A Micro-Extended Belief Rule-Based System for Big Data Multi-Class Classification Problems

Long-Hao Yang, Jun Liu, Member, IEEE, Ying-Ming Wang, and Luis Martínez, Member, IEEE

Abstract—Big data classification problems have drawn great attention from diverse fields and many classifiers have been developed. Among those classifiers, the extended belief rule-based system (EBRBS) has shown its potential in both big data and multi-class situations while time complexity and computing efficiency are two challenging issues to be handled in EBRBS. As such, three improvements of EBRBS are proposed firstly in the present paper to decrease the time complexity and computing efficiency of EBRBS for multi-class classification under the assumption of large amount of data, including the strategy to skip rule weight calculation, a simplified evidential reasoning algorithm, and the domain division-based rule reduction method. This turns out to be a micro version of the EBRBS, classed Micro-EBRBS. Moreover, one of commonly used cluster computing, named Apache Spark, is then applied to implement the parallel rule generation and inference schemes of the Micro-EBRBS for big data multi-class classification problems. The comparative analyses of experimental studies demonstrate that the Micro-EBRBS not only can obtain a desired accuracy, but also has the comparatively better time complexity and computing efficiency than some popular classifiers, especially for multi-class classification problems.

Index Terms—Apache Spark, Big data, Extended belief rule-based system (EBRBS), Multi-class,

I. INTRODUCTION

CLASSIFICATION problems are the common and fundamental ones involved in various real-world applications, such as intrusion detection [1], pattern recognition [19], image processing [2], and DNA sequence classification [5]. Classification becomes much more complex under big data and multi-class situations. The former always implies a high requirement of the computing efficiency of classifiers, while the need of a cluster computing for implementing the classifiers is getting popular. The latter implies many overlaps among the data of different classes, which requires the classifiers to have a powerful ability to differentiate the class boundaries.

Among many methodologies for multi-class classification problems, such as support vector machine (SVM) [34], ensemble learning [21], and others, rule-based systems (RBSs) are one kind of useful tools and have been a popular framework for designing classifiers in past decades. Basically, these RBSs can be divided into two categories depending on the construction methods of rule bases: RBSs based on the optimization model and iterative algorithm to determine the optimal values of parameters involved [33], [36], [39], or RBSs where rules are generated from sample data without the optimal model [8], [22], [28]. While handling big data multi-class classification problems, it is obvious that the RBSs without the optimal model are a better choice because of its high computing efficiency [26].

Some popular RBSs without the optimal model include fuzzy rule-based classification system (FRBCS) based on Chi et al. algorithm (Chi-FRBCS) [8] and the extended belief rule-based system (EBRBS) proposed by Liu et al. [22], both of them are originated from the work by Wang and Mendel [31]. It is worth noting that Chi-FRBCS has been applied to deal with big data classification problems recently (see details in Section II-B). Although there are some existing works about EBRBS [6], [11], [38], [39], it has not been developed yet as an efficient classifier to deal with big data classification problems. Hence, the goal of this study is to propose a novel big data EBRBS classifier and show its performance, in terms of accuracy and efficiency, in big data multi-class classification problems.

EBRBS has shown its potential to address multi-class classification problems because of its rules with belief structure and the evidential reasoning (ER) algorithm used in the inference scheme to collectively handle multi-class information for classification [20], [23], [24], [25], however, two challenges must be addressed to handle the big data situation:
(1) The time complexity of the EBRBS must be reduced to ensure the high computing efficiency under big data problems. The proposed solution in the present paper is to optimize the procedures of the EBRBS so that its time complexity can be reduced, which are mainly focused on three key procedures of the EBRBS: rule weight calculation, the ER algorithm, and the size of rule base. The corresponding improvements are proposed respectively, which forms a micro version of the EBRBS with much higher computing efficiency enables to deal with big data multi-class classification problems, called the Micro-EBRBS.

(2) The cluster computing must be applied to implement the parallel computing to improve the computing efficiency.

Chi-FRBCS classifiers usually involve the cluster computing to deal with big data classification problems [10], [26], [29]. Likewise, it is necessary to propose a solution of implementing the EBRBS classifier by using the cluster computing to handle the big data problem. Apache Spark [4] is an open-source framework that supports the processing of large datasets in a distributed computing environment and provides primitives for in-memory cluster computing and APIs in Scala, Java, and Python. As such, Apache Spark-based implementation of the parallel rule generation and inference scheme is proposed to improve the computing efficiency of Micro-EBRBS in big data situation.

To verify the effectiveness and computing efficiency of the Micro-EBRBS, three experiments based on 14 classification datasets, in which 4 of these datasets have relatively large number of data, are carried out to test the performance of the Micro-EBRBS. Two main aspects, namely accuracy and computing time, are used to compare the Micro-EBRBS with the EBRBS, the conventional FRBCS and machine-learning classifiers, and the big data FRBCS classifiers.

The remainder of this paper is organized as follows: Section II briefly reviews the background and challenges of the EBRBS for classification. Section III introduces the Micro-EBRBS for big data multi-classification problems. Section IV discusses experiments to demonstrate the performance of the Micro-EBRBS, and the paper is concluded in Section V.

II. BACKGROUND AND CHALLENGES

In this section, the EBRBS for classification problems is reviewed firstly to provide the basic knowledge of this study.

(1) The time complexity of the EBRBS must be reduced to ensure the high computing efficiency under big data problems.

The proposed solution in the present paper is to optimize the procedures of the EBRBS so that its time complexity can be reduced, which are mainly focused on three key procedures of the EBRBS: rule weight calculation, the ER algorithm, and the size of rule base. The corresponding improvements are proposed respectively, which forms a micro version of the EBRBS with much higher computing efficiency enables to deal with big data multi-class classification problems, called the Micro-EBRBS.

(2) The cluster computing must be applied to implement the parallel computing to improve the computing efficiency.

Chi-FRBCS classifiers usually involve the cluster computing to deal with big data classification problems [10], [26], [29]. Likewise, it is necessary to propose a solution of implementing the EBRBS classifier by using the cluster computing to handle the big data problem. Apache Spark [4] is an open-source framework that supports the processing of large datasets in a distributed computing environment and provides primitives for in-memory cluster computing and APIs in Scala, Java, and Python. As such, Apache Spark-based implementation of the parallel rule generation and inference scheme is proposed to improve the computing efficiency of Micro-EBRBS in big data situation.

To verify the effectiveness and computing efficiency of the Micro-EBRBS, three experiments based on 14 classification datasets, in which 4 of these datasets have relatively large number of data, are carried out to test the performance of the Micro-EBRBS. Two main aspects, namely accuracy and computing time, are used to compare the Micro-EBRBS with the EBRBS, the conventional FRBCS and machine-learning classifiers, and the big data FRBCS classifiers.

The remainder of this paper is organized as follows: Section II briefly reviews the background and challenges of the EBRBS for classification. Section III introduces the Micro-EBRBS for big data multi-classification problems. Section IV discusses experiments to demonstrate the performance of the Micro-EBRBS, and the paper is concluded in Section V.

II. BACKGROUND AND CHALLENGES

In this section, the EBRBS for classification problems is reviewed firstly to provide the basic knowledge of this study.

(1) The time complexity of the EBRBS must be reduced to ensure the high computing efficiency under big data problems.

The proposed solution in the present paper is to optimize the procedures of the EBRBS so that its time complexity can be reduced, which are mainly focused on three key procedures of the EBRBS: rule weight calculation, the ER algorithm, and the size of rule base. The corresponding improvements are proposed respectively, which forms a micro version of the EBRBS with much higher computing efficiency enables to deal with big data multi-class classification problems, called the Micro-EBRBS.

(2) The cluster computing must be applied to implement the parallel computing to improve the computing efficiency.

Chi-FRBCS classifiers usually involve the cluster computing to deal with big data classification problems [10], [26], [29]. Likewise, it is necessary to propose a solution of implementing the EBRBS classifier by using the cluster computing to handle the big data problem. Apache Spark [4] is an open-source framework that supports the processing of large datasets in a distributed computing environment and provides primitives for in-memory cluster computing and APIs in Scala, Java, and Python. As such, Apache Spark-based implementation of the parallel rule generation and inference scheme is proposed to improve the computing efficiency of Micro-EBRBS in big data situation.

To verify the effectiveness and computing efficiency of the Micro-EBRBS, three experiments based on 14 classification datasets, in which 4 of these datasets have relatively large number of data, are carried out to test the performance of the Micro-EBRBS. Two main aspects, namely accuracy and computing time, are used to compare the Micro-EBRBS with the EBRBS, the conventional FRBCS and machine-learning classifiers, and the big data FRBCS classifiers.

The remainder of this paper is organized as follows: Section II briefly reviews the background and challenges of the EBRBS for classification. Section III introduces the Micro-EBRBS for big data multi-classification problems. Section IV discusses experiments to demonstrate the performance of the Micro-EBRBS, and the paper is concluded in Section V.
It is clear from the comparison of Eqs. (1) and (2) that the fuzzy rule is a special case of the EBR. Moreover, the EBR is more flexible to express multi-class information under uncertainty and incompleteness, e.g., \((D_1, 0.6), (D_2, 0.4)\) means 60\% sure that the class is \(D_1\), 40\% sure that it is \(D_2\). \[
\sum_{j=1}^{c_i} \alpha_{i,j} = 0.9 \text{ means } 100\% - 90\% = 10\% \text{ ignorance in the } i\text{th antecedent attribute of the } k\text{th EBR.}
\]

Based on the above EBRB, the ER algorithm based inference scheme is applied to integrate EBRs to produce estimated classes, i.e., the integrated result is belief distribution \((D_1, 0.4008), (D_2, 0.4275), (D_3, 0.1718)\) and finally produces the class \(D_2\) as the output. A simple methodological framework of the EBRBS for classification is shown in Fig. 1 and the detailed step procedure can be referred to [22]. Additionally, the rule generation scheme is an indispensable part of the EBRBS and illustrated in Fig. 2.

