Mining Association Rules Restricted on Constraint

Conference Paper · February 2012

DOI: 10.1109/rivf.2012.6169825

4 authors:

Anh Tran
University of Dalat

Tin C Truong
University of Dalat

Bac Le
Ho Chi Minh City University of Science

Hai Duong
University of Dalat, Vietnam

Mining Association Rules Restricted on Constraint

Anh N. Tran
Department of Mathematics and Computer Science, University of Dalat, Dalat, Vietnam
anhtn@dlu.edu.vn

Tin C. Truong
Department of Mathematics and Computer Science, University of Dalat, Dalat, Vietnam
tintc@dlu.edu.vn

Bac H. Le
Department of Information Technology, University of Natural Science, HCM, Vietnam
lhbac@fit.hcmus.edu.vn

Hai V. Duong
Department of Mathematics and Computer Science, University of Dalat, Dalat, Vietnam
haidv@dlu.edu.vn

Abstract—The aim of this paper is to solve the problem of mining association rule restricted on a given constraint itemset which often changes. At the time of building the system, on given database, we mine first the lattice of closed itemsets. Based on that lattice, whenever the constraint or the minimum support changes, the lattice of all restricted frequent closed itemsets is obtained. The set of all association rules restricted on constraint partitions into disjoint equivalence classes. Each class is represented by a pair of two nested frequent closed itemsets. Then, we just mine independently each rule class. Users can select the rule class that they are interested in. Spending only a little of time, we can mine and figure out the basic rules of that class. They are useful for users because their left-handed sides are minimal and their right-handed sides are maximal. When necessary, the set of all remaining consequence ones together with their confidences can be quickly generated from the basic ones. This consequence set also splits into the different subsets according to different generating operators. Hence, our approach is very efficient and close to user! The theoretical affirmations and experimental results prove that.

Data mining with constraint; restricted association rule; restricted frequent itemset; basic rule; consequence rule

I. INTRODUCTION

Being firstly introduced and researched by Agrawal et al. [2] in 1993, association rule mining is one of the important problems in data mining. The numbers of mined frequent itemsets and association rules are usually enormous, especially in dense databases as well as with the small minimum supports and confidences. As usual, users only take care of the smaller sets of frequent itemsets and association rules that satisfy given properties or constraints. The post-processing to select them wastes much time and can be repeated many times until users get the results that they desire. Hence, the problem of mining frequent itemsets and association rules with constraints is reality. It has been receiving attentions of many researchers. In [1, 3, 5, 8, 9, 12, 13] some authors researched on mining frequent itemsets and association rules from the viewpoint of the user's interaction with the system. Srikant et al. [13] considered the problem of integrating constraints in the form of Boolean expression that appoint the presence or absence of items in rules. Bayardo et al. [3] took care of mining association rules with constraints in their right-handed sides and considered additionally mining with the minimum improvement. In [9], Nguyen et al. proposed the mining with constraints such as monotone, anti-monotone, etc. They suggested the algorithm CAP for processing those constraints. Pei et al. [12] considered the convertible constraints and integrated them into the mining of FP-growth algorithm. In [5], Cong and Liu suggested the concept of tree boundary to utilize previous mining result. Lee et al. [8] obtained the algorithms for mining association rules with multi-dimensional constraints based on FP-growth.

This paper focuses on the problem of mining association rules for online users. The data of websites are usually obtained in the tables that the numbers of their columns (items) are enormous. However, at a given time, users are only interested in items contained a given itemset C. Showing immediately association rules between itemsets contained in C is an important task, especially when C often changes. The traditional approach [2] solves this problem in two phases. The first one is mining frequent itemset restricted on constraint C. The second is generating restricted association rules from those frequent itemsets. Although restricted by constraint, the cardinality of mined frequent itemsets is still enormous. Then, generating association rules from them spends much time.

Recent works [10, 14, 15, 17] figured out that mining association rules directly from the frequent closed itemsets and their generators is more efficient. In [1], we proposed the efficient approach to mine frequent closed itemsets and their generators when C often changes. Based on those two results, the paper suggests the model for mining restricted association rules. It is described as follows. First, at the time that the system is built, using two algorithms of *Charm-L* [16] and *MinimalGenerators* [17], we mine first the lattice $\mathcal{LG}_A$ of closed itemsets (and their generators). Due to this lattice, when users select minimum support and constraint C, the class $\mathcal{FLG}_C$ of all restricted frequent closed itemsets (and their generators) is quickly determined by applying algorithms of *MineCGCons* [1] and *CreateLattice*. Fixing the minimum confidence, based on $\mathcal{FLG}_C$, the min-max basis of restricted association rules in the form of min-max is mined (using the algorithm *MineBasis*). Having minimal antecedents and maximal consequents, those rules are useful for users. Moreover, the size of this basis is smaller than the one of the set of all restricted association rules. When necessary, all remaining rules together with their confidences are quickly generated from the basic ones by applying algorithms of *MineCSet1* and *MineCSet2*.

