each generation, new individuals are generated by crossover and mutation. In addition, a single iteration of the Michigan approach is used for generating some rules from the existing ones in an offspring by crossover and mutation. Some other rules are also generated from misclassified patterns in a heuristic manner. The generation update is $(\mu+\mu)$-ES type. See [6] in more detail.

IV. Computational Experiments

We examine the effects of the number of divisions of the population and the training data. We also compare between GAssist and our fuzzy GBML using ten large data sets.

A. Experimental Settings

All experiments in this paper are performed on a normal workstation with Intel Xeon CPU E5-2660 (8 core 2.20 GHz) × 2 and 32 GB RAM. That is, the workstation has 16 CPU cores in total. The population size of non-parallel non-distributed models is specified as 200. For parallel distributed models, the sub-population size is evenly divided amongst the used CPU cores. It is not always possible to evenly divide amongst the used CPU cores. In such cases, we round the population size up to the nearest divisible number. We examine eight specifications on the number of divisions (i.e., the used CPU cores) and the (sub-)population size shown in Table I. The number of training data subsets is the same as the number of sub-populations in this paper. The termination condition is 10,000 generations.

The data sets used in this paper are listed in Table II. Ten-fold cross-validation is performed five times using different data partitions. That is, the average results are calculated over 50 runs of each algorithm (i.e., 5 x 10CV).

The parameters of GAssist and our fuzzy GBML are listed in Tables III and IV, respectively.

TABLE I. The Number of Used CPU Cores and The Population Size

No. CPU cores	1	3	5	7	9	11	13	15
Population size	200	201	200	203	207	209	208	210
Sub-population size	-	67	40	29	23	19	16	14

TABLE II. Data Sets Used in This Study

Data name	# patterns	# attributes	# classes
Yeast	1,484	8	10
Segment	2,310	19	7
Phoneme	5,404	5	2
Page-blocks	5,472	10	5
Texture	5,500	40	11
Satimage	6,435	36	7
Ring	7,400	20	2
Twonorm	7,400	20	2
PenBased	10,992	16	10
Magic	19,020	10	2

TABLE III. The Parameter Setting of GAssist

Parameter	Value
Default class	Majority
Initial number of rules	20
Probability crossover	0.6
Probability mutation	0.6
Probability one	0.9
MDL initial ratio	0.075
MDL starting iteration	25
MDL weight relax factor	0.9
Tournament size	3
Class wise initialization	Yes
Smart initialization	Yes
Probability merge	0.05
Probability split	0.05
Probability reinitialize	0.03
Probability reinitialize at end	0.00
Rule deletion starting iteration	5
Rule deletion minimum rules	12
Penalize rule sets with less rules than	4

TABLE IV. THE PARAMETER SETTING OF OUR FUZZY GBML

Parameter	Value
Initial number of rules	30
Crossover probability	0.9
Pittsburgh mutation probability	$1 / (\|S\| n)$
Michigan mutation probability	$1 / n$
Rule set upper limit	60
Michigan style probability	0.5
"*don't care*" probability	$(n - 5) / n$

* n: Number of attributes

B. Training Data Division

In [21], several schemes for dividing data in k-fold cross-validation are examined. These methods can also be utilized for dividing the training data set in our parallel distributed implementation. To investigate this possibility, we compare two methods: the standard stratified cross-validation (SCV) and the distribution optimally balanced SCV (DOB-SCV).

SCV is the most commonly used method which simply counts how many patterns of each class there are in the data set and then distributes them evenly in the subsets in a random manner. This method does not consider the properties of the data set. Since this method performs the division by random, training data subsets may inappropriately be biased.

DOB-SCV assigns close-by patterns to different subsets in order to avoid the bias of the pattern distribution. By using similar training data subsets, the fitness of each individual is expected to be almost the same even if the assignment of the training data subset is changed.

Table V shows the test data accuracy when the number of divisions was specified as seven. The results totally depended on the data sets and the algorithms. While the difference between SCV and DOB-SCV was clear on average in GAssist, it was very slight in our fuzzy GBML. From the results, we use DOB-SCV as the training data division method for the following experiments.

