Structures of frequent itemsets and classifying structures of association rule set by order relations

Article *in* International Journal of Intelligent Information and Database Systems · January 2014

DOI: 10.1504/IJIIDS.2014.068339

CITATIONS
0

READS
53

3 authors:

Anh Tran
University of Dalat
11 PUBLICATIONS **65** CITATIONS
SEE PROFILE

Tin C Truong
University of Dalat
21 PUBLICATIONS **117** CITATIONS
SEE PROFILE

Bac Le
Ho Chi Minh City University of Science
62 PUBLICATIONS **300** CITATIONS
SEE PROFILE

Structures of frequent itemsets and classifying structures of association rule set by order relations

Anh Tran* and Tin Truong

Department of Mathematics and Computer Science,
University of Dalat,
01 Phu Dong Thien Vuong,
Dalat, Vietnam
Email: anhtn@dlu.edu.vn
Email: tintc@dlu.edu.vn
*Corresponding author

Bac Le

Department of Computer Science,
University of Science,
VNU – Ho Chi Minh,
227 Nguyen Van Cu, District 5,
Ho Chi Minh city, Vietnam
Email: lhbac@fit.hcmus.edu.vn

Abstract: This paper shows a mathematical foundation for almost important features in the problem of discovering knowledge by association rules. The sets of itemsets and association rules are partitioned into disjoint classes by two appropriate equivalence relations based on closures. The structure and unique representation of frequent itemsets are figured out through their generators and corresponding eliminable itemsets. Due to this structure, each equivalence rule class is split into different sets of basic and consequence rules according to an order relation. Indeed, the basic set comprises minimal elements (basic rules) whose forms are explicitly showed. Then, we propose operators to non-repeatedly deduce all consequence rules by adding, deleting or moving appropriate eliminable itemsets in both sides of basic rules. Further, we show that mining association rules based on a new order relation, min relation, is better than four other ones in terms of reductions in the time to extract basic rules, their cardinalities and rule lengths. These theoretical results are proven to be reliable. Experimental study on many benchmark databases shows the efficiency of the corresponding algorithms. Our approach (e.g., the partitions of the frequent itemset and association rule sets) is suitable to deal with big data because it can be exploited in parallel and distributed environment.

Keywords: association rule; basic rule; generator; equivalence relation; order relation; frequent itemset; intelligent information.

Reference to this paper should be made as follows: Tran, A., Truong, T. and Le, B. (2014) 'Structures of frequent itemsets and classifying structures of association rule set by order relations', *Int. J. Intelligent Information and Database Systems*, Vol. 8, No. 4, pp.295–323.

Biographical notes: Anh Tran is currently a PhD student at the Department of Mathematics and Computer Science, University of Dalat, Dalat, Vietnam. He received his BSc in 1999 and MSc in 2004. His interesting researches include data mining and artificial intelligence.

Tin Truong is with the Department of Mathematics and Computer Science, University of Dalat, Dalat, Vietnam. He received his BSc in 1983 and PhD in Stochastic Optimal Control in 1990. His interesting researches are in artificial intelligence, data mining and stochastic analysis.

Bac Le received his BSc in 1984, MSc in 1990 and PhD in Computer Science in 1999. He is an Associate Professor and works at the Department of Computer Science, University of Science, VNU-Ho Chi Minh, Ho Chi Minh City, Vietnam. His interesting researches include artificial intelligence, soft computing and knowledge discovery and data mining.

This paper is a revised and expanded version of a paper entitled 'Structures of association rule set' presented at ACIIDS 2012, Taiwan, 20 March 2012.

1 Introduction

The problem of discovering association rules is one of the important problems in data mining, and has been widely used in various domains such as market business, risk management, etc. An example of association rule extracted from a dataset of supermarket is: "milk and bread → egg (support = 80%, confidence = 90%)". This rule states that the proportion of customers who buy milk, bread and eggs is 80% and the proportion of customers who buy egg among those who bought milk and bread is 90%. This rule can be used to rearrange the way of how to place the foods such as: milk, bread, eggs. It can be used to help the store manager to make market strategies such as: by promotion of milk and bread, it can blow up the sales of the eggs.

The problem is stated as follows: Given a transaction database, the task is to determine all association rules that satisfy two user-defined minimum support and confidence thresholds. Researchers usually decompose it into two sub-problems. One is to extract frequent itemsets, of which the occurrences are greater than or equal to minimum support, in the data. The other is to exploit association rules from those frequent itemsets with the given minimum confidence.

The algorithms for mining frequent itemsets can be divided into three categories. The first comprises the well-known Apriori property-based algorithms, such as AIS (Agrawal et al., 1993) and Apriori (Agrawal and Srikant, 1994). The second one (Han et al., 2000; Das et al., 2001) implements the search over FP tree or the similar structures. The last one mines frequent itemsets on vertical data format, e.g., Eclat (Zaki, 2000). However, the cardinality of frequent itemset class $\mathcal{FS}$ is usually enormous (Bayardo, 1998), especially if the minimum support threshold is small. Thus, the process of finding frequent itemsets usually produces an exponential number of frequent itemsets. As an evident consequence, the set $\mathcal{ARS}$ of association rules grows to be unwieldy. The traditional algorithms, for instances, Apriori (Agrawal and Srikant, 1994), Ap-genrules (Agrawal and Srikant, 1994), to determine frequent itemsets and association rules usually

result in a lot of wasteful computation since they not only check many necessary conditions but also generate a large number of redundant candidates.

One of the solution to overcome disadvantages mentioned in previous paragraph is to use maximal frequent itemsets [see in Bayardo (1998) and Burdick et al. (2001)], which are typically orders of magnitude much fewer than frequent itemsets. While, the class of all maximal frequent itemsets is a condensed representation of the frequent itemset class, they lead to the loss of the supports. More details, though all the sub-itemsets of a maximal itemsets are frequent but we do not know the actual support of those sub-itemsets.

Hence, maximal itemsets are not suitable for association rule generation. In order to avoid those issues, without losing any useful information, mining frequent closed itemsets seems to be the most appropriate approach. The concept of frequent closed itemset comes from the lattice theory (Birkhoff, 1967) and was applied for formal concept analysis (Wille, 1992; Ho, 1995). Besides, when the number of association rules extracted from (dense) databases is very large, the running time and the memory requirement are usually enormous. Moreover, it is too complicated for users to understand and manage a huge number of results. Thus, it is important to obtain a small set of useful association rules that contains much essential information.

1.1 Related work

A common approach is to filter the found rule set via constraints on additional interestingness measures until it becomes manageable (Klemettinen et al., 1994; Srikant et al., 1997; Bayardo and Agrawal, 1999; Bayardo et al., 2000). Other approaches have proposed inference ways to prune the association rules that can be derived from the other rules (Cristofor and Simovici, 2002; Li and Hamilton, 2004).

Recently, a new framework for mining association rules based on frequent closed itemsets and their generators is proposed. Touch (Szathmary et al., 2009) and GenClose (Tran et al., 2013), recently proposed, are two efficient algorithms for mining frequent closed itemsets and their generators. We can also execute a hybrid algorithm, called CharmL-MG, which comprises Charm-L (Zaki and Hsiao, 2005) and Minimal Generator (Zaki, 2004), for doing that. Since the number of closed itemsets and generators is typically orders of magnitude fewer than all frequent itemsets, discovering them can be of a great help to purge many redundant itemsets. Moreover, they do not lose the supports, i.e., the frequent itemsets of the same closure have the same support (see Bastide et al., 2000; Zaki, 2004; Szathmary et al., 2009; Truong and Tran, 2010; Truong et al., 2010; Tran et al., 2012a).

Indeed, some authors proposed mining the smaller and understandable sets of rules. When it is necessary, all remained rules can be derived from them, without accessing the data. Zaki (2004) considered most-general rules that have minimal antecedents and minimal consequents (in terms of subset relation) in the collection of the rules with the same support and confidence. The author also presented how to find these rules. However, his method generates many candidate rules. Furthermore, there is no algorithm to discover the rest rules, some of which can be interesting rules to users (see Truong et al., 2010).

Likewise, using the bases of Duquenne and Guigues (1986) and Luxenburger (1991), Pasquier et al. (2005) introduced ‘minimal’ rules those have minimal antecedents and

maximal consequents in the class of rules with the same support and confidence. They proposed algorithms to mine minimal rules and derive other rules from those minimal rules. However, those algorithms miss some rules and take a lot of time to enumerate redundant rules as well as to delete the repeated ones (as shown in Truong and Tran, 2010).

Based on the lattice of frequent closed itemsets and their generators, Truong and Tran (2010) and Truong et al. (2010) proposed algorithms to enumerate basic rules as minimal and most-general forms. The ones for mining all their consequence rules are also obtained.

1.2 Contribution

To overcome the disadvantages mentioned above, understanding structures of frequent itemsets and association rules is essential. Based on frequent closed itemset lattice, we use two appropriate equivalence relations on $\mathcal{FS}$ and $\mathcal{ARS}$ to partition them into disjoint classes $\mathcal{FS}(L)$ and $\mathcal{AR}(L, S)$, where (L, S) is a pair of nested non-empty frequent closed itemsets. All itemsets in each class have the same closure, so the same support. Similarly, all rules in each rule class have the same support and confidence.

For each class $\mathcal{FS}(L)$, we show that the generators are minimal itemsets according to an appropriate order relation. In order to discover all remained itemsets in the same class, we only need to add appropriate eliminable itemsets, which relate to sets of zero probability measure (Truong and Tran, 2010; Truong et al., 2010), to generators of L. To avoid generating and testing the redundant itemsets in this process, we propose and apply a sufficient condition (related to generators and eliminable itemsets). Then, each itemset has a unique representation.

For each pair (L, S), based on structure of $\mathcal{FS}(L)$ and different order relations on each class $\mathcal{AR}(L, S)$, we show respective explicit representations of different basic rule sets. All rules in each basic set are minimal and all rest consequence rules (non-minimal ones) are sufficiently deduced by adding, deleting or moving appropriate eliminable itemsets in both sides of the basic ones. Based also on the set techniques, some simple sufficient conditions related to the generators and eliminable itemsets of their both sides are checked in order to avoid the duplications. Additionally, experiments figure out that association rule mining based on order relation $\prec_{\min}$ is better than the different ones.

1.3 Organisation

The remainder of this paper is organised as follows. Section 2 reminds some elementary concepts of concept lattice, frequent itemsets, association rules and generators. Section 3 shows the structures of itemsets with the same closure via explicit and unique representations based on their generators and eliminable itemsets. Section 4 classifies structures of association rule set by using different order relations on each equivalence rule class. Sections 5 and 6 show the experimental results and conclusions.

2 Preliminaries

2.1 Concept lattice and probability space

Given non-empty sets $\mathcal{O}$ containing objects (or transactions), $\mathcal{A}$ containing attributes (or items) related to objects $o \in \mathcal{O}$ and $\mathcal{R}$ is a binary relation on $\mathcal{O} \times \mathcal{A}$. A triple $\mathcal{T} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$ is called a data mining context. Consider two set functions:

1 $\lambda: 2^{\mathcal{O}} \to 2^{\mathcal{A}}$

2 $\rho: 2^{\mathcal{A}} \to 2^{\mathcal{O}}$ determined as follows:

$$\forall A \subseteq \mathcal{A}, O \subseteq \mathcal{O} : \lambda(O) = \{a \in \mathcal{A} \mid (o,a) \in \mathcal{R}, \forall o \in O\},$$
$$\rho(A) = \{o \in \mathcal{O} \mid (o,a) \in \mathcal{R}, \forall a \in A\},$$

where $2^{\mathcal{O}}$, $2^{\mathcal{A}}$ are respectively the classes of all subsets of $\mathcal{O}$ and $\mathcal{A}$. Assign that, $\lambda(\varnothing) = \mathcal{A}$, $\rho(\varnothing) = \mathcal{O}$, $h = \lambda o \rho$, $h' = \rho o \lambda$. Thus, $h'(O), h(A)$ are respectively called the closures of O and A. Itemsets A and O are closed iff $h(A) = A$ and $h'(O) = O$. Then, we have $A = \lambda(O)$, $O = \rho(A)$ and the pair $C = (O, A) \in \mathcal{O} \times \mathcal{A}$ is called a concept. In the class of concepts $\mathcal{C} = \{C = (O, A) \in \mathcal{O} \times \mathcal{A}\}$, one define the order relation $\prec$ as relation $\supseteq$ between subsets of $\mathcal{O}$, then $\mathcal{L} \equiv (\mathcal{C}, \prec)$ is the concept lattice [the reader should see Birkhoff (1967), Ho (1950), Wille (1992) for more details].

