Expert Systems with Applications 41 (2014) 655–662

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An effective parallel approach for genetic-fuzzy data mining

Tzung-Pei Hong [a], Yeong-Chyi Lee [b,*], Min-Thai Wu [c]

[a] Dept. of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan
[b] Dept. of Information Management, Cheng Shiu University, Kaohsiung, Taiwan
[c] Dept. of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan

ARTICLE INFO

Keywords:
Data mining
Fuzzy set
Genetic algorithm
Parallel processing
Association rule

ABSTRACT

Data mining is most commonly used in attempts to induce association rules from transaction data. In the past, we used the fuzzy and GA concepts to discover both useful fuzzy association rules and suitable membership functions from quantitative values. The evaluation for fitness values was, however, quite time-consuming. Due to dramatic increases in available computing power and concomitant decreases in computing costs over the last decade, learning or mining by applying parallel processing techniques has become a feasible way to overcome the slow-learning problem. In this paper, we thus propose a parallel genetic-fuzzy mining algorithm based on the master–slave architecture to extract both association rules and membership functions from quantitative transactions. The master processor uses a single population as a simple genetic algorithm does, and distributes the tasks of fitness evaluation to slave processors. The evolutionary processes, such as crossover, mutation and production are performed by the master processor. It is very natural and efficient to run the proposed algorithm on the master–slave architecture. The time complexities for both sequential and parallel genetic-fuzzy mining algorithms have also been analyzed, with results showing the good effect of the proposed one. When the number of generations is large, the speed-up can be nearly linear. The experimental results also show this point. Applying the master–slave parallel architecture to speed up the genetic-fuzzy data mining algorithm is thus a feasible way to overcome the low-speed fitness evaluation problem of the original algorithm.

1. Introduction

As information technology (IT) progresses rapidly, its capacity to store and manage data in databases is becoming important. Though IT development facilitates data processing and eases demands on storage media, extraction of available implicit information to aid decision making has become a new and challenging task. Vigorous efforts have thus been devoted to designing efficient mechanisms for mining information and knowledge from large databases. As a result, data mining, first proposed by Agrawal, Imielinksi, and Swami (1993), has become a central field of study in the database and artificial intelligence areas.

Deriving association rules from transaction databases is most commonly seen in data mining (Chen, Han, & Yu, 1996; Famili, Shen, Weber, & Simoudis, 1997; Han & Fu, 1995). It discovers relationships among items such that the presence of certain items in a transaction tends to imply the presence of certain other items. In the past, Agrawal and his co-workers proposed a method (Srikant & Agrawal, 1996) for mining association rules from data sets using quantitative and categorical attributes. Their proposed method first determines the number of partitions for each quantitative attribute, and then maps all possible values of each attribute onto a set of consecutive integers.

Recently, the fuzzy set theory (Zadeh, 1965) has been used more and more frequently in intelligent systems because of its simplicity and similarity to human reasoning (Kandel, 1992). Many fuzzy learning algorithms for inducing rules from given sets of data have been designed and used to good effect with specific domains (Hong & Lee, 1996; Yuan & Shaw, 1995). Several fuzzy mining algorithms for managing quantitative data have also been proposed (Cai, Fu, Cheng, & Kwong, 1998; Kaya & Alhajj, 2003; Luan, Sun, Zhang, Yu, & Zhang, 2012; Mangalampall & Pudi, 2009; Mohamadlou, Ghodsi, Razmi, & Keramati, 2009; Ouyang & Huang, 2009; Wang, Su, Liu, & Cai, 2012), where the membership functions were assumed to be known in advance. The given membership functions may, however, have a critical influence on the final mining results. Genetic algorithms (GAs) Holland, (1975) have also recently been used in the field of data mining since they are powerful search techniques in solving difficult problems and can provide feasible solutions in a limited amount of time. Hong et al. thus proposed a GA-based fuzzy data-mining method (Hong, Chen, Wu, & Lee, 2006) for extracting both association rules and membership functions from quantitative transactions. In that method, the fitness evaluation is based on the suitability of derived membership

* Corresponding author. Tel.: +886 77310606.
E-mail addresses: tphong@nuk.edu.tw (T.-P. Hong), yeongchyi@csu.edu.tw (Y.-C. Lee), d53040015@student.nsysu.edu.tw (M.-T. Wu).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.07.090

functions and the number of large itemsets. The evaluation for fitness values is, however, quite time-consuming.

Due to dramatic increases in available computing power and concomitant decreases in computing costs over last decades, learning or mining by applying parallel processing techniques has become a feasible way of overcoming the problem of slow learning (Cordón, Herrera, & Villar, 2001; Herrera, Lozano, & Verdegay, 1997; Wang, Hong, & Tseng, 1998). Several parallel approaches to speed up the process of data-mining were also proposed (Agrawal & Shafer, 1996; Chen, Wang & Chen, 2012; Joshi, Han, Karypis, & Kumar, 2000; Veloso, Meira, & Parthasarathy, 2003). In addition, some parallel methods with genetic algorithms were also suggested (Abramson & Abela, 1992; Araujo, Lopes, & Freitas, 1999). They have been applied to solving timetable scheduling and discovering classification rules.

Among the parallel architectures, the master–slave architecture is particularly easy to implement. It also usually promises substantial gains in performance (Cantu-Paz, 1998). The master processor allocates the tasks to the slave processors and collects the results from them. It can also do its own work if necessary. As mentioned before, the fitness evaluation in genetic-fuzzy data mining is usually very time-consuming. In this paper, we thus extend our previous work (Hong et al., 2006) by using the master–slave parallel architecture to dynamically adapt membership functions and to use them to deal with quantitative transactions in fuzzy data mining. It is very natural and efficient to design a parallel GA-fuzzy mining algorithm based on the master–slave architecture. The master processor uses a single population as a simple genetic algorithm does, and distributes the tasks of fitness evaluation for suitability of membership functions and large itemsets to slave processors. The evolutionary processes, such as crossover, mutation and production are performed by the master processor. We expect that by appropriately allocating the tasks among the different types of processors, the efficiency of the proposed genetic-fuzzy mining algorithm can greatly be raised.