The paper is organized as follows. Section II recalls some concepts of association rule mining and some features of association rule mining with constraint. Section III presents the structure of restricted frequent itemsets. Sections IV proposes the structure of the restricted association rule set based on min-max basis. It also indicates efficient algorithms for finding rule subsets of basic and consequence. Sections V and VI show experimental results and the conclusion. Since the size of paper is limited, we do not figure out some proofs.

II. PRELIMINAIRES

A. Concepts of association rule mining

Given sets of $\mathcal{O}$ containing records or transactions of a database $\mathcal{T}$ and $\mathcal{A}$ containing attributes or items related to each of transaction $o\in\mathcal{O}$. Let $\mathcal{R}$ be a binary relation in $\mathcal{O}\mathrm{x}\mathcal{A}$. Consider two operators: λ: $2^{\mathcal{O}}\rightarrow 2^{\mathcal{A}}$, ρ: $2^{\mathcal{A}}\rightarrow 2^{\mathcal{O}}$ determined by $\lambda(O) = \{a\in\mathcal{A} \mid (o, a) \in \mathcal{R}, \forall o\in O\}$, $\forall O\subseteq\mathcal{O}$; $\rho(A) = \{o\in\mathcal{O} \mid (o, a) \in \mathcal{R}, \forall a\in A\}$, $\forall A\subseteq\mathcal{A}$. Defining closure operator h in $2^{\mathcal{A}}$ [4] by: $h = \lambda \circ \rho$, h(A) is called the closure of itemset $A \subseteq\mathcal{A}$. If A = h(A), A is called closed itemset. Let $\mathcal{CS}$ be the class of all closed itemsets. The lattice of all closed itemsets (and their corresponding generators) is denoted by $\mathcal{LG}_{\mathcal{A}}$. The support of itemset A is defined by: $s(A) = |\rho(A)|/|\mathcal{O}|$. Denote that s_0 is minimum support, $s_0 \in [1/|\mathcal{O}|; 1]$, if $s(A) \geq s_0$ then A is called frequent itemset [2]. Let $\mathcal{FS}$ and $\mathcal{FCS} = \mathcal{CS}\cap\mathcal{FS}$ be the classes of all frequent itemsets and all frequent closed ones. For two non-empty itemsets G, A: $\emptyset\neq G\subseteq A\subseteq\mathcal{A}$, G is called a generator of A [10] iff: $h(G) = h(A)$ and $(\forall\ \emptyset\neq G'\subset G \Rightarrow h(G') \subset h(G))$. Let $\mathcal{G}(A)$ be the class of all generators of A. In $2^{\mathcal{A}}$, an itemset R is called eliminable [14] in S if $R\subset S$ and $\rho(S) = \rho(S\backslash R)$. Denote that $\mathcal{N}(S)$ is the class of all eliminable itemsets in S. For any frequent itemset S (according to s_0), we take a non-empty, strict subset L from S ($\emptyset\neq L\subset S$), $R = S\backslash L$. Denote that r: $L\rightarrow R$ is the rule created by L, R (or by L, S). Then, $s(r) = s(S)$ and $c(r) = |\rho(S)|/|\rho(L)| = s(S)/s(L)$ are the support and the confidence of r. Let c_0 be the minimum confidence, $c_0 \in (0; 1]$. The rule r is an association rule iff $c(r) \geq c_0$ [2].