On $\mathcal{O}$, get the σ-field (Feller, 1950) $\mathcal{F} = 2^{\mathcal{O}}$ including all subsets of $\mathcal{O}$. Let $\mathcal{P}$ be a countable probability measure:

$$\mathcal{P}(O) = |O| / |\mathcal{O}|, \forall O \subseteq \mathcal{O}.$$

We have probability space $(\mathcal{O}, \mathcal{F}, \mathcal{P})$. A set of items, which appears in at least one transaction, is said to be an itemset. The class of all itemsets according to $\mathcal{R}$ is denoted as $\mathcal{IC}$, formerly

$$\mathcal{IC} := \{A \subseteq \mathcal{A} \mid \exists o \in \mathcal{O} : (o,a) \in \mathcal{R}\}.$$

2.2 Frequent itemsets, association rules and generators

Let minsup s_0 and minconf c_0 be the minimum support and minimum confidence respectively. For any itemset $S \in \mathcal{IC}$, the probability $\mathcal{P}(\rho(S)) = |\rho(S)| / |\mathcal{O}|$ is called the support of S, denoted by *supp(S)*. An itemset S is called frequent iff[1] $supp(S) \geq s_0$ (Agrawal et al., 1993). We use $\mathcal{CS}$, $\mathcal{FS}$ and $\mathcal{FCS}$ to denote the classes of all closed itemsets, all frequent itemsets with threshold s_0, and all frequent closed itemsets ($\mathcal{FCS} = \mathcal{CS} \cap \mathcal{FS}$).

For every non-empty, strict subset L from S ($\varnothing \neq L \subset S$), $S \in \mathcal{FS}$, and $R = S \backslash L$ (i.e., $S = L +^2 R$) denote r: $L \to R$ the rule determined by L, R (or by L, S). The conditional probability of $\rho(R)$ given $\rho(L)$:

$$c(r) \equiv \mathcal{P}[\rho(S) \mid \rho(L)] = \mathcal{P}[\rho(L) \cap \rho(R)] / \mathcal{P}(\rho(L)) = |\rho(S)| / |\rho(L)|$$

is called the confidence of r. The rule r is said to be an association rule iff $c(r) \geq c_0$ (Agrawal et al., 1993). Let $\mathcal{ARS}$ be the set of all association rules corresponding with thresholds s_0 and c_0.

When the confidence of association rule $r: L \to R$ is equal to 1, the logical reasoning and the reasoning based on association rule are coincide. In other words, the objects containing itemset L also comprises itemset R.

Proposition 1: We have: $\rho(L) \subseteq \rho(R) \Leftrightarrow c(r: L \to R) = 1$.

Proof: $c(r: L \to R) = \mathcal{P}(\rho(L + R) \mid \rho(L)) = 1 \Leftrightarrow \rho(L + R) = \rho(L) \cap \rho(R) = \rho(L) \Leftrightarrow \rho(L) \subseteq \rho(R)$. □

For two non-empty itemsets G, A: $\varnothing \neq G \subseteq A \subseteq \mathcal{A}$, G is called a generator (Pasquier et al., 2005) of A if $h(G) = h(A)$ and $(\forall G' : \varnothing \neq G' \subset G \Rightarrow h(G') \subset h(G))$. Its other definitions can be found in Bastide et al. (2000), Boulicaut et al. (2003), Zaki (2004). Let $\mathcal{G}(A)$ be the class of all generators of A. They are numbered 1, 2, …: $\mathcal{G}(A) = \{A_i, i \in I = \{1, 2, \ldots, n_A\}, n_A \leq |A|\}$.

To mine closed itemsets and their generators, we can apply CharmL-MG, Touch or GenClose. This paper focuses on the exploration of structures of frequent itemsets and classifying structures of association rule set by different order relations.

3 Structures of itemsets

We consider the structures of the itemsets in general. An equivalence relation, based on the closures of itemsets, is used to partition the class of all itemsets $\mathcal{IC}$ into disjoint equivalence classes. Each class comprises the itemsets with the same support. Therefore, we only need to investigate structure of each class independently. We propose the concept of eliminable itemset and show how to recognise it. Based on sufficient conditions on generators, theorems 2, 3 help us to understand the structures of itemsets with and without constraints. Those structures allow us to design two corresponding efficient algorithms *GFS*, *GCFS* for sufficiently, non-repeatedly generating all itemsets with and without constraint.

3.1 Partition of the class of itemsets by equivalence relation

Definition 1 (Equivalence relation on the itemset class, Truong and Tran (2010)). Closed mapping $h: 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ generates the following binary relation $\sim_{\mathcal{A}}$ on $2^{\mathcal{A}}$: *for any two itemsets* $A, B \subseteq \mathcal{A}$:

$$A \sim_{\mathcal{A}} B \text{ iff } h(A) = h(B).$$

Theorem 1 [Partition of itemset class, Truong and Tran (2010)]. Relation $\sim_{\mathcal{A}}$ is an equivalence relation. It partitions $\mathcal{IC}$ into disjoint equivalence classes. All itemsets in each class have the same closure so the same support. We have:

$$\mathcal{IC} = \sum_{A \in \mathcal{CS}} [A] = \sum_{A \in \mathcal{CS}} \{\varnothing \neq X \subseteq A \mid h(X) = A\}, \mathcal{FS} = \sum_{A \in \mathcal{FCS}} [A]$$

where $[A]$ denotes the equivalence class comprising A and '+' denotes union operator of two disjoint sets.

Proof: $\forall B \in [A]$, $h(B) = h(A) \Leftrightarrow \rho(B) = \rho(A)$. Thus, $supp(B) = supp(A)$. The remained assertions are obvious. □

From this partition, all itemsets can be mined in a parallel and distributed environment.

3.2 Generators and eliminable itemsets

We define a partial order relation on each equivalence class $[A]$.

Definition 2. Consider an order relation $\prec_A$ (on the set $2^A \setminus \{\varnothing\}$) defined as follows:

$$\forall A, B : \varnothing \neq A, B \subseteq \mathcal{A} :$$
$$A \prec_A B \Leftrightarrow A \subseteq B \text{ and } h(A) = h(B).$$

Proposition 2 (Generator is a minimal element with respect to $\prec_A$). Relation $\prec_A$ is partial order relation. For every non-empty itemset $A \subseteq \mathcal{A}$, a non-empty subset $G \subseteq A$ in the equivalence class $[A]$ is a minimal element with respect to $\prec_A$ iff it is a generator of A. Moreover,

$$\mathcal{G}(A) \neq \varnothing.$$

Proof:

- $G \subseteq A$ is a minimal element iff $G \in \mathcal{G}(A)$:

  '$\Rightarrow$': *if* G is a minimal itemset in $[A]$ with respect to $\prec_A$, then $\varnothing \neq G \subseteq A$, $h(G) = h(A)$. If $\varnothing \neq G' \subset G$, then $h(G') \subseteq h(G)$. Since G is a minimal, so if $h(G') = h(G)$, then $G' \equiv G$! Thus, $h(G') \subset h(G)$, i.e., G is a generator of A.

  '$\Leftarrow$': if G is a generator of A and $G' \subset G$, then $h(G') \subset h(G) = h(A)$. Assume that $\exists G' \prec_A G$ and $G' \neq G$, i.e., $G' \subset G$ and $h(G') = h(G) = h(A)$: it is absurd! Thus, G is a minimal itemset in $[A]$.

- $\mathcal{G}(A) \neq \varnothing$: assume that $|A| = m$. Consider subsets of A: $A_{i1} = A \setminus \{a_{i1}\}$, $a_{i1} \in A$, $\forall i_1 = 1..m$.
  a If $h(A_{i1}) \subset h(A)$, $\forall i_1 = 1..$m, then A is its generator.
  b Otherwise, if there exists $i_1 = 1..m$: $h(A_{i1}) = h(A)$, then we repeat this process with A_{ij} ($j = 1..m-1$) until:
  Either case *i)* occurs with some $j \in \{1,\ldots, m-1\}$, then $A_{i,j-1} \in \mathcal{G}(A)$
  Or case *ii)* occurs such that $|A_{i,m-1}| = 1$ and $h(A_{i1}) = h(A_{i2}) = \ldots = h(A_{i,m-1}) = h(A)$, then $A_{i,m-1} \in \mathcal{G}(A)$.

Since A is finite, so this process will finish after a finite number of steps and we always find out a generator G of A. □

Example 1. Consider database $\mathcal{T}$ in Table 1. Figure 1 shows the lattice of closed itemsets discovered from $\mathcal{T}$ and their generators, where: closed itemsets are underlined, their supports are the outside numbers and their generators are italicised. This lattice is used in all examples of the paper. The class $\mathcal{IC}$ of all itemsets containing in $\mathcal{A}$ = {*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*} (briefly $abcdefgh^3$) is partitioned into the disjointed equivalence classes:

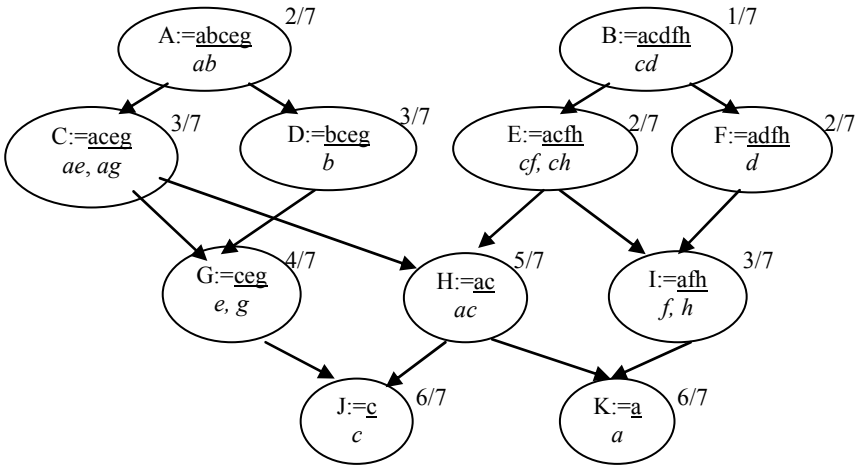
$$\mathcal{IC} = [A]+[B]+[C]+[D]+[E]+[F]+[G]+[H]+[I]+[J]+[K].$$

For minimum support s_0 = 2/7, we have $\mathcal{FS} = \mathcal{IC} \setminus [B]$. The itemsets *ae*, *ag*, *aec*, *aeg*, *agc*, *aceg* have the same closure *C* = *aceg*. We realise that *ae*, *ag* are minimal ones and they are also the generators of *C*, i.e., $\mathcal{G}(C)$ = {*ae*, *ag*}. The reason is that $h(a) = a \subset C$ and $h(e) = h(g) = ceg \subset C$.

Table 1 An example database $\mathcal{T}$

Transactions	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	1	0	1	0	1	0	1	0
2	1	0	1	0	0	1	0	1
3	1	0	0	1	0	1	0	1
4	0	1	1	0	1	0	1	0
5	1	1	1	0	1	0	1	0
6	1	0	1	1	0	1	0	1
7	1	1	1	0	1	0	1	0

Figure 1 The lattice of closed itemsets and corresponding generators mined from $\mathcal{T}$



Definition 3. [Eliminable itemset, Truong and Tran (2010)]. In $2^{\mathcal{A}}$, a subset *R* is called eliminable itemset in *S* iff

$$R \subset S \text{ and } \rho(S \setminus R).$$

Denote the class of all eliminable itemsets in *S* by $\mathcal{N}(S)$ and assign that

$$\mathcal{N}^*(S) := \mathcal{N}(S) \setminus \{\varnothing\}.$$

This concept relates to the sets with the zero probability. We show bellow the criteria for recognising an eliminable itemset is as follows.

Proposition 3. [How to recognise an eliminable itemset, Truong and Tran (2010)]. $\forall\, R \subset S$.

a $$R \in \mathcal{N}(S) \Leftrightarrow \rho(S \setminus R) \subseteq \rho(R) \Leftrightarrow h(S) = h(S \setminus R) \Leftrightarrow supp(S) = supp(S \setminus R) \Leftrightarrow c(r: S \setminus R \to R) = 1 \Leftrightarrow \mathcal{P}(\rho(S \setminus R) \setminus \rho(S)) = 0.$$

b $$\mathcal{N}(S) = \{A : A \subseteq S \setminus G, G \in \mathcal{G}(S)\}.$$

Proof:

a $\forall R \subset S$, since $\rho(S) \subseteq \rho(S\backslash R)$ and $h(S) \supseteq h(S\backslash R)$ so $R \in \mathcal{N}(S) \Leftrightarrow \rho(S) \supseteq \rho(S\backslash R) \Leftrightarrow$ supp(S) = supp($S\backslash R$) $\Leftrightarrow h(S) = h(S\backslash R) \Leftrightarrow \rho(S\backslash R) \subseteq \rho(R) \Leftrightarrow c(r: S\backslash R \to R) = \mathcal{P}(\rho(R) \mid \rho(S\backslash R)) = 1 \Leftrightarrow \mathcal{P}(\rho(S\backslash R) \setminus \rho(S)) = 0$.

b – '$\supseteq$': $\forall A \subseteq S\backslash G$, $G \in G(S)$, we have $h(S) = h(G) \subseteq h(S\backslash A) \subseteq h(S)$. Thus, $h(S\backslash A) = h(S)$. By a), we have $A \in \mathcal{N}(S)$.