TABLE V. TEST DATA ACCURACY USING SCV AND DOB-SCV.

	GAssist Test data accuracy (%)		Fuzzy GBML Test data accuracy (%)	
Data Set	SCV	DOB-SCV	SCV	DOB-SCV
Yeast	**56.78**	56.75	54.89	**56.24**
Segment	92.89	**92.95**	**92.91**	92.62
Phoneme	83.98	**84.37**	83.31	**83.78**
Page-blocks	**94.55**	94.53	96.06	**96.18**
Texture	81.92	**82.71**	94.30	**94.41**
Satimage	**84.03**	83.88	**85.85**	85.82
Ring	91.74	**91.77**	**93.31**	92.21
Twonorm	85.46	**85.65**	**95.57**	95.11
Penbased	88.44	**89.05**	**95.29**	95.20
Magic	84.31	**84.33**	84.68	**84.74**
Average	83.22	**84.60**	87.62	**87.63**

C. Experimental Results

Figs. 2-5 show the training data accuracy, the test data accuracy, the number of rules, and the computation time for each data set. The horizontal dashed lines represent the results by the non-parallel non-distributed models with a single population and a single training data set. The line charts represent the results by the parallel distributed models with a different number of divisions.

With respect to the training data accuracy in Fig. 2, the increase in the number of divisions tended to the deterioration in the training data accuracy in general. However, we can observe the improvement of the training data accuracy by using our parallel distributed implementation of GAssist for Texture data (Fig. 2 (e)) and Penbased data (Fig. 2 (i)) and our fuzzy GBML for Satimage data (Fig. 2 (f)) and Twonorm data (Fig. 2 (h)). For some data sets, the deterioration by the divisions of the training data was not clear.

Fig. 3 shows the test data accuracy (%) for each data set. Comparing with Fig. 2, we can see almost the same effect on the number of divisions for each data set. An interesting observation is that there are more cases where the test data accuracy by the parallel distributed models was better than that by the non-parallel non-distributed models. The effect of the parallel distributed implementation on GAssist was almost the same as our fuzzy GBML for all data sets except for Twonorm data. When we applied our parallel distributed models to larger data sets (e.g., Penbased data and Magic data), the effect of the number of divisions was not clear. This may indicate that our parallel distributed models with a larger number of divisions with CPU cores can reduce the computation time for larger data sets (than Magic data) while maintaining the test data accuracy.

Fig. 4 shows the number of rules in the obtained classifiers for each data set. We can see that the number of rules in the classifiers obtained by the parallel distributed GAssist monotonically decreased as the number of divisions increased for all data sets except for Twonorm data. For data sets with more than two classes (i.e., Yeast data, Segment data, Page-blocks data, Twonorm data, Satimage data, Penbased data), the parallel distributed GAssist reduced the number of rules to almost the same or less number of classes. This means that the obtained classifiers cannot classify some of the classes. That is, the special treatment should be considered in the parallel distributed GAssist. The number of rules in the classifiers obtained by the parallel distributed fuzzy GBML was also less than that obtained by the non-parallel non-distributed fuzzy GBML for all data sets except for Satimage data.

Fig. 5 shows the computation time (min.) for each data set. For both parallel distributed models, we can observe the drastic decrease in the computation time according to the number of divisions. Although the synchronization of sub-populations may become serious bottleneck for a large number of sub-populations, the reduction in the number of rules by our parallel distributed implementation may also contribute to the reduction in the computation time.

Fig. 6 shows the relative average results over ten data sets. First, for each data, the result obtained by the parallel distributed model is normalized by that obtained by the non-parallel non-distributed model in percentage. Then the average value is calculated over ten data sets.