As shown in Fig. 2, there are two kinds of parameters involved in the rule generation scheme of the EBRB. The first one is named as the basic parameters, including attribute weights, reference values and utility values of antecedent attributes, and classes of the consequent attribute. All these basic parameters are always determined by using expert knowledge. The second one is the generated parameters, including rule weights and belief distributions of antecedent and consequent attributes. All these generated parameters have to be initialized according to the sample input-output data and the basic parameters, including two steps: 1) generation of belief distributions for antecedent and consequent attributes using transformation techniques; and 2) calculation of rule weights using consistency measures. The detailed description of those steps can be found in Appendix B.

B. Chi-FRBCS in big data classification problems

FRBCSs are popular methods for classification problems with many versions developed so far, e.g., Chi-FRBCS [8], structural learning algorithm on vague environment (SLAVE) [15], fuzzy hybrid genetic-based machine learning algorithm (FH-GBML) [18], fuzzy unordered rule induction algorithm (FURIA) [16], and fuzzy association rule-based classification method for high-dimensional problems (FARC-HD) [3]. However, considering the limitations of standard fuzzy rule base learning approaches for large number of samples, the Chi-FRBCS was recognized by many researchers as the suitable FRBCS to handle big data classification problems [26], [29].

In the last few years, several big data classifiers based on the Chi-FRBCS have been proposed and made use of the Apache Hadoop to deploy the distributed system. For example, Lopez et al. [26] proposed the first FRBCS capable of addressing big data and imbalance datasets, called Chi-FRBCS-BigDataCS, which utilized the Apache Hadoop to distribute the computational operations of Chi-FRBCS and also included cost-sensitive learning techniques to address imbalanced big data. After that, Rio et al. [29] developed a more general big data classifier based on the Chi-FRBCS and Apache Hadoop, called Chi-FRBCS-BigData, which includes two versions: Chi-FRBCS-BigData-max and Chi-FRBCS-BigData-Ave. Both show the ability to deal with big data problems providing competitive results and reasonable computing efficiencies. Fernandez et al. [12] studied the relationship between the granularity and data scattering for Chi-FRBCS in big data classification problems and made use of Chi-FRBCS-BigData to accomplish their analysis. Later on, Fernandez et al. [13] carried out many experimental studies regarding the use of Chi-FRBCS-BigData to analyze the differences in performance with respect to the lack of data for the learning stage, how rules are distributed among Maps, and their influences on the classification stage. Elkano et al. [10] proposed a global version of Chi-FRBCS-BigDataCS in order to address the problem of previous big data classifiers that it would become less accurate when more computing nodes are added into the cluster.

The above literatures review shows many potential applications of Chi-FRBCS in big data classification problems. However, they were mainly focused on two-class classification problems. Although the decomposition strategies [27], such as One-Versus-One (OVO) and One-Versus-All (OVA) schemes, can be used to decompose a multi-class problem into multiple two-class problems, it is unavoidable to cause the increase of the time complexity. Considering that the EBRBS has an effective rule representation scheme better than the one in Chi-FRBCS owning to embedding the belief structure into both
antecedent and consequent attributes, the present work aims to propose a novel classifier based on the EBRBS for big data problems in comparison with Chi-FRBCS in terms of accuracy and efficiency.

C. Challenges of EBRBS for big data problems

Due to the importance of computing efficiency in big data problems, in the context of the EBRBS, the time complexity of the rule generation and inference scheme are analyzed in this subsection. For the discussion purposes, Figs. 3 and 4 provide the pseudocode and the time complexity of those two schemes involved in the EBRBS, respectively.

From Fig. 3, the time complexity of generating belief distributions shown in the 1st line to the 8th line is \(O(T \times (\Sigma_{i=1,..., M} J_i + N))\) and the time complexity of calculating rule weights shown in the 9th line to the 14th line is \(O(T^2 \times (\Sigma_{i=1,..., M} J_i + N))\), where \(T\) is the number of sample data, \(M\) is the number of antecedent attributes, \(J_i\) is the number of reference values in the \(i\)th antecedent attribute, and \(N\) is the number of classes. Clearly, for the rule generation scheme of the EBRBS, the calculation of rule weights requires the most computing time and sometimes it would be unacceptable while there are large amounts of sample data. For example, the dataset Poker has 1,025,010 samples (i.e., \(T=1,025,010\)), 10 attributes (i.e., \(M=10\)), and 10 classes (i.e., \(N=10\)). If the number of reference values is assumed to be 3 for each attribute (i.e., \(J_i=3; i=1,..., M\)) and the computing time of each operation is \(10^6\) second, then the total computing time of generating rule weights is 11,673.8 hours.

From Fig. 4, the time complexity of the inference scheme is \(O(S \times L \times (\Sigma_{i=1,..., M} J_i + N))\), where \(S\) is the number of test data, \(L\) is the number of EBRs in the EBRB. Considering that one EBR is directly transformed from one sample data [22], so \(T=L\). Therefore the time complexity of the inference scheme can be expressed as \(O(S \times T \times (\Sigma_{i=1,..., M} J_i + N))\). Obviously, the computing time would also be unacceptable while there is large amount of sample data involved in the rule generation scheme, i.e., while the 10-fold cross validation is utilized to test the dataset Poker, the number of sample data and test data is therefore 922,509 and 102,501, respectively. Finally, the computing time of the inference scheme is 1,050.6 hours.

The above discussions clearly show that although the EBRBS is the RBS without the optimal model, the time-consuming process found in the calculation of rule weights and the inference scheme would be serious challenges while the EBRBS is applied to address the classification problem with a large amount of data. Therefore, the present work aims to address these challenges.

III. A NOVEL EBRBS FOR BIG DATA MULTI-CLASS CLASSIFICATION PROBLEMS

According to the challenges pointed out in Section II-C, the possible approaches are investigated to reduce the time complexity of the rule generation and the inference schemes, followed by a new rule reduction method to downsize the EBRB. Based on these achievements, a novel EBRBS and its Apache Spark-based implementation are developed to deal with big data multi-class classification problems.

A. Analysis of rule weight calculation and the ER algorithm involved in EBRBS

In order to reduce the time complexity of the EBRBS, the properties of the rule weight calculation and the ER algorithm are investigated as follows.

**Theorem 1.** The rule weight of each EBR will approximate to 1 while using large number of sample data to generate EBRs.

**Proof.** Suppose there are \(L\) EBRs. Based on the calculation of rule weights in Eq. (B6) in Appendix B, we can get the first order partial derivative of the rule weight \(\theta_k\) (\(k=1,..., L\)) with respect to the inconsistency degree \(\text{Incons}(R_k)\) as follows:

\[
\frac{\partial \theta_k}{\partial \text{Incons}(R_k)} = \frac{- \sum_{j=1}^{L} \text{Incons}(R_j) - \text{Incons}(R_k)}{[\text{Incons}(R_k) + \sum_{j=1, j \neq k}^{L} \text{Incons}(R_j)]^2} \leq 0
\]

(3)

It follows that \(\theta_k\) decreases while its \(\text{Incons}(R_k)\) increases. In
other words, when Incons($R_k$) is equal to 1, $R_i$ has a minimum rule weight. This makes sense that if this rule causes the contradiction, then it will be useless.

Apparently, when the number of rules is increasing, the inconsistency degree of each rule is also increasing. It follows that when $L$ is approaching to $\infty$, the inconsistency degree of each rule is approaching to 1. Without loss of generality, we consider the rule weight of $R_k$ as follows:

$$\theta_k = \lim_{L \to \infty} \left[ 1 - \frac{\text{Incons}(R_k)}{\sum_{i=1}^{L} \text{Incons}(R_i)} \right] = 1 - \lim_{L \to \infty} \frac{1}{L} = 1$$

(4)

Now that $L$ is equal to the number of sample data, this concludes the proof.

**Example 1.** Suppose there are $L$ EBRs and two different EBRs, namely the $k$th rule and the $i$th rule ($k, i \in \{1, \ldots, L\}; i \neq k$) with their inconsistency degree Incons($R_k$) and Incons($R_i$), respectively. The relationship between rule weights and inconsistency degrees can be shown in Fig. 5: while Incons($i$) is fixed, $\theta_i$ increases with the decrease of Incons($k$). Moreover, while Incons($k$)=1, the $k$th rule has a minimum rule weight.

Secondly, assume that Incons($R_k$)=1 ($k \in \{1, \ldots, L\}$) and the inconsistency degree of other rules are set by using random values. Then the relationship between $\theta_i$ and $L$ (from 1 to 1000) is shown in Fig. 6, in which each line (twenty lines in total) denotes an independent experiment of using random values to determine $\theta_i$ ($i = 1, \ldots, L; i \neq k$). From Fig. 6, Incons($k$)=1 leads to the minimum rule weight for the $k$th rule. However, with the increase of $L$, it turned out that $\theta_i$ approximates to 1.

**Remark 2:** From Theorem 1, it is unnecessary to calculate each rule weight because its value will approximate to 1 while a large number of data are used to generate EBRs. As a result, the time complexity of the rule generation scheme is reduced from $O(T^2 \times (\sum_{i=1}^{L} \sum_{j=1}^{M_j} + N)$ to $O(T \times (\sum_{i=1}^{L} \sum_{j=1}^{M_j} + N)$.

**Theorem 2.** For classification problems, the following ER algorithm for classification (denoted as ER-C) is the same as the analytical ER algorithm shown in Eq. (A1) in Appendix A.

$$\beta_n^{ER-C} = \prod_{i=1}^{L} \left( w_i \beta_{n,k} + 1 - w_k \sum_{i=1}^{N} \beta_{i,k} \right)$$

(5)

**Proof.** Suppose a classification problem has $N$ classes and the $n$th ($n=1, \ldots, N$) class is denoted as $D_n$. Assume the estimated class of the EBRBS is the $n$th class for the test input data $x$. Hence, according to Eq. (A3) in Appendix A, we have

$$\beta_n > \beta_i; i = 1, \ldots, N; i \neq n$$

(6)

Based on the analytical ER algorithm in Eq. (A1), we assume

$$\chi^1 = \prod_{i=1}^{L} \left( 1 - w_i \sum_{i=1}^{N} \beta_{i,k} \right)$$

(7)

$$\chi^2 = \prod_{i=1}^{L} \left( 1 - w_i \right)$$

(8)

Hence, we can obtain

$$\beta_n = \frac{\beta_n^{ER-C} - \chi^1}{\sum_{i=1}^{N} \beta_i^{ER-C} - (N - 1) \chi^1 - \chi^2}$$

(9)

$$\beta_i^{ER-C} = \frac{\beta_i^{ER-C} - \chi^1}{\sum_{i=1}^{N} \beta_i^{ER-C} - (N - 1) \chi^1 - \chi^2}$$

$$\Leftrightarrow \beta_n^{ER-C} > \beta_i^{ER-C}; i = 1, \ldots, N; i \neq n$$

It follows that the ER-C algorithm is the same as the analytical ER algorithm in EBRBS for classification.