– '$\subseteq$': inversely assume that for every generator $G \in \mathcal{G}(S)$, there exists $a \in A \cap G$. Let $G_0 \in \mathcal{G}(S\backslash A)$, then $h(G_0) = h(S\backslash A) = h(S)$ and $G_0 \in \mathcal{G}(S)$. Thus, $a \in A \cap G_0 \subseteq A \cap (S\backslash A) = \varnothing$: It is the contradiction! □

Obviously,

$$\mathcal{N}(S) = \bigcup_{G \in \mathcal{G}(S)} \mathcal{N}(S, G) \tag{1}$$

where

$$\mathcal{N}(S, G) = \{A : A \subseteq S \setminus G\}$$

and

$$\mathcal{N}^*(S, G) = N(S, G) \setminus \{\varnothing\}.$$

For two different generators G_1, $G_2 \in \mathcal{G}(S)$, it could be that

$$\mathcal{N}(S, G_1) \cap \mathcal{N}(S, G_2) \neq \varnothing,$$

i.e., eliminable itemsets generated by (1) could be duplicated! For example, with two generators *ae*, *ag* of *C*, we have

$$\mathcal{N}^*(C, ae) = \{cg, c, g\}, \mathcal{N}^*(C, ag) = \{ce, c, e\}$$

where

$$\mathcal{N}^*(C, ae) \cap \mathcal{N}^*(C, ag) = \{c\} \neq \varnothing$$

and

$$\mathcal{N}^*(C) = \mathcal{N}^*(C, ae) \cup \mathcal{N}^*(C, ag) = \{cg, c, g, e, ce\}.$$

3.3 Representation of itemsets via their generators and eliminable itemsets

This part shows representations and structures of itemsets having the same closure with and without constraint. For any non-empty itemset X: $\varnothing \neq X \subseteq \mathcal{A}$, we denote the equivalence class restricted on X as follows:

$$\begin{aligned} &\lfloor X \rfloor = \{X' \subseteq X \mid X' \neq \varnothing, h(X') = h(X)\} \text{ or} \\ &\lfloor X \rfloor = \{X' \subseteq X \mid X' \in [X] \setminus \{\varnothing\}\}. \end{aligned} \tag{2}$$

Proposition 4. The following statements hold:

a $\lfloor X \rfloor \subseteq [X]$, i.e., all itemsets in $\lfloor X \rfloor$ and $[X]$ have the same support

b $\forall X' \in \lfloor X \rfloor, \mathcal{G}(X') \subseteq \mathcal{G}(X)$

c $X \in \mathcal{CS} \Leftrightarrow \lfloor X \rfloor = [X]$.

d *Closure-preserved property of eliminable itemset:*

$\forall X' = X_0 + Y \in \lfloor X \rfloor$,

where

$X_0 \in \mathcal{G}(X), Y \subseteq X \setminus X_0$,

then:

$X' \setminus X'' \in \lfloor X \rfloor, \forall X'' \subseteq X' \setminus X_0$.

$\forall X' \in \lfloor X \rfloor, X'' \subseteq X \setminus X' \Rightarrow X' + X'' \in \lfloor X \rfloor$,

i.e., when deleting an eliminable itemset X'' from set X' or adding it to X', we do not change its closure and support.

e Non-unique representation of itemsets:

$X' \in \lfloor X \rfloor \Leftrightarrow \exists X_i \in \mathcal{G}(X), X'' \in \mathcal{N}(X, X_i) : X' = X_i + X''$.

Proof:

a $h(X') = h(X) \Leftrightarrow \rho(X') = \rho(X) \Leftrightarrow \text{supp}(X') = \text{supp}(X)$.

b $\forall X_0 \in \mathcal{G}(X'), h(X_0) = h(X') = h(X)$. Further, $\forall X_1 \subset X_0 \Rightarrow h(X_1) \subset h(X_0)$, i.e., $X_0 \in \mathcal{G}(X)$.

c $X \in \mathcal{CS} \Leftrightarrow X = h(X). \forall X' \in [X], X' \subseteq h(X') = h(X) = X$. Thus, $X' \in \lfloor X \rfloor$. $h(X) \in [X] \subseteq \lfloor X \rfloor \Rightarrow X \subseteq h(X) \subseteq X \Rightarrow X \in \mathcal{CS}$.

d Since $X' \setminus X'' = X_0 + (Y \setminus X'') \subseteq X, h(X) = h(X_0) \subseteq h(X' \setminus X'') \subseteq h(X)$, thus $X' \setminus X'' \in \lfloor X \rfloor$. Since $X' + X'' \subseteq X$ and $h(X) = h(X') \subseteq h(X' + X'') \subseteq h(X)$, then $X' + X'' \in \lfloor X \rfloor$.

e '$\Rightarrow$': If $X' \in \lfloor X \rfloor$, there exists $X_i \in G(X')$. Therefore, $h(Xi) = h(X') = h(X)$, so $X_i \in G(X)$. Let $X'' = X' \setminus X_i$. Then $X'' \subseteq X \setminus X_i$, so $X'' \in N(X, X_i)$ and $X' = X_i + X''$.

'$\Leftarrow$': If $X' = X_i + X''$, where $X_i \in G(X), X'' \in N(X, X_i)$, then $X' \subseteq X$ and $h(X) = h(X_i) \subseteq h(X') \subseteq h(X)$, i.e., $X' \in \lfloor X \rfloor$. □

Since an itemset could have different generators, the itemsets coming from Proposition 4.e could be duplicated. For example, let us consider itemset $X = aeg$ with $G(X) = \{ae, ag\}$. It has two following representations: $X = ae + g$ for $g \in N(X, ae) = \{g\}$ and $X = ag + e$ with $e \in N(X, ag) = \{e\}$. To avoid the duplication in the mining of itemsets, it is necessary to find a unique representation for each itemset, i.e., a more explicit structure of itemsets in $\lfloor X \rfloor$. As shown in Section 4, this representation results an efficient method to discover consequence rules.

3.3.1 Representation of itemsets with constraint

For any two itemsets X, $Y \subseteq A$: $Y \neq \emptyset$ and $X \cap Y = \emptyset$, we use $\lfloor Y \rfloor_X$ to denote the class of itemsets Y' containing in Y such that the closure of such itemset Y' with X is identical to the one of the union of Y with X:

$$\lfloor X \rfloor_X := \{Y' \subseteq Y \mid Y' \neq \emptyset, h(X + Y') = h(X + Y)\}. \tag{3}$$

For instance, we have $\lfloor aeg \rfloor_c = \{ae, ag\}$ because $h(ae + c) = h(ag + c) = h(aeg + c) = aegc$ but the closures of itemsets of $eg + c$, $a + c$, $e + c$ and $g + c$ differ from $aegc$. Further, $\lfloor ceg \rfloor_a = \{e, ec, eg, egc, g, gc\}$.

Obviously, if X is empty, (2) and (3) are coincide. Now, we take the differences of X from the generators Z_k of $X + Y$. The class of their minimal elements (according to the set containment relation '$\subseteq$') is called $Y_{\min,X}$:

$$Y_{\min,X} = Minimal\left\{Y_k \equiv Z_k \setminus X \,\middle|\, Z_k \in G(X + Y)\right\}.$$

Let

$$Y_{X,U} = \bigcup_{Y_k \in Y_{\min,X}} Y_k, Y_{X,U,k} = Y_{X,U} \setminus Y_k, Y_{X\_} = Y \setminus Y_{X,U},$$

we denote:

$$IS(Y)_X = \left\{ \begin{array}{l} Y' = Y_K + Y_k^{'} + Y^{\sim} \,\middle|\, Y_k \in Y_{\min,X}, Y^{\sim} \subseteq Y_{X\_}, Y_k^{'} \subseteq Y_{X,U,K} \text{ and} \\ \left(Y_j \not\subset Y_k + Y_k^{'}, \forall Y_j \in Y_{\min,X} : 1 \leq j \leq k\right)^{(A)}, Y' \neq \emptyset \end{array} \right\}. \tag{4}$$

We have the following theorem.

Theorem 2 (Structure and representations of itemsets with constraint). $\forall X, Y \subseteq A$: $\emptyset \neq Y$ and $X \cap Y = \emptyset$:

a Non-unique representation of itemset with constraint X:

$$Y' \in \lfloor Y \rfloor_X \Leftrightarrow \exists Z_0 \in \mathcal{G}(X+Y), Y'' \in \mathcal{N}(X+Y) : Y' = (Z_0 + Y'') \setminus X, Y' \neq \varnothing.$$

b All itemsets of $\mathcal{IS}(Y)_X$ are non-repeatedly generated, i.e., they are uniquely represented.

c $\lfloor Y \rfloor_X = \mathcal{IS}(Y)_X$.

Proof:

a – '$\Rightarrow$': If $Y' \in \lfloor Y \rfloor_X$, then $X + Y' \neq \varnothing$. Let $Z_0 \in \mathcal{G}(X+Y') \subseteq \mathcal{G}(X+Y)$, $Y'' = (X+Y') \setminus Z_0 \subseteq (X+Y) \setminus Z_0$, so $Y'' \in \mathcal{N}(X+Y)$. Thus $Y' = (Z_0 + Y'') \setminus X$.

– '$\Leftarrow$': If $Y' = (Z_0 \setminus X) + (Y'' \setminus X)$, where $Z_0 \in \mathcal{G}(X+Y), Y'' \in \mathcal{N}(X+Y)$, then $Y' \subseteq Y$, because $X \cap Y = \varnothing$. Otherwise, $Z_0 = (Z_0 \cap X) + (Z_0 \setminus X) \subseteq X + Y'$, so $h(X+Y) = h(Z_0) \subseteq h(X+Y') \subseteq h(X+Y)$. Hence $h(X+Y') = h(X+Y)$ and $Y' \in \lfloor Y \rfloor_X$.

b Assume that there exist k, j such that $k > j \geq 1$ and $Y_k + Y_k' + Y_k^{\sim} \equiv Y_j + Y_j' + Y_j^{\sim}, Y_k$, $Y_j \in Y_{\min,X}, Y_k^{\sim}, Y_j^{\sim} \subseteq Y_{X\_}, Y_k' \subseteq Y_{X,U,k}, Y_j' \subseteq Y_{X,U,j}$. Since $Y_j \cap Y_k^{\sim} = \varnothing$, so $Y_j \subset Y_k + Y_k'$ (the equality does not come since Y_k and Y_j are two different minimal itemsets). It contradicts to the selection of the index k! Thus, all itemsets of $\mathcal{IS}(Y)_X$ are totally different.

c – '$\subseteq$': If $Y' \in \lfloor Y \rfloor_X$, then $X + Y' \neq \varnothing$. Let $Z_k \in \mathcal{G}(X+Y') \subseteq (X+Y), Y_k = Z_k \setminus X$, $B \equiv \{Y_j = Z_j \setminus X \mid Z_j \in \mathcal{G}(X+Y')\}$ and $C \equiv \{Y_j = Z_j \backslash X \mid Z_j \in \mathcal{G}(X+Y)\}$, then $Y_k \in B \subseteq C$. We always select the minimum index k such that $Y_k \in Y'_{\min,X} \equiv Minimal(B)$, then $Y_k \in Y_{\min,X}$. Indeed, assume that $Y_k \notin Y_{\min,X}$, then there exists $Y_j \in Y_{\min,X} \equiv$ Minimal(C): $Y_j \subset Y_k$, $Y_j = Z_j \backslash X$ and $Z_j \in \mathcal{G}(X+Y)$. On the other hand, $Z_j \subseteq X + Y_j \subseteq X + Y_k \subseteq X + Y' \subseteq X + Y$, so $h(X+Y) = h(Z_j) = h(X+Y')$, $Z_j \in \mathcal{G}(X+Y')$ and $Y_j \in B$: this is opposite to the minimal definition of Y_k in B. Let $Y_k'' = Y' \setminus Y_k$, since $Y_k \subseteq Y' \subseteq Y$, then $Y' = Y_k + Y_k' + Y^{\sim}$, where $Y_k' = Y_k'' \cap Y_{X,U} \subseteq Y_{X,U,k}, Y^{\sim} = Y_k'' \setminus Y_{X,U} \subseteq Y_{X\_}$. Assume that there exists j such that $1 \leq j < k$, $Y_j \in Y_{\min,X}$ and $Y_j \subset Y_k + Y_k', Y_j = Z_i \setminus X$ and $Z_j \in G(X+Y)$. On the other hand, $Z_j \subseteq X + Y_j \subseteq X + Y_k + Y_k' \subseteq X + Y' \subseteq X + Y$, so $h(X+Y) = h(Z_j) = h(X+Y')$, $Z_j \in \mathcal{G}(X+Y'), Y_i \in B \cap Y_{\min,X}$. Hence $Y_j \in Y'_{\min,X}$ and $j < k$: it contradicts to the selection of the index k! Thus $Y' \in \mathcal{IS}(Y)_X$.