B. Features of association rule mining restricted on constraint

For every $\mathcal{C}\in 2^{\mathcal{A}}\backslash\{\emptyset\}$, let us consider connection operators: $\rho_{\mathcal{C}}$: $2^{\mathcal{C}}\rightarrow 2^{\mathcal{O}}$, $\lambda_{\mathcal{C}}$: $2^{\mathcal{O}}\rightarrow 2^{\mathcal{C}}$ and $h_{\mathcal{C}}$: $2^{\mathcal{C}}\rightarrow 2^{\mathcal{C}}$ defined [1] as follows: $\emptyset\neq C'\subseteq\mathcal{C}$, $\emptyset\neq O\subseteq\mathcal{O}$: $\rho_{\mathcal{C}}(C') = \{o\in\mathcal{O}: (o, a)\in\mathcal{R}, \forall a\in C'\}$ ($\rho_{\mathcal{C}}(\emptyset) := \mathcal{O}$), $\lambda_{\mathcal{C}}(O) = \{a\in\mathcal{C}: (o, a)\in\mathcal{R}, \forall o\in O\}$ ($\lambda_{\mathcal{C}}(\emptyset) := \mathcal{C}$), and $h_{\mathcal{C}} = \lambda_{\mathcal{C}} \circ \rho_{\mathcal{C}}$. For every G, C': $\emptyset\neq G\subseteq C'\subseteq\mathcal{C}$, G is called a generator restricted on $\mathcal{C}$ of C' [1] iff: $h_{\mathcal{C}}(G) = h_{\mathcal{C}}(C')$ and $(\forall\emptyset\neq G'\subset G \Rightarrow h_{\mathcal{C}}(G') \subset h_{\mathcal{C}}(G))$.

Theorem 1 (Relations between connection operators and Galois ones) [1]: For every $\mathcal{C}\in 2^{\mathcal{A}}\backslash\{\emptyset\}$, $\emptyset\neq C'\subseteq\mathcal{C}$, $O\subseteq\mathcal{O}$, one have: $\rho_{\mathcal{C}}(C') = \rho(C')$, $\lambda_{\mathcal{C}}(O) = \lambda(O) \cap \mathcal{C}$, $h_{\mathcal{C}}(C') = h(C') \cap \mathcal{C}$ and $\mathcal{G}_{\mathcal{C}}(C') = \mathcal{G}(C')$.

While considering itemsets contained in $\mathcal{C}$, theorem 1 allows to apply the following concepts of association rule mining to the one with constraint $\mathcal{C}$ (association rule mining restricted on $\mathcal{C}$) such as support, frequent itemset, generators, eliminable itemset, confidence and association rule. An itemset $C'\subseteq\mathcal{C}$ is called a restricted closed itemset if $h_{\mathcal{C}}(C') = C'$ [1]. The class $\mathcal{FCS}(\mathcal{C})$ contains all restricted frequent closed itemsets. The class of all restricted frequent itemsets is denoted by $\mathcal{FS}_{\mathcal{C}}$. The set of all association rules coming from restricted frequent itemsets is denoted by $\mathcal{ARS}_{\mathcal{C}}$.

III. STRUCTURE OF RESTRICTED FREQUENT ITEMSETS

First, we recall the important result of generating quickly (non-repeatedly) all frequent closed itemsets restricted on $\mathcal{C}$ and their generators from lattice $\mathcal{LG}_{\mathcal{A}}$ without the need of accessing the data. Next, an equivalence relation partitions the class of all restricted frequent itemsets into disjoint equivalence classes. Without lost of generality, we only need to investigate the structure of frequent itemsets in each class. Generators and their corresponding eliminable itemsets represent frequent itemsets in a class. Finally, we suggest the way of generating non-repeatedly restricted frequent itemsets in each class and the sub ones. This allows deriving non-repeatedly (so quickly) all consequence rules restricted on $\mathcal{C}$ together with their confidences.

Proposition 1 (Generating non-repeatedly all restricted frequent closed itemsets and their generators from lattice $\mathcal{LG}_{\mathcal{A}}$) [1]: Assign that $\mathcal{FCS}_{\mathcal{C}} := \{C' = L\cap\mathcal{C} \mid L\in\mathcal{FCS}, \exists L_i\in\mathcal{G}(L): L_i \subseteq \mathcal{C}\}$, the following statements hold: (a) $\mathcal{FCS}_{\mathcal{C}} = \mathcal{FCS}(\mathcal{C})$, (b) *all elements of $\mathcal{FCS}_{\mathcal{C}}$ are non-repeatedly generated*, (c) $\mathcal{G}(C') = \{L_i\in\mathcal{G}(L): L_i \subseteq C'\}$.

