Tạp chí Tin học và Điều khiển học, T.20, S.2 (2004), 121–136

PARALLEL MINING FOR FUZZY ASSOCIATION RULES

PHAN XUAN HIEU, HA QUANG THUY

Faculty of Technology, Vietnam National University, Hanoi

Abstract. In this article, we focus on mining fuzzy association rules. First, we specially highlight the delicate relation between fuzzy association rules and fuzzy theory. As a result, we will recommend a method to convert the fuzzy association rules into quantitative ones. More remarkably, we propose a new parallel algorithm for mining fuzzy association rules. The algorithm has been experimented on PC-Cluster (using MPI standard) and returned optimistic results. The testing tools named FuzzyARM and ParallelFARM were also developed and run on serial and parallel systems respectively.

Keyword: association rule, data mining, fuzzy association rule, mining fuzzy association rule, parallel algorithm, serial algorithm.

Tóm tắt. Bài báo này định hướng tới khai phá luật kết hợp mờ. Đầu tiên chúng tôi nêu bật mối liên hệ tinh tế giữa luật kết hợp mờ với lý thuyết mờ. Từ đó, chúng tôi đề nghị phương pháp chuyển đổi luật kết hợp mờ thành luật kết hợp định lượng. Đáng kể hơn, chúng tôi đề xuất thuật toán song song mới khai phá luật kết hợp mờ. Thuật toán đã được thử nghiệm trên cụm PC-cluster (dùng chuẩn MPI) và cho kết quả hiệu quả. Hai công cụ FuzzyARM và ParallelFARM đã được phát triển và chạy tương ứng trên các hệ thống tuần tự và song song.

1. INTRODUCTION AND RELATED WORD

Association rule is the form of "70 percent of customers that *purchase beer* also *purchase dry beef*, 20 percent of customers purchase both". "Purchase beer" and "purchase dry beef" are called the antecedent and the consequent of the association rule respectively. 20% is called *support* factor (the percentage of transactions or records that contain both antecedent and consequent of a rule) and 70% is called *confident* factor (the percentage of transactions or records that hold the antecedent also hold the consequent of a rule).

Almost all previous algorithms deal with binary association rules [11, 23, 24]. In binary association rules, an item is only determined whether it is present or not. The quantity associated with each item is fully ignored, e.g. a transaction buying twenty bottles of beer is the same a transaction that buys only one bottle. However, attributes in real world databases may be binary, quantitative, or categorical, etc. To discover association rules that involve these data types, quantitative and categorical attributes need to be discretized to convert into binary ones. There exist some of discretization methods that are proposed in [22] [26]. An example of this kind of rule is "*sex* = *'male'* and *age* $\in$ *'50..65'* and *weight* $\in$ *'60..80'* and *sugar in blood* > *120mg/ml* $\Rightarrow$ *blood pressure* = *'high'*, with support 30% and confidence 65%". However, quantitative association rule expose several shortcomings such as "sharp boundary

problem" and meaning interpretation due to the traditional methods of data discretization. Fuzzy association rule was suggested to overcome these drawbacks in quantitative association rules. Fuzzy association rule is more natural and intuitive to users thanks to its "fuzzy" characteristics. An example is "*dry cough* and *high fever* and *muscle aches* and *breathing difficulties* $\Rightarrow$ *get SARS* = 'yes', with support 4% and confidence 80%". *High fever* in the above rule is a fuzzy attribute. We measure the body temperature based on a fuzzy concept.

In this article, we concentrate on fuzzy association rule and the new parallel algorithm for mining it. The rest of article is organized as follows: The section 2 formally describes the issue of mining binary association rules. Some methods of data discretization based on fuzzy concepts are mentioned in the section 3. The section 4 presents the fuzzy association rule and serial algorithm for mining this kind of rule. A new parallel algorithm for mining fuzzy association rule is proposed in the following section. And, the last section makes a conclusion by reviewing the achievements obtained throughout the article and stating the future work.

2. MINING ASSOCIATION RULES

Let $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of n items or attributes (in transactional or relational databases) and $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ be a set of m transactions or records. Each transaction is identified with its unique TID number. A (transactional) database $\mathbf{D}$ is a binary relation δ on the Descart multiplication $\mathbf{I}\times\mathbf{T}$ (or also written $\delta \subseteq \mathbf{I}\times\mathbf{T}$). We say $(\mathbf{i}, \mathbf{t}) \in \delta$ (or $\mathbf{i}\delta\mathbf{t}$) if an item $\mathbf{i}$ occurs in a transaction $\mathbf{t}$. Generally speaking, a transactional database is a set of transactions, where each transaction $\mathbf{t}$ contains a set of items or $\mathbf{t} \in 2^I$ (where $2^I$is the power set of $\mathbf{I}$) [13, 24].

$\mathbf{X} \subseteq \mathbf{I}$ is called an itemset. The *support factor* of an itemset $\mathbf{X}$, denoted as $s(\mathbf{X})$, is the percentage of transactions that contains $\mathbf{X}$. $\mathbf{X}$ is *frequent* if its support is greater than or equal to a user-specified minimum support (*minsup*) value, i.e. $s(\mathbf{X}) \geq$ *minsup* [24]. Association rule is an implication in the form of $X \xrightarrow{c} Y$, where $\mathbf{X}$ and $\mathbf{Y}$ are frequent itemsets that disjoint, i.e. $\mathbf{X} \cap \mathbf{Y} = \varnothing$, and c, the confidence factor of the rule, is the conditional probability that a transaction contains $\mathbf{Y}$, given that it contains $\mathbf{X}$, i.e. $c = s(\mathbf{X}\cup\mathbf{Y})/s(\mathbf{X})$. A rule is *confident* if its confidence factor is larger or equal to a user-specified minimum confidence (*minconf*) value, i.e. $c \geq$ *minconf* [24]. A rule $X \xrightarrow{c} Y$ is frequent if the itemset $\mathbf{X}\cup\mathbf{Y}$ is frequent. The association rules mining task can be stated as follows:

> Let $\mathbf{D}$ be a database, *minsup* and *minconf* are the minimum support and the minimum confidence respectively. The mining task tries to discover all *frequent* and *confident* association rules $X \rightarrow Y$, i.e. $s(\mathbf{X}\cup\mathbf{Y}) \geq$ *minsup* and $c(X \rightarrow Y) = s(\mathbf{X}\cup\mathbf{Y})/s(\mathbf{X}) \geq$ *minconf*.

Most of the previously proposed algorithms decompose this mining task into two separated phases [3, 4, 11, 13, 22, 23]: (1) finding all possible frequent itemsets and (2) generating all possible frequent and confident rules from frequent itemsets.