– '$\supseteq$': If $Y' \in \mathcal{IS}(Y)_X$, then there exists $Y_k = Z_k \backslash X \in Y_{\min,X}, Z_k \in \mathcal{G}(X+Y)$, $Y_k' \subseteq Y_{X,U,k}, Y^{\sim} \subseteq Y_{X\_} : Y' = Y_k + Y_k' + Y^{\sim} \subseteq Y$ and $Y' \neq \varnothing$. Since $Y_k \subseteq Y$, so $Y_k' \subseteq Y_{X,U,k} \subseteq Y_{X,U} \subseteq Y, Y^{\sim} \subseteq Y$ and $Y' \subseteq Y$. Moreover, $Z_k \subseteq X + Y_k \subseteq X + Y' \subseteq X + Y$, so $h(X+Y) = h(X_k) = h(X+Y')$. Hence, $Y' \in \lfloor Y \rfloor_X$. □

The first assertion shows non-unique representation of itemset $Y' \in \lfloor Y \rfloor_X$, because Y' could have many generators $Z_0 \in \mathcal{G}(X + Y)$ such that $Z_0 \subseteq Y'$. But, by two last ones, the representation of Y' in $\mathcal{IS}(Y)_X$ is unique. Based on this theorem, we suggest the algorithm *GCFS*, whose pseudo-code is shown in Figure 2, for deriving all itemsets in $\mathcal{FS}(Y)_X = \mathcal{IS}(Y)_X$ where $\text{supp}(X + Y) \geq s_0$, with $X \cap Y = \varnothing$, $S = X + Y$.

Figure 2 The algorithm *GCFS* for non-repeatedly generating frequent itemsets with constraint

Input: $\varnothing \neq X, Y \subseteq \mathcal{A}$: $X \cap Y = \varnothing$, $S = X + Y \in \mathcal{FCS}$, $\mathcal{G}(S)$.
Output: $\mathcal{FS}(Y)_X$.
$\mathcal{FS}(Y)_X$ **GCFS**$(X, Y, \mathcal{G}(S))$
1. $\mathcal{FS}(\text{Y})_\text{X} = \varnothing$;
2. $\text{Y}_{\text{min,X}} = \text{Minimal}\{\text{Y}_\text{k} \equiv \text{Z}_\text{k} \backslash \text{X} \mid \text{Z}_\text{k} \in \mathcal{G}(\text{S})\}$;
3. $\text{Y}_{\text{X,U}} = \bigcup_{\text{Y}_\text{k} \in Y_{\text{min},X}} \text{Y}_\text{k}$; $\text{Y}_{\text{X\_}} = \text{Y} \backslash \text{Y}_{\text{X,U}}$;
4. for each (k=1; $\text{Y}_\text{k} \in \text{Y}_{\text{min,X}}$; k++) do
5. $\quad \text{Y}_{\text{X,k}} = \text{Y}_{\text{X,U}} \backslash \text{Y}_\text{k}$;
6. $\quad$ for each ($\text{Y'}_\text{k} \subseteq \text{Y}_{\text{X,k}}$) do
7. $\quad\quad$ IsDuplicate = false;
8. $\quad\quad$ for (j=1; j<k; j++) do
9. $\quad\quad\quad$ if ($\text{Y}_\text{j} \subset \text{Y}_\text{k} + \text{Y'}_\text{k}$) then
10. $\quad\quad\quad\quad$ IsDuplicate = true;
11. $\quad\quad\quad\quad$ break;
12. $\quad\quad$ if (not(IsDuplicate)) then
13. $\quad\quad\quad$ for each ($\text{Y}^{\sim} \subseteq \text{Y}_{\text{X\_}}$) do
14. $\quad\quad\quad\quad \mathcal{FS}(\text{Y})_\text{X} = \mathcal{FS}(\text{Y})_\text{X} + \{\text{Y}_\text{k} + \text{Y'}_\text{k} + \text{Y}^{\sim}\}$;
15. return $\mathcal{FS}(\text{Y})_\text{X}$;

Example 2. Let us consider the generation of $\mathcal{IS}(aeg)_c$. We have $Y_{\text{min},c} = \{ae, ag\}$, $Y_{c,U} = aeg$, $Y_{c\_} = \varnothing$. For $Y_1 = ae$, we have $Y_{c,U,1} = g$. Since $2^{\{g\}} = \{\varnothing, g\}$, the following itemsets belong to $\mathcal{IS}(aeg)_c$: $ae + \varnothing$, $ae + g$. Now, let us consider $Y_2 = ag$. Because of the fact that $Y_{c,U,2} = e$, the next itemset in $IS(aeg)_c$ is $ag + \varnothing$. This process makes no duplication since we always check the condition (A) in (4). Otherwise, itemset ag+e will be derived again [see Figure 3(a)]. Figure 3(b) illustrates how to non-repeatedly exploit all itemsets of $\mathcal{IS}(ceg)_a = \{e, ec, eg, egc, g, gc\}$, where $Y_{\text{min},a} = \{Y_1 = e, Y_2 = g\}$, $Y_{a,U} = eg$ and $Y_{a\_} = c$.

3.3.2 Representation of itemsets (without constraint)

As a consequence of theorem 2, when $X = \varnothing$, i.e., $\lfloor Y \rfloor = \lfloor Y \rfloor_\varnothing$, $Y_{\text{min},X} = \mathcal{G}(Y)$. Let

$$Y_U = \bigcup_{Y_k \in \mathcal{G}(Y)} Y_k, Y_{U,j} = Y_U \setminus Y_i, Y_{\_} = Y \setminus Y_U,$$

$$IS(Y) = IS(Y)_{\varnothing} = \left\{ \begin{array}{l} Y' = Y_j + Y_j' + Y^{\sim} \,\middle|\, Y_i \in G(Y), Y^{\sim} \subseteq Y_{\_}, Y_j' \subseteq Y_{U,i} \text{ and} \\ \left(Y_k \not\subset Y_i + Y_j', \forall Y_k \in G(Y) : 1 \le k < i\right)^{(B)}, Y' \neq \varnothing \end{array} \right\} \quad (5)$$

and $FS(Y) = IS(Y)$ for $\text{supp}(Y) \ge s_0$. We have the following theorem 3.

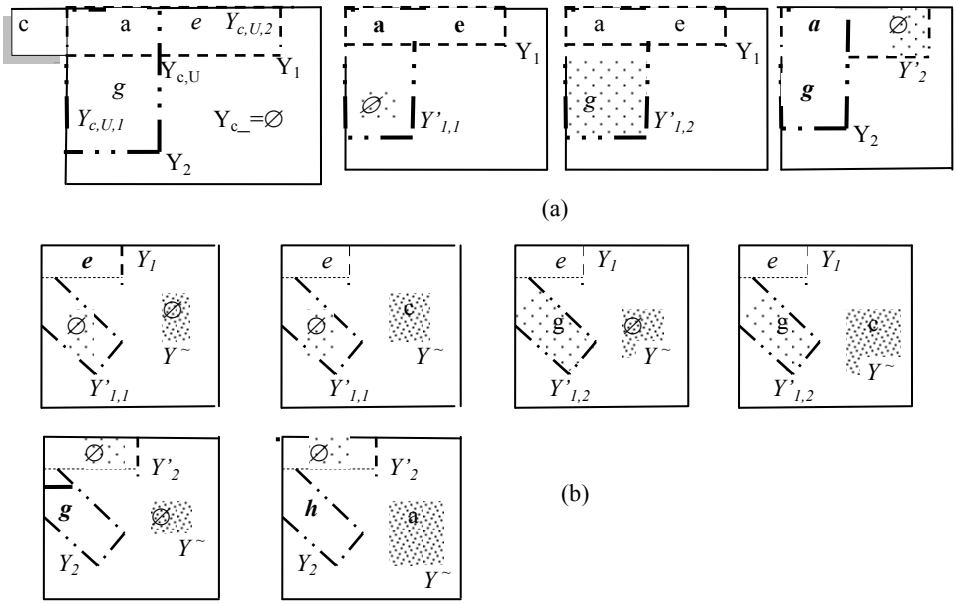
Theorem 3 (Structure and unique representation of itemsets).

a All itemsets of $IS(Y)$ are non-repeatedly generated.

b $\lfloor Y \rfloor = IS(Y)$.

Proof: It is a consequence of theorem 2 for $X = \varnothing$. □

Figure 3 An illustration for the process of generating non-repeatedly the itemset in $IS(Y)_X$, (a) for generating $IS(\text{aeg})_c$ (b) For generating $IS(\text{ceg})_a$ process of generating $IS(\text{ceg})_a$



The theorem shows the explicit structure and the unique representation of all itemsets X' in $\lfloor X \rfloor$ through $IS(X)$. Based on this structure, it is easy to obtain the algorithm $GFS(L, G(L)) \equiv GCFS(\varnothing, L, G(L))$ (see Figure 4) for generating non-repeatedly all frequent itemsets in each equivalence class $[L]$, $\varnothing \neq L \in FCS$.

Example 3. Consider again class $[G]$ in example 1, with $G(G) = \{X_1 = e, X_2 = g\}$. Thus, $X_U = eg$, $X_{U,1} = g$, $X_{U,2} = e$, $X_{\_} = c$. By theorem 3, itemset $X' = ceg \in IS(G)$ is uniquely obtained as follows:

$$X' = X_1 + X_1' + X^{\sim},$$

where

$$X_1' = g \subseteq X_{U,1}, X^{\sim} = c \subseteq X_{\_}.$$

By Proposition 4.e, X' has two duplicate representations:

$$X' = e + gc = g + ec.$$

If the condition (*B*) in (5) is absent, X' is generated again:

$$X' = X_2 + X_2' + X^{\sim},$$

where

$$X_2' = e \subseteq X_{U,2}.$$

Hence, all itemsets in $\lfloor G \rfloor = [G] = \mathcal{IS}(G) = \mathcal{FS}(G) = \{e, ec, eg, egc, g, gc\}$ are non-repeatedly enumerated.

4 Classifying structures of association rule set

We begin this chapter by using an equivalence relation based on the closure *L* of left side and the one *S* of both sides of association rules to partition the association rule set into disjoint equivalence rule classes. Hence, we only independently consider each equivalence class $\mathcal{AR}(L, S)$[4]. Since, the size of each class could be still big, we should take care of the mining of basic sets with the smaller size, compared to the class. When it is necessary, the other rules could be derived from them. Pasquier et al. (2005), Zaki (2004) and Truong and Tran (2010) and Truong et al. (2010) considered the mining task for basic rules in forms of minimal and most general. From general view, one can see that there are different forms of basic rules according to different order relations. Then, we consider five order relations. For a given relation, the basic set of each class comprises minimal elements according to that relation. Based on structures of itemsets proposed in previous section, we describe how to deduce all remained consequence rules non-repeatedly from that basic set.

Figure 4 The algorithm *GFS* to generate frequent itemsets non-repeatedly

Input: $\varnothing \neq L \in \mathcal{FCS}, \mathcal{G}(L)$.
Output: $\mathcal{FS}(L) = [L]$.
$\mathcal{FS}(L)$ **GFS**$(L, \mathcal{G}(L))$
1. $\mathcal{FS}(L) = \varnothing$; $L_U = \bigcup_{L_i \in \mathcal{G}(L)} L_i$; $L\_ = L \backslash L_U$;
2. for each (i=1; $L_i \in \mathcal{G}(L)$; i++) do
3. $L_{U,i} = L_U \backslash L_i$;
4. for each ($L'_i \subseteq L_{U,i}$) do
5. IsDuplicate = false;
6. for (k=1; $L_k \in \mathcal{G}(L)$, k<i; k++) do
7. if ($L_k \subset L_i + L'_i$) then
8. IsDuplicate = true;
9. break;
10. if (not(IsDuplicate)) then
11. for each ($L^{\sim} \subseteq L\_$) do
12. $\mathcal{FS}(L) = \mathcal{FS}(L) + \{L_i + L'_i + L^{\sim}\}$;
13. return $\mathcal{FS}(L)$;

4.1 Partition of association rule set by equivalence relation

Definition 4 [Equivalence relation on association rule set, Truong and Tran (2010)]. Let $\sim_r$ be a binary relation on $\mathcal{ARS}$ defined as follows:

$$\forall L', S', Ls, Ss \subseteq \mathcal{A}, \varnothing \neq L' \subset S', \varnothing \neq Ls \subset Ss, r: L' \to S' \setminus L', s: Ls \to Ss \setminus Ls:$$
$$s \sim_r r \text{ iff } (Ls \in [L'] \text{ and } Ss \in [S']).$$

Let $\mathcal{NFCS}$ be the class containing all pairs of two nested frequent closed itemsets:

$$\mathcal{NFCS} = \{(L, S) \mid \varnothing \neq L \subseteq S : L, S \in \mathcal{FCS}, \text{supp}(S) / \text{supp}(L) \geq c_0\}.$$

It follows from definition 4, we have the following theorem which plays an important role in partitioning association rule set.