The remainders of this paper are organized as follows. Related works about parallel processing applied to data mining and genetic algorithms are reviewed in Section 2. A parallel genetic-fuzzy mining framework based on the master–slave architecture is described in Section 3. The adopted representation of chromosomes and the genetic operators used in this paper are stated in Section 4. A parallel genetic-fuzzy mining algorithm is proposed in Section 5. A simple example for demonstrating the proposed algorithm is given in Section 6. The time complexity of the proposed algorithm is analyzed in Section 7. The experimental results are shown in Section 8. Finally, conclusion and future work are given in Section 9.

2. Review of related works

Some related works about data mining and genetic algorithms are first reviewed below.

2.1. Data mining

The goal of data mining is to discover important associations among items such that the presence of some items in a transaction will imply the presence of some other items. To achieve this purpose, Agrawal and his co-workers proposed several mining algorithms based on the concept of large itemsets to find association rules in transaction data (Agrawal & Srikant, 1994; Agrawal et al., 1993). They divided the mining process into two phases. In the first phase, candidate itemsets were generated and counted by scanning the transaction data. If the number of an itemset appearing in the transactions was larger than a pre-defined threshold value (called minimum support), the itemset was considered a large itemset. Itemsets containing only one item were processed first. Large itemsets containing only single items were then combined to form candidate itemsets containing two items. This process was repeated until all large itemsets had been found. In the second phase, association rules were induced from the large itemsets found in the first phase. All possible association combinations for each large itemset were formed, and those with calculated confidence values larger than a predefined threshold (called minimum confidence) were output as association rules.

Srikant and Agrawal then proposed a mining method (Srikant & Agrawal, 1996) to handle quantitative transactions by partitioning the possible values of each attribute. Hong et al. proposed a fuzzy mining algorithm to mine fuzzy rules from quantitative data (Hong, Kuo, & Chi, 2001). They transformed each quantitative item into a fuzzy set and used fuzzy operations to find fuzzy rules. Cai et al. proposed weighted mining to reflect different importance to different items (Cai et al., 1998). Each item was attached a numerical weight given by users. Weighted supports and weighted confidences were then defined to determine interesting association rules. Yue et al. then extended their concepts to fuzzy item vectors (Yue, Tsang, Yeung, & Shi, 2000). Fuzzy mining has also been widely adopted in many research fields, such as sequential pattern mining, intrusion detection, biological knowledge extraction, and so on (Chen & Huang, 2008; Lopez, Blanco, Garcia, & Marin, 2007; Romsaiyud1 & Premchaiswadi, 2011; Tajbakhsh, Rahmati, & Mirzaei, 2009; Wang et al., 2012; Watanabe & Fujioka, 2012). In the above fuzzy mining approaches, the membership functions were assumed to be known in advance. Wang et al. used GAs to tune membership functions for intrusion detection systems based on similarity of association rules (Wang & Bridges, 2000). Kaya and Alhajj (2003) proposed a GA-based clustering method to derive a predefined number of membership functions for getting a maximum profit within an interval of user specified minimum support values.

As to parallel data mining, Agrawal and Shafer proposed three parallel mining algorithms based on the Apriori algorithm for speeding up the mining process (Agrawal & Shafer, 1996). The first one was called count distribution, in which the counting task for itemsets was distributed in different processors. The second one was called data distribution, in which itemsets were distributed in different processors and the results were broadcast to each processor for generating globe candidate itemsets in the next phase. The third one was called candidate distribution, which reduced the problem of synchronization between processors by repartitioning transactions according to the itemsets allocated to distinct processors. In addition, two parallel algorithms for mining frequent itemsets were also proposed based on the data-distribution and the candidate-distribution approaches by using the lattice data structure (Veloso et al., 2003).

2.2. Genetic algorithms

Genetic algorithms (GAs) have become increasingly important for researchers in solving difficult problems since they could provide feasible solutions in a limited amount of time (Homaifar, Guan, & Liepins, 1993). They were first proposed by Holland (1975) and have been successfully applied to the fields of optimization, machine learning, neural network, fuzzy logic controllers, and so on (Alcala, Alcala-Fdez, Gacto, & Herrera, 2007; Gautam, Khare & Pardasani, 2010). GAs are developed mainly based on the ideas and the techniques from genetic and evolutionary theory (Grefenstette, 1986). According to the principle of survival of the fittest, they generate the next population by several operations, with each individual in the population representing a possible solution. There are three principal operations in a genetic algorithm.

1. The crossover operation generates offspring from two chosen individuals in the population, by exchanging some bits in the two individuals. The offspring thus inherit some characteristics from each parent.
2. The mutation operation generates offspring by randomly changing one or several bits in an individual. The offspring may thus possess different characteristics from their parents. Mutation prevents local searches of the search space and increases the probability of finding global optima.
3. The selection operation chooses some offspring for survival according to predefined rules. This keeps the population size within a fixed constant and puts good offspring into the next generation with a high probability.

On applying genetic algorithms to solving a problem, the first step is to define a representation that describes the problem states. The most common way used is the bit string representation. An initial population of individuals, called chromosomes, is then defined and the three genetic operations (crossover, mutation, and selection) are performed to generate the next generation. Each chromosome in the population is evaluated by a fitness function to determine its goodness. This procedure is repeated until a user-specified termination criterion is satisfied.

As to parallel GA, three types were proposed as follows (Cantu-Paz, 1998).

1. Single-population master–slave GAs: This type of parallel GAs uses a single population as simple GAs do, but evaluates fitness values by distributing chromosome among several processors. In addition, genetic operators such as crossover and mutation consider the entire population and are implemented in a main processor.
2. Single-population fine-grained GAs: This type of parallel GAs needs great amounts of parallel processors and consists of one spatially structured population. Genetic operators are constrained to a small neighborhood.
3. Multiple-population coarse-grained GAs: This type of parallel GAs consists of several subpopulations and their individuals can occasionally be migrated.