**Example 2.** Suppose there are two EBRs and their belief distributions of the consequent attribute are shown as follows:

$$R_1 = \{(D_n, \beta_{n,k}); n = 1, 2\} \text{ with } \sum_{i=1}^{N} \beta_{n,k} = 1$$

(10)

$$R_i = \{(D_n, \beta_{n,i}); n = 1, 2\} \text{ with } \sum_{i=1}^{N} \beta_{n,i} = 1$$

(11)

The activation weight of the rule $R_i$ and $R_i$ are assumed to be
To illustrate the possible values of \( d_{ER} \) and \( d_{ER-C} \), we consider nine illustrative cases under the assumptions that \( \beta_{1,1} = \beta_{1,} \leq \beta_{1,k} \) and \( \beta_{1,1} > \beta_{1,k} \) while the value of \( w_k \) lies between 0 and 1. Without loss of generality, the value of both \( \beta_{1,1} \) and \( \beta_{1,k} \) is assumed as 0.3, 0.5, and 0.7, respectively. Hence, the curves of \( d_{ER} \) and \( d_{ER-C} \) are shown in Fig. 7. Fig. 7 shows that \( d_{ER} \) and \( d_{ER-C} \) have same negative and positive symbols for nine combinations based on different belief degrees and activation weights. Hence, the ER-C algorithm can produce the same estimated class as the ER algorithm.

**Remark 3.** From Theorem 2, the ER-C algorithm can be used to replace the ER algorithm in the inference scheme of the EBRBS while facing classification problems. Additionally, the existing studies of using the ER algorithm as inference engine for classification problems, such as [6] and [7], can also use the ER-C algorithm to replace the inference engine.

**Remark 4.** The advantages of the ER-C algorithm over the ER algorithm can be summarized as follows:

1. The ER-C algorithm has a much clean and simple formula than the ER algorithm because it derives from the core part of the ER algorithm.
2. The ER-C algorithm is more efficient than the ER algorithm according to their time complexity, in which the ER-C algorithm is \( O(L \times N) \) and the ER algorithm is \( O(L \times N^2) \).
3. In term of independence, the calculation of the integrated belief degree for all classes is independent of each other in the ER-C algorithm so that it is possible to have more solutions of parallelization for the EBRBS.

**B. Domain division-based rule reduction method for EBRBS**

In order to further reduce the time complexity of the EBRBS, in this subsection, a domain division-based rule reduction method is proposed for the rule generation scheme. Firstly, the main idea, which follows the similar way to the fuzzy partition [17] and the Wang-Mendel model [31], of the proposed rule reduction method are given based on the following definitions.

**Definition 1 (Division point).** The division point is the intersection between transform functions used to calculate the belief degree to which the input data belongs to the reference value of antecedent attributes. For convenience, \( P(A_{i,j}, A_{j+1}) \) \((i=1, \ldots, M; j=1, \ldots, J-1)\) is defined to express the division point between the \( A_{i,j} \) and \( A_{i,j+1} \) in the \( i \)th antecedent attribute.

**Definition 2 (Division domain).** The division domain is the local input space constructed by the two adjacent division points of each antecedent attribute. For convenience, \( D(A_{i,j}, \ldots, A_{M,j}) \) \((j=1, \ldots, J; i=1, \ldots, M)\) is defined to express the division domain constructed by the division point regarding the reference values \( A_{i,n}, \ldots, A_{M,j} \).

**Example 3.** Suppose that an EBRB includes one antecedent attribute \( U_i \) with three reference values \( A_{i,j} \) and their utility values \( u(A_{i,j}) \) \((i=1, 2, 3)\). Without loss of generality, the order of those utility values is \( u(A_{i,1}) < u(A_{i,2}) < u(A_{i,3}) \). \( x_1 \) is assumed to be the input variable of the attribute \( U_i \) and the utility function used to generate belief degrees of the reference value \( A_{i,j} \) \((i=1, 2, 3)\), is assumed to be piecewise linear [35], namely \( \alpha_{i,j}(x_1) \), shown in Fig. 8. From Fig. 8, there are two division points, namely \( P(A_{i,1}, A_{i,2}) \) and \( P(A_{i,2}, A_{i,3}) \), and three division domains, namely \( D(A_{i,1}), D(A_{i,2}), \) and \( D(A_{i,3}) \).

**Remark 5.** Based on Definitions 1 and 2 together with Example 3, when an input data falls into a division domain, the relatively maximal belief degree can be generated from the input data for all reference values belonging to the division domain. In other words, each division domain can be regarded as the clustering center of the input data which most likely belong to the reference value of division domain in the form of belief distribution.

**Definition 3 (Rule clustering strategy).** The rule clustering strategy is the map relationship between EBRs and division domains based on the relatively maximal belief degree in each antecedent attribute so that it can be defined as

\[ R_k \rightarrow D(A_{i,j}, \ldots, A_{M,j}) \]  \( (14) \)

where \( j_i = \arg \max_{j=1,\ldots,J} \{ \alpha_{i,j}^k \} \) \((k=1, \ldots, L; i=1, \ldots, M)\); \( \alpha_{i,j}^k \) denotes the belief degree of the reference value \( A_{i,j} \) in the \( k \)th EBR \((i=1, \ldots, J; j=1, \ldots, M)\); \( M \) is the number of antecedent attributes, \( L \) is the number of EBRs, and \( J_i \) is the number of reference values used for the \( i \)th antecedent attribute.

**Definition 4 (Rule reduction strategy).** The rule reduction strategy is the combination strategy for the EBRs which are assigned to the same division domain so that it can be defined as

\[ \bar{\alpha}_{i,j}^k = \frac{\sum_{l=1}^{L_i} \alpha_{i,j}^k}{L_i}, \overline{\beta_{n,k}} = \frac{\sum_{i=1}^{N} \beta_{n,k}}{L} \quad \text{while} \quad L_i > 0 \]  \( (15) \)

where \( L_i \) is the number of the rules gathered at the \( i \)th \((i=1, \ldots, M)\) division domain and \( \overline{\beta_{n,k}} \) \((n=1, \ldots, N; k=1, \ldots, L)\) denote the belief degrees of the reference value \( A_{i,j} \) and \( D_n \) in the \( k \)th rule, respectively.

**Remark 6.** As shown in Eq. (15), different denominators are
used to calculate the belief degree of antecedent and consequent attributes. The main reason is that: (1) the belief degree of antecedent attribute reflects the space relationship between different division domains in a sense, so the belief degree is calculated by using $L_i$; (2) the belief degree of consequent attribute reflects how many rules have the same class, thus the belief degree is calculated by using $L$.

**Example 4.** Suppose that an EBRB includes two antecedent attributes $U_1$ and $U_2$ with three reference values $A_{ij}$ and their utility values $u(A_{ij})$ ($i=1, 2$; $j=1, 2, 3$). Without loss of generality, the order of those utility values is $u(A_{i1})<u(A_{i2})<u(A_{i3})$ in the $i$th antecedent attribute. In addition, the EBRB has three EBRs $R_1$, $R_2$, and $R_3$, respectively, as shown in Fig. 9.

From Fig. 9, four division points and nine division domains can be constructed based on Definitions 1 and 2. After that, according to Definition 3, all three EBRs are assigned to the division domain $D(A_{12}, A_{22})$. Afterwards, based on Definition 4, the EBRs gathered in $D(A_{12}, A_{22})$ are all combined as a new rule, denoted as $\overline{R}$, which is obtained via Eq. (15), e.g., $\overline{c}_{11}=(0.1+0.0+0.2)/3=0.1$, $\overline{c}_{13}=(0.9+0.7+0.8)/3=0.8$, and $\overline{c}_{13}=(0.0+0.3+0.0)/3=0.1$ in the antecedent attribute $U_1$, and $\overline{c}_{21}=(1+1+0)/3=0.6667$ and $\overline{c}_{21}=(0+0+1)/3=0.3333$ in the attribute $D$.

**Remark 7.** Based on Definitions 3 and 4 together with Example 4, when one more EBR is assigned to the same division domain, a new EBR would be generated, in which the information of the new EBR is considered complete because it is generated by using the belief distribution of the antecedent and consequent attribute of all original rules.

Based on the above Definitions 1 to 4 and Examples 3 to 4, the steps of the domain division-based rule reduction method are described as follows:

**Step 1:** To generate division points for each antecedent attribute by using the transform functions. Suppose that there are $M$ antecedent attributes with $J$ reference values for the $i$th ($i=1, \ldots, M$) antecedent attribute. Based on Definition 1, $J_i-1$ division points, namely $\{P(A_{ij}, A_{ij+1}); j=1, \ldots, J_i-1\}$, are generated for the $i$th antecedent attribute.

**Step 2:** To generate division domains for the EBRBS by using the division points. Based on Definition 2, $\prod_{i=1}^{M} J_i$ division domains, namely $\{D(A_{1j}, \ldots, A_{Mj}) \}: j=1, \ldots, J_i; i=1, \ldots, M\}$, are generated for the EBRBS, in which the division domain is the clustering center according to Remark 5.

**Step 3:** To assign all EBRs to the division domains based on the rule clustering strategy. Suppose that there are $T$ EBRs transformed from $T$ sample data. Based on Definition 3, these $T$ EBRs are all assigned to the $\prod_{i=1}^{M} J_i$ division domains.