Theorem 4 [Partition of the association rule set, Truong and Tran (2010)]. Relation $\sim_r$ is an equivalence relation. It partitions $\mathcal{ARS}$ into disjoint equivalence rule classes $\mathcal{AR}(L, S)$. All rules in each class have the same support and confidence.

$$\mathcal{ARS} = \sum_{(L,S) \in \mathcal{NFCS}} \mathcal{AR}(L, S)$$

Proof: $\forall L', S', Ls, Ss \subseteq \mathcal{A}, \varnothing \neq L' \subset S', \varnothing \neq Ls \subset Ss, r: L' \to S' \setminus L', s: Ls \to Ss \setminus Ls$: $s \sim_r r$, we have $Ls \in [L']$ and $Ss \in [S']$. Then, $\text{supp}(s) = \text{supp}(Ss) = \text{supp}(S') = \text{supp}(r)$ and $c(s) = \text{supp}(s) / \text{supp}(Ls) = \text{supp}(S') / \text{supp}(L') = c(r)$. The remained affirmations are obvious. □

Based on this partition, we can designed the parallel algorithms to mine association rules. This is the strength of the equivalence relations and is a typical instance of the divide-and-conquer method, widely used in computer science. Naturally, $\mathcal{AR}(L, L) = \varnothing$, $\forall L \in \mathcal{FCS}$: $L \in \mathcal{G}(L)$. Hence, for $L \equiv S$, we always suppose that $L \notin \mathcal{G}(L)$.

Example 4. From the lattice in Figure 1, for $s_0 = 2/7$ and minimum confidence $c_0 = 3/6$, we have the partition of association rule set (the support and confidence of the rules are superscripted), given in Figure 5. The rule class $\mathcal{AR}(G, G)$ consists of the rules with the same support 4/7 and confidence 1. The rules $e \to a$, $e \to ac$ $e \to ag$, $e \to acg$, $g \to a$, $g \to ac$, $g \to ae$, $g \to ace$, $ec \to a$, $ec \to ag$, $gc \to a$, $gc \to ae$, $eg \to a$, $eg \to ac$ and $ecg \to a$ belong to the class $\mathcal{AR}(G, C)$. The closures of their left sides are the same – G, the ones of the two-side unions are also the same – C. Hence, they all have support 3/7 and confidence 3/4.

Figure 5 The partition of association rule set $\mathcal{ARS}$ with $s_0 = 2/7$ and $c_0 = 3/6$

$\mathcal{ARS}$

$\mathcal{AR}$(C, C) $^{3/7, 1}$	$\mathcal{AR}$(D, D) $^{3/7, 1}$	$\mathcal{AR}$(A, A) $^{2/7, 1}$	$\mathcal{AR}$(E, E) $^{2/7, 1}$	$\mathcal{AR}$(F, F) $^{2/7, 1}$
$\mathcal{AR}$(G, G) $^{4/7, 1}$	$\mathcal{AR}$(H, H) $^{5/7, 1}$	$\mathcal{AR}$(I, I) $^{3/7, 1}$	$\mathcal{AR}$(J, J) $^{6/7, 1}$	$\mathcal{AR}$(K, K) $^{6/7, 1}$
$\mathcal{AR}$(G, C) $^{3/7, 3/4}$	$\mathcal{AR}$(G, D) $^{3/7, 3/4}$	$\mathcal{AR}$(H, C) $^{3/7, 3/4}$	$\mathcal{AR}$(J, C) $^{3/7, 3/6}$	$\mathcal{AR}$(K, C) $^{3/7, 3/6}$
$\mathcal{AR}$(J, G) $^{3/7, 3/6}$	$\mathcal{AR}$(J, H) $^{5/7, 5/6}$	$\mathcal{AR}$(K, H) $^{5/7, 5/6}$	$\mathcal{AR}$(K, I) $^{3/7, 3/6}$	
$\mathcal{AR}$(C, A) $^{2/7, 2/3}$	$\mathcal{AR}$(D, A) $^{2/7, 2/3}$	$\mathcal{AR}$(G, A) $^{2/7, 2/4}$	$\mathcal{AR}$(I, F) $^{2/7, 2/3}$	

Over each equivalence rule class $\mathcal{AR}(L, S)$, we consider the set of minimal rules $\mathcal{B}_{name}(L, S)$ with respect to an order relation $\prec_{name}$, called basic rule set. Thus, corresponding consequence rule set is denoted as:

$$\mathcal{C}_{name}(L,S) \coloneqq \mathcal{AR}(L,S) \setminus \mathcal{B}_{name}(L,S).$$

It contains all consequence rules that are not minimal rules, i.e., for every consequence rule r_c, there exists a basic rule r_b such that:

$$r_b \prec_{name} r_c.$$

We need to answer two following questions:

1 How to determine quickly basic rules r_b?

2 How to derive non-repeatedly their all consequence rules r_c?

4.2 Basic rules as minimal rules with respect to order relation

We usually expect the basic rules with the sides are minimal or maximal. In fact, Zaki (2004) concentrated on the mining of most-general rules those have minimal antecedents and minimal consequents. Pasquier et al. (2005) considered minimal rules of which, the antecedents are minimal and the consequents are maximal. In the paper, we consider five relations of $\prec_{mM}$ (min-Max), $\prec_{mm}$ (min-min), $\prec_{Mm}$ (Max-min), $\prec_{\mathrm{MM}}$ (Max-Max) and $\prec_{\min}$ (min).

Definition 5. We define five binary relations on $\mathcal{AR}(L, S)$ as follows:

$$\forall r_j : L_j \to R_j \in \mathcal{AR}(L,S), S_j = L_j + R_j, j = 1,2:$$

a $r_1 \prec_{mM} r_2$ iff ($L_1 \subseteq L_2$ and $R_1 \supseteq R_2$)

b $r_1 \prec_{mm} r_2$ iff ($L_1 \subseteq L_2$ and $R_1 \subseteq R_2$)

c $r_1 \prec_{MM} r_2$ iff ($L_1 \supseteq L_2$ and $S_1 \supseteq S_2$)

d $r_1 \prec_{Mm} r_2$ iff ($L_1 \supseteq L_2$ and $R_1 \subseteq R_2$)

e $r_1 \prec_{\min} r_2$ iff ($L_1 \supseteq L_2$ and $S_1 \supseteq S_2$, for $L \subset S$) or ($L_1 \subseteq L_2$ and $R_1 \supseteq R_2$, for $L = S$).

It is easy to see that those relations are partial order relations over $\mathcal{AR}(L, S)$. Theorem 5 shows the explicit forms of the corresponding basic rules.

Theorem 5. For (L, S), we have:

a $\mathcal{B}_{mM}(L, S) = \{r_b: L_i \to S \backslash L_i \mid L_i \in \mathcal{G}(L)\}\}$

$\mathcal{B}_{mM}(L, L) = \{r_b: L_i \to L \backslash L_i \mid L_i \in \mathcal{G}(L)\}$, if $L \notin \mathcal{G}(L)$

b $\mathcal{B}_{mm}(L, S) = \{r_b: L_i \to R_k \mid L_i \in \mathcal{G}(L), R_k \in R_{\min,Li}\}$, where
$R_{\min,Li} = Minimal\{S_k \backslash L_i \mid S_k \in \mathcal{G}(S)\}$

$\mathcal{B}_{mm}(L, L) = \{r_b: L_i \to \{a\} \mid L_i \in \mathcal{G}(L), a \in L \backslash L_i\}$, if $L \notin \mathcal{G}(L)$

c $\mathcal{B}_{MM}(L, S) = \{r_b: L \to S\backslash L\}$

$\mathcal{B}_{MM}(L, L) = \{r_b: L\backslash\{a\} \to \{a\} \mid a \in L\backslash L_i, L_i \in \mathcal{G}(L)\}$, if $L \notin \mathcal{G}(L)$

d $\mathcal{B}_{Mm}(L, S) = \{r_b: L \to R \mid R \in R_{\min,L}\}$, where $R_{\min,L} = Minimal\{R_k = S_k\backslash L \mid S_k \in \mathcal{G}(S)\}$

$\mathcal{B}_{Mm}(L, L) = \{r_b : L \setminus \{a\} \to \{a\} \mid a \in L \setminus L_i, L_i \in \mathcal{G}(L)\}$

$= \{r_b : L \setminus \{a\} \to \{a\} \mid a \notin L_U\}, L_U = \bigcap_{L_i \in \mathcal{G}(L)} L_i$, *if* $L \notin \mathcal{G}(L)$.

e $\mathcal{B}_{\min}(L, S) = \{r_b: L \to S\backslash L\}$

$\mathcal{B}_{\min}(L, L) = \{r_b: L_i \to L\backslash L_i \mid L_i \in \mathcal{G}(L)\}$, if $L \notin \mathcal{G}(L)$.

Proof: To show that $r_b \in \mathcal{B}_{name}(L, S)$ is a minimal rule with respect to order relation $\prec_{name}$, it needs only to prove that if there exists a rule $r_0 \prec r_b$, where $r_0 : L' \to R' \in \mathcal{AR}(L, S)$, then $r_0 \equiv r_b$. Here, we only prove case *e)* according to the min basic rules. The ones are similar.

- In the case $L \subset S$: First, we prove that every minimal rule $r : L' \to R' \in \mathcal{B}_{\min}(L, S)$ is always in form r_b: $L \to S\backslash L$. Indeed, if $L' \subset L$ and $L^{\sim} = (L \setminus (L' \cup R')) \neq \varnothing$, then $r_0 \prec_{\min} r$ and $r_0 \neq r$, where $r_0 : L' + L^{\sim} \to R'$! If $L' \subset L$ and $L^{\sim} = (L \setminus (L' \cup R')) = \varnothing$, then $r_0' \prec_{\min} r$ and $r_0' \neq r$, for $r_0' : L \to R' \setminus L$! Thus, $L' = L$. If $R' \subset S \setminus L$, then $r_1 \prec_{\min} r$ and $r_1 \neq r$, with r_1: $L \to S\backslash L$! Thus $R' = S \setminus L$, i.e., $r \equiv r_b$. Conversely, to show that r_b is a minimal rule with respect to order relation $\prec_{\min}$, we only need to prove that if rule $r_0 \prec_{\min} r_b$, where $r_0 : L' \to R' \in \mathcal{AR}(L, S)$, then $r_0 \equiv r_b$. Indeed, $L' \supseteq L$ and $L' + R' \supseteq S$. Since $L = h(L') \supseteq L' \supseteq L$ and $S = h(L' + R') \supseteq L' + R' \supseteq S$, so $L' \equiv L$ and $R' = S \setminus L$. Hence, $r_0 \equiv r_b$.
- In the case $L \equiv S$: Assume that $L \notin \mathcal{G}(L)$. First, we prove that every minimal rule $r : L' \to R' \in \mathcal{B}_{\min}(L, L)$ is always in form r_b: $L_i \to L\backslash L_i$, where $L_i \in \mathcal{G}(L') \subseteq \mathcal{G}(L)$. Indeed, if $L_i \subset L'$ then $r_0 \prec_{\min} r$ and $r_0 \neq r$, where $r_0 : L_i \to R'$! Thus, $L' = L_i$. If $R' \subset L \setminus L_i$, then $r_1 \prec_{\min} r$ and $r_1 \neq r$, where r_1: $L_i \to L\backslash L_i$! Thus, $R' = S \setminus L_i$, i.e., $r \equiv r_b$. Conversely, assume that there exists $L_i \in \mathcal{G}(L)$, r_b: $L_i \to L\backslash L_i \in \mathcal{B}_{\min}(L, L)$ and $r_0 : L' \to R' \in \mathcal{AR}(L, L)$ such that $r_0 \prec_{\min} r_b$. Then $L' \subseteq L_i$ and $R' \supseteq L \setminus L_i$. Since $L_i \in \mathcal{G}(L), h(L') = L$, thus $L' = L_i, L \setminus L_i = L \setminus L' \supseteq R' \supseteq L \setminus L_i$. Hence $R' = L \setminus L_i$ and $r_0 \equiv r_b$. □

We take care of two following properties of basic sets: the rule length, called L, and the maximum number of rules, called N. The upper estimation of the product $L.N$ is shown in column LN of Table 2. The works related to relations $\prec_{mM}$ and $\prec_{mm}$ have been considered by Pasquier et al. (2005) and Zaki (2004). Overcoming weakness in their results, in Truong and Tran (2010) and Truong et al. (2010), we indicated the better algorithms for quickly mining association rules based on those relations. In the rest of the paper, we

only consider relation $\prec_{\min}$ (the results for two relations of $\prec_{Mm}$ and $\prec_{MM}$ can be obtained in the similar way).