Among the above three types, the type of master–slave parallel GAs consists of a simple structure and uses less parameter to control the process of evolution. This type of parallel GAs has been successfully applied for solving timetable scheduling and discovering classification rules (Abramson & Abela, 1992; Araujo et al., 1999). In this paper, we will use this parallel architecture to fuzzy data mining due to its suitability.

3. A parallel genetic-fuzzy mining framework

In Hong et al. (2006), we used the fuzzy and GA concepts to discover both useful association rules and suitable membership functions from quantitative values. The proposed approach in that paper is shown in Fig. 1, where a genetic algorithm was proposed for searching membership functions suitable for mining problems and then the final best set of membership functions was used to mine association rules.

In Fig. 1, the proposed approach maintains a population of sets of membership functions, and uses the genetic algorithm to automatically derive the resulting one. It first transforms each set of membership functions into a fixed-length string. It then chooses appropriate strings for "mating", gradually creating good offspring membership function sets. The offspring membership function sets then undergo recursive "evolution" until a good set of membership functions has been obtained. The fitness evaluation is based on the suitability of derived membership functions and the number of large itemsets. The evaluation for fitness values is, however, quite time-consuming since it must find large 1-itemsets for each chromosome in each generation.

In this section, a parallel genetic-fuzzy framework based on the master–slave architecture is thus proposed to speed up the mining process. The proposed framework is shown in Fig. 2.

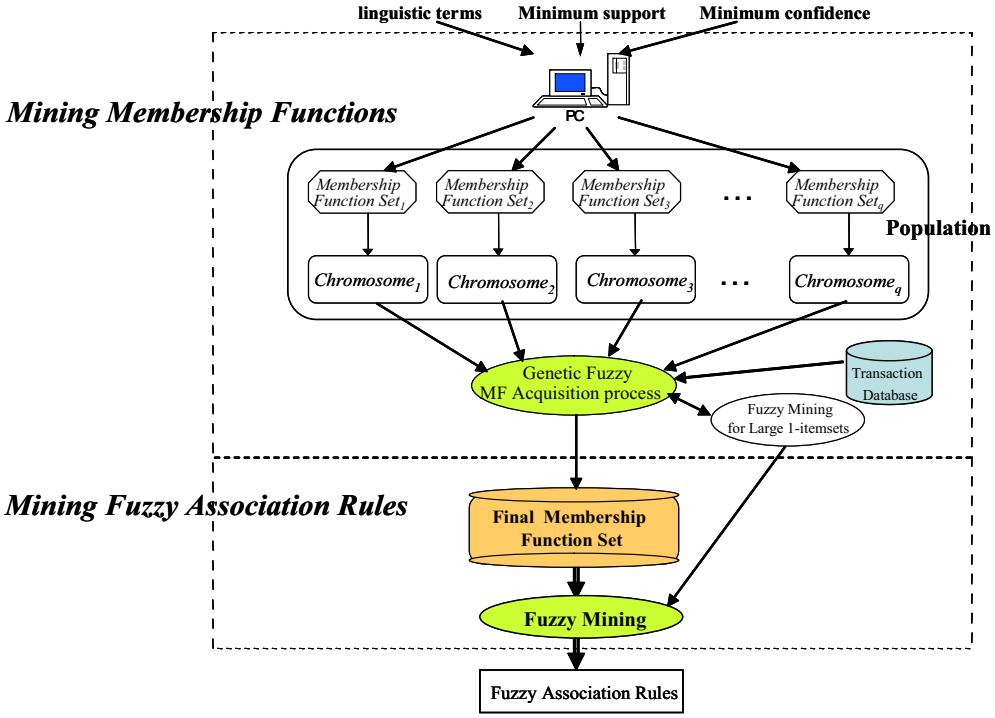

Fig. 1. A GA-based approach for fuzzy data mining.

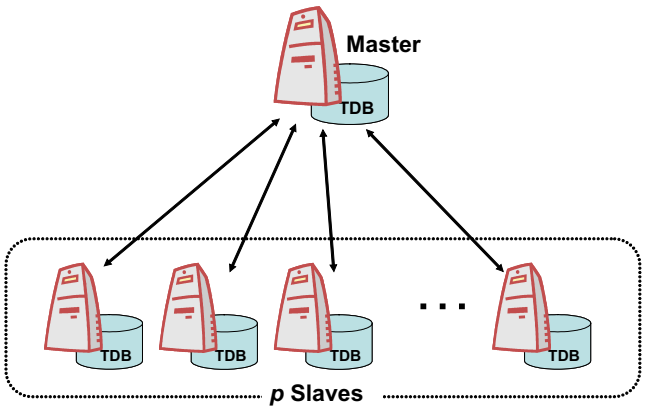

Fig. 2. The proposed parallel genetic-fuzzy mining framework.

In Fig. 2, there are $p + 1$ processors composed of one master and p slaves. Each processor has an identical transaction database for mining. Since the fitness evaluation process is the most time-consuming part in the entire genetic-fuzzy mining process, it is thus processed in parallel by the slave processors. The other part is processed by the master processor. The parallel genetic-fuzzy data mining framework includes two phases, mining membership function and mining fuzzy association rules. In the first phase, the master and the slaves cooperate to gradually discover a set of adaptive membership functions. The master processor maintains a population of sets of membership functions and performs the genetic operators to select, mate and evolve chromosomes. The master processor distributes the tasks of fitness evaluation for suitability of membership functions and large itemsets to slave processors. Each slave processor with an identical transaction database then calculates the fitness value of a chromosome that is assigned by the master. The evaluation results from the slaves are then sent back to the master. The master processor then performs the evolutionary processes, such as crossover, mutation and production according to the evaluation results collected. After undergoing recursive evolutions, a good set of membership functions can be obtained. The good set of membership functions is then used in the second phase by the master processor to mine fuzzy association rules. The second phase is not necessary to be processed in parallel since the most time-critical part is phase 1. The time complexity analyzed later will show this.

4. Chromosome representation and fitness evaluation

It is important to encode membership functions as string representation for GAs to be applied. Several possible encoding approaches have been described in Cordón et al. (2001), Parodi and Bonelli (1993) and Wang et al. (1998). In this paper, each set of membership functions is encoded as a chromosome and handled as an individual with real-number schema.