**Step 4:** To generate new EBRs from the EBRs gathered in each division domain based on the rule reduction strategy. Suppose that there are $L$ division domains which include at least one EBR, based on Definition 4, all EBRs in each division domain are used to generate $L$ new EBR, respectively.

**Remark 8.** After utilizing the domain division-based rule reduction method to downsize the EBRB, $L$ new EBRs shown in Step 4 can construct a reduced EBRB, in which the number of rules regarding the reduced EBR should be no more than both the number of sample data $T$ and the number of division domains $\prod_{i=1}^{M} J_i$.

C. **Micro-EBRBS: EBRBS with rule reduction and ER-C algorithm but without rule weight calculation**

Based on the above analysis and new improvements, a novel EBRBS, called **Micro-EBRBS** which has a simplified rule generation and inference schemes and a downsized EBR comparing to EBRBS, is developed and its methodological framework is shown in Fig. 10.

It is clear from Fig. 10 that, comparing to the EBRBS shown in Figs. 1 and 2, the Micro-EBRBS includes the process of rule reduction but exclude the process of rule weight calculation. Additionally, the ER-C algorithm is used to replace the ER algorithm in the process of activated rule integration. More specifically, the rule generation and the inference scheme of the Micro-EBRBS are described as follows:

For the rule generation scheme of the Micro-EBRBS, it consists of the following two steps:

**Step 1:** Generation of belief distributions using the transformation technique.
After determining the basic parameters based on expert knowledge, including attribute weights, utility values of the reference values used for antecedent and consequent attributes, the belief distributions of the antecedent and consequent attributes can be generated. Suppose that \( \{u(A_i); j=1,\ldots, J_i\} \) is a set of given utility values used for the \( i \)th \((i=1,\ldots, M)\) antecedent attribute, and \( x_{kj} \) is the \( k \)th \((k=1,\ldots, L)\) sample input data of the \( i \)th antecedent attribute. Thus, the belief distribution of the \( i \)th antecedent attribute generated using the utility-based equivalence transformation technique [35] is:

\[
S(x_{ki}) = \{(A_{ij}, \alpha^*_{ij}); j=1,\ldots, J_i\} \tag{16}
\]

where

\[
\alpha^*_{ij} = \frac{u(A_{ij}) - x_{kj}}{u(A_{ij}) - u(A_{ij+1})} \tag{17}
\]

\[
\alpha^*_{ij+1} = 1 - \alpha^*_{ij}, \text{ if } u(A_{ij}) \leq x_{kj} \leq u(A_{ij+1})
\]

\[
\alpha^*_{ij} = 0, \text{ for } t = 1,\ldots, J_i \text{ and } t \neq j, j+1 \tag{18}
\]

where \( \alpha^*_{ij} \) is the belief degree of \( A_{ij} \) in the \( k \)th EBR obtained from the sample input data \( x_{kj} \).

Next, when the \( k \)th sample output data is \( y_k \) and the given utility values attached to the consequent attribute \( D \) are \( \{u(D_n); n=1,\ldots, N\} \), the belief distribution of the consequent attribute is:

\[
S(y_k) = \{ (D_x, \beta^k_x); n=1,\ldots, N \} \tag{19}
\]

Finally, all belief distributions shown in Eqs (16) and (19) together with attribute weights determined by experts are used to construct an EBRB.

**Step 2**: Rule reduction based on domain divisions.

For the EBRB generated by Step 1, the domain-division-based rule reduction method shown in Section III-B is then used to reduce the number of rules. Suppose there are \( M \) antecedent attributes with \( J_i \) reference values \( A_{ij} \) \((i=1,\ldots, M)\) and one consequent attribute \( D \) with \( N \) classes \( D_n \) \((n=1,\ldots, N)\), the \( k \)th \((k=1,\ldots, \bar{L})\) rule of the reduced EBRB can be written as

\[
R_k: IF \ U_i \ is \ \{(A_{ij}, \overline{\alpha}^i_{ij}); j=1,\ldots, J_i\} \ and \ \cdots \ and \ U_M \ is \ \{(A_{im}, \overline{\alpha}^m_{im}); j=1,\ldots, J_m\}, \ THEN \ D \ is \ \{(D_x, \overline{\beta}_x); n=1,\ldots, N; \text{ with } \theta_k = 1 \ and \ \{\delta_1,\ldots, \delta_M\} \tag{20}
\]

where \( \overline{\alpha}^i_{ij} \) and \( \overline{\beta}_x \) are the integrated belief degree of the \( A_{ij} \) and the class \( D_n \) using the rule reduction strategy.

**Remark 9.** As shown in the proposed rule reduction method, the most complex step is to generate \( \overline{\alpha}^i_{ij} \) \((k=1,\ldots, \bar{L} ; i=1,\ldots, M; j=1,\ldots, J_i)\) and \( \overline{\beta}_x \) \((n=1,\ldots, N)\) by using \( L \) rules of EBRB. Furthermore, Section II-C shows that the time complexity of generating belief distributions is \( O(L \times \sum_{i=1}^{M} J_i + N) \). Hence, the time complexity for the rule generation scheme of the Micro-EBRBS is \( O(L \times \bar{L} \times \sum_{i=1}^{M} J_i + N) \).

For the inference scheme of the Micro-EBRBS, it consists of the following two steps:

**Step 1**: Calculation of activation weights using the distance measure.

While a test input data is provided for the Micro-EBRBS, the activation weights can be calculated for each EBR of the reduced EBRB. Suppose that \( x=(x_1, \ldots, x_M) \) is a test input data, each input \( x_i \) \((i=1,\ldots, M)\) will be firstly transformed into a belief distribution of the reference values of the \( i \)th antecedent attribute using Eqs. (17) and (18).

\[
S(x_i) = \{(A_{ij}, \alpha^i_{ij}); j=1,\ldots, J_i\} \tag{21}
\]

Next, the individual matching degree of the \( i \)th antecedent attribute in the \( k \)th rule, denoted as \( S^k_i(x, U) \), is calculated by using the Euclidean distance:

\[
S^k_i(x, U_i) = 1 - d^k_i(x, U_i) = 1 - \min\left\{ \left[ \sqrt{\sum_{j=1}^{M} (\alpha^i_{ij} - \alpha^k_{ij})^2} \right]^2 \right\} \tag{22}
\]

where \( d^k_i(x, U_i) \) is the distance measurement.

Finally, the activation weight of the \( k \)th EBR, denoted as \( w_k \), is calculated by

\[
w_k = \frac{\theta_k \prod_{i=1}^{M} \left(S^k_i(x, U_i) \right)^5}{\sum_{k=1}^{\bar{L}} \theta_k \prod_{i=1}^{M} \left(S^k_i(x, U_i) \right)^5} \delta_k = \max_{i=1,\ldots, M} \{\delta_i \} \tag{23}
\]

where \( \delta_k \) is the weight of the \( k \)th rule; \( \delta_k \) is the weight of the \( i \)th antecedent attribute.

**Step 2**: Integration of activated rules using the ER-C algorithm.

After performing Step 1, all activated rules can be integrated using the ER-C algorithm shown in Eq. (5) and the integrated belief distribution of the test input data \( x \) can be represented as follows:

\[
f(x) = \{(D_x, \beta^{ER-C}_x); n=1,\ldots, N\} \tag{24}
\]

Afterwards, the estimated class for the test input data \( x \) can be obtained as follows:

\[
f(x) = D_n, n = \arg \max_{i=1,\ldots, N} \beta^{ER-C}_x \tag{25}
\]

**Remark 10.** Considering that the inference scheme of the Micro-EBRBS is based on the reduced EBRB, which only has \( \bar{L} \) rules, the time complexity of the inference scheme
regarding the Micro-EBRBS is $O(\bar{L} \times (\Sigma_{i=1,...,M} J_i + N))$ for classifying each test data.

D. Apache Spark-based implementation of Micro-EBRBS for big data multi-class classification

In order to further improve the computing efficiency of the Micro-EBRBS, the Apache Spark is used to achieve the parallel rule generation and inference schemes. As an in-memory big data platform, the Apache Spark has been proven that it supports a much wider range of functionality than the Apache Hadoop [9]. The fundamental data structure of the Apache Spark is the resilient distributed dataset (RDD), which represents a collection of distributed items that can be manipulated across many computing nodes concurrently. Hence, the RDD allows the data cache to be stored in memory and perform computations for the same data directly from memory. After the RDD is constructed, the program can perform the following two operations:

1) Transformations: this kind of operation is to create a new RDD from existing RDD and the concrete function includes `map` (which is to return a new RDD formed by passing each element of the source through a function), `mapToPair` (which is to return a new RDD of key-value pairs by using a function), `reduceByKey` (which is to return a new RDD of key-value pairs where the values for each key are aggregated using the given reduce function), and so on. The detailed description of those functions can be found in [4].

2) Actions: this kind of operation is to return the final results of RDD computations and the concrete function includes `reduce` (which is to aggregate the elements of the RDD using a function), `collect` (which is to return all the elements of the RDD as an array at the driver program), and others. The detailed description of those functions can be found in [4].

Based on the functions of transformation and action, the pseudocodes are provided to illustrate the Apache Spark-based implementation of the parallel rule generation and inference schemes of Micro-EBRBS below respectively.

IV. EXPERIMENTS

The performance of the Micro-EBRBS is empirically assessed through three different experiments with 14 classification datasets from the well-known UCI repository of machine learning databases [30]. The EBRBS, the conventional FRBCS and machine-learning classifiers, and the big data FRBCS classifiers are used to compare in terms of the accuracy and computing efficiency, respectively.

A. Datasets and experiment conditions

Fourteen classification datasets obtained from UCI are used to evaluate the performance of the Micro-EBRBS. The main
Pseudocode of parallel rule generation scheme of Micro-EBRBS

Input: SampleDataSet denotes the set of sample input-output data, each sample input-output data of SampleDataSet is denoted as sampleData, rule denotes the EBR shown in Eq. (20), tuple1, tuple2, and tuple3 denote the 2-tuple composing of rule and its division domain.