Table 2 An evaluation about the basic sets based on different order relations

Basic sets	*L*	*N*	*LN*
$\mathcal{B}_{mM}$(L, S)	|S|	$\|\mathcal{G}(L)\|$	$\|S\|.\|\mathcal{G}(L)\|$
$\mathcal{B}_{mm}$(L, S)	$\|L_i \cup S_k\|$	$\|\mathcal{G}(L)\|.\|\mathcal{G}(S)\|$	$\|S\|.\|\mathcal{G}(L)\|.\|\mathcal{G}(S)\|$
$\mathcal{B}_{MM}$(L, S)	|S|	$1 \vee \left\|L \setminus \bigcap_{L_i \in \mathcal{G}(L)} L_i\right\|$	$\|S\| \left\|L \setminus \bigcap_{L_i \in \mathcal{G}(L)} L_i\right\|$
$\mathcal{B}_{Mm}$(L, S)	$\|L \cup S_k\|$	$\|\mathcal{G}(S)\|$	$\|S\|.\|\mathcal{G}(S)\|$
$\mathcal{B}_{\min}$(L, S)	|S|	$1 \vee \|\mathcal{G}(L)\|$	$\|S\|.\|\mathcal{G}(L)\|$

Following directly from the explicit form of min basic rules, we obtain the algorithms for mining them, given in Figure 6. For example, we have the min basic set:

$$\mathcal{B}_{\min}(G, C) = \{ceg \to a\}$$

and

$$\mathcal{B}_{\min}(G, G) = \{e \to cg, g \to ce\}.$$

Figure 6 The algorithms of *MINBasicSet1* and *MINBasicSet2* for mining min basic rules

Input: (L, S) $\in \mathcal{NFCS}$*: L* $\subset$ *S.*
Output: $\mathcal{B}_{\min}$*(L, S).*
$\mathcal{B}_{\min}$*(L, S)* **MINBasicSet1** *(L, S)*
1. $\mathcal{B}_{\min}(L, S) = \{L \to S \backslash L\}$;
2. return $\mathcal{B}_{\min}(L, S)$;

Input: $\varnothing \neq L \in \mathcal{FCS}$*,* $\mathcal{G}(L)$*. Output:* $\mathcal{B}_{\min}(L, L)$*.*
$\mathcal{B}_{\min}(L, L)$ **MINBasicSet2** *(L,* $\mathcal{G}(L)$*)*
1. $\mathcal{B}_{\min}(L, L) = \varnothing$;
2. for each ($L_i \in \mathcal{G}(L)$) do
3. $\mathcal{B}_{\min}(L, L) = \mathcal{B}_{\min}(L, L) + \{L_i \to L \backslash L_i\}$;
4. return $\mathcal{B}_{\min}(L, L)$;

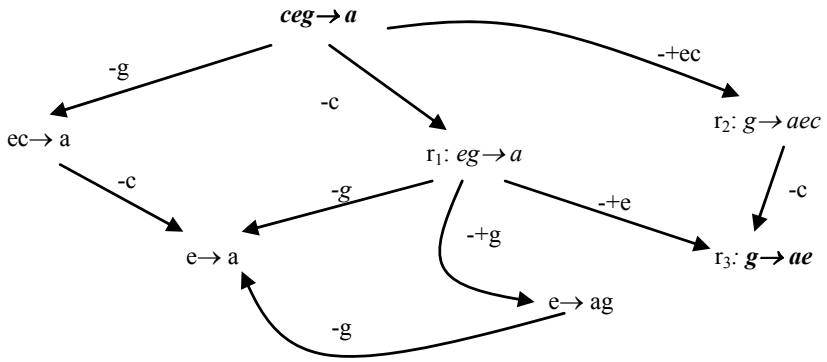
4.3 Deriving non-repeatedly all min consequence rules

How the users deduce all consequence rules in $\mathcal{C}_{\min}(L, S)$ from the basic set $\mathcal{B}_{\min}(L, S)$? Based on closure-preserved property of eliminable itemset (proposition 4d), from basic rules or their consequence rules, formerly r_0, for generating consequence ones that belong to the same class, we only need to delete (–) or to move (–+) eliminable itemsets in both sides of r_0. These operators are used (by users) for generating preserved-confidence consequence rules.

Figure 7 illustrates a way to derive consequence rules from the basic rules. From basic rule r_0: $ceg \to a$, we deduce consequence rule r_1: $eg \to a$ by deleting eliminable itemset c from r_0's left side. Consequence rule r_2: $g \to aec$ is derived by moving itemset ec from r_0's left side to its right side. Rule r_1 deduces r_3: $g \to ae$ and so on. Thus, we have the consequence set $\mathcal{C}_{\min}(G, C) = \{ec \to a, r_1, r_2, e \to a, e \to ag, r_3\}$. However, a consequence rule derived from a rule can be identical to a rule coming from a different

rule. In fact, r_3 is derived in turn from r_1 and r_2. That makes the duplications in the process of deriving consequence rules.

Figure 7 The basic and consequence rules of rule class $\mathcal{AR}(G, C)$



To speed up the mining, we replace the generation of consequence rules from each rule r_0 with the one from each rule set comprising the rules with the same left or right side. As follows, we propose two operators for deleting, or moving appropriate eliminable itemsets on both sides of basic rules for deriving their all preserved-confidence and non-repeated consequence rules in each equivalence rule class $\mathcal{AR}(L, S)$.

Definition 6 (Operators for deriving non-repeatedly all consequence rules).

a For $L \subset S$:

- Deleting eliminable itemsets in two sides of basic rules:

$$\mathcal{B}_{\min-L-R}(L,S) = \left\{ \begin{matrix} r_c : L' \to R' \mid R' \in \mathcal{FS}(S \setminus L)_L, L' \in \mathcal{FS}(L) \text{ and} \\ (L' \subset L \text{ or } R' \subset S \setminus L) \end{matrix} \right\}.$$

- Moving eliminable itemsets from left-hand side to right side:

$$\mathcal{B}_{\min-L+R}(L,S) = \left\{ \begin{matrix} r_c : L'' \to R' + (L' \setminus L'') \mid L' \in \mathcal{FS}(L), R' \in \mathcal{FS}(S \setminus L)_L, \\ L'' \in FS(L') \setminus \{L'\} \end{matrix} \right\}$$

b For $L = S$:

- Deleting eliminable itemsets in right side of basic rules:

$$\mathcal{B}_{\min-R}(L,L) = \{ r_c : L_i \to R' \mid \varnothing \neq R' \subset L \setminus L_i, L_i \in \mathcal{G}(L) \}$$

- Moving eliminable itemsets from right side to left side:

$$\mathcal{B}_{\min-R+L}(L,L) = \left\{ \begin{matrix} r_c : L_i + R'' \to R' \setminus R'' \mid L_i \in \mathcal{G}(L), \varnothing \neq R' \subseteq L \setminus L_i, \\ \varnothing \neq R'' \subset R', i = 1 \text{ or } (i > 1 : L_k \not\subset L_i + R'', \forall k : 1 \leq k < i \end{matrix} \right\}.$$

Theorem 6 (Structure and representation of min consequence set).

a In the case $L \subset S$:

1 all rules in each set $\mathcal{B}_{\text{min-}L\text{-}R}(L, S)$, $\mathcal{B}_{\text{min-}L\text{+}R}(L, S)$ are non-repeatedly generated

2 all rules in $\mathcal{B}_{\text{min-}L\text{-}R}(L, S)$, $\mathcal{B}_{\text{min-}L+R}(L, S)$, $\mathcal{B}_{\text{min}}(L, S)$ are totally different

3 $\mathcal{C}_{\text{min}}(L, S) = \mathcal{B}_{\text{min-}L\text{-}R}(L, S) + \mathcal{B}_{\text{min-}L+R}(L, S)$.

b In the case $L = S$:

1 all rules in each set $\mathcal{B}_{\text{min-}R}(L, L)$, $\mathcal{B}_{\text{min-}R+L}(L, L)$ are non-repeatedly generated

2 all rules in $\mathcal{B}_{\text{min-}R}(L, L)$, $\mathcal{B}_{\text{min-}R+L}(L, L)$, $\mathcal{B}_{\text{min}}(L, S)$ are totally different

3 $\mathcal{C}_{\text{min}}(L, L) = \mathcal{B}_{\text{min-}R}(L, L) + \mathcal{B}_{\text{min-}R+L}(L, L)$.

Proof:

a *In the case 'L ⊂ S'*:

1 From theorems of 2 and 3.

2 It is obvious since either their left sides or right sides are different.

3 – '⊇': For every $r_c : L' \to R' \in \mathcal{B}_{\text{min}-L-R}(L,S)$, where $R' \in \mathcal{FS}(S \setminus L)_L$, $L' \in \mathcal{FS}(L)$ and $(L' \subset L \text{ or } R' \subset S \setminus L)$, then $R' \in \mathcal{FS}(S \setminus L)_{L'}, h(L' + R') = S$ and $r_c \in \mathcal{C}_{\text{min}}(L,S)$. For every $r_c : L'' \to R'(L'' \setminus L'') \in \mathcal{B}_{\text{min}-L+R}(L,S)$, where $L' \in \mathcal{FS}(L), L'' \in \mathcal{FS}(L') \setminus \{L'\}, R' \in \mathcal{FS}(S \setminus L)_L$, we have $R' \in \mathcal{FS}(S \setminus L)_{L'}$, $h(L'' + R' + (L'' \setminus L'')) = h(L' + R') = h(L + R') = S$ and $r_c \in \mathcal{C}_{\text{min}}(L,S)$.

– '⊆': For every $r_c : L'' \to R'' \in \mathcal{C}_{\text{min}}(L,S) : h(L'' + R'') = S, h(L'') = L$ and $(L'' \subset L \text{ or } R'' \neq S \setminus L)$, we have $R'' = R_0^{'} + R'$, where $R_0^{'} = R'' \cap L, R' = R'' \setminus L$, $L' = L'' + R_0^{'} \supseteq L'', L'' + R'' \subseteq L \cup R'' = L + R' \subseteq S$. Then $R_0^{'} = L' \setminus L''$, $h(L'') = h(L') = L$ and $h(L + R') = S$, i.e., $L' \in \mathcal{FS}(L), L'' \in \mathcal{FS}(L') \subseteq \mathcal{FS}(L)$, $R' \in \mathcal{FS}(S \setminus L)_L$ and $r_c : L'' \to R' + (L' \setminus L'')$.

If $R_0^{'} = (L' \setminus L'') = \varnothing$, then $(R' = R'' \neq S \setminus L \text{ or } L'' \subset L)$. Thus $r_c : L'' \to R' \in \mathcal{B}_{\text{min-}L\text{-}R}(L, S)$.

If $R_0^{'} = (L' \setminus L'') \neq \varnothing$, then $L'' \in \mathcal{FS}(L') \setminus \{L'\}, r_c : L'' \to R' + (L' \setminus L'') \in \mathcal{B}_{\text{min}-L+R}(L,S)$.

b *In the case 'L = S':*

1 All rules in $\mathcal{B}_{\text{min-}R}(L, L)$ are non-repeatedly enumerated by its definition. Assume that there exists $i_2 > i_1 \geq 1 : r_{cj} : L_{ij} + R_j'' \to R_j \setminus R_j'' \mid L_{ij} \in \mathcal{G}(L), R_j^{'} \subset L \setminus L_{ij}$, $\varnothing \neq R_j'' \subset R_j^{'}, \forall j = 1,2$ and $r_{c1} = r_{c2}$. Then $L_{i1} \subset L_{i1} + R_1'' = L_{i2} + R_2''$. It contradicts the selection of index i_2! Thus, all rules in $\mathcal{B}_{\text{min}-R+L}(L, L)$ are also non-repeatedly enumerated.

2 It is obvious from their definitions.

3 – '⊇': If $r_c : L_i \to R' \in \mathcal{B}_{\text{min}-R}(L,L)$, where $L_i \in \mathcal{G}(L), R' \subset L \setminus L_i$, then $h(L_i) = h(L_i + R') = L$, so $r_c \in \mathcal{C}_{\text{min}}(L, L)$. If $r_c : L_i + R'' \to R \setminus R'' \in \mathcal{B}_{\text{min}-R+L}(L,L)$, where $L_i \in \mathcal{G}(L)$, $R' \subseteq L \setminus L_i, \varnothing \neq R'' \subset R'$, then $L_i + R'' + (R \setminus R'') = L_i + R' \subseteq L$ and $L = h(L_i) \subseteq h(L_i + R'' + (R \setminus R'')) = h(L_i + R') \subseteq L$, so $r_c \in \mathcal{C}_{\text{min}}(L, L)$.

– '⊆': For every $r_c : L' \to R \in \mathcal{C}_{\text{min}}(L,L), L' \in \mathcal{FS}(L), h(L' + R) = L, R \neq \varnothing$ and

$(L' \neq L_i$ or $R \subset L \setminus L_i)$, then $L' = L_i + L''$, where $L_i \in \mathcal{G}(L)$, $L' = L' \setminus L_i \subset L \setminus L_i$.

If $L'' = \emptyset$, then $R \subset L \setminus L_i$, so $r_c \in \mathcal{B}_{\text{min-}R}(L, L)$.