In order to effectively encode the associated membership functions, we use two parameters to represent each membership function, as Parodi and Bonelli (1993) did. Membership functions applied to a fuzzy rule set are then assumed to be isosceles-triangle functions as shown in Fig. 3, where R_{jk} denotes the membership function of the k-th linguistic term of item I_j, c_{jk} indicates the center abscissa of fuzzy region R_{jk}, and w_{jk} represents half the spread of fuzzy region R_{jk}.

As Parodi and Bonelli did, we then represent each membership function as a pair (c,w). Thus, all pairs of (c,w)'s for a certain item are concatenated to represent its membership functions. Thus the set of membership functions MF_1 for the first item I_1 is then represented as a substring of $c_{11}w_{11},\ldots,c_{1|I1|}w_{1|I1|}$, where $|I_1|$ is the number of terms of I_1. The entire set of membership functions is then encoded by concatenating substrings of $MF_1, MF_2, \ldots, MF_j$. Since c and w are both numeric values, a chromosome is thus encoded as a fixed-length real-number string rather than a bit string.

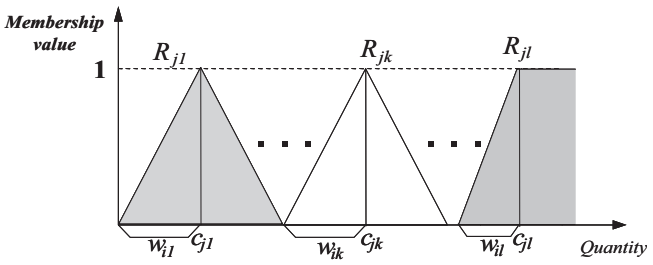

Fig. 3. Membership Functions of item I_j.

Note that other types of membership functions (e.g. non-isosceles trapezes) can also be adopted in our method. For coding non-isosceles triangles and trapezes, three and four points are needed instead of two for isosceles triangles.

According to the proposed representation, each chromosome thus consists of a set of membership functions for all the items. This representation allows genetic operators (defined later) to search for appropriate solutions.

In order to develop a good set of membership functions from an initial population, the genetic algorithm selects *parent* membership function sets with high fitness values for mating. Note that the selection of membership function sets is performed by the master processor, and the evaluation of each membership function set is processed by the slave processors. An evaluation function is defined to qualify the derived membership function sets. The evaluation results are then sent back to the master processor to control how the solution space is searched to promote the quality of the membership functions. The fitness value of a chromosome C_q is defined as follows:

$$f(C_q) = \frac{|L_1|}{suitability\,(C_q)},$$

where suitability (C_q) is the suitability of the membership functions in a chromosome C_q and $|L_1|$ is the number of large 1-itemsets obtained by using the set of membership functions. The suitability is defined according to the two factors – overlap ratio and coverage ratio. The overlap ratio of two membership functions is defined as the overlap length divided by half the minimum span of the two functions. The coverage ratio of a set of membership functions for an item is defined as the coverage range of the functions divided by the maximum quantity of that item in the transactions. The suitability factor used in the fitness function can reduce the occurrence of the two bad kinds of membership functions shown in Fig. 4, where the first one is too redundant, and the second one is too separate. Details can be referred to in Hong et al. (2006).

Besides, using the number of large 1-itemsets can achieve a trade-off between execution time and rule interestingness. Usually, a larger number of 1-itemsets will result in a larger number of all itemsets with a higher probability, which will thus usually imply more interesting association rules. The evaluation by 1-itemsets is, however, faster than that by all itemsets or interesting association rules. It can be further speeded up by the parallel processing approach proposed in this paper.

Two genetic operators, the *max–min–arithmetical* (*MMA*) *crossover* proposed in Herrera et al. (1997) and the *one-point mutation*, are used in the genetic fuzzy mining framework. Note that the genetic operations are performed by the master processor.

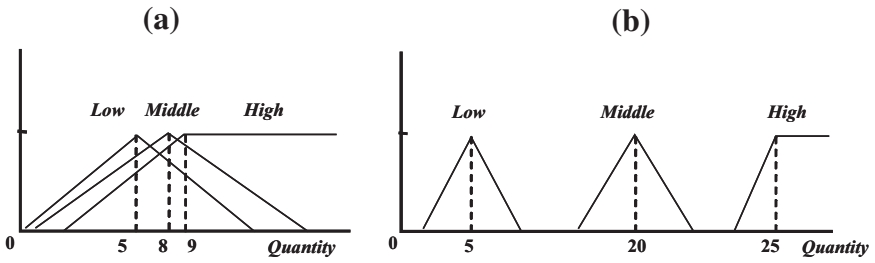

Fig. 4. Two bad membership functions.

5. The proposed parallel genetic-fuzzy mining algorithm

According to the above description, the proposed parallel algorithm for mining both fuzzy association rules and membership functions is described below.

5.1. The proposed parallel genetic-fuzzy mining algorithm

INPUT: One master processor, p slave processors (p is the maximum number of individuals to be evaluated in each generation), a body of n quantitative transaction data stored in each processor, a set of m items, a support threshold α, and a confidence threshold λ.

OUTPUT: A set of fuzzy association rules with its associated set of membership functions.

STEP 1: Randomly generate a population of individuals by the master processor; each individual is a set of membership functions for all the m items.

STEP 2: Encode each set of membership functions into a string representation by the master processor.

STEP 3: Distribute the individuals from the master processor to the slave processors.

STEP 4: Calculate the fitness value of each chromosome by each corresponding slave processor by the following substeps:

STEP 4.1: For each transaction datum D_i, $i = 1$ to n, and for each item I_j, $j = 1$ to m, transfer the quantitative value $v_j^{(i)}$ into a fuzzy set $f_j^{(i)}$ represented as:

$$\left(\frac{f_{j1}^{(i)}}{R_{j1}}+\frac{f_{j2}^{(i)}}{R_{j2}}+\cdots+\frac{f_{jl}^{(i)}}{R_{jl}}\right),$$

using the corresponding membership functions represented by the chromosome, where R_{jk} is the k-th fuzzy region (term) of item I_j, $f_{jl}^{(i)}$ is $v_j^{(i)}$'s fuzzy membership value in region R_{jk}, and $l(=|I_j|)$ is the number of linguistic terms for I_j.