3. DATA DISCRETIZATION BASED ON FUZZY SETS

3.1. Traditional methods of data discretization

Binary association rules mining algorithms [11, 13, 23, 24] work with databases containing

Table 1. Diagnostic database of heart disease

Age	**Sex**	**Chest pain type (1,2,3,4)**	**Serum cholesterol (mg/ml)**	**Fasting blood sugar (>120mg/ml)**	**Resting electrocardio-graphics (0,1,2)**	**Maximum heart rate**	**Heart disease**
60	1 (f)	4	206	0(<120mg/ml)	2	132	2 (yes)
29	0 (m)	3	274	1(>120mg/ml)	2	150	2
54	1	3	273	0	2	152	1 (no)

only binary attributes. Hence, they cannot be directly applied to practical databases as shown in table 1. In order to conquer this obstacle, quantitative and categorical columns must first be converted into binary ones [22, 26]. *The first case*: let A be a discrete quantitative or categorical attribute with finite value domain $\{v_1, v_2, \ldots, v_k\}$ and k is small enough ($k < 20$). After being discretized, the original attribute is developed into k new binary attributes named $A\_V_1, A\_V_2, \ldots A\_V_k$. Value of a record at column A_V_i is equal to *True* (Yes or 1) if the original value of this record at attribute A is equal to v_i, and equal to *False* (No or 0) otherwise. The attributes *Chest pain type* and *Resting electrocardiographics* in table 1 belong to this case. *The second case*: if A is a continuous and quantitative attribute or a categorical one having value domain $\{v_1, v_2, \ldots, v_p\}$ (p is relatively large). A will be mapped to q new binary columns in the form of $\langle A$: start$_1$..end$_1\rangle, \langle A$: start$_2$..end$_2\rangle, \ldots, \langle A$: start$_q$..end$_q\rangle$. Value of a given record at column $\langle A$: start$_i$..end$_i\rangle$ is *True* (Yes or 1) if the original value v at this record of A is between $start_i$ and end_i, $\langle A$: start$_i$..end$_i\rangle$ will receive *False* (No or 0) value for vice versa. The attributes *Age*, *Serum cholesterol*, and *Maximum heart rate* in table 1 belong to this form.

Unfortunately, the mentioned discretization methods encounter some pitfalls such as "sharp boundary problem" [3, 5]. The figure below indicates the support distribution of an attribute A having the value domain ranging from 1 to 10. Supposing that we divide A into two separated intervals [1..5] and [6..10] respectively. If the *minsup* value is 41%, the range [6..10] will not gain sufficient support. Therefore [6..10] cannot satisfy *minsup* ($40\% < minsup = 41\%$) even though there is a large support near its left boundary. For example, [4..7] has support 55%, [5..8] has support 45%. So, this partition results in a "sharp boundary" between 5 and 6, and therefore mining algorithms cannot generate confident rules involving the interval [6..10].

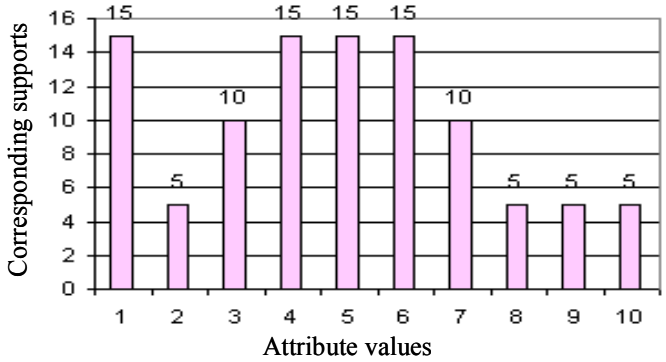


Figure 1. "Sharp boundary problem"

Another disadvantage is that partitioning value domain into separated ranges results in a problem in rule interpretation. Supposing that the range [1..29] denotes young people, [30..59] for middle-aged people, and [60..120] for old ones, so the age of 59 implies a middle-aged person whereas the age of 60 implies an old person. This is not intuitive and natural in understanding the meaning of quantitative association rules. Fuzzy association rule was recommended to overcome the above shortcomings [3, 5]. This kind of rule not only successfully improves "sharp boundary problem" but also help us to express association rules in a more intuitive and a friendly format.

3.2. Data discretization using fuzzy sets

In the fuzzy set theory [12, 28], an element can belongs to a set with a membership value in [0, 1]. This value is assigned by the membership function associated with each fuzzy set. For attribute x and its domain D_x (also known as universal set), the mapping of the membership function associated with fuzzy set f_x is as follow:

$$m_{f_x}(x) : D_x \rightarrow [0, 1] \tag{3.1}$$

The fuzzy set provides a smooth change over the boundaries and allows us to express association rules in a more expressive form. Let's use the fuzzy set in data discretizing to make the most of its benefits. For xample, for the attribute *Age* and its universal domain [0, 120], we attach with it three fuzzy sets *Age_Young*, *Age_Middle-aged*, and *Age_Old*. The graphic representations of these fuzzy sets are shown in the following figure.

Figure 2. Membership functions of fuzzy sets associated with "Age" attribute

3.3. Data discretization using fuzzy sets can bring the following benefits

Firstly, smooth transition of membership functions should help us eliminate the "sharp boundary problem". Besides, fuzzy association rule is more intuitive, and natural than known ones. Also, data discretization by using fuzzy sets assists us significantly reduce the number of new attributes because number of fuzzy sets associated with each original attribute is relatively small comparing to that of an attribute in quantitative association rules. For instance, if we use normal discretization methods over attribute *Serum cholesterol*, we will obtain five sub-ranges (also five new attributes) from its original domain [100, 600], whereas we will create only two new attributes *Cholesterol_Low* and *Cholesterol_High* by applying fuzzy sets. This advantage is very essential because it allows us to compact the set of candidate itemsets, and therefore shortening the total mining time. Moreover, all values of records at fuzzy attributes are in [0, 1]. As a result, this offers an exact method to measure the contribution or the impact of each record to the overall support of an itemset. The final advantage, that we will see more

clearly in the next section, is fuzzified databases still hold "downward closure property" if we have a wise choice for T-norm operator. Thus, conventional algorithms such as Apriori also work well upon fuzzified databases with just slight modifications.