Output: A set of extended belief rules EBRSet
01 EBRSet = new JavaSparkContext().parallelize( SampleDataSet ).mapToPair( sampleData -> { 
02 Generate rule from sampleData by using Steps 1 to 2 shown in the rule generation scheme of the Micro-EBRBS;
03 Generate divisionDomain for rule by using Steps 1 to 3 shown in the domain division-based rule reduction method;
04 Return new Tuple2<divisionDomain, rule>
05 }).reduceByKey( (tuple1, tuple2) -> {
06 Generate tuple3 by using Step 4 shown in the domain division-based rule reduction method;
07 Return tuple3;
08 }).map( tuple3 -> {
09 Obtain rule from tuple3;
10 Return rule;
11 }).collect();

Pseudocode of parallel inference scheme of Micro-EBRBS

Input: TestDataSet denotes the set of test input data, each test input data of TestDataSet is denoted as testData, class1 denotes the estimated class of the Micro-EBRBS, and class2 denotes the actual class of test input data, a and b denote the integer variable.

Output: The total number of test input data correctly classified by the Micro-EBRBS totalCorrect.
01 totalCorrect = new JavaSparkContext().parallelize( TestDataSet )
02 .map(testData -> {
03 Generate class1 for testData by using Steps 1 to 2 shown in the inference scheme of the Micro-EBRBS;
04 Obtain class2 from testData;
05 Return class1==class2?1:0;
06 }).reduce( (a, b) -> a+b );

<table>
<thead>
<tr>
<th>No.</th>
<th>Dataset</th>
<th>No. of data</th>
<th>No. of attributes</th>
<th>No. of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diabetes</td>
<td>393</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Cancer</td>
<td>569</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Transfusion</td>
<td>748</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Banknote</td>
<td>1,372</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Magic</td>
<td>19,020</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>Waveform</td>
<td>5,000</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>Glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>Red wine</td>
<td>1,599</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>Satimage</td>
<td>6,435</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>Census</td>
<td>95,130</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>Gas sensors</td>
<td>928,991</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>Covtype</td>
<td>581,012</td>
<td>54</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>Poker</td>
<td>1,025,010</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

TABLE I

characteristics of these datasets are summarized in Table I. Notice that for the datasets Diabetes, Cancer, and Census, we have removed the data with missing attribute values.

To develop the comparison in multiple aspects, k-fold cross-validation (K-CV) is considered in the experiments, where each dataset is divided into k blocks, with k-1 blocks as training data, namely sample input-output data, and the remaining block as testing data. Additionally, the nonparametric statistical analysis is used to assess if significant differences exist among different classifiers at a level of significance of α=0.1. For conducting multiple statistical comparisons over multiple datasets, as suggested in [14], the Friedman and Holm tests are employed.

For the first and the second experiments (Sections IV-B and IV-C), the datasets with relatively small number of data, including the 1st to the 10th datasets, are used to compare the performance of the Micro-EBRBS with the EBRBS and the conventional FRBCS and machine-learning classifiers. All these classifiers are implemented using Java programming (JDK 1.8.0) and the open source software (Weka and KEEL) on Intel (R) Core (TM) i5-4300U CPU at 1.90GHz and 4GB RAM with Windows 7. For each dataset, the average results of the 10 runs of each classifier are used to compare their performances.

For the third experiment (Section IV-D), the datasets with relatively large number of data, including the 11th to the 14th datasets, are used to compare the performance of the Micro-EBRBS with the big data FRBCS classifiers, and all these classifiers are executed in the 17 nodes cluster connected via 8Gt/s Ethernet LAN network, where the master node is composed of 1 Intel Xeon E5-2640 4 cores at 2.5GHz and 16GB RAM and the slave nodes are composed of 2 Intel Xeon E5-2670v2 10 cores at 2.5GHz and 64GB RAM. The entire cluster runs under Red Hat 7.3 and Apache Spark 2.1.0.

B. Comparative analysis between Micro-EBRBS and EBRBS

The first experiment aims to compare the accuracy and computing efficiency of the Micro-EBRBS with the EBRBS, respectively, and the comparisons are based on the 2/4/6/8/10-CV to investigate the influences on the accuracy and computing efficiency of these EBRBSs by using different numbers of training and testing data.

For the basic parameters of the Micro-EBRBS and EBRBS, suppose that all attribute weights are 1, namely

\[ \delta_i = 1; i=1,...,M \]  

(26)

where \( M \) is the number of antecedent attributes. The number of reference values is three for each antecedent attribute, and the utility value of these reference values is defined as follows:

\[ \{u(A_{ij}); j=1,2,3\} = \left\{ lb_i, \frac{lb_i + ub_i}{2}, ub_i \right\}; i=1,...,M \]  

(27)

where \( u(A_{ij}) \) denotes the utility value of the reference value \( A_{ij} \), \( lb_i \) and \( ub_i \) denote the lower and upper bounds of the \( i \)th antecedent attribute, respectively.
For the multi-class datasets, including Wine, Waveform, Glass, Red wine, the similar conclusions are obtained in terms of the computing time and the accuracy.

In order to show the detailed comparison of the Micro-EBRBS and the EBRBS, Tables II and III provide the results of rule generation time, inference time, total time, number of rules, number of activated rules, and the accuracy, in which the ratio is the result of the Micro-EBRBS divided by the result of the EBRBS. Hence, the Micro-EBRBS with a larger ratio in terms of accuracy and a smaller ratio in term of computing time is better than the EBRBS.

Table II shows that the computing time of the Micro-EBRBS is much less than that of the EBRBS for all two and multi-class datasets, where the minimum ratio of computing time is obtained from the dataset Banknote and its ratios are 0.002,

Figs. 13 and 14 show the computing time and the accuracy regarding the Micro-EBRBS in comparison with the EBRBS over two and multi-class datasets, respectively, in which the computing time includes the time of rule generation scheme, inference scheme and total time.

For the two-class datasets, including Diabetes, Cancer, Transfusion, and Banknote, the computing time of rule generation scheme regarding the EBRBS is increasing with the increasing number of training data used to generate EBRs, e.g., 50% data are regarded as the training data in 2-CV and 90% data as the training data in 10-CV. Additionally, there are slight differences between the Micro-EBRBS and the EBRBS in term of the computing time of the inference scheme and the accuracy. As a result, the total computing time of the EBRBS is much more than that of the Micro-EBRBS.

### Table II

<table>
<thead>
<tr>
<th>No. of classes</th>
<th>Dataset</th>
<th>Rule generation (ms)</th>
<th>Inference (ms)</th>
<th>Total (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Micro-EBRBS</td>
<td>EBRBS</td>
<td>Ratio</td>
</tr>
<tr>
<td>Two-class</td>
<td>Diabetes</td>
<td>6.1</td>
<td>429.0</td>
<td>0.014</td>
</tr>
<tr>
<td></td>
<td>Cancer</td>
<td>40.8</td>
<td>1,475.5</td>
<td>0.028</td>
</tr>
<tr>
<td></td>
<td>Transfusion</td>
<td>3.2</td>
<td>1,709.6</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>Banknote</td>
<td>12.4</td>
<td>6,645.2</td>
<td>0.002</td>
</tr>
<tr>
<td>Multi-class</td>
<td>Wine</td>
<td>4.6</td>
<td>81.1</td>
<td>0.058</td>
</tr>
<tr>
<td></td>
<td>Waveform</td>
<td>750.5</td>
<td>205,530.3</td>
<td>0.004</td>
</tr>
<tr>
<td></td>
<td>Glass</td>
<td>3.0</td>
<td>164.4</td>
<td>0.018</td>
</tr>
<tr>
<td></td>
<td>Red Wine</td>
<td>40.7</td>
<td>13,291.8</td>
<td>0.003</td>
</tr>
</tbody>
</table>

Fig. 13. Comparison of computing time and accuracy between Micro-EBRBS and EBRBS for two-class datasets.

Fig. 14. Comparison of computing time and accuracy between Micro-EBRBS and EBRBS for multi-class datasets.
Note that the accuracy and inference time of the Micro-EBRBS closely approximate the EBRBS for multi-class datasets, with a minimum ratio of 0.875 for the number of rules and 0.86 for the number of activated rules. Furthermore, it is clear from Table III that the accuracy of the Micro-EBRBS closely approximates that of the EBRBS and the range of the ratio can be expressed as [0.963, 1.000] for eight classification datasets.

In order to further compare the accuracy of the Micro-EBRBS and EBRBS, Friedman and Holm tests are applied to provide the statistical analysis based on two and multi-class datasets, respectively. From Table IV, although the EBRBS can obtain the best accuracy at most of two and multi-class datasets, none of hypotheses is in favor of the significant difference between EBRBS and Micro-EBRBS. However, the computing efficiency of Micro-EBRBS is much better than EBRBS.
In summary, for the comparison of the Micro-EBRBS and EBRBS, the experiment results have shown that the former has much less computing time than the latter. Moreover, the Micro-EBRBS is comparable to the EBRBS in terms of accuracy while much less number of rules and activated rules are used to address many classification datasets.

C. Comparative analysis between Micro-EBRBS and conventional classifiers

The second experiment aims to compare the accuracy of the Micro-EBRBS with the conventional classifiers, which include the FRBCS and the conventional machine-learning classifiers. Apart from the setting of the Micro-EBRBS introduced in Eqs. (26) and (27), the other classifiers are shown as follows:

1. Chi-FRBCS [29]: it was proposed by Chi et al., where the Penalized Certainty Factor (PCF) is used to calculate rule weights, the winner rule strategy is used as the fuzzy reasoning method, and the OVO is used to improve the performance of Chi-FRBCS in dealing with multi-classification problems. Here, assume that the number of fuzzy labels is three for each attribute and these fuzzy labels modeled as triangular membership function.

2. Other FRBCS classifiers: structural learning algorithm in vague environment (SLAVE), fuzzy hybrid genetic-based machine learning algorithm (FH-GBML), fuzzy unordered rule induction algorithm (FURIA), and fuzzy association rule-based classification method for high-dimensional problems (FARC-HD), they are all obtained from KEEL software. The setting of these FRBCS classifiers follows the previous study in [3].