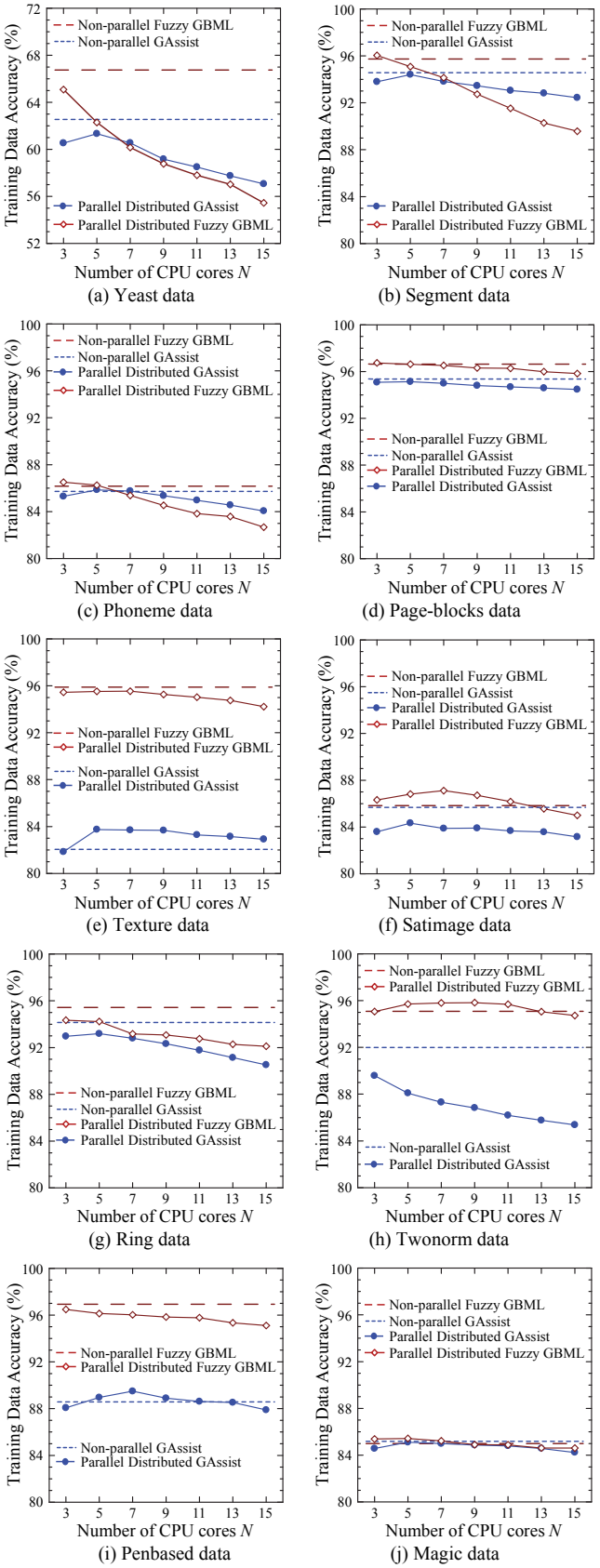

Fig. 2. Training data accuracy for both algorithms.