4. MINING FUZZY ASSOCIATION RULES

Table 2. Diagnostic database about heart disease of 4 patients

Age	**Serum cholesterol (mg/ml)**	**Fasting blood sugar (>120mg/ml)**	**Heart disease**
60	206	0 (<120mg/ml)	2 (yes)
52	255	0	2
68	274	1 (>120mg/ml)	2
54	288	1	1 (no)

Let $\mathbf{D}$ be a relational database, $\mathbf{I} = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_n\}$ be a set of n attributes, denoting that i_u is the u^{th} attribute in $\mathbf{I}$. And $\mathbf{T} = \{\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_m\}$ is a set of m records, and t_v is the v^{th} record in $\mathbf{T}$. The value of record t_v at attribute i_u can be refered to as $t_v[i_u]$. For instance, in the table 2, the value of $t_3[i_2]$ (also the value of $t_3[Serum\ cholesterol]$) is 274 (mg/ml). Using fuzzification method in the previous section, we associate each attribute i_u with a set of fuzzy sets:

$$F_{i_u} = \left\{ f_{i_u}^1, f_{i_u}^2, ..., f_{i_u}^k \right\}$$

For example, with the database in table 2, we have: $F_{Age} = \{Age\_Young, Age\_Middle\text{-}aged, Age\_Old\}$

A fuzzy association rule stated in [3, 5] is an implication in the form of:

$$\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B} \tag{4.1}$$

Where:

- $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{I}$ are itemsets. $\mathbf{X} = \{x_1, x_2, \ldots, x_p\}$ and $\mathbf{Y} = \{y_1, y_2, \ldots, y_q\}$.
- $\mathbf{A} = \{f_{x1}, f_{x2}, \ldots, f_{xp}\}$, $\mathbf{B} = \{f_{y1}, f_{y2}, \ldots, f_{yq}\}$ are sets of fuzzy sets corresponding to attributes in $\mathbf{X}$ and $\mathbf{Y}$, $f_{xi} \in F_{xi}$ v ah $f_{yj} \in F_{yj}$.

We can rewrite the fuzzy association rules as two following forms:

$$\mathbf{X} = \{x_1, \ldots, x_p\} \text{ is } \mathbf{A} = \{f_{x1}, \ldots, f_{xp}\} \Rightarrow \mathbf{Y} = \{y_1, \ldots, y_q\} \text{ is } \mathbf{B} = \{f_{y1}, \ldots, f_{yq}\} \tag{4.2}$$

or

$$(x_1 \text{ is } f_{x1}) \text{ AND } \ldots \text{ AND } (x_p \text{ is } f_{xp}) \Rightarrow (y_1 \text{ is } f_{y1}) \text{ AND } \ldots \text{ AND } (y_q \text{ is } f_{yq}) \tag{4.3}$$

A fuzzy itemset is now defined as a pair $\langle \mathbf{X}, \mathbf{A} \rangle$, in which $\mathbf{X}$ ($\subseteq \mathbf{I}$) is an itemset and $\mathbf{A}$ is a set of fuzzy sets associated with attributes in $\mathbf{X}$. The support of a fuzzy itemset $\langle \mathbf{X}, \mathbf{A} \rangle$ is denoted $fs(\langle \mathbf{X}, \mathbf{A} \rangle)$ and determined by the following formula:

$$fs(\langle X, A \rangle) = \frac{\sum_{v=1}^{m} \left\{ \alpha_{x_1}(t_v[x_1]) \otimes \alpha_{x_2}(t_v[x_2]) \otimes ... \otimes \alpha_{x_p}(t_v[x_p]) \right\}}{|T|} \tag{4.4}$$

Where:

- $\mathbf{X} = \{x_1, \ldots, x_p\}$ and $t_v$is the v^{th} record in $\mathbf{T}$.
- $\otimes$ is the T-norm operator in fuzzy logic theory. Its role is similar to that of logic operator AND in traditional logic.
- $\alpha_{x_u}(t_v[x_u])$is calculated as:

$$\alpha_{x_u}(t_v[x_u]) = \begin{cases} m_{x_u}(t_v[x_u]) & if\ m_{x_u}(t_v[x_u]) \geq w_{x_u} \\ 0 & if\ vice\ versa \end{cases} \tag{4.5}$$

  $m_{x_u}$is the membership function of fuzzy set f_{x_u} associated with x_u, and $w_{x_u}$is a threshold of membership function m_{x_u} and specified by users.
- $|\mathbf{T}|$ (card of $\mathbf{T}$) is the total number of records in $\mathbf{T}$ (also equal to m).

A frequent fuzzy itemset: a fuzzy itemset $\langle \mathbf{X}, \mathbf{A} \rangle$ is frequent if its support is greater or equal to a fuzzy minimum support (*fminsup*) specified by users, i.e. $fs(\langle \mathbf{X}, \mathbf{A} \rangle) \geq fminsup$. The support of a fuzzy association rule is defined as:

$$fs(\langle \mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B} \rangle) = fs(\langle \mathbf{X} \cup \mathbf{Y}, \mathbf{A} \cup \mathbf{B} \rangle) \tag{4.6}$$

A fuzzy association rule is frequent if its support is larger or equal to *fminsup*, i.e. $fs(\langle \mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B} \rangle) \geq fminsup$. Confidence factor of a fuzzy association rule is denoted $fc(\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B})$ and defined as:

$$fc(\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B}) = fs(\langle \mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B} \rangle) / fs(\langle \mathbf{X}, \mathbf{A} \rangle) \tag{4.7}$$

A fuzzy association rule is considered frequent if its confidence greater or equal to a fuzzy minimum confidence (*fminconf*) threshold specified by users. This means that the confidence must satisfy the condition: $fc(\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B}) \geq fminconf$.

T-norm operator ($\otimes$): there are various ways to choose T-norm operator [1, 2, 12, 28] for formula (3.6) such as: (1) min function ($a \otimes b = min(a, b)$); (2) normal multiplication ($a \otimes b = a.b$); (3) limited multiplication ($a \otimes b = max(0, a + b - 1)$); (4) drastic multiplication ($a \otimes b = a(if\ b = 1), = b(if\ a = 1), = 0(if\ a, b < 1)$); etc.

Based on experiments, we see that the *normal multiplication* is the most preferable choice for T-norm operator because they are convenient to calculate support factors as well as can highlight the logical relations among fuzzy attributes in frequent fuzzy itemsets. The following formula (4.8) is derived from the formula (4.4) by applying the *normal multiplication*.

$$fs(\langle X, A \rangle) = \frac{\sum_{v=1}^{m} \prod_{x_u \in X} \{\alpha_{x_u}(t_v[x_u])\}}{|T|}$$

Algorithm for mining fuzzy association rules:

The inputs of the algorithm are a database $\mathbf{D}$ with attribute set $\mathbf{I}$ and record set $\mathbf{T}$, and *fminsup* as well as *fminconf*. The outputs of the algorithm are all possible confident fuzzy association rules.