STEP 4.2: For each item region R_{jk}, calculate its scalar cardinality $count_{jk}$ on the transactions as follows:

$$count_{jk}=\sum_{i=1}^{n} f_{jk}^{(i)}.$$

STEP 4.3: For each R_{jk}, $1 \leqslant j \leqslant m$ and $1 \leqslant k \leqslant |I_j|$, check whether its $count_{jk}$ is larger than or equal to the minimum support threshold α. If R_{jk} satisfies the above condition, put it in the set of large 1-itemsets (L_1). That is:

$$L_1 = \{R_{jk} | count_{jk} \geqslant \alpha,\ 1 < j < m \text{ and } 1 < k < |I_j|\}.$$

STEP 4.2: Set the fitness value of the chromosome as the number of large itemsets in L_1 divided by *suitability* (C_q). That is:

$$f(C_q)=\frac{|L_1|}{suitability(C_q)}.$$

STEP 1: Send the fitness value $f(C_q)$ of each chromosome C_q from each slave processor to the master processor.

STEP 2: Execute the crossover operations on the population by the master processor.

STEP 3: Execute the mutation operations on the population by the master processor.

STEP 4: Distribute the individuals to be evaluated from the master processor to the slave processors.

STEP 5: Calculate the fitness value of each chromosome by each corresponding slave processor as in STEP 4.

STEP 6: Send the fitness value $f(C_q)$ of each chromosome C_q from each slave processor to the master processor.

STEP 7: Use the defined selection criteria to choose suitable individuals for the next generation by the master slave.

STEP 8: If the termination criterion is not satisfied, go to Step 3; otherwise, do the next step.

STEP 9: Use the set of membership functions with the highest fitness value for finding all fuzzy large itemsets by the master processor.

STEP 10: Find the fuzzy association rules from the fuzzy large itemsets by the master processor.

In Steps 13 and 14, our fuzzy mining algorithm proposed in Hong et al. (2001) can be used to find the results.

6. An example

In this section, an example is given to illustrate the proposed parallel fuzzy mining algorithm. The data set in this example includes six transactions shown in Table 1. This is a simple example to show how the proposed algorithm can be used to mine membership functions and fuzzy association rules from data.

STEP 1: Ten individuals are randomly generated as the initial population by the master processor. Each individual represents a set of membership functions for all the four items including milk, bread, cookies, and beverage.

STEP 2: Each set of membership functions is encoded into a chromosome according to the representation proposed in Section 4. Assume the ten individuals are generated as follows:

C_1: **5, 5, 10, 5, 15, 5,** 6, 6, 12, 6, 18, 6, **3, 3, 6, 3, 9, 3,** 4, 4, 8, 4, 12, 4;

C_2: **5, 5, 10, 5, 15, 5,** 4, 6, 10, 6, 16, 6, **4, 3, 7, 3, 10, 3,** 4, 4, 8, 4, 12, 4;

C_3: **4, 3, 7, 3, 10, 3,** 6, 6, 12, 6, 18, 6, **6, 5, 11, 5, 16, 5,** 6, 4, 10, 4, 14, 4;

C_4: **5, 2, 7, 2, 9, 2,** 5, 4, 9, 4, 13, 4, **6, 5, 11, 5, 16, 5,** 6, 4, 10, 4, 14, 4;

C_5: **4, 3, 7, 3, 10, 3,** 6, 6, 12, 6, 18, 6, **5, 3, 8, 3, 11, 3,** 3, 4, 7, 4, 11, 4;

C_6: **6, 3, 9, 3, 12, 3,** 5, 5, 10, 5, 15, 5, **4, 4, 8, 4, 12, 4,** 6, 4, 10, 4, 14, 4;

Table 1
Six transactions in this example.

TID	Items
*T*1	(Milk, 5); (bread, 10); (cookies, 7), (beverage, 7)
*T*2	(Milk, 7); (bread, 14); (cookies, 12).
*T*3	(Bread, 15); (cookies, 12)
*T*4	(Milk, 2); (bread, 5); (cookies, 5).
*T*5	(Bread, 9)
*T*6	(Milk, 13); (beverage, 12)

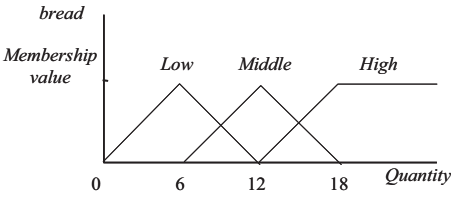

Fig. 5. The membership functions for bread in the first slave processor.

C_7: **3, 3, 6, 3, 9, 3,** 6, 2, 8, 2, 10, 2, **6, 5, 11, 5, 16, 5,** 4, 4, 8, 4, 12, 4;
C_8: **4, 3, 7, 3, 10, 3,** 6, 6, 12, 6, 18, 6, **5, 6, 11, 6, 17, 6,** 6, 4, 10, 4, 14, 4;
C_9: **3, 3, 6, 3, 9, 3,** 6, 3, 9, 3, 12, 3, **6, 5, 11, 5, 16, 5,** 6, 2, 8, 2, 10, 2;
C_{10}: **4, 3, 7, 3, 10, 3,** 6, 2, 8, 2, 10, 2, **6, 5, 11, 5, 16, 5,** 6, 3, 9, 3, 12, 3.

Each chromosome represents the membership functions of the four items and each item has three membership functions.

STEP 3: The ten individuals are distributed to the slave processors. Each slave processor then has its own chromosome to process.

STEP 4: The fitness value of each chromosome is then calculated by a slave processor by the following substeps:

STEP 4.1: The quantitative values of each transaction datum are transformed into a fuzzy set according to the membership functions in each chromosome. Take the first slave processor as an example. Its membership functions for bread in C_1 are represented as (6,6,12,6,18,6), which are shown in Fig. 5.