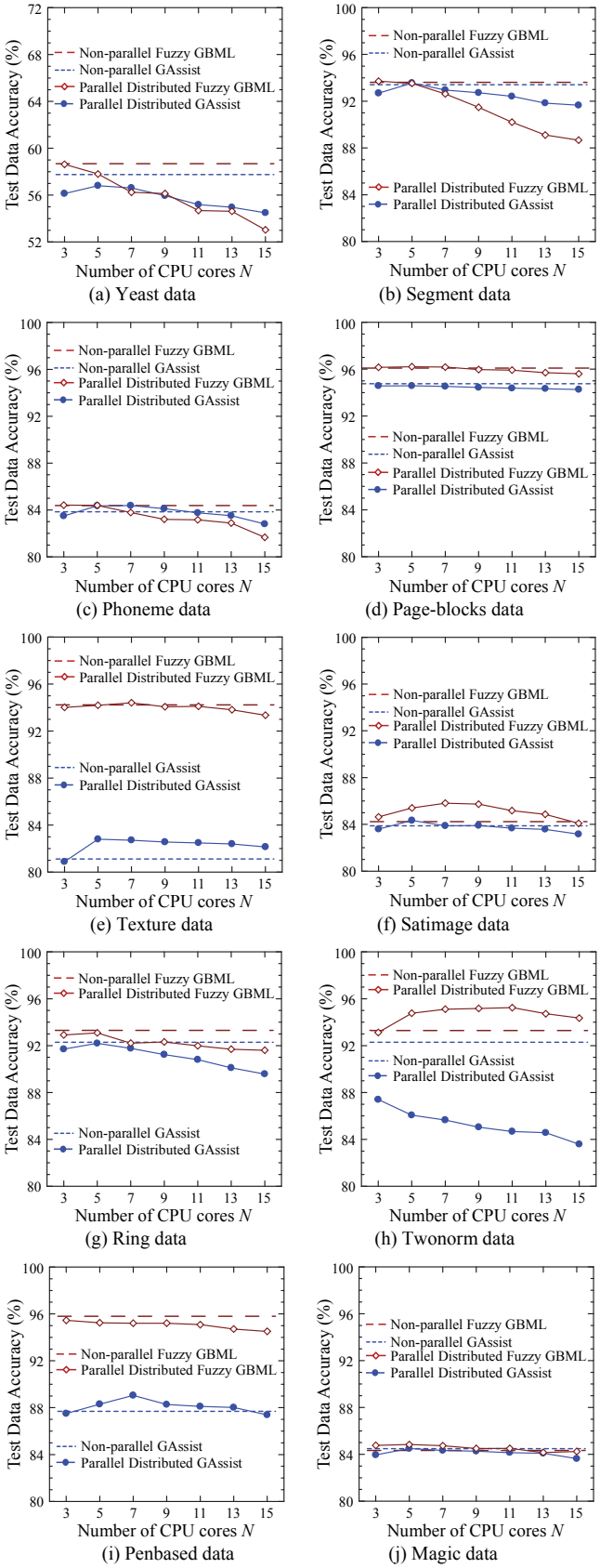

Fig. 3. Test data accuracy for both algorithms.