The algorithm in table 3 uses the following sub-programs:

- $(\mathbf{D}_F, \mathbf{I}_F, \mathbf{T}_F)$ = **FuzzyMaterialization**($\mathbf{D}$, $\mathbf{I}$, $\mathbf{T}$): this function is to convert the original database $\mathbf{D}$ into the fuzzified database $\mathbf{D}_F$. Afterwards, $\mathbf{I}$ and $\mathbf{T}$ are also transformed to $\mathbf{I}_F$ and $\mathbf{T}_F$ respectively. In addition, the function **FuzzyMaterialization** also converts $\mathbf{T}$ into $\mathbf{T}_F$.

Table 3. Algorithm for mining fuzzy association rules

```
1    BEGIN
2    (D_F, I_F, T_F) = FuzzyMaterialization(D, I, T);
3    F_1 = Counting(D_F, I_F, T_F, fminsup);
4    k = 2;
5    while (F_{k-1} ≠ ∅) {
6        C_k = Join(F_{k-1});
7        C_k = Prune(C_k);
8        F_k = Checking(C_k, D_F, fminsup);
9        F = F ∪ F_k;
10       k = k + 1;
11   }
12   GenerateRules(F, fminconf);
13   END
```

- $\mathbf{F}_1$ = **Counting**($\mathbf{D}_F$, $\mathbf{I}_F$, $\mathbf{T}_F$, ***fminsup***): this function is to generate $\mathbf{F}_1$, that is set of all frequent fuzzy 1-itemsets. All elements in $\mathbf{F}_1$ must have supports greater or equal to *fminsup*.

- $\mathbf{C}_k$ = **Join**($\mathbf{F}_{k-1}$): this function is to produce the set of all fuzzy candidate k-itemsets ($\mathbf{C}_k$) based on the set of frequent fuzzy $(k-1)$-itemsets ($\mathbf{F}_{k-1}$) discovered in the previous step. The following SQL statement indicates how elements in $\mathbf{F}_{k-1}$ are combined to form candidate k-itemsets.

  INSERT INTO C_k
  SELECT $p.i_1, p.i_2, \ldots, p.i_{k-1}, q.i_{k-1}$
  FROM $\mathrm{L}_{k-1}p$, $\mathrm{L}_{k-1}q$
  WHERE $p.i_1 = q.i_1, \ldots, p.i_{k-2} = q.i_{k-2}, p.i_{k-1} < q.i_{k-1}$ AND $p.i_{k-1}.o \neq q.i_{k-1}.o$;

  In which, $p.i_j$ and $q.i_j$ are index number of j^{th} fuzzy attributes in itemsets p and q respectively. $p.i_j.o$ and $q.i_j.o$ are the index number of original attribute. Two fuzzy attributes sharing a common original attribute must not exist in the same fuzzy itemset.

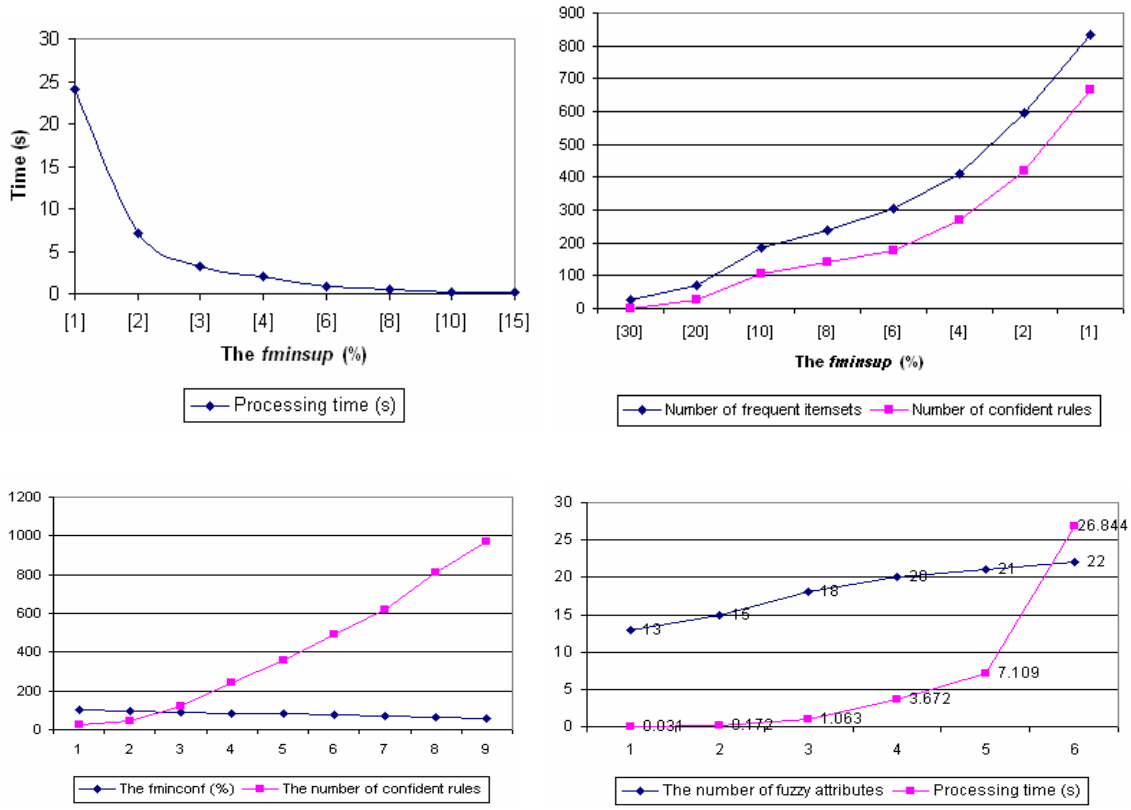
- $\mathbf{C}_k$ = **Prune**($\mathbf{C}_k$): this function helps us to prune any unnecessary candidate k-itemset in $\mathbf{C}_k$ thanks to the downward closure property "*all subsets of a frequent itemset are also frequent, and any superset of a non-frequent itemset will be not frequent*". To evaluate the usefulness of any k-itemset in $\mathbf{C}_k$, the **Prune** function must make sure that all $(k-1)$-subsets of $\mathbf{C}_k$ are present in $\mathbf{F}_{k-1}$.

- $\mathbf{F}_k$ = **Checking**($\mathbf{C}_k$, $\mathbf{D}_F$, ***fminsup***): this function first scans over the whole records or transactions in the datatabase to update support factors for candidate itemsets in $\mathbf{C}_k$. Afterwards, **Checking** eliminates any infrequent candidate itemset, i.e. whose support is smaller than ***fminsup***. All frequent itemsets are retained and put into $\mathbf{F}_k$.

- **GenerateRules**(**F**, ***fminconf***): this function generates all possible confident fuzzy association rules from the set of all frequent fuzzy itemsets **F**.