For transaction T5, the amount "9" of item *bread* is converted into the fuzzy set $\left(\frac{0.5}{bread.Low}+\frac{0.5}{bread.Middle}\right)$ using the above membership functions. The results for all the transactions are shown in Table 2, where the notation *item.term* is called a fuzzy region.

STEP 4.2: The scalar cardinality of each fuzzy region in the transactions is calculated as the *count* value. Take the fuzzy region *milk.Low* as an example. Its scalar cardinality = (1.0 + 0.6 + 0.0 + 0.4 + 0.0 + 0.0) = 2.0. The counts for all the fuzzy regions are shown in Table 3.

STEP 4.3: The count of any fuzzy region is checked against the predefined minimum support value α. Assume in this example, α is set at 0.2. The minimum count is then $10 * 0.2(=2)$. Since all the count values of *milk.Low*, *bread.Middle* and *cookies.High* are larger than 2.0, these items are then put in L_1 as shown in Table 4.

STEP 4.4: Since there are three large 1-itemsets for the membership functions of C_1 and its suitability is calculated as 4 according to the formula for the membership functions. The fitness value of C_1 is thus 3/4(=0.75). The fitness values of all the chromosomes in the slave processors are shown in Table 5.

Table 2
The fuzzy sets transformed from the data in Table 1.

TID	Fuzzy set
*T*1	$\left(\frac{1.0}{milk.Low}\right)\left(\frac{0.33}{bread.Low}+\frac{0.67}{bread.Middle}\right)\left(\frac{0.67}{cookies.Middle}+\frac{0.33}{cookies.High}\right)\left(\frac{0.25}{beverage.Low}+\frac{0.75}{beverage.Middle}\right)$
*T*2	$\left(\frac{0.6}{milk.Low}+\frac{0.4}{milk.Middle}\right)\left(\frac{0.67}{bread.Middle}+\frac{0.33}{bread.High}\right)\left(\frac{1}{cookies.High}\right)$
*T*3	$\left(\frac{0.5}{bread.Middle}+\frac{0.5}{bread.High}\right)\left(\frac{1}{cookies.High}\right)$
*T*4	$\left(\frac{0.4}{milk.Low}\right)\left(\frac{0.83}{bread.Low}\right)\left(\frac{0.33}{cokkies.Low}+\frac{0.67}{cokkies.Middle}\right)$
*T*5	$\left(\frac{0.5}{bread.Low}+\frac{0.5}{bread.Middle}\right)$
*T*6	$\left(\frac{0.4}{milk.Middle}+\frac{0.6}{milk.High}\right)\left(\frac{1}{beverage.High}\right)$

Table 3
The counts of the fuzzy regions.

Item	Count	Item	Count
milk.Low	2.00	cookies.Low	0.33
milk.Middle	0.80	cookies.Middle	1.33
milk.High	0.60	cookies.High	2.33
bread.Low	1.67	beverage.Low	0.25
bread.Middle	2.33	beverage.Middle	0.75
bread.High	0.83	beverage.High	1.00

Table 4
The set of the large 1-itemsets (L_1) in this example.

Itemset	Count
milk.Low	2.0
bread.Middle	2.33
cookies.High	2.33

Table 5
The fitness value of each chromosome in each slave processor.

Chromosome	*f*	Chromosome	*f*
C_1	0.75	C_6	0.54
C_2	0.53	C_7	0.32
C_3	0.27	C_8	0.54
C_4	0.3	C_9	0.31
C_5	0.58	C_{10}	0.32

STEP 5: The fitness value of each chromosome is sent from each slave processor to the master processor.

STEP 6: After collecting all the fitness values from the slave processors, the master processor executes the crossover operations on the population.

STEP 7: The master processor the mutation operator to generate possible offspring. The operation is the same as the traditional one except that rearrangement may need to be done.

STEP 8: The master processor distributes all the possible offspring to the slave processors.

STEP 9: The fitness value of each possible offspring is then calculated by a slave processor as in STEP 4.

STEP 10: Each slave processor sends its fitness value to the master processor.

STEP 11: The master processor then selects ten chromosomes according to the given selection criteria. For example, the best ten chromosomes can be selected from the mix of the parents and the offspring.

STEP 12: The same procedure is then executed until the termination criterion is satisfied. The best chromosome (with the highest fitness value) is then output as the membership functions for deriving fuzzy rules.

STEP 13: The fuzzy large itemsets are then derived by the fuzzy mining method proposed in Hong and Lee (1996).
STEP 14: The fuzzy association rules are then derived from the fuzzy large itemsets.

7. Time complexity analysis

The time complexities of both the sequential and the parallel genetic-fuzzy mining algorithms are first analyzed. The speed-up of the parallel mining algorithm is then derived. Some notation is first defined as follows:

p:	the number of slave processors (individuals);
n:	the number of generations;
f:	the average execution time of calculating the fitness value of an individual in each generation;
g:	the average execution time of processing all genetic operations during a generation;
c:	the average communication time between a master processor and a slave processor during a generation;
s:	the average execution time of mining the association rules in the second phase;
T^s_{ave}:	the average execution time of the sequential GA-fuzzy mining algorithm;
T^p_{ave}:	the average execution time of the parallel GA-fuzzy mining algorithm;
S_{ave}:	the average speed-up calculated by T^s_{ave} over T^p_{ave}.