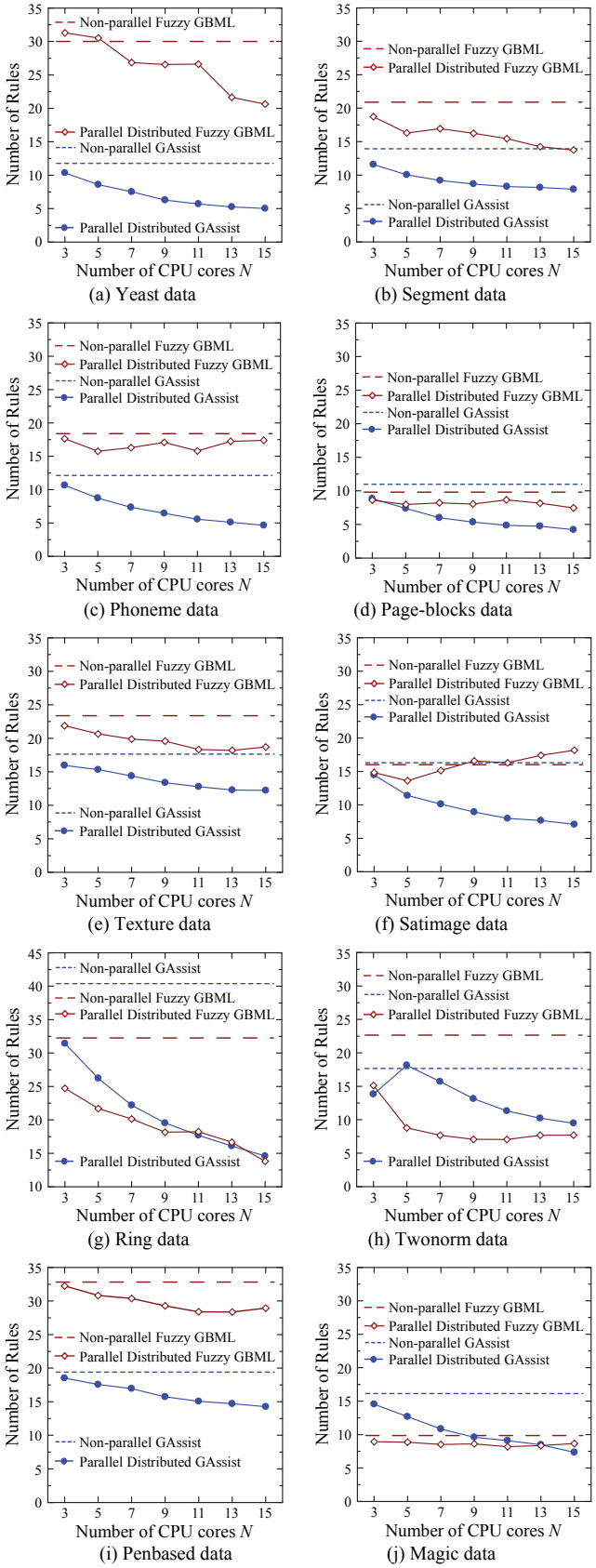

Fig. 4. Number of rules in both algorithms.

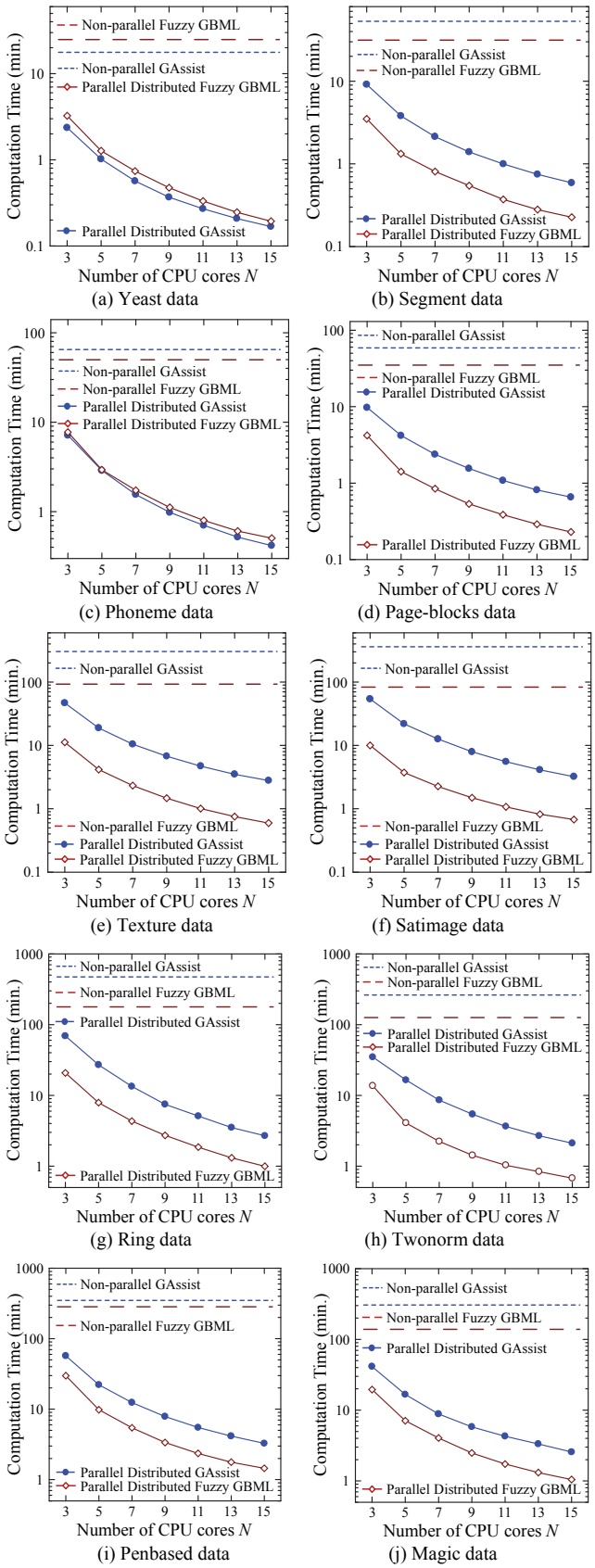

Fig. 5. Computation time for both algorithms.