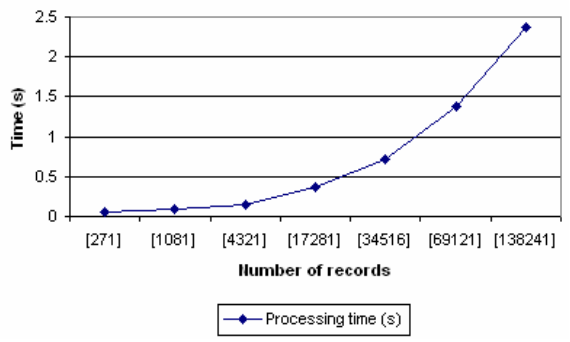
Convert fuzzy association rule into quantitative one: according to the formula (4.5), the member-ship function of each fuzzy set f is attached with a cut w_f. Based on this threshold, we can defuzzify to convert association rule into another form similar to quantitative one. For example, the fuzzy rule "*Old people* $\Rightarrow$ *Blood sugar* $\leq$ *120 mg/ml*, with support 62% and confidence 82%" should be changed to the rule "*Age* $\geq$ *46* $\Rightarrow$ *Blood sugar* $\leq$ *120 mg/ml*, with support 62% and confidence 82%". We see the minimum value of attribute [*Age*, *Age_Old*] that greater or equal to w_{Age_Old} (=0.5) is 0.67. The age corresponding to the fuzzy value 0.67 is 46, so any person whose age is larger or equal to 46 will have fuzzy value greater or equal to 0.67. Therefore, we substitute "*Age_Old*" by "*Age* $\geq$ *46*". Similarly, we can change any fuzzy association rule to quantitative one.

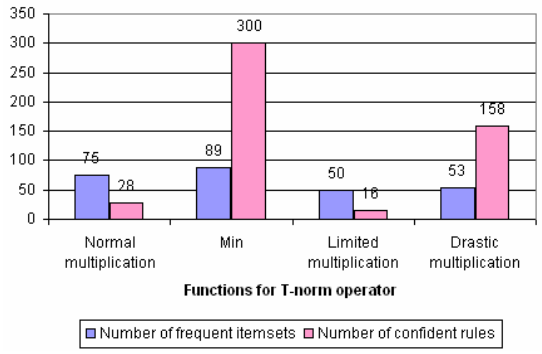
Experiments

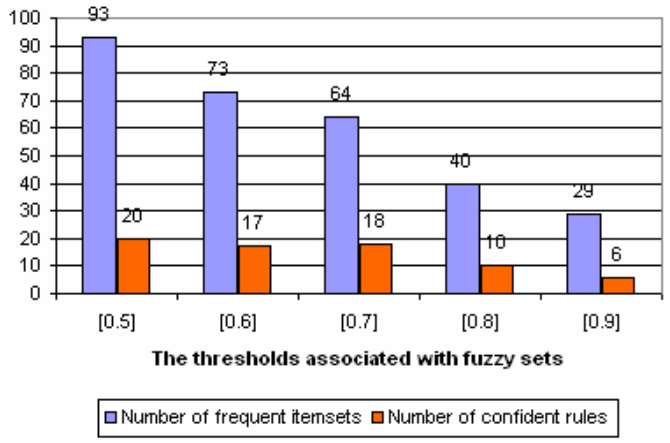


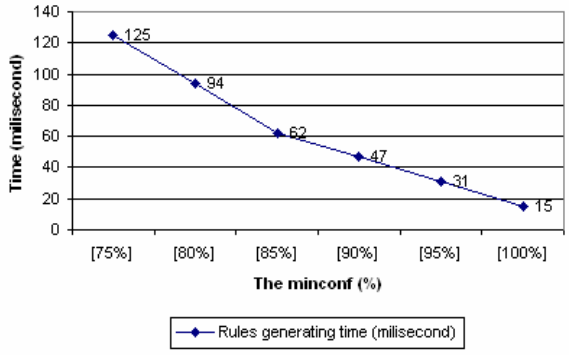
The FuzzyARM tool was developed for the purpose of experiment. It was written in MS Visual C++ language and run on IBM PC Pentium IV, 1.5 GHz, 512 Mb RAM. The testing data are the databases of heart disease diagnosis (created by George John, October 1994, statlog-adm@ncc.up.pt, bob@stams.strathclyde.ac.uk), diabetes disease, auto and vehicle (Drs.Pete Mowforth and Barry Shepherd, Turing Institute George House 36 North Hanover St. Glasgow G1 2AD). The algorithm for mining fuzzy association rules is tested in various aspects such as processing time, number of frequent itemsets and confident rules, the effect of *fminsup* and *fminconf*, the influence of number of records and number of attributes, the

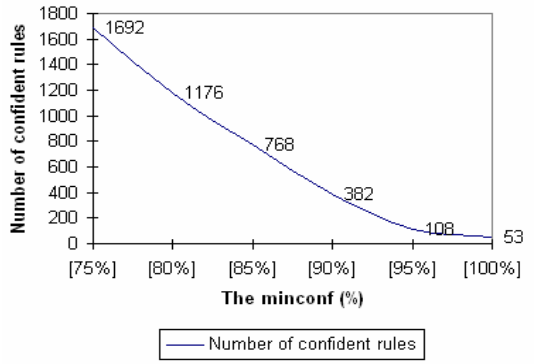
efficiency of each choice for T-norm operator, etc.











5. PARALLEL MINING FOR FUZZY ASSOCIATION RULES

One of the most essential and time-consuming tasks in association rules mining is finding all possible frequent itemsets from immense volumes of data. It needs much CPU time (CPU-bound) and I/O operation (I/O-bound). Thus, researchers have been trying their best to improve the existing algorithms or devise new ones in order to speed up the whole mining process [6, 8, 11, 13, 23]. Most of these algorithms are sequential and work efficiently upon small or medium databases (the sizes of databases are recognized based on their number of

attributes and records). However, they lose their performance and expose some disadvantages while working with extremely large databases (usually hundreds of megabytes or more) due to the limitations in the processor's speed as well as the capacity of internal memory of a single computer.

Fortunately, with the explosive development in hardware industry, high performance computing systems are introduced to the market. This has opened up an opportunity for a new research direction in data mining community. Since 1995, researchers continually devise efficient parallel and distributed algorithms for the issue of association rules mining [4, 7, 10, 15, 19, 20, 22]. These algorithms are diverse because of their tight dependences upon architectures of various parallel computing systems. We would like to recommend a novel parallel algorithm for mining fuzzy association rules. It has been experimented on a Windows-based PC-Cluster system using MPI standard [16–18] and returns optimistic results. This algorithm is relatively optimal because it strongly reduces the data communication and synchronization among processors. However, it can only mine the fuzzy or quantitative association rules as well as suite for relational rather than transactional databases. Almost all known parallel algorithms, more or less, need the data communication and synchronization among processors. This leads to an additional complexity in real implementations of these algorithms. Hence, they are not considered to be "ideal" parallel computing problems. Based on the approach in fuzzy association rules mentioned above, we would like to suggest a new parallel algorithm for mining this kind of rule. It is ideal that little communication needs to be taken place during the processing time. Data communication is made only twice: one at the startup for dividing and delivering fuzzy attributes among processors, and one for rules gathering as the algorithm finishes.