The average time complexity of the sequential mining algorithm is:

$$T^s_{ave} = n^*(p^*f + g) + s.$$

The parallel mining algorithm can distribute the fitness-evaluation tasks to the slave processors. Therefore, the execution time for the fitness evaluation in each generation needs only f. The parallel algorithm, however, needs additional computation time c between a master processor and slave processors. The average time complexity of the paralleled mining algorithm is thus:

$$T^p_{ave} = n^*(f + g + c) + s.$$

The average speed-up is thus:

$$S_{ave} = \frac{T^s_{ave}}{T^p_{ave}} = \frac{n(p^*f + g) + s}{n(f + g + c) + s}.$$

In this paper, the fitness evaluation is based on the suitability of derived membership functions and the number of large 1-itemsets. The entire database must be scanned to find the large 1-itemsets. The average evaluation time f is thus much larger than the average execution time for genetic operations g, especially when the processed dataset is large. In addition, since the master only distributes the code of a chromosome to a slave and receives a number (the evaluation value) from the slave, the communication time is thus very little. Therefore, the speed-up can be further simplified as the following:

$$S_{ave} = \frac{n(p^*f + g) + s}{n(f + g + c) + s} \approx \frac{n^*p^*f + s}{n^*f + s}.$$

The average evaluation time f includes the time of finding large 1-itemsets. The average execution time s of mining association rules in the second phase includes finding all large itemsets and deriving rules from them. s is thus larger than f. However, because the number of items in the longest large itemsets is usually small, some pruning techniques may be used to reduce the mining time s, and the number of generations is usually set at more than one hundred, s may thus be much smaller than $n * f$, especially when n is large. In this case, the above speed-up can be further simplified as:

$$S_{ave} \approx \frac{n^*p^*f + s}{n^*f + s} \approx \frac{n^*p^*f}{n^*f} = p.$$

Thus, when the number of generations is large, the speed-up is nearly linear.

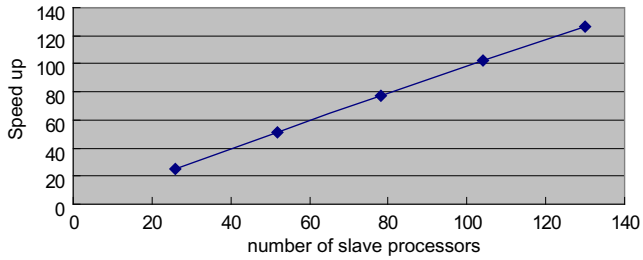

Fig. 6. The experimental results for speed-up with transaction I/O time.

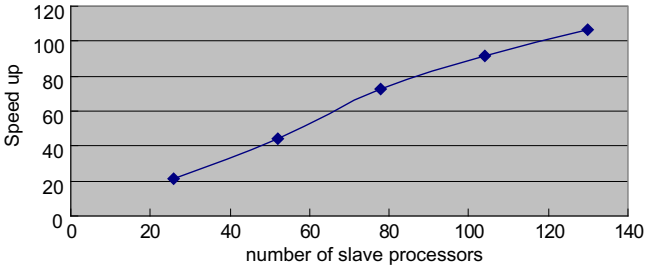

Fig. 7. The experimental results for speed-up without transaction I/O time.

8. Experimental results

In this section, the experiments made to show the performance of the proposed approach are described. They were simulated in Java on a personal computer with Intel Pentium IV 3.2 GHz and 512 MB RAM. 64 items and 10,000 transactions were used in the experiments. In each data set, the numbers of purchased items in transactions were first randomly generated. The purchased items and their quantities in each transaction were then generated. An item could not be generated twice in a transaction. The crossover rate p_c is set at 0.8, and the mutation rate p_m is set at 0.01. The minimum support α is set at 400. Experiments with different population sizes from 10 to 50 were made to show the speed-up trend of the proposed algorithm. The communication time was not considered. Note that by the genetic-fuzzy mining algorithm proposed in Hong et al. (2006), the maximum number p of individuals to be evaluated in each generation was $2.6 * r$, where r is the number of individuals in a population. The experimental results with and without transaction I/O time are shown respectively in Figs. 6 and 7.

It can be easily observed from the above two figures that the speed-up increases nearly linearly along with the number of slave processors. The speed-up is also close to the number of slave processors. It is quite consistent with our analysis in Section 6. Also note that the speed-up with I/O is larger than that without I/O. This is due to the increased evaluation time in slave processors when transactions must be read. Speed-up will thus be more apparent according to the speed-up analysis.

9. Conclusions and future works

In this paper, we have proposed a parallel genetic-fuzzy mining algorithm based on the master–slave architecture to extract both association rules and membership functions from quantitative transactions. The master and the slaves first cooperate to discover a set of suitable membership functions. The set of membership functions found is then used by the master processor to mine fuzzy association rules. The second phase is not necessary to be processed in parallel since the most time-critical part lies in phase 1. The time complexities for both sequential and parallel genetic-fuzzy mining algorithm have been analyzed, with results showing the good effect of the proposed approach. When the number of generations is large, the speed-up can be nearly linear. The experimental results have also shown this point. Applying the master–slave parallel architecture to speed up the genetic-fuzzy data mining algorithm is thus a feasible way to overcome the low-speed fitness evaluation problem of the original algorithm. The proposed parallel mining algorithm can also be easily modified for execution on a bounded number of processors to meet real-world requirements. In the future, we will continuously attempt to enhance the GA-based framework for more complex mining problems.

References

Abramson, D., & Abela, J. (1992). A parallel genetic algorithm for solving the school timetabling problem. In *The fifteenth australian computer science conference* (pp. 1–11).

Agrawal, R., & Srikant, R. (1994). Fast algorithm for mining association rules. In *The international conference on very large databases* (pp. 487–499).

Agrawal, R., Imielinksi, T., & Swami, A. (1993). Mining association rules between sets of items in large database. In *The 1993 ACM SIGMOD conference* (pp. 207–216).

Agrawal, R., & Shafer, J. C. (1996). Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering, 8*(6), 962–969.

Alcala, R., Alcala-Fdez, J., Gacto, M. J., & Herrera, F. (2007). Genetic learning of membership functions for mining fuzzy association rules. In *The IEEE international conference on fuzzy systems* (pp. 1–6).

Araujo, D. L. A., Lopes, H. S., & Freitas, A. A. (1999). A parallel genetic algorithm for rule discovery in large databases. In *The IEEE international conference on systems, man and cybernetics conference* (Vol. 3, pp. 940–945).

Cai, C. H., Fu, W. C., Cheng, C. H., & Kwong, W. W. (1998). Mining association rules with weighted items. In *The international database engineering and applications symposium* (pp. 68–77).

Cantu-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis, 10*(2), 141–171.