If $L'' \neq \emptyset$, then $R'' := L'' \subset L \setminus L_i, R \subseteq L \setminus L' \subset L \setminus L_i$, so $R' := R + R'' \not\subset L \setminus L_i$, $\emptyset \neq R'' \subset R'$. Thus, $r_c \in \mathcal{B}_{\text{min}-R+L}(L, L)$. □

Example 5. Figure 8 shows an illustration for the derivation of consequence rules from $\mathcal{B}_{\text{min}}(G, C) = \{ceg \rightarrow a\}$. We have $\mathcal{FS}(G) = \{e, ec, eg, g, gc, egc\}$ (from example 3) and $\mathcal{FS}(C \setminus G)_G = \{a\}$. Hence, $\mathcal{B}_{\text{min-}L\text{-}R}(G, C) = \{r_i: i = 1, .., 5\}$. Each itemset L' in $\mathcal{FS}(G)$ must be considered with $R' = a$. For $L' = e, g : \mathcal{FS}(L') \setminus \{L'\} = \emptyset$. For $L' = eg, FS(L') \setminus \{L'\} = \{e, g\}$, we have r_7: $e \rightarrow a + eg \setminus e$ and r_8: $g \rightarrow a + eg \setminus g$. For $L' = ec, gc$: we have r_6 and r_9. Similarly, for $L' = egc$, we get r_{10}, r_{11}, r_{12}, r_{13} and r_{14}. Therefore, $\mathcal{B}_{\text{min}-L+R}(G, C) = \{r_i: i = 6, .., 14\}$.

Figure 8 An illustration for the process of deriving non-repeatedly all consequence rules of $\mathcal{AR}(G, C)$

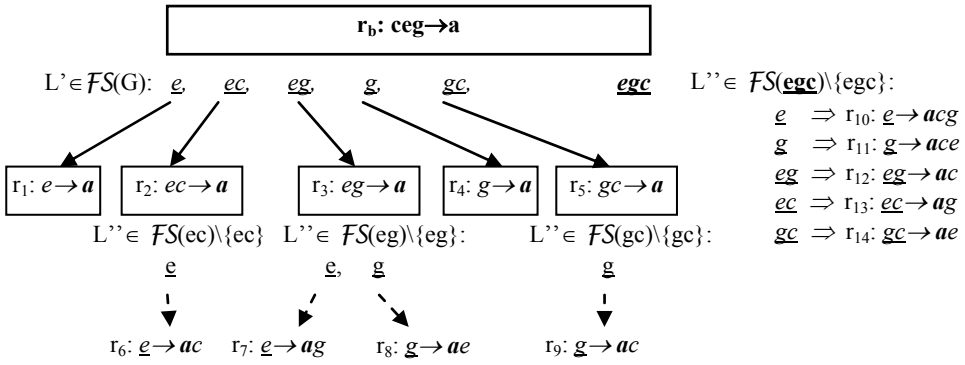


Figure 9 The algorithms for generating all min consequence rules non-repeatedly

Input: $(L, S) \in \mathcal{NFCS}, \mathcal{G}(L), \mathcal{G}(S)$: $L \subset S$.
Output: $C_{min}(L, S)$.
$C_{min}(L, S)$ **MINConsSet1** *(L, S)*

```
1.  C_min(L, S) = ∅;
2.  FS(L)      = GFS(L, G(L));
3.  FS(S\L)_L  = GCFS(L, S\L, G(S));
4.  for each (L'∈FS(L)) do
5.     for each (R'∈FS(S\L)_L) do
6.        if (L'⊂L or R'⊂S\L) then
7.           C_min(L, S) = C_min(L, S) + {r_c:L'→R'};
8.  for each (L'∈FS(L)) do
9.     FS(L') = GFS (L', G(L'));
10.    for each (L''∈FS(L')\{L'}) do
11.       for each (R∈FS(S\L)_L) do
12.    C_min(L,S) = C_min(L,S) + {r_c:L''→R+(L'\L'')};
13. return C_min(L, S);
```

Input: $\emptyset \neq L \in \mathcal{FCS}, \mathcal{G}(L)$. *Output:* $C_{min}(L, L)$.
$C_{min}(L, L)$ **MINConsSet2** *(L, G(L))*

```
1.  C_min(L, L) = ∅;
2.  for each (L_i∈G(L)) do
3.     for each (R⊂L\L_i) do
4.        C_min(L, L) = C_min(L, L) + {r_c:L_i→R};
5.  for each (L_i∈G(L)) do
6.     for each (∅≠R⊆L\L_i) do
7.        for each (∅≠R'⊂R) do
8.           IsDuplicate = false;
9.           for each (L_k∈G(L) | k<i) do
10.             if (L_k⊂L_i+R') then
11.                IsDuplicate = true;
12.                break;
13.          if (not(IsDuplicate)) then
14.    C_min(L,L) = C_min(L,L) + {r_c:L_i+R'→(R\R')};
15. return C_min(L, L);
```

Based on theorem 6, we obtain the algorithms of *MINConsSet1* and *MINConsSet2* (shown in Figure 9) for generating non-repeatedly all consequence ones.

5 Experimental study

The experiments below were carried out on a 2.93 GHz Pentium(R) Dual-Core CPU E6500 with 1.94GB of RAM, running under Linux, Cygwin. The source of CharmL-MG (at http://www.cs.rpi.edu/~zaki) is used to obtain the lattice of frequent closed itemsets and their generators. Five following benchmark databases (at http://fimi.cs.helsinki.fi/data/): T10i4d100k, Retail, Pumsbstar, Mushroom and Connect are used where two at first are sparse databases and the last three are dense ones. Their characteristics are shown in Table 3. For an order relation $\prec_{name}$, we take the disjoint union of all basic sets of all rule classes, i.e., $\sum_{(L,S)\in\mathcal{FCS}} \mathcal{B}_{name}(L,S)$, to create the corresponding basis. Thus, we have five bases of *min*, *mm*, *mM*, *Mm* and *MM*.

Table 3 Database characteristics

Database (DB)	*#Transaction*	*# Items*	*Average size*
Connect (C)	67,557	129	43
Mushroom (M)	8,124	119	23
Retail (R)	88,162	16,470	10
Pumsbstar (P)	49,046	7,117	50
T10I4D100K (T)	100,000	1,000	10

Mining association rules based on the relations $\prec_{mM}$ and $\prec_{mm}$ has been considered by Pasquier et al. (2005) and Zaki (2004). However, there are some drawbacks in those results. Indeed, the algorithm for mining *minmin* (most-general) basic rules proposed by Zaki (2004) generates many redundant candidates. Further, the algorithms for mining the *minMax* (minimal) rules and deriving the other ones given by Pasquier et al. (2005) miss some rules and are in a lot wasteful time to generate candidates. Indeed, theoretical and experimental results in Truong and Tran (2010) and Truong et al. (2010) showed them. We also proposed the better algorithms to mine association rules based on two those relations. In this experiment, we compare the mining basic rules based on relation $\prec_{min}$ against four relations of $\prec_{mm}$, $\prec_{mM}$, $\prec_{Mm}$ and $\prec_{MM}$ on the properties of the cardinality, the mining time and the rule length. Keep in mind that the outputs of the mining processes based on five different order relations are the same.

At first experiments, for each database, we fix a value of minimum confidence c_0 (%) and select three different typical values of minimum support s_0 *(%)*. The cardinalities of the bases are shown in Table 4 where the cardinality of min basis is denoted as N_{min}. The percent ratios of N_{min} to the cardinalities of four bases of *mm*, *mM*, *Mm* and *MM* are figured out in columns RN_{mm} (RN_{mm} := (N_{min} / N_{mm}) * 100%), NC_{mM}, NC_{Mm} and NC_{MM} respectively. It follows from this table that the cardinality of min basis is always smaller than the four ones. In fact, the values in the columns RN_{mm}, RN_{mM}, RN_{Mm} and RN_{MM} are all less than 100%. The reductions in the cardinalities are 24%, 8%, 15% and 3% (followed from the last row in the table). For each database (DB), we select two (low and

average) values of s_0. For each couple (*DB*, s_0), we consider four values of c_0 ranging from high values down to low ones. Observing these experiments, we also realise that the cardinality of min basis is smallest on five. Figure 10 and Figure 11 illustrate that for two databases of *mushroom* and *retail*.

Table 4 Comparison of the cardinalities of the bases

DB, c_0 (%)	s_0 (%)	N_{min}	RN_{mm} (%)	RN_{mM} (%)	RN_{Mm} (%)	RN_{MM} (%)
C, 94	95	25,548	98.7	100.0	98.7	98.7
	90	215,505	97.5	100.0	97.5	97.5
	85	452,180	95.5	100.0	95.5	95.5
M, 50	30	4,842	63.1	84.0	75.2	92.2
	20	17,714	43.0	80.5	56.8	91.6
	10	176,082	42.9	79.9	56.1	95.1
R, 60	0.01	190,463	88.3	96.1	94.4	97.9
	0.0075	329,807	80.5	93.4	90.2	96.3
	0.006	685,470	69.4	90.3	84.3	93.9
P, 60	50	3,107	90.4	97.3	93.9	97.5
	40	159,757	63.8	85.1	74.4	98.5
	35	651,953	58.3	80.6	71.0	98.9
T, 60	0.1	306,705	99.9	99.9	99.9	100.0
	0.01	757,988	88.1	96.7	94.9	99.2
	0.001	1,491,506	63.5	92.0	85.6	99.6
Average			76	92	85	97

Figure 10 The cardinalities of five bases for *mushroom* with s_0 = *20%* and *10%* (see online version for colours)

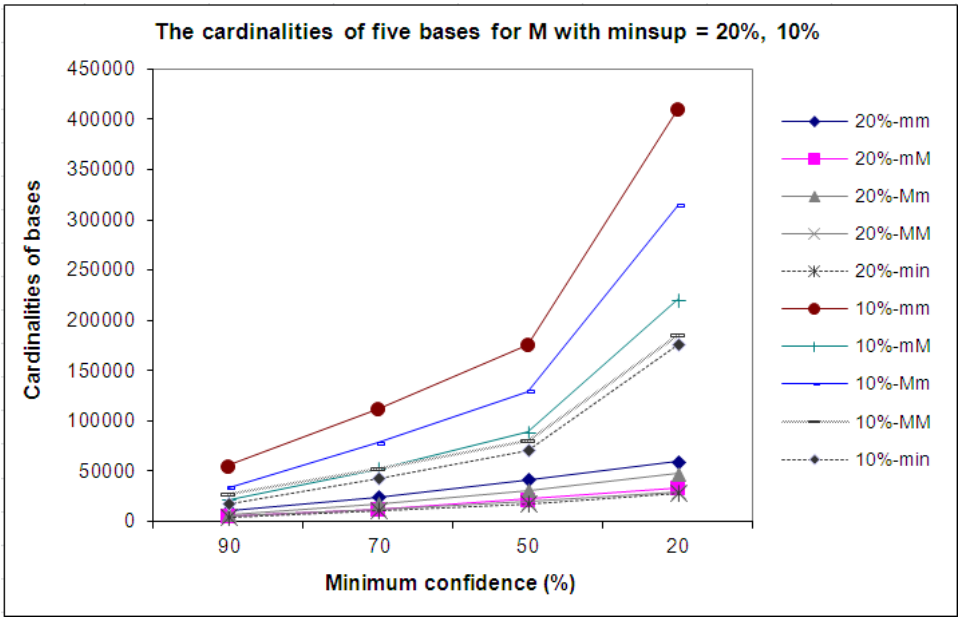


Figure 11 The cardinalities of five bases for *retail* with $s_0 = 0.0075\%$ and 0.006% (see online version for colours)

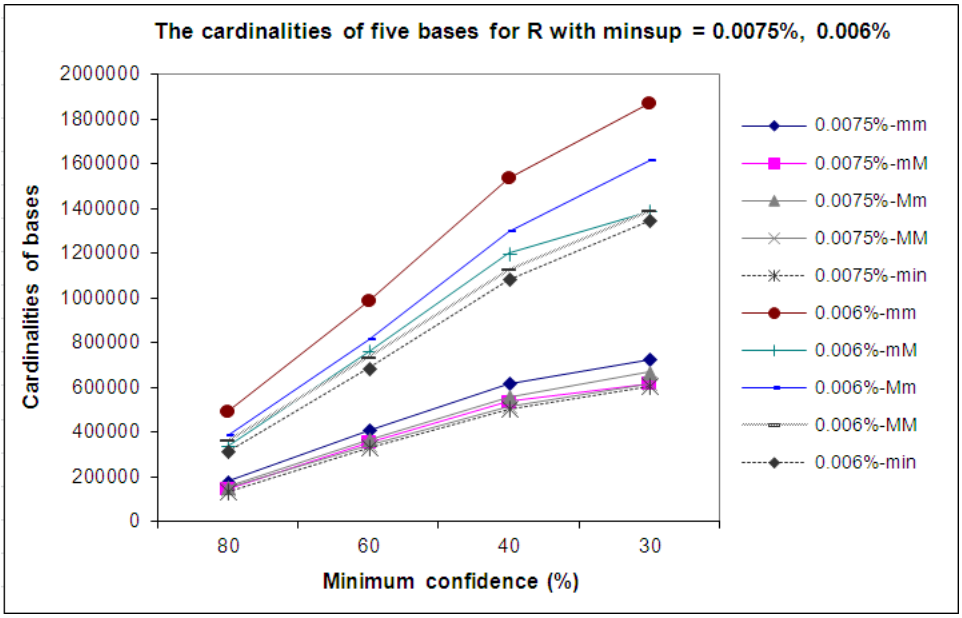


Figure 12 The times for mining bases for *pumsbstar* with $s_0 = 40\%$ and 35% (see online version for colours)