5.1. Our approach

Each fuzzy attribute is a pair of attribute name accompanied by fuzzy set name. For instance, with $\mathbf{I} = \{Age, SerumCholesterol, BloodSugar, HeartDisease\}$, we now have the set of fuzzy attributes $\mathbf{I}_F$ as:

$$\begin{aligned}\mathbf{I}_F = \{&[Age, Age\_Young](1), [Age, Age\_Middle\text{-}aged](2),\\ &[Age, Age\_Old](3), [Cholesterol, Cholesterol\_Low](4),\\ &[Cholesterol, Cholesterol\_High](5), [BloodSugar, BloodSugar\_0](6),\\ &[BloodSugar, BloodSugar\_1](7), [HeartDisease, HeartDisease\_No](8),\\ &[HeartDisease, HeartDisease\_Yes](9)\}.\end{aligned}$$

We totally perceive that any fuzzy association rule (both antecedent and consequent) never contains two fuzzy attributes that share a common original attribute in $\mathbf{I}$. For example, the rule such "*Age_Old* and *Cholesterol_High* and *Age_Young* $\Rightarrow$ *HeartDisease_Yes*" is invalid because it contains both *Age_Old* and *Age_Young* (derived from a common attribute *Age*). There are two chief reasons for the above supposition. First, fuzzy attributes sharing a common original attribute are usually mutually exclusive in meaning so that they will largely decrease the support of rules in which they are contained together. For example, the *Age_Old* is opposite in semantics with *Age_Young* because no person in the world is "both *young* and *old*". Second, such rule is not worthwhile and carries little meaning. Thus, we can conclude that all fuzzy

attributes in the same rule are independent in that there is no pair of fuzzy attribute whose original attribute is identical. This observation is the foundation of our new parallel algorithm.

We will roughly describe our idea via a simple example. Suppose that we will run our algorithm on the database in table 2 and on a 6-processor parallel system. We need to divide the set of fuzzy attributes $\mathbf{I}_F$ among processors so that processors can operate in parallel and independently as follows.

$$\begin{gathered}\text{The processor } \mathbf{P}^1\text{: } \mathbf{I}_F^1 = \{1, 4, 6, 7, 8, 9\}\text{; for } \mathbf{P}^2\text{: } \mathbf{I}_F^2 = \{1, 5, 6, 7, 8, 9\}\text{;} \\ \text{for } \mathbf{P}^3\text{: } \mathbf{I}_F^3 = \{2, 4, 6, 7, 8, 9\}\text{; for } \mathbf{P}^4\text{: } \mathbf{I}_F^4 = \{2, 5, 6, 7, 8, 9\}\text{;} \\ \text{for } \mathbf{P}^5\text{: } \mathbf{I}_F^5 = \{3, 4, 6, 7, 8, 9\}\text{; for } \mathbf{P}^6\text{: } \mathbf{I}_F^6 = \{3, 5, 6, 7, 8, 9\}.\end{gathered}$$

We divide the $\mathbf{I}_F$ based on the first two attributes *Age* and *Cholesterol*. The 9 initial fuzzy attributes are now distributed among 6 processors and each processor receives 6 fuzzy attributes. This division is "ideal" because the number of processors (6) is equal to the multiplication of number fuzzy sets associated with attribute *Age* (3) and number of fuzzy sets associated with attribute *Cholesterol* (2) (i.e. 6 = 3*2). The optimal division is where we could equally disperse fuzzy attributes to all processors in the system. In the case of being unable to obtain an optimal division, we will use "the most reasonable" one. This means that several processors are in idle state while others work hard. I would like to present an algorithm used for fuzzy attributes division. It first tries to find the optimal solution, if not it will return "the most reasonable" one. The division algorithm is formally described below:

Given a database $\mathbf{D}$ with $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ is set of n attributes, and $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ is set of m records. After being fuzzified, $\mathbf{D}$, $\mathbf{I}$, and $\mathbf{T}$ are converted into $\mathbf{D}_F$, $\mathbf{I}_F$, and $\mathbf{T}_F$ respectively.

$$\mathbf{I}_F = \{[i_1, f_{i1}^1], \ldots, [i_1, f_{i1}^{k1}], [i_2, f_{i2}^1], \ldots, [i_2, f_{i2}^{k2}], \ldots, [i_n, f_{in}^1], \ldots, [i_n, f_{in}^{kn}]\}.$$

Where, f_{ij}^u and $k_j$are the u^{th} fuzzy set and the number of fuzzy sets associated with attribute i_j. For example, the database in table 2, we have $\mathbf{I} = \{$*Age*, *SerumCholesterol*, *BloodSugar*, *HeartDisease*$\}$ and after converting we receive the $\mathbf{I}_F$. In this case, $k_1 = 3$, $k_2 = 2$, $k_3 = 2$, $k_4 = 2$ are numbers of fuzzy sets associated with original attributes in $\mathbf{I}$.

Let $\mathbf{FN} = \{k_1\} \cup \{k_2\} \cup \ldots \cup \{k_n\} = \{s_1, s_2, \ldots, s_v\}$ ($v \leq n$ as the may be pairs such as (k_i, k_j) that equal) and $\mathbf{N}$ is the number of processors in the system. The division algorithm is changed to the problem stated as follows:

> *Find the non-empty subset* ***Fn*** *of* ***FN*** *such that the multiplication among elements in* ***Fn*** *is equal to* ***N*** *(this is the optimal solution). In the case of being unable to obtain the optimal solution, the algorithm will return "the most reasonable" solution. This means that the multiplication among elements in* ***Fn*** *is a lower approximation of* ***N****.*

The strategy for searching the optimal solution is that the algorithm must lookup the support counts of frequent 1-itemsets (returned by **Counting** function) during its execution to decide which attributes are suited for being divided. Attributes used to divide should be well balanced among their fuzzy attributes in the terms of support count. This strategy helps maintain load-balancing among processors.

The parallel algorithm:

Inputs: The database **D** with the attribute set **I** and the record set **T,** the number of processors is refered to as **N,** The *minsup* and *minconf* indicate the minimum support and minimum confidence threshold respectively.

Outputs: All possible confident fuzzy association rules.