3. Machine-learning classifiers: k nearest neighbor (KNN), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), Artificial Neural Network (ANN), and Support Vector Machine (SVM), are all obtained from WEKA software. Apart from the default setting, 20% number of training data is set as neighbors for the KNN, 5% number of training data is set as the minimum number of data per leaf for the DT, the number of random trees is set as 5 for the RF, and the number of iterations is set as 10 for the ANN.

Table V shows the accuracy of Micro-EBRBS in comparison with five FRBCS classifiers, including SLAVE, FH-GBML, FURIA, FARC-HD, and Chi-FRBCS, over two and multi-class datasets, respectively, where the result of the best accuracy is highlighted in bold-face and the number in brackets denotes the rank of each classifier. For the two-class datasets, the accuracy

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of classes</th>
<th>Average rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>Two-class</td>
<td>77.10% (1.5)</td>
</tr>
<tr>
<td>Cancer</td>
<td>Two-class</td>
<td>92.33% (3)</td>
</tr>
<tr>
<td>Transfusion</td>
<td>Two-class</td>
<td>76.60% (5)</td>
</tr>
<tr>
<td>Banknote</td>
<td>Two-class</td>
<td>91.33% (6)</td>
</tr>
<tr>
<td>Magic</td>
<td>Two-class</td>
<td>74.96% (6)</td>
</tr>
<tr>
<td>Wine</td>
<td>Multi-class</td>
<td>89.47% (6)</td>
</tr>
<tr>
<td>Waveform</td>
<td>Multi-class</td>
<td>81.48% (4)</td>
</tr>
<tr>
<td>Glass</td>
<td>Multi-class</td>
<td>58.05% (4)</td>
</tr>
<tr>
<td>Red Wine</td>
<td>Multi-class</td>
<td>55.60% (6)</td>
</tr>
<tr>
<td>Satimage</td>
<td>Multi-class</td>
<td>81.69% (4)</td>
</tr>
</tbody>
</table>

Table VI shows the computing time in seconds for Micro-EBRBS and five FRBCS classifiers.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of classes</th>
<th>Average rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>Two-class</td>
<td>0.7999</td>
</tr>
<tr>
<td>Cancer</td>
<td>Two-class</td>
<td>0.1000</td>
</tr>
<tr>
<td>Transfusion</td>
<td>Two-class</td>
<td>0.1000</td>
</tr>
<tr>
<td>Banknote</td>
<td>Two-class</td>
<td>0.1000</td>
</tr>
<tr>
<td>Magic</td>
<td>Two-class</td>
<td>0.1000</td>
</tr>
<tr>
<td>Wine</td>
<td>Multi-class</td>
<td>0.0180</td>
</tr>
<tr>
<td>Waveform</td>
<td>Multi-class</td>
<td>0.0250</td>
</tr>
<tr>
<td>Glass</td>
<td>Multi-class</td>
<td>0.0250</td>
</tr>
<tr>
<td>Red Wine</td>
<td>Multi-class</td>
<td>0.0250</td>
</tr>
<tr>
<td>Satimage</td>
<td>Multi-class</td>
<td>0.0250</td>
</tr>
</tbody>
</table>
of FARC-HD is better than other FRBCS classifiers as well as Micro-EBRBS ranked at the 3rd place in term of average rank. For the multi-class datasets, the FARC-HD remains its advantages in dealing with classification problems over other classifiers. Despite the fact that the FARC-HD outperforms the Micro-EBRBS, it is still possible to see a considerable decrease in the average rank of Micro-EBRBS, namely from 4.0 to 2.0. This is so because the distributed belief degree is used in each EBR to express multiple classes and Micro-EBRBS therefore has excellent abilities to deal with multi-class problems.

In addition to the accuracy and average rank for each dataset and each classifier shown in Table V, Table VI provides the statistical analysis of accuracy while the Micro-EBRBS is selected as the control method for the Friedman and Holm tests. As shown in Table VI, apart from the SLAVE and Chi-FRBCS in the case of multi-class datasets, all hypotheses regarding five FRBCS classifiers are accepted, which means that although some of FRBCS classifiers, such as the FARCH-HD, FURIA, and FH-GBML, are better than Micro-EBRBS, none of hypotheses is in favor of the significant differences between the FRBCS classifiers and the Micro-EBRBS. For the SLAVE and Chi-FRBCS in the case of multi-class datasets, the hypothesis is rejected, which means that the significant differences can be found to show a better accuracy of Micro-EBRBS comparing to the SLAVE and Chi-FRBCS.

Table VII shows the computing time of Micro-EBRBS in comparison with the SLAVE, FH-GBML, FURIA, FARC-HD, and Chi-FRBCS, where the result of the best computing time is highlighted in bold-face and the number in brackets denotes the rank of each classifier. In both of two and multi-class datasets, the computing time of Micro-EBRBS is close to that of Chi-FRBCS, and is significantly faster than the other FRBCS classifiers. This is because some additional methodologies were used to improve the FRBCS classifiers, e.g. the genetic algorithm, which is an iterative optimization algorithm, and is one of the components of SLAVE and FARC-HD, leading to a time-consuming process while using those FRBCS classifiers.

Hence, in the application of FRBCS classifiers for addressing big data classification problems, the related works introduced in Section II-B were all based on Chi-FRBCS owing to its high efficient process of dealing with data.

In order to further compare with the Micro-EBRB and Chi-FRBCS, Figs. 15 and 16 show their time of rule generation scheme, inference scheme, and total as well as the accuracy over two and multi-class relatively large datasets, respectively. For the two-class datasets Banknote and Magic, the computing time of rule generation scheme regarding the Chi-FRBCS closely approximates to that regarding the Micro-
EBRBS from 2-CV to 10-CV. Meanwhile, for the dataset Banknote, the Micro-EBRBS is slightly better than the Chi-FRBCS in terms of the computing time of inference scheme and the accuracy. But for the dataset Magic, the Chi-FRBCS is slightly better than the Micro-EBRBS regarding the computing time and accuracy. For the multi-class datasets Waveform and Satimage, apart from the computing time of rule generation scheme, the Micro-EBRBS is much better than the Chi-FRBCS in term of the computing time of the inference scheme and the accuracy.

In order to show the detailed comparison of Micro-EBRBS and Chi-FRBCS, Tables VIII and IX provide the results and their ratio obtained from 10-CV, such as the rule generation time, inference time, total time, number of rules, number of activated rules, and accuracy. Table VIII shows that the Chi-FRBCS has less computing time than the Micro-EBRBS in terms of the rule generation scheme for both two and multi-class datasets and the inference scheme for two-class datasets. However, the Micro-EBRBS has less computing time of the inference scheme than the Chi-FRBCS in the multi-class datasets. As a result, the Chi-FRBCS has better total time and their ratio are 1.184 and 1.142 for the two-class datasets Banknote and Magic, and the Micro-EBRBS has better total time and their ratio are 0.626 and 0.141 for the multi-class datasets Waveform and Satimage.

Table IX shows that the number of rules in the Chi-FRBCS is the same as the Micro-EBRBS, but the number of activated rules in the Chi-FRBCS is smaller than the Micro-EBRBS, where the maximum ratio is obtained from the multi-class dataset Waveform and its ratio is 3.307, and except for the dataset Magic, the accuracy of the Chi-FRBCS is worse than the Micro-EBRBS for all two and multi-class datasets, where the maximum ratio is obtained from the multi-class dataset Satimage and its ratio is 1.142.

In order to compare the accuracy of the Micro-EBRBS with the conventional machine-learning classifiers, Table X shows the accuracy of seven classifiers for ten classification datasets. The number in brackets denotes the rank of each classifier and the best result is marked as bold in Table X. For the two-class datasets, the accuracy of the RF and ANN are better than the Micro-EBRBS ranked at the 3rd place, and are further better than the KNN, NB, DT, and SVM. For the multi-class dataset,
the Micro-EBRBS can reach the second best average rank better than the KNN, DT, NB, ANN, and SVM. From Table X, RF obtains the best average rank in both two and multi-class datasets. This is partly because RF is an ensemble learning method that operates by constructing a multitude of DTs. Hence, in one sense, the ensemble learning used in the RF can also use to improve the Micro-EBRBS.

Table XI shows the statistical analysis of accuracy while the Micro-EBRBS is selected as the control method for the Friedman and Holm tests. From Table XI, all hypotheses regarding the six machine-learning classifiers are accepted, namely the Micro-EBRBS and the machine-learning classifiers have a similar accuracy for two and multi-class datasets without significant differences.

In summary, for the comparison of the Micro-EBRBS and the conventional classifiers, the experiment results have proven that the Micro-EBRBS has the similar accuracies with the conventional FRBCS and the conventional machine-learning classifiers but its computing time is much less than the conventional FRBCS classifiers except for the Chi-FRBCS.

It is worth noting that the Micro-EBRBS has the higher accuracy but less computing time than the Chi-FRBCS for multi-class datasets.

D. Comparative analysis between Micro-EBRBS and big data FRBCS classifiers

The third experiment aims to compare the accuracy and the computing efficiency of the Micro-EBRBS with the big data FRBCS classifiers, where the core supporting theory of the big data FRBCS classifiers is shown in Section IV-C and they include the following two versions [29]:

1) Chi-FRBCS-BigData-Max (Chi-Max for short): In this big data classifier, the rule generation scheme searches for the fuzzy rules with the same fuzzy label. Among these fuzzy rules, only the fuzzy rule with the highest rule weight is maintained in the fuzzy rule base.

2) Chi-FRBCS-BigData-Ave (Chi-Ave for short): In this big data classifier, the rule generation scheme also searches for the fuzzy rules with the same fuzzy label. Afterwards, the average rule weight of the fuzzy rules that have the same class is computed. Finally, the fuzzy rules with the greatest average rule weight is kept in the fuzzy rule base.

Fig. 17 shows the computing time and the accuracy of the Micro-EBRBS in comparison with the big data FRBCS classifiers, including Chi-Max and Chi-Ave, over two-class and multi-class datasets, respectively.