Chen, M. S., Han, J., & Yu, P. S. (1996). Data mining: an overview from a database perspective. In *IEEE transactions on knowledge and data engineering* (Vol. 8). No. 6.

Chen, J. S., Wang, J. Y., & Chen, F. G. (2012). Enhance the multi-level fuzzy association rules based on cumulative probability distribution approach. In *13th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing* (pp. 89–94).

Chen, Y. L., & Huang, T. C. K. (2008). A novel knowledge discovering model for mining fuzzy multi-level sequential patterns in sequence databases. *Data & Knowledge Engineering, 66*(3), 349–367.

Cordón, O., Herrera, F., & Villar, P. (2001). Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. *IEEE Transactions on Fuzzy Systems, 9*(4), 667–674.

Famili, A., Shen, W. M., Weber, R., & Simoudis, E. (1997). Data preprocessing and intelligent data analysis. In *Intelligent data analysis* (Vol. 1). No. 1.

Gautam, P., Khare, N., & Pardasani, K. R. (2010). A model for mining multilevel fuzzy association rule in database, CoRR abs/1001.3488.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Trans System Man, and Cybernetics, 16*(1), 122–128.

Han, J., & Fu, Y. (1995). Discovery of multiple-level association rules from large database. In *The international conference on very large databases*.

Herrera, F., Lozano, M., & Verdegay, J. L. (1997). Fuzzy connectives based crossover operators to model genetic algorithms population diversity. *Fuzzy Sets and Systems, 92*(1), 21–30.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Homaifar, A., Guan, S., & Liepins, G. E. (1993). A new approach on the traveling salesman problem by genetic algorithms. In *The fifth international conference on genetic algorithms*.

Hong, T. P., Chen, C. H., Wu, Y. L., & Lee, Y. C. (2006). A GA-based fuzzy mining approach to achieve a trade-off between number of rules and suitability of membership functions. *Soft Computing, 10*(11), 1091–1101.

Hong, T. P., Kuo, C. S., & Chi, S. C. (2001). Trade-off between time complexity and number of rules for fuzzy mining from quantitative data. *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, 9*(5), 587–604.

Hong, T. P., & Lee, C. Y. (1996). Induction of fuzzy rules and membership functions from training examples. *Fuzzy Sets and Systems, 84*, 33–47.

Joshi, M., Han, E. H., Karypis, G., & Kumar, V. (2000). Efficient parallel algorithms for mining associations. *Parallel and Distributed Systems, 1795*, 418–429.

Kandel, A. (1992). *Fuzzy expert systems*. Boca Raton: CRC Press, pp. 8–19.

Kaya, M., & Alhajj, R. (2003). A clustering algorithm with genetically optimized membership functions for fuzzy association rules mining. In *The IEEE international conference on fuzzy systems* (pp. 881–886).

Lopez, F. J., Blanco, A., Garcia, F., & Marin, A. (2007). Extracting biological knowledge by fuzzy association rule mining. In *IEEE international fuzzy systems conference* (pp. 1–6).

Luan, R. P., Sun, S. F., Zhang, J. F., Yu, F., & Zhang, Q. (2012). A dynamic improved a priori algorithm and its experiments in web log mining. In *9th International conference on fuzzy systems and knowledge, discovery* (pp. 2161–2164).

Mangalampalli, A., & Pudi, V. (2009). Fuzzy association rule mining algorithm for fast and efficient performance on very large datasets. In *IEEE international conference on fuzzy systems* (pp. 1163–1168).

Mohamadlou, H., Ghodsi, R., Razmi, J., & Keramati, A. (2009). A method for mining association rules in quantitative and fuzzy data. In *International conference on, computers & industrial engineering* (pp. 453–458).

Ouyang, W., & Huang, Q. (2009). Mining direct and indirect weighted fuzzy association rules in large transaction databases. *International Conference on Fuzzy Systems and Knowledge Discovery, 3*, 128–132.

Parodi, A., & Bonelli, P. (1993). A new approach of fuzzy classifier systems. In *Proceedings of fifth international conference on genetic algorithms* (pp. 223–230). Morgan Kaufmann.

Romsaiyud1, W., & Premchaiswadi, W. (2011). Applying mining fuzzy sequential patterns technique to predict the leadership in social networks. In *Ninth international conference on ICT and knowledge engineering* (pp. 134–137).

Srikant, R., & Agrawal, R. (1996). Mining quantitative association rules in large relational tables. In *The 1996 ACM SIGMOD international conference on management of data* (pp. 1–12).

Tajbakhsh, A., Rahmati, M., & Mirzaei, A. (2009). Intrusion detection using fuzzy association rules. *Applied Soft Computing, 9*(2), 462–469.

Veloso, A., Meira, W., Jr., & Parthasarathy, S. (2003). New parallel algorithms for frequent itemset mining in very large databases. In *The fifteenth symposium on computer architecture and high performance computing* (pp. 158–166).

Wang, W., & Bridges, S. M. (2000). Genetic algorithm optimization of membership functions for mining fuzzy association rules. In *The international joint conference on information systems, fuzzy theory and technology* (pp. 131–134).

Wang, M. H., Su, X. B., Liu, F. M., & Cai, R. C. (2012). A cancer classification method based on association rules. In *9th International conference on fuzzy systems and knowledge, discovery* (pp. 1094–1098).

Wang, C. H., Hong, T. P., & Tseng, S. S. (1998). Integrating fuzzy knowledge by genetic algorithms. *IEEE Transactions on Evolutionary Computation, 2*(4), 138–149.

Watanabe, T., & Fujioka, R. (2012). Fuzzy association rules mining algorithm based on equivalence redundancy of items. In *IEEE international conference on systems, man, and, cybernetics* (pp. 1960–1965).

Yuan, Y., & Shaw, M. J. (1995). Induction of fuzzy decision trees. *Fuzzy Sets and Systems, 69*, 125–139.

Yue, S., Tsang, E., Yeung, D., & Shi, D. (2000). Mining fuzzy association rules with weighted items. In *The IEEE international conference on systems, man and cybernetics* (pp. 1906–1911).

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), 338–353.