The parallel algorithm for mining fuzzy association rules includes the following steps:

(1) Converting the database **D**, attribute set **I,** and record set **T** into $\mathbf{D}_F$, $\mathbf{I}_F$, and $\mathbf{T}_F$ respectively. This is fuzzification process. This step is very similar to the function **Fuzzy-Materialization** in the algorithm in the table 3.

(2) Call the function **Counting** in the algorithm in table 3 to count support factor for all 1-itemsets. After this step, only frequent 1-itemsets are retained for subsequent steps.

(3) Using the division algorithm for fuzzy attributes scattering among **N** processors in the system.

(4) Each processor $\mathbf{P}^i$ use the sequential algorithm in table 3 to mine frequent itemsets and the rule generating algorithm to generate confident fuzzy association rules.

(5) Collecting discovered rules from all processors in the system.

Proof of correctness:

The division process is merely recursive. This means that each dividing step is very similar to the previous or next step. For this reason, we need to prove for only one step, and the proof for the general case is reduced to a more simple proof below:

Let $\mathbf{N} = k_1$ be the number of processors in the system and $s = \{k_1\}$ is an optimal solution. After being partitioning, k_1 processors receive:

$\mathbf{I}_F^1 = \{[i_1, f_{i1}^1], [i_2, f_{i2}^1], \ldots, [i_2, f_{i2}^{k2}], \ldots, [i_n, f_{in}^1], \ldots, [i_n, f_{in}^{kn}]\}$ (for processor 1)

$\mathbf{I}_F^2 = \{[i_1, f_{i1}^2], [i_2, f_{i2}^1], \ldots, [i_2, f_{i2}^{k2}], \ldots, [i_n, f_{in}^1], \ldots, [i_n, f_{in}^{kn}]\}$ (for processor 2)

$\vdots$

$\mathbf{I}_F^{k1} = \{[i_1, f_{i1}^{k1}], [i_2, f_{i2}^1], \ldots, [i_2, f_{i2}^{k2}], \ldots, [i_n, f_{in}^1], \ldots, [i_n, f_{in}^{kn}]\}$ (for processor k_1).

We have to prove that the discovered frequent itemsets from parallel algorithm and those from sequential algorithm are the same. This means that any frequent itemset resulted from the sequential algorithm will belong to the set of frequent itemsets returned by the parallel algorithm and vice versa.

Proof:

(1) Obviously, any frequent itemset generated from the parallel algorithm belongs to the set of frequent itemsets generated from the sequential algorithm.

(2) Any frequent itemset generated from the sequential algorithm is classified into $(k_1 + 1)$ categories as: If it contains the fuzzy attribute $[i_1, f_{i1}^1]$, then it will be generated from the processor 1. If containing the fuzzy attribute $[i_1, f_{i1}^2]$, it will be generated from the processor 2, etc. If containing the fuzzy attribute $[i_1, f_{i1}^{k1}]$, it will be generated from the processor k_1. And, if it contain no fuzzy attribute in $\{[i_1, f_{i1}^1], [i_1, f_{i1}^2], \ldots, [i_1, f_{i1}^{k1}]\}$, it will be generated from all k_1 processors in the system. Thereby, we can conclude that any frequent itemset generated from the sequential algorithm also belongs to the set of frequent itemsets returned

by the parallel algorithm.

Computational complexity:

Almost all serial Apriori-like algorithms are classified into NP-complete. This assertion was stated by M. J. Zaki in [14] when he transform the problem of association rules mining into an equivalent issue in bipartite graph: *determining the number of maximal bipartite cliques in a bipartite graph.* Hence, the algorithm shown in table 3 also has a computational complexity of NP-complete. Nevertheless, the result is very encouraging because most of the real world databases are very sparse. The experiments indicate that the time complexity is usually polynomial according to the database size (i.e., number of attributes and number of records).

Supposing that the computational complexity of sequential algorithm for mining association rules is denoted as $\mathbf{C} = f(|\mathbf{I}|, |\mathbf{T}|)$, where $|\mathbf{I}|$ and $|\mathbf{T}|$ are cardinalities of $\mathbf{I}$ and $\mathbf{T}$ respectively. We now try to estimate the time complexity of our new parallel algorithm based on $\mathbf{C} = f(|\mathbf{I}|, |\mathbf{T}|)$.

Assume that our system has $\mathbf{N}$ processors and the partitioning algorithm finds out an optimal solution $s = \{k_1, k_2, \ldots, k_m\}$. Not losing the generality, the system initially has only k_1 processors (i.e., $\mathbf{N} = k_1$). The set of frequent 1-itemsets is partitioned equally among processors. As a result, the number of candidates generated during mining time at each processor is equal to $1/k_1$ of that of the serial algorithm. The time complexity will, therefore, be reduced k_1 times comparing to that of the sequential algorithm. In other words, the time complexity, denoted as $\mathbf{PC}$, is $\mathbf{PC} = \mathbf{C} / \mathbf{N} = f(|\mathbf{I}|, |\mathbf{T}|)/k_1$. Similarly reasoning, if the system is now added $(k_2-1)*k_1$ new processors (i.e., $\mathbf{N} = k_1*k_2$). The new computational complexity is $\mathbf{PC} = \mathbf{C}/\mathbf{N} = f(|\mathbf{I}|, |\mathbf{T}|)/ (\mathrm{k_1}^* \mathrm{k_2})$. In general, if the system includes $\mathbf{N} = (k_1*k_2*\ldots*k_m)$ processors, the complexity will be $\mathbf{PC} = \mathbf{C} / \mathbf{N} = f(|\mathbf{I}|, |\mathbf{T}|)/(k_1*k_2*\ldots*k_m)$. In conclusion, the time complexity of our parallel algorithm reduces $\mathbf{N}$ times comparing to that of serial algorithm, where $\mathbf{N}$ is the number of processors in the system.