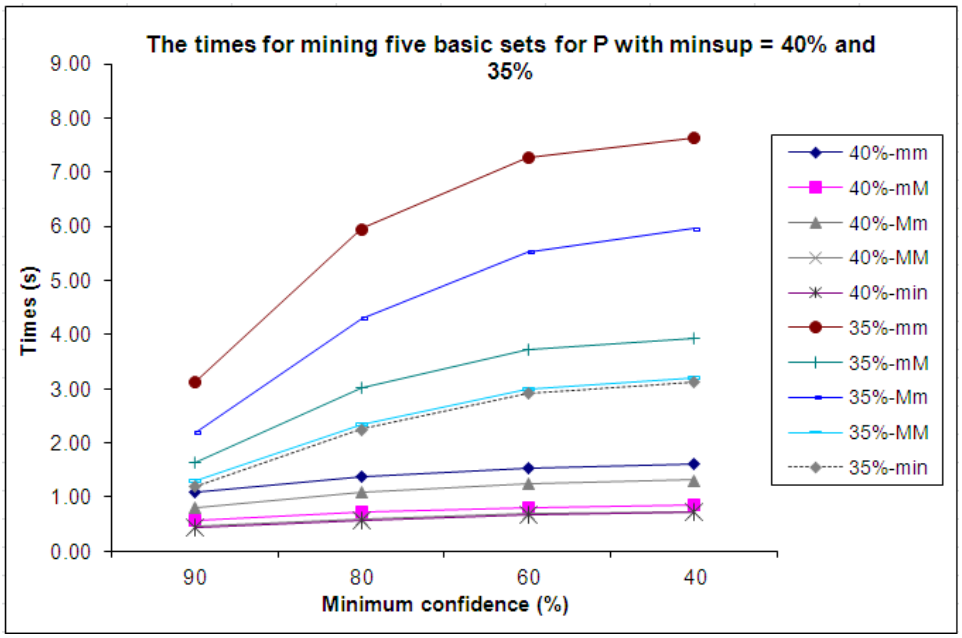
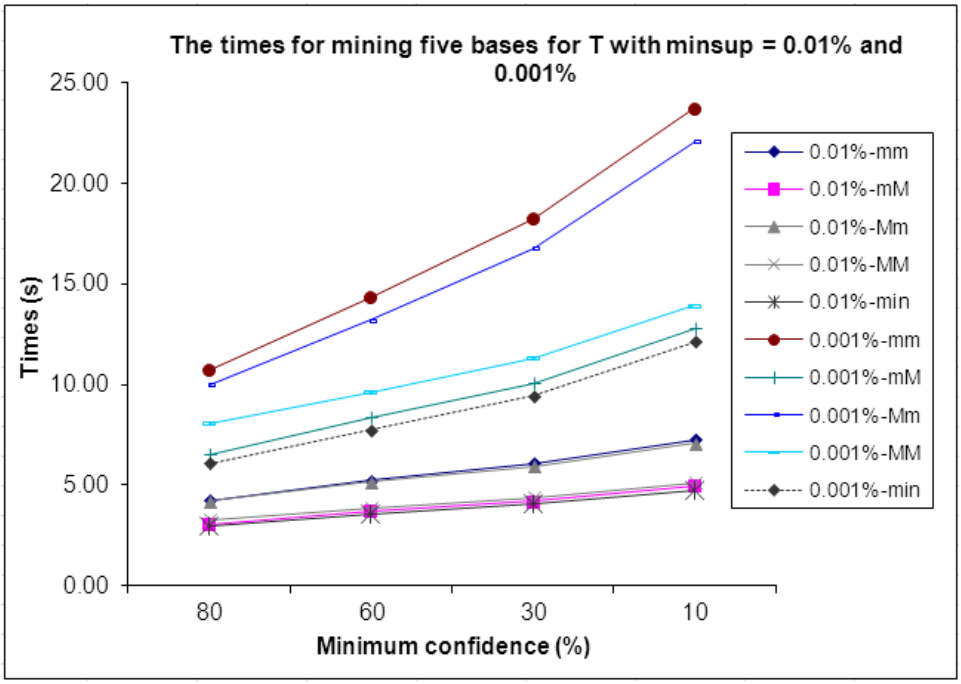


Figure 13 The times for mining bases for *T10i4d100k* with $s_0 = 0.01\%$ and *0.001%* (see online version for colours)



As a result, the time for mining min basis is smaller than the ones for mining other bases. Figure 12 and Figure 13 show this fact on the databases *Pumsbstar*, *T10i4d100k*.

In the last experiments, we compare the average length of min basic rules against the ones of basic rules in the forms *mm*, *mM*, *Mm* and *MM*. Table 5 includes the winner in each experiment. The table follows that in almost cases (by a ratio of 7/10) min basis is the winner. The *MM* basis wins only one time. The *mm* basis wins twice, however, it is not the second in the races of the mining times and the cardinalities.

Table 5 The winners in the experiments for comparison the average length of basic rules

DB	s_0 *(%)*	*Minimum confidence values*			
		1	*2*	*3*	*4*
C	90	mm	mm	mm	mm
	85	mm	mm	mm	mm
M	20	min	min	min	min
	10	min	min	min	min
R	0.0075	min	min	min	min
	0.006	min	min	min	min
P	40	min	min	min	min
	35	min	min	min	min
T	0.01	min	min	min	min
	0.001	MM	MM	MM	MM

6 Conclusions

We approached to the problem of mining frequent itemsets and association rules by the lattice of frequent closed itemsets and their generators. Using an equivalence relation based on frequent closed itemsets, we partition the frequent itemset class into disjoint equivalence classes where each class comprises frequent itemsets of the same closure. In addition, based on the pairs of nested frequent closed itemsets, we also partition the association rule set into equivalence rule classes. Two rules are in the same class if and only if the closures of their left sides are the same and the ones of their right sides are the same. Hence, the mining of frequent itemsets and association rules can avoid much duplication. In an itemset class, the unique representation of each itemset is described through the closure and their generators. For the target of mining association rules, in each rule class, we propose the unique representations of right side rules, which have the same left side. From those representations, we propose two algorithms *GFS*, *GCFS*, which can be applied to efficiently mine all frequent itemsets as well all association rules.

Toward condensed representations of association rule set, we propose five order relations to obtain five basic sets of each rule class. According to an order relation, the basic rule set consists of minimal elements that can be explicitly described via the generators and the closures. The consequence rules are completely, non-repeatedly generated by the operators of adding, deleting or moving appropriate eliminable itemsets in both sides of basic rules. Especially, we show that mining association rules based on *min* order relation is better than four other ones in terms of reductions in the running time for mining basic rules, their cardinalities and rule lengths. The corresponding algorithms of *MINBasicSet1*, *MINBasicSet2*, *MINConSet1* and *MINConSet2* are proposed.

Proposed algorithms can be converted in parallel and distributed environment. As an interesting extension, we plan to extend them to Hadoop Map/Reduce framework.

References

Agrawal, R. and Srikant, R. (1994) 'Fast algorithms for mining association rules', *Proceedings of the 20th International Conference on Very Large Data Bases*, pp.478–499.

Agrawal, R., Imielinski, T. and Swami, A. (1993) 'Mining association rules between sets of items in large databases', *Proceedings of the 1993 ACM SIGMOID Conference*, pp.207–216, Washington DC, USA.

Bastide, Y., Taouil, R., Pasquier, N., Stumme, G. and Lakhal, L. (2000) 'Mining frequent patterns with counting inference', *SIGKDD Explorations* Vol. 2, No. 2, pp.66–75.

Bayardo, R.J. (1998) 'Efficiently mining long patterns from databases', *Proceedings of the SIGMOD Conference*, pp.85–93.

Bayardo, R.J. and Agrawal, R. (1999) 'Mining the most interesting rules', *Proceedings of the KDD Conference*, pp.145–154.

Bayardo, R.J., Agrawal, R. and Gunopulos, D. (2000) 'Constraint-based rule mining in large, dense databases', *Data Mining and Knowledge Discovery*, Vol. 4, Nos. 2/3, pp.217–240.

Birkhoff, G. (1967) *Lattice Theory*, 3rd ed., American Mathematical Society, Providence, RI.

Boulicaut, J., Bykowski, A. and Rigotti, C. (2003) 'Free-sets: a condensed representation of Boolean data for the approximation of frequency queries', *Data Mining and Knowledge Discovery*, Vol. 7, No. 1, pp.5–22.

Burdick, D., Calimlim, M. and Gehrke, J. (2001) 'MAFIA: a maximal frequent itemset algorithm for transactional databases', *Proceedings of ICDE'01*, pp.443–452.

Cristofor, L. and Simovici, D. (2002) 'Generating an informative cover for association rules', *Proceeding of the IEEE International Conference on Data Mining*.

Das, A., Ng, W.K. and Woon, Y.K. (2001) 'Rapid association rule mining', *Proceedings of 10th International Conference on Information and Knowledge Management*, pp.474–481, ACM Press.

Duquenne, V. and Guigues, J-L. (1986) 'Famille minimale d'implications informatives re'sultant d'un tableau de donne´es binaires', *Math. et Sci. Hum.*, Vol. 24, No. 95, pp.5–18.

Feller, W. (1950) *An Introduction to Probability Theory and Its Applications*, Vol. 1, New York, John Wiley & sons. Inc., Chapman & Hall. Ltd. London.

Han, J., Pei, J. and Yin, J. (2000) 'Mining frequent itemsets without candidate generation', *Proceedings of SIGMOID'00*, pp.1–12.

Ho, B. (1995) 'An approach to concept formation based on formal concept analysis', *IEICE Trans. Information and Systems*, Vol. E78-D, No. 5, pp.553–579.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H. and Verkamo, A.I. (1994), 'Finding interesting rules from large sets of discovered association rules', *Proceeding of the 3rd CIKM Conference*, pp.401–407.

Li, G. and Hamilton, H.J. (2004) 'Basic association rules', *Proceeding of the 4th SIAM International Conference on Data Mining*, pp.166–177.

Luxenburger, M. (1991) 'Implications partielles dans un contexte', *Math., Inf. et Sci. Hum.*, Vol. 29, No. 113, pp.35–55.

Pasquier, N., Taouil, R., Bastide, Y., Stumme, G. and Lakhal, L. (2005) 'Generating a condensed representation for association rules', *J. of Intelligent Information Systems*, Vol. 24, No. 1, pp.29–60.

Srikant, R., Vu, Q. and Agrawal, R. (1997) 'Mining association rules with item constraints', *Proceeding KDD'97*, pp.67–73.

Szathmary, L., Valtchev, P. and Napoli, A. (2009) 'Efficient vertical mining of frequent closed itemsets and generators', *IDA 2009*, pp.393–404.

Tran, A., Duong, H., Truong, T. and Le, B. (2012a) 'Mining frequent itemsets with dualistic constraints', *Proceedings of PRICAI 2012*, pp.807–813, LNAI 7458, Springer-Verlag.

Tran, A., Truong, T. and Le, B. (2013) 'An approach for mining concurrently closed itemsets and generators', *ICCSAMA 2013, Advanced Computational Methods for Knowledge Engineering,* Vol. 479, pp.355–366, SCI.

Truong, T. and Tran, A. (2010) 'Structure of set of association rules based on concept lattice', *ACIIDS 2010*, *Adv. in Intelligent. Information and Database System*, Vol. 283, pp.217–227, SCI.

Truong, T., Tran, A. and Tran, T. (2010) 'Structure of association rule set based on min-min basic rules', *Proceedings of the 2010 IEEE-RIVF International Conference on Computing and Communication Technologies*, pp.83–88.

Wille, R. (1992) 'Concept lattices and conceptual knowledge systems', *Computers and Mathematics with Applications* Vol. 23, Nos. 6–9, pp.493–515.

Zaki, M.J. (2000) 'Scalable algorithms for association mining', *IEEE Trans. Knowledge and Data Engineering*, Vol. 12, No. 3, pp.372–390.

Zaki, M.J. (2004) 'Mining non-redundant association rules', *Data Mining and knowledge Discovery*, Vol. 9, No. 3, pp.223–248.

Zaki, M.J. and Hsiao, C.J. (2005) 'Efficient algorithms for mining closed itemsets and their lattice structure', *IEEE Trans. Knowledge and Data Engineering*, Vol. 17, No. 4, pp.462–478.

Notes

1 We write 'if and only if' simply 'iff'

2 The notation '+' represents the union of two disjoint sets.

3 We write the set $\{a_1, a_2, .., a_n\}$ simply $a_1a_2..a_n$.

4 (L, S) denote the pair of two non-empty, nested frequent closed itemsets L, S, i.e., $L, S \in \mathcal{FCS}$ and $\varnothing \subset L \subseteq S$ such that supp(S)/supp(L) $\geq c_0$.