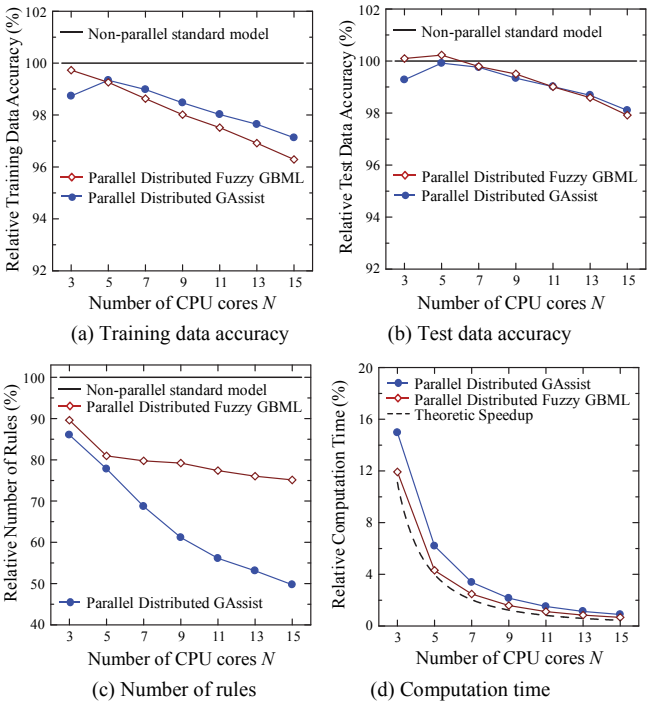

Fig. 6. Average results over ten data sets with the average non-parallel results set to 100% for comparison.

Fig. 6 (a) shows clear deteriorations on the training data accuracy by the parallel distributed models. That is, the number of divisions clearly affected the training data accuracy. On the contrary, the test data accuracy was improved by the parallel distributed fuzzy GBML, when the number of divisions was three or five as shown in Fig. 6 (b). The degree of the deterioration in the relative test data accuracy was smaller than that of the relative training data accuracy for both parallel distributed models.

From Fig. 6 (c), it is clear that the number of rules decreased as the number of divisions increased. With seven or more CPU cores, the effect was very different between the parallel distributed GAssist and the parallel distributed fuzzy GBML. The parallel distributed GAssist reduced the number of rules too much.

Fig. 6 (d) shows the relative computation time. As we mentioned earlier, the theoretic speedup that can be obtained by our parallel distributed implementation is N^2, where N is the number of used CPU cores (i.e., the number of sub-populations and the number of training data subsets). We can see that both parallel distributed models achieved a clear speedup not far from the theoretic speedup. One may feel that such a speedup is unrealistic because the increase of the computational overhead is unavoidable by the increase in the number of sub-populations. The obtained speedup in Fig. 6 (d) is explained by the decrease in the number of rules. As shown in Fig. 6 (c), the increase in the number of sub-population leads to the decrease in the number of rules in each individual in our computational experiments. The decrease in the number of rules in each individual leads to the decrease in the computation load for classifying each training pattern. As a result, the speedup for fitness evaluation of each individual is more than the decrease of the training data. For example, the following situation can happen: The computation time for fitness evaluation using 1/9 of the training data is shorter than 1/9 of that for fitness evaluation using all training data. The decrease in the number of rules explains why larger speedup than the theoretical speedup was obtained for some data sets in our computational experiments.

It should be mentioned that there is a large possibility that GAssist can be improved by changing its parameters and adjusting the basic framework of our parallel distributed implementation specialized for GAssist.