From Fig. 17, the Micro-EBRBS has the significantly better computing efficiency for the seven-class dataset Covtype and the ten-class dataset Poker, similar one for the three-class.
dataset Gas sensors and worse one for the two-class dataset Census comparing to big data FRBCS classifiers. For the ten, seven, and three-class datasets, the computing efficiency differences between these classifiers diminish gradually from 4 cores to 128 cores. Additionally, from the datasets Census and Gas sensors, while 128 cores are used in the cluster computing, the computing time is more than the result obtained from 64 cores mainly because of the increasing cluster costs. So, determination of the number of cores is important to improve the computing efficiency of the Micro-EBRBS.

From the comparison of accuracy, the Micro-EBRBS reflects a better robustness than the big data FRBCS classifiers because the accuracy is consistent for all two and multi-class datasets when varying the number of cores used in the cluster computing, but the accuracy of the big data FRBCS classifiers are changeable. Therefore, comparing to the big data FRBCS classifiers, the Micro-EBRBS is able to provide exactly the same classifier while implementing by the Apache Spark. More specifically, Table XII shows the value and ratio of the Micro-EBRBS, Chi-Max, and Chi-Ave under 4 cores, where the best result is marked as bold.

In term of the computing time, the big data FRBCS classifiers have a slight advantage while addressing the two-class dataset Census and all ratios of rule generation, inference, and total time are less than 1.056. However, the computing time of the Micro-EBRBS is much better than both Chi-Max and Chi-Ave while increasing number of classes, i.e. ten-class dataset Poker whose ratios are all smaller than 0.1 for the rule generation, inference, and total time. In term of the accuracy, the Micro-EBRBS is better than the big data FRBCS classifiers and the range of the ratio can be expressed as [1.039, 1.176] for four datasets.

Additionally, in order to detect significant differences among the obtained results with a level of significance of $\alpha=0.1$. Hence, in the datasets Census, Gas sensors, Covtype, and Poker, the accuracy of the Micro-EBRBS is better than that of the Chi-Max and Chi-Ave.

In summary, according to the comparison of the Micro-EBRBS and the big data FRBCS classifiers, it is evident that the Micro-EBRBS has the advantage of using less computing time and obtaining better accuracy and robustness for the big data multi-class datasets.

**E. Time complexity Comparison**

In this subsection, a comparison of the Micro-EBRBS, EBRBS, and Chi-FRBCS is provided to show which one has a better time complexity to deal with big data multi-class classification problems.

Suppose that there are $L$ rules in EBRB (or sample data), $L$ rules in the reduced EBRB, $S$ testing data, $M$ antecedent attribute with $J_i$ reference values, and $N$ classes. The time complexity of different schemes in the EBRBS, Micro-EBRBS, and Chi-FRBCS is shown in Table XIV based on discussions in Section II-B, Section III-C, and [29]. Additionally, in order to clearly compare three classifiers, their time complexity can be simplified under the assumptions: (1) the number of sample data $L$ is much bigger than the square of the number of classes $N^2$; and (2) the total number of reference values for antecedent attributes $\sum_{i=1}^{M} J_i$ is much bigger than the number of classes $N$.

**Remark 11.** It is clear from Table XIV that the time complexity of the Micro-EBRBS is better than the EBRBS in both the rule generation and the inference schemes due to the following three reasons:

1. The Micro-EBRBS has a simple process in the rule generation scheme because of excluding the calculation of rule weights comparing to the EBRBS.
2. The Micro-EBRBS has less number of rules in the reduced EBRB owning to using the proposed rule reduction method to downsize the EBRB comparing to the EBRBS.
3. The Micro-EBRBS can be implemented by using the Apache Spark to generate rules and classify test input data in parallel thanks to the better performance of independence in the rule generation and the inference schemes.

**Remark 12.** It is clear from Table XIV that the Chi-FRBCS is more efficient than the Micro-EBRBS in the rule generation

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Classifier</th>
<th>Census</th>
<th>Gas sensors</th>
<th>Covtype</th>
<th>Poker</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td>22,269</td>
<td>50,062</td>
<td>36,099</td>
<td>-</td>
</tr>
<tr>
<td>Rule generation (ms)</td>
<td>Micro-EBRBS</td>
<td>19,817</td>
<td>24,349</td>
<td>68,246</td>
<td>388,942</td>
</tr>
<tr>
<td></td>
<td>Chi-Max</td>
<td>20,007</td>
<td>23,000</td>
<td>67,656</td>
<td>376,419</td>
</tr>
<tr>
<td></td>
<td>Chi-Ave</td>
<td>18,759</td>
<td>20,916</td>
<td>70,127</td>
<td>368,942</td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td>12,879</td>
<td>34,170</td>
<td>31,258</td>
<td>-</td>
</tr>
<tr>
<td>Inference (ms)</td>
<td>Micro-EBRBS</td>
<td>199,324</td>
<td>129,858</td>
<td>1,818,237</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Chi-Max</td>
<td>191,375</td>
<td>17,845</td>
<td>8,310,398</td>
<td>0.016</td>
</tr>
<tr>
<td></td>
<td>Chi-Ave</td>
<td>190,816</td>
<td>17,223</td>
<td>8,307,228</td>
<td>0.016</td>
</tr>
<tr>
<td>Total (ms)</td>
<td>Micro-EBRBS</td>
<td>219,141</td>
<td>35,148</td>
<td>179,920</td>
<td>1,854,334</td>
</tr>
<tr>
<td></td>
<td>Chi-Max</td>
<td>211,382</td>
<td>42,194</td>
<td>8,378,644</td>
<td>0.022</td>
</tr>
<tr>
<td></td>
<td>Chi-Ave</td>
<td>209,575</td>
<td>40,223</td>
<td>8,374,884</td>
<td>0.022</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>Micro-EBRBS</td>
<td>89.69</td>
<td>48.60</td>
<td>57.20</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Chi-Max</td>
<td>86.34</td>
<td>41.34</td>
<td>52.79</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Chi-Ave</td>
<td>86.14</td>
<td>41.68</td>
<td>53.53</td>
<td>1.069</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Chi-Max</th>
<th>Chi-Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$ value</td>
<td>0.0133</td>
<td>0.0771</td>
</tr>
<tr>
<td>Critical value</td>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>Hypothesis</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
</tbody>
</table>
scheme. However, the Micro-EBRBS is more efficient than the Chi-FRBCS in the inference scheme, especially for the multi-classification problems. Additionally, the computing efficiency of both the two big data classifiers can be further improved by using the Apache Spark.

V. CONCLUSIONS

In this study, the analysis of the rule weight calculation and the ER algorithm were carried out to investigate the approach of reducing the time complexity of the EBRBS, a popular advanced rule-based system, followed by a ER-C algorithm and a domain division-based rule reduction method proposed for developing a micro version of EBRBS with high computing efficiency, called Micro-EBRBS. Furthermore, the Apache Spark was introduced to implement the Micro-EBRBS for better dealing with big data multi-classification problems. 14 classification datasets were used to validate the accuracy and computing efficiency of the Micro-EBRBS in comparison with the EBRBS, the conventional FRBCS and machine-learning classifiers, and the big data FRBCS classifiers. The detailed contributions are summarized as follows:

1) The non-necessity of the rule weight calculation and the ER algorithm involved in the EBRBS were investigated, in which the former demonstrates that it is unnecessary to calculate rule weights for each EBR under the assumption of large amount of data, the latter proves that the ER-C algorithm has the same functionality as the ER algorithm under the assumption of classification problems.

2) The division point and division domain were defined to divide the input space of the EBRBS into multiple local input spaces. Accordingly, the rule clustering strategy and rule reduction strategy were further defined to propose a domain division-based rule reduction method to downsized EBRB.

3) The Micro-EBRBS, which includes the rule reduction and the ER-C algorithm but excludes the rule weight calculation in comparison with the EBRBS, and its implementation based on the Apache Spark were developed to deal with big data multi-classification problems, which were then validated through the detailed case studies. The results have shown advantages of the Micro-EBRBS over the existing methods in terms of computing efficiency and classification accuracy.

For the future research, the application of Micro-EBRBS and further improvement to make it more effective to deal with the practical problem with uncertain and imbalance data.

APPENDIX A. INFERENCE SCHEME OF EBRBS FOR CLASSIFICATION PROBLEMS

The inference scheme of the EBRBS mainly includes two steps [22]: (1) calculation of activation weights for each EBR using distance measure and (2) integration of activated rules for estimating classes using the ER algorithm.

One thing to note is that the procedure of the first step is the same as the Micro-EBRBS. Hence, the activation weight $w_k$ ($k=1,...,L$) can be obtained by using Eq. (23), which shows a positive correlation between the $w_k$ and rule weight $\theta_i$. After calculating activation weights, all activated rules should be integrated using the analytical ER algorithm [32, 37]:

$$\beta_n = \left( \prod_{i=1}^{L} (w_i \beta_{ik} + 1-w_i \sum_{j=1}^{N} \beta_{jk}) \right) - \frac{\left( \prod_{i=1}^{L} (1-w_i \sum_{j=1}^{N} \beta_{jk}) \right)}{\left( \prod_{i=1}^{N} \prod_{k=1}^{L} (w_i \beta_{ik} + 1-w_i \sum_{j=1}^{N} \beta_{jk}) \right) - \left( \prod_{i=1}^{L} (1-w_i \sum_{j=1}^{N} \beta_{jk}) \right) - w_i (N-1) \prod_{i=1}^{L} \left( 1-w_i \sum_{j=1}^{N} \beta_{jk} \right) - \prod_{i=1}^{L} (1-w_i) \right.$$  \hspace{1cm} (A1)

The integrated belief distribution of the test input data $x$ is:

$$f(x) = \{(D_n, \beta_n); n=1,...,N\}$$  \hspace{1cm} (A2)

For classification problems, suppose $D_n$ denotes the $n$th class, the estimated class of the EBRBS can be obtained by seeking the greatest belief degree.

$$f(x) = D_n, n = \arg \max_{i=1,...,N} \{\beta_i\}$$  \hspace{1cm} (A3)

APPENDIX B. BELIEF DISTRIBUTION GENERATION AND RULE WEIGHT CALCULATION OF EBRBS

The belief distribution generation and the rule weight calculation are important processes in the rule generation scheme of the EBRBS and their details can be refer to [22].