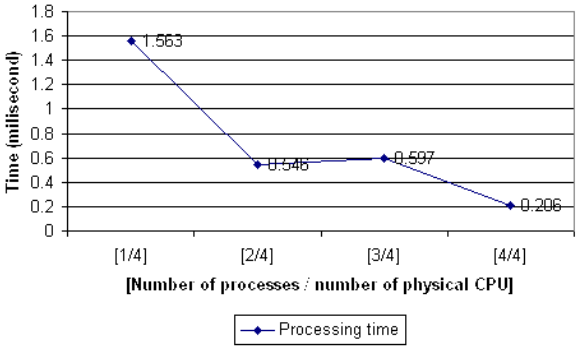
Experiments:

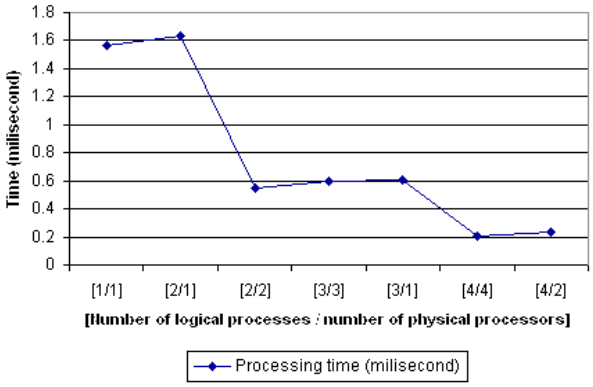
The tool ParallelFARM (Parallel Fuzzy Association Rules Mining) was developed for the purpose of experiment. It was written in MS Visual C++ and run on Windows-based PC-Cluster using MPI standard (Message Passing Interface). Our PC-Cluster includes four Windows-based nodes that run on IBM PC Pentium IV, 1.5 GHz, 512 Mb RAM. The testing data are the databases of heart disease (created by George John, October 1994, statlog-adm@ncc.up.pt, bob@stams.strathclyde.ac.uk), auto and vehicle (Drs.Pete Mowforth and Barry Shepherd, Turing Institute George House 36 North Hanover St. Glasgow G1 2AD), etc.

6. CONCLISION AND FUTURE WORD

The target of section 3 is to deeply study a special kind of rule - the fuzzy association rule. This type of rule is much more flexible and intuitive comparing to the elementary kind of rule described in the previous section. The depiction of this kind of rule in [3,5] is so brief that they could not emphasize the sensitive relation between fuzzy association rules and fuzzy logic theory. The article also explains why we choose the *algebraic multiplication* for

T-norm operator in formula (4.4). In addition, this section restates the algorithm for mining fuzzy association rules in [3,5] based on Apriori with just slight customizations. Finally, this section offers a transformation method for converting fuzzy association rule into quantitative ones. We also suggest a novel parallel algorithm for mining fuzzy association rules. In this algorithm, processors will largely reduce the amount of information need to be communicated during processing time. The algorithm is considered to be "ideal" thanks to its wise strategy in deliver the original set of fuzzy attributes for processors. This division method is both balanced and intelligent in that partitions after dividing are equal and each processor can operate upon its partition independently.





REFERENCES

[1] Phan Dinh Dieu, *Logic in Knowledge Systems,* Faculty of Technology, Hanoi National University, 1999.

[2] Dinh Manh Tuong, *Artificial Intelligence* Faculty of Technology, Hanoi National University, 2003.

[3] Attila Gyenesei, *A Fuzzy Approach for Mining Quantitative Association Rules,* Turku Centre for Computer Science, TUCS Technical Reports, No 336, March 2000.

[4] Andreas Mueller, "Fast sequential and parallel algorithms for association rule mining: A comparison", Department of Computer Science, University of Maryland-College Park, MD 20742.

[5] Chan Man Kuok, Ada Fu, and Man Hon Wong, "Mining fuzzy association rules in databases", Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong.

[6] Doug Burdick, Manuel Calimlim, and Johannes Gehrke, "MAFIA: A maximal frequent itemset algorithm for transactional databases", Department of Computer Science, Cornell University.

[7] Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar, "Scalable Parallel Data Mining for Association Rules", Department of Computer Science, University of Minnesota, 4-192 EECS Building, 200 Union St. SE, Minneapolis, MN 55455, USA - 1997.

[8] Jian Pei, Jiawei Han, and Runying Mao, "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. Intelligent Database Systems Research Lab, School of Computing Science, Simon Fraser University, Burnaby, B. C., Canada."

[9] Jochen Hipp, Ulrich Guntzer, and Gholamreza Nakhaeizadeh. Algorithms for Association Rule Mining – A General Survey and Comparison. *ACM SIGKDD* (2& 1) (2000) 58–64.

[10] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu, Efficient parallel data mining for association rules. *Fourth International Conference on Information and Knowledge Management,* Baltimore, Maryland, Nov 1995.

[11] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient Algorithms for Discovering Association Rules. In *KDD-1994: AAAI Workshop on Knowledge Discovery in Databases,* pages 181-192, Seattle, Washington, July 1994, pages 181-192.

[12] L. A. Zadeh, *Fuzzy sets. Informat. Control* (1965) 338–353.

[13] Mohammed J. Zaki, Ching-Jui Hsiao, CHARM: An Efficient Algorithm for Closed Association Rules Mining, *RPI Technical Report* (1999) 99–10.

[14] Mohammed J. Zaki, Mitsunori Ogihara. Theoretical Foundations of Association Rules. In 3^{rd} *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery,* June 1998.

[15] Mohammed J. Zaki, Srinivasan Parthasarathy, and Mitsunori Ogihara. Parallel Data Mining for Association Rules on Shared-Memory Systems. In *Knowledge and Information Systems,* (3&1) (2001) 1–29.

[16] *MPI: A Message-Passing Interface Standard*, Message Passing Interface Forum. June 12, 1995.

[17] *MPI-2: Extensions to the Message-Passing Interface*, Message Passing Interface Forum, July 18, 1997.

[18] *MPI-2 Journal of Development*, Message Passing Interface Forum, July 18, 1997.

[19] Osmar R. Zaiane, Mohammad El-Hajj, and Paul Lu, "Fast Parallel Association Rule Mining Without Candidacy Generation", University of Alberta, Edmonton, Alberta, Canada.

[20] Qin Ding, William Perrizo, "Using Active Networks in Parallel Mining of Association Rules", Computer Science Department, North Dakota State University, Fargo ND 58105-5164.

[21] Rakesh Agrawal and John Shafer, Parallel mining of association rules: Design, implementation and experience, *Research Report RJ 10004*, IBM Almaden Research Center, San Jose, California, February 1996.

[22] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the* 20^{th} *International Conference on Very Large Databases,* Santiago, Chile, Sep 1994.

[23] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of theACM SIGMOD Conference on Management of Data,* Washington, D. C., May 1993 (207–216).

[24] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami (1993), "Mining association rules between sets of items in large databases", *In Proc. of theACM SIGMOD Conference on Management of Data,* Washington, D. C, pages 207-216,

[25] Ramakrishnan Srikant and Rakesh Agrawal, Mining Quantitative Association Rules in Large Relational Tables, IBM Almaden Research Center, San Jose, CA 95120.

[26] R. J. Miller, Y. Yang, "Association Rules over Interval Data", Department of Computer & Information Science, Ohio State University, USA.

[27] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy,*Advances in Knowledge Discovery and Data Mining*, AAAI Press / The MIT Press 1996.

[28] H. J. Zimmermann, *Fuzzy Set Theory and Its Applications*, Kluwer Academic Publishers, 1991.

Received on September 20 - 2003