V. Conclusions

In this paper, we applied our parallel distributed implementation to the representative Pittsburgh-style GBML, GAssist for data mining from large data. We examined the effects of the divisions of the population and the training data set on the search ability, the classifier complexity, and the computation time for ten data sets. We also compared the behavior between the parallel distributed GAssist and the parallel distributed fuzzy GBML. From the experimental results, the performance of both parallel distributed models was totally dependent on the data sets. For some data sets, the deterioration in the test data accuracy by a large number of divisions was not observed. When we appropriately specified the number of divisions, the test data accuracy was better than that by the non-parallel non-distributed model. This means that our parallel distributed implementation can drastically reduce the computation time while maintaining or improving the generalization ability of the classifiers for unseen patterns. The speedup rate by the parallel distributed models was almost the same as the theoretical rate. The experimental results on the parallel distributed GAssist also indicate that a special treatment is necessary in order to avoid the serious reduction in the number of rules in the classifiers.

In big data era, many researchers are developing data mining techniques for big data [22]-[24]. There are many future research topics. An example is the use of a very large number of training data subsets. The training data can be divided into a larger number of subsets than the number of sub-populations. Then, those training data subsets can be rotated in the same way [25]. A dynamic subset assignment is also an interesting way [26]. Another example is to use high performance computing environments such as GPGPU and Hadoop ecosystems. In [27] and [28], the fitness calculation of GBML is parallelized by GPGPU.

An application of our parallel distributed implementation to evolutionary multiobjective data mining [29]-[31] is also left for a future research topic. Although we have already applied our parallel distributed implementation to a multiobjective fuzzy GBML algorithm [14]-[15], we still need further improvements to obtain very accurate classifiers.

Acknowledgment

The authors would like to thank Mr. Patrik Ivarsson for his help with computational experiments.

REFERENCES

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

[2] A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla, "Genetic-based machine learning systems are competitive for pattern recognition," *Evolutionary Intelligence*, vol. 1, pp. 209-232, July 2008.

[3] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, and F. Herrera, "Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study," *IEEE Trans. on Evolutionary Computation*, vol. 14, no. 6, pp. 913-941, December 2010.

[4] T. Kovacs, "Genetics-based machine learning," In G. Rozenberg, T. Bäck, and J. Kok (eds) *Handbook of Natural Computing: Theory, Experiments, and Applications*, pp. 937-986, Springer, 2012.

[5] Y. Nojima, H. Ishibuchi, and I. Kuwajima, "Parallel distributed genetic fuzzy rule selection," *Soft Computing*, vol. 13, no. 5, pp. 511-519, March 2009.

[6] H. Ishibuchi, S. Mihara, and Y. Nojima, "Parallel distributed hybrid fuzzy GBML models with rule set migration and training data rotation," *IEEE Trans. on Fuzzy Systems*, vol. 21, no. 2, pp. 355-368, 2013.

[7] O. Cordón, F. Herrera, F. Hoffman, and L. Magdalena, *Genetic fuzzy systems*, World Scientific, 2001.

[8] A. Fernández, V. López, M. J. del Jeses, and F. Herrera, "Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges," *Knowledge-Based Systems*, vol. 80, pp. 109-121, May 2015.

[9] Y. Nojima, P. Ivarsson, and H. Ishibuchi, "Application of parallel distributed implementation to GAssist and its sensitivity analysis on the number of sub-populations and training data subsets," *Proc. of 14th International Symposium on Advanced Intelligent Systems*, Daejeon, Korea, November 13-16, 2013.

[10] J. Bacardit, *Pittsburgh Genetic-based Machine Learning in the Data Mining Era: Representations, Generalization and Runtime*, PhD Thesis, Ramon Llull University, Barcelona, Spain, 2004.

[11] Y. Nojima, S. Mihara, and H. Ishibuchi, "Application of parallel distributed genetics-based machine learning to imbalanced data sets," *Proc. of 2012 IEEE International Conference on Fuzzy Systems*, pp. 928-933, Brisbane, Australia, June, 10-15, 2012.

[12] Y. Nojima, S. Mihara, and H. Ishibuchi, "Ensemble classifier design by parallel distributed implementation of genetic fuzzy rule selection for large data sets," *Proc. of 2010 IEEE Congress on Evolutionary Computation*, pp. 2113-2120, Barcelona, Spain, July 18-23, 2010.