One thing to note is that the detailed procedure of the belief distribution generation is the same as the Micro-EBRBS. Hence, for the $k$th ($k=1,...,L$) EBR, the belief distributions of the $i$th antecedent attribute $S_i^k = \{ (A_{ij}, \alpha_{ij}^k); j=1,...,J_i \}$ ($i=1,...,M$) and the consequent attribute $S^r = \{ (D_n, \beta_{nk}); n=1,...,N \}$ can be obtained by using Eqs. (16) and (19).

Definition B.1 (Similarity of two belief distributions). Suppose two belief distribution $P = (p_1,...,p_r)$ and $Q = (q_1,...,q_s)$, then the similarity of $P$ and $Q$ can be calculated as follows:

<table>
<thead>
<tr>
<th>Rule generation scheme</th>
<th>Inference scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(L \cdot \sum_{i=1}^{M} J_i + L \cdot N)$</td>
<td>$O(S \cdot L \cdot (\sum_{i=1}^{M} J_i + N))$</td>
</tr>
<tr>
<td>$O(L \cdot L \cdot \sum_{i=1}^{M} J_i + L \cdot N)$</td>
<td>$O(S \cdot L \cdot (\sum_{i=1}^{M} J_i + L \cdot L \cdot N))$</td>
</tr>
<tr>
<td>$O(L \cdot \sum_{i=1}^{M} J_i + L \cdot L \cdot N)$</td>
<td>$O(S \cdot L \cdot (\sum_{i=1}^{M} J_i + L \cdot L \cdot N))$</td>
</tr>
<tr>
<td>$O(S \cdot L \cdot \sum_{i=1}^{M} J_i + L \cdot L \cdot N)$</td>
<td>$O(S \cdot L \cdot (\sum_{i=1}^{M} J_i + L \cdot L \cdot N))$</td>
</tr>
</tbody>
</table>
Pseudocode of rule generation scheme of the Chi-FRBCS

**Input:** \( \mu_i(x_i) \) and \( R_k(x_i) \) denote the membership degree and the fuzzy label set of the fuzzy rule which is transformed from the sample input data \( x_i \) and the class \( y_i \), \( \forall \in \{D_1, \ldots, D_N\} \); \( Class_i(R_k) \) and \( w_{ij}(R_k) \) denote the class and the rule weight of the fuzzy rule \( R_k \) while considering the \( i \)th and the \( j \)th class as a two-class classification problem.

**Output:** the set of fuzzy rules \( FRB \)

1. \( Initialize \ FRB = \{\} \); 
2. For each sample input data \( x_i \in \{x_1, \ldots, x_m\} \):
   1. \( Initialize \ \mu_{ij}(x_i) = 1 \) and \( R_i(x_i) = \{\} \); 
   2. For each input data \( x_i, x_i \in \{x_1, \ldots, x_m\} \):
      1. Calculate \( \mu_{ij}(x_i) = \mu_{ij}(x_i) \cdot \max\{ \mu_{kj}(x_i) : j = 1, \ldots, J \} \); 
      2. Add \( R_i(x_i) = R_i(x_i) \cup \{A_{ij}\} \); \( s = \arg \max_{m=1, \ldots, M} \{ \mu_{ij}(x_i) \} \); 
      3. End for
3. If \( R_i \notin FRB \) and \( R_i \notin R_i(x_i) \) then:
   1. Update \( \mu_{ij}(R_i) = \mu_{ij}(R_i) + \mu_{ij}(x_i) \); 
   2. Else if \( R_i(x_i) \notin FRB \) then:
      1. Add \( FRB = FRB \cup \{R_i(x_i)\} \); 
      2. Initialize \( \mu_{ij}(R_i) = \mu_{ij}(x_i) \); 
      3. End if
4. End for
5. For each fuzzy rule \( R_i \) in \( FRB \):
   1. For each class \( D_i \) in \( D_i \) (i \( < \) j) in \( D_1, \ldots, D_N \):
      1. Initialize \( Class_i(R_i) = D_j = \max_{x_i \in D_j} \{ \mu_{ij}(R_i) \} \); 
      2. Initialize \( w_{ij}(R_i) = \mu_{ij}(R_i) \); 
   2. End for
6. End for

**Sim** \((P, Q) = 1 - d(P, Q) \)

\[
Sim(P, Q) = 1 - \min\{1, \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2}\} \quad (B1)
\]

where \( d(P, Q) \) denotes the distance between \( P \) and \( Q \).

Based on Definition B.1, for the \( k \)th \((k=1, \ldots, L)\) EBR, the similarity of rule antecedent (SRA) and the similarity of rule consequent (SRC) can be calculated as follows:

\[
SRA(R_i, R_j) = \min_{l=1, \ldots, M} \{\text{Sim}(S^l_i, S^l_j)\}
\]

\[
= \min_{l=1, \ldots, M} \left\{1 - \min\{1, \sqrt{\sum_{i=1}^{n} (\alpha_{i,j}^l - \alpha_{i,j}^l)^2}\}\right\}; \quad (B2)
\]

\[
SRC(R_i, R_j) = \text{Sim}(S^i, S^j)
\]

\[
= 1 - \min\{1, \sqrt{\sum_{i=1}^{n} (\beta_{i,j} - \beta_{i,j})^2}\}; \quad (B3)
\]

where \( l=1, \ldots, L \) and \( l \neq k \); \( S^l \) denotes the belief distribution of the \( l \)th antecedent attribute in the \( k \)th EBR; \( S^i \) denotes the belief distribution of the consequent attribute in the \( k \)th EBR.

**Definition B.2 (Consistency of EBRs).** Suppose the SRA and the SRC of the \( l \)th \((l=1, \ldots, L)\) and the \( k \)th \((k=1, \ldots, L; l \neq k)\) EBRs are \( SRA(R_i, R_j) \) and \( SRC(R_i, R_j) \), respectively, then the consistency of the rules \( R_i \) and \( R_j \) can be calculated as follows:

\[
Cons(R_i, R_j) = \exp\left(\frac{\left(\frac{SRA(R_i, R_j)}{SRC(R_i, R_j)} - 1\right)^2}{\left(1 - \frac{SRA(R_i, R_j)}{SRC(R_i, R_j)}\right)^2}\right) \quad (B4)
\]

Based on Definition B.2, the inconsistency degree of the \( k \)th EBR can be calculated as follows:

\[
Incons(R_i) = \sum_{l=1, l \neq k}^{L} \left(1 - Cons(R_i, R_j)\right)
\]

\[
= \sum_{l=1, l \neq k}^{L} 1 - \exp\left(\frac{\left(\frac{SRA(R_i, R_j)}{SRC(R_i, R_j)} - 1\right)^2}{\left(1 - \frac{SRA(R_i, R_j)}{SRC(R_i, R_j)}\right)^2}\right) \quad (B5)
\]

Finally, the rule weight is calculated as follows:

\[
\theta_k = \frac{Incons(R_i)}{\sum_{l=1}^{L} Incons(R_j)} \quad (B6)
\]

**APPENDIX C. RULE GENERATION AND INFERENCE SCHEMES OF CHI-FRBCS**

**A. Time complexity of the rule generation scheme**

The rule generation scheme of the Chi-FRBCS, which consists of the PCF and OVO, is introduced as follows:

As shown in the pseudocode of the rule generation scheme, the time complexity of calculating \( \mu_{ij}(x_i) \) shown in the 5th line and updating the set \( FRB \) shown in the 8th to the 13th lines
is \( O(\sum_{m=1}^{M} J_m) \) and \( O(L) \), respectively, for each sample data, where \( L \) is the number of fuzzy rules. In addition, from the 15th to the 20th lines, its time complexity is \( O(L \times N^2) \). As a result, the time complexity of the rule generation scheme involved in the Chi-FRBCS is shown as follows:

\[
O(L \times (\sum_{m=1}^{M} J_m + L) + L \times N^2) \quad (C1)
\]

\[ \text{C1:}
\]

**B. Time complexity of the inference scheme**

The inference scheme of the Chi-FRBCS, which consists of the winning rule strategy and OVO, is introduced as follows:

From the pseudocode of the rule generation scheme, the time complexity of calculating \( \mu(x) \) shown in the 7th line is \( O(M) \) for each test input data, fuzzy rule, and two-class classification problem. Hence, the time complexity of the inference scheme involved in the Chi-FRBCS is shown as follows:

\[
O(S \times L \times M \times N^2) \quad (C2)
\]

**REFERENCES**


**Long-Hao Yang** received the B.Eng. and M.Eng. degrees from Fuzhou University, China, in 2012 and 2015, respectively, and he is currently pursuing the Ph.D. degree at Fuzhou University, China.

He has published over 10 articles in BRB-related studies. His current research interests include BRB learning, machine learning and artificial intelligence, and BRB applications in complex systems modeling.

**Jun Liu** (M’09) received the BSc and MSc degrees in applied mathematics, and the PhD degree in information engineering from Southwest Jiaotong University, Chengdu, China, in 1993, 1996, and 1999, respectively.

He is currently a Reader in Computer Science at Ulster University, Northern Ireland, UK. His current research interests include logic and reasoning methods for intelligent systems and formal verification; intelligent DSSs and information management, with applications in health care, engineering, and industry field, etc.; applied computational intelligence for uncertainty analysis and optimisation. He has published more than 80 papers in journals indexed by the SCI and more than 80 contributions in International Conferences.

**Ying-Ming Wang** received his MSc in Systems Engineering from Huazhong University of Science and Technology, China, in 1987, and Ph.D. in Automatic Control Theory and Application from Southeast University, China, in 1991.

He is currently a full distinguished professor of Changjiang Scholars Program at Fuzhou University, Fuzhou, China. He has published over 123 SCI and 28 SSCI-indexed journal papers and has been the most cited Chinese researchers since 2014. His research interests include multiple criteria decision analysis, data envelopment analysis, rule-based inference, and quality function deployment.

**Luis Martínez** (M’10) received the M.Sc. and Ph.D. degrees in computer sciences from the University of Granada, Granada, Spain, in 1993 and 1999, respectively.