[13] H. Ishibuchi and Y. Nojima, "Handling a training dataset as a black-box model for privacy preserving in fuzzy GBML algorithms," *Proc. of 2015 IEEE International Conference on Fuzzy Systems*, 8 pages, Istanbul, Turkey, August 2-5, 2015.

[14] Y. Nojima, Y. Takahashi, and H. Ishibuchi, "Application of parallel distributed implementation to multiobjective fuzzy genetics-based machine learning," *Proc. of 7th Asian Conference on Intelligent Information and Database Systems*, Part I, pp. 462-471, Bali, Indonesia, March 23-25, 2015.

[15] Y. Takahashi, Y. Nojima, and H. Ishibuchi, "Rotation effects of objective functions in parallel distributed multiobjective fuzzy genetics-based machine learning," *Proc. of 10th Asian Control Conference*, 6 pages, Kota Kinabalu, Malaysia, May 31-June 3, 2015.

[16] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, nos. 2-3, pp. 255-287, 2011.

[17] J. Bacardit and J. M. Garell, "Evolving multiple discretizations with adaptive intervals for a Pittsburgh rule-based learning classifier system," *Proc. of Genetic and Evolutionary Computation – GECCO 2003*, pp. 1818-1831, 2003.

[18] J. Bacardit, D. E. Goldberg, and M. V. Butz, "Improving the performance of a Pittsburgh learning classifier system using a default rule," *Learning Classifier Systems*, Springer, pp. 291-307, 2007.

[19] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol 14, no. 5, pp. 465-471, September 1978.

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

[21] J. G. Moreno-Torres, J. A. Saez, and F. Herrera, "Study on the impact of partition-induced dataset shift on k-fold cross-validation," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1304-1312, 2012.

[22] A. Fernandez, S. Río, V. López, A. Bawakid, M. J. del Jesus, J. M. Benítez, F. Herrera, "Big data with cloud computing: An insight on the computing environment, MapReduce and programming frameworks," *WIREs Data Mining and Knowledge Discovery,* vol. 4, no. 5, pp. 380-409, 2014.

[23] Y. Zhai, Y.-S. Ong, I. W. Tsang, "The emerging "big dimensionality"," *IEEE Computational Intelligence Magazine*, vol. 9, no. 3, pp. 14–26, 2014.

[24] Z.-H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, "Big data opportunities and challenges: Discussions from data analytics perspectives," *IEEE Computational Intelligence Magazine*, vol. 9, no. 4, pp. 62-74, 2014.

[25] Y. Nojima, H. Ishibuchi, and S. Mihara, "Use of very small training data subsets in parallel distributed genetic fuzzy rule selection," *Proc. of 4th International Workshop on Genetic and Evolutionary Fuzzy Systems*, pp. 27-32, Mieres, Spain, March 17-19, 2010.

[26] C. Gathercole and P. Ross, "Dynamic training subset selection for supervised learning in genetic programming," *Proc. of Parallel Problem Solving from Nature III* (Y. Davidor and H.-P. Schwefel and R. Manner, eds., LNCS 866), pp. 312-321, 1994.

[27] A. Cano, A. Zafra, and S. Ventura, "Parallel evaluation of Pittsburgh rule-based classifiers on GPUs," *Neurocomputing*, vol. 126, pp. 45-57, 2014.

[28] M. A. Franco, N. Krasnogor, and J. Bacardit, "Speeding up the evaluation of evolutionary learning systems using GPGPUs," *Proc. of 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 1039-1046, 2010.

[29] M. Fazzolari, R. Alcala, Y. Nojima, H. Ishibuchi, and F. Herrera, "A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions," *IEEE Trans. on Fuzzy Systems*, vol. 21, no. 1, pp. 45-65, February 2013.

[30] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C. A. Coello Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE Trans. on Evolutionary Computation,* vol. 18, no. 1, pp. 4-19, 2014.

[31] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C. A. C. Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part II," *IEEE Trans. on Evolutionary Computation*, vol. 18, no. 1, pp. 20-35, 2014.