Definition 1 (Equivalence relation on $2^{\mathcal{C}}$) [1]: $\forall$ A, B $\in 2^{\mathcal{C}}$:

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

Theorem 2 (Partition of $\mathcal{FS}_{\mathcal{C}}$) [1]: Equivalence relation $\sim_{\mathcal{C}}$ partitions $\mathcal{FS}_{\mathcal{C}}$ into the equivalence classes. Each class contains the restricted frequent itemsets having the same closure:

$$\mathcal{FS}_{\mathcal{C}} = \sum_{C'\in\mathcal{FCS}_{\mathcal{C}}} [C']_{\sim_{\mathcal{C}}}.$$

Example 1. Consider database T in Fig. 1.a. The lattice $\mathcal{LG}_{\mathcal{A}}$ mined from T by applying *Charm-L* and *MinimalGenerators* is shown in Fig. 2, where: closed itemsets are underlined, their supports are the superscript numbers and their generators are italicized. Fix now $s_0 = 2/7$ and $\mathcal{C}$ = *acegh* ($\{a_1, a_2, \ldots, a_n\}$ is abbreviated that $a_1a_2\ldots a_n$), applying *MineCGCons*, one have $\mathcal{FCS}_{\mathcal{C}}$ = {<C'=ach, $\mathcal{G}(C')$={ch}>, <aceg, {ae, ag}>, <ah, {h}>, <ceg, {e, g}>, <ac, {ac}>, <c, {c}>, <a, {a}>}. The partition of the class of all frequent itemsets restricted on $\mathcal{C}$ is pointed out in Fig. 1.b. This example is used as support for the remaining ones of the paper.

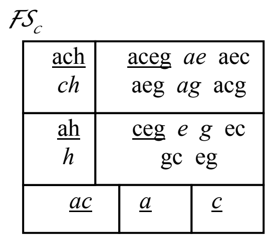
Proposition 2 (Representation of restricted frequent itemsets having the same closure): For every $C'\in\mathcal{FCS}_{\mathcal{C}}$:

$$X \in [C']_{\sim C} \Leftrightarrow \exists X_0 \in \mathcal{G}(C'), \exists X' \in \mathcal{N}(C'): X = X_0 + X'$$

(The symbol + is denoted as the union of two disjoint sets).

Trans	Items
1	aceg
2	acfh
3	adfh
4	bceg
5	aceg
6	bceg
7	acfh

(a) *Database T*



(b) *The partition of* $\mathcal{FS}_c$

Figure 1. Database T and the partition of $\mathcal{FS}_c$.

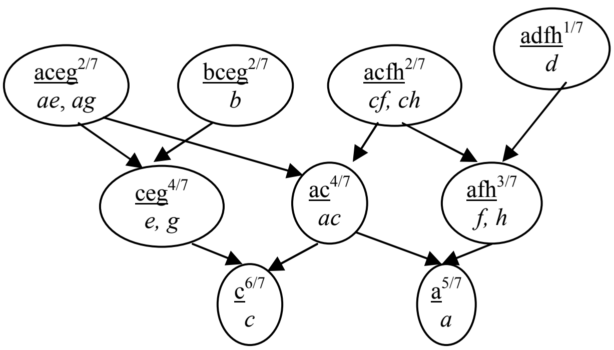


Figure 2. The lattice $\mathcal{LG}_A$ according to database T.

Proof: If X ∈ $[C']_{\sim C}$, there exists $X_0 \in \mathcal{G}(X)$. Then, $h_C(X_0) = h_C(X) = h_C(C')$, so $X_0 \in \mathcal{G}(C')$. Let X'' = X\\$X_0$. Hence, X'' ⊆ X\\$X_0$, so X'' ∈ $\mathcal{N}$(X) and X' = X_i+X''.

Proposition 3 (Generating non-repeatedly all restricted frequent itemsets of each class). $\forall L \in \mathcal{FCS}_C$, let $\mathcal{L}_U = \bigcup_{L_i \in \mathcal{G}(L)} \mathcal{L}_i$, $\mathcal{L}^{\sim} = L \backslash \mathcal{L}_U$, $\mathcal{L}_{U,Li} = \mathcal{L}_U \backslash L_i$,

$$\mathcal{FS}_C(L) \equiv \{L' := L_i + R_i + R_i' \mid L_i \in \mathcal{G}(L), R_i \subseteq \mathcal{L}_{U,Li}, R_i' \subseteq \mathcal{L}^{\sim},$$
$$i=1 \text{ or } (i>1: L_k \not\subset L_i+R_i, L_k \in \mathcal{G}(L), \forall 1 \le k<i)\}.$$

We conclude that: (a) *all sets in* $\mathcal{FS}_C(L)$ *are non-repeatedly generated*, (b) $\mathcal{FS}_C(L) \equiv [L]_{\sim C}$.

Proof: (a) Assume that $\exists i>k \ge 1$: $L_i+R_i+R_i' \equiv L_k+R_k+R_k'$, $R_i \subseteq \mathcal{L}_{U,Li}$, $R_i' \subseteq \mathcal{L}^{\sim}$, $R_k \subseteq \mathcal{L}_{U,Lk}$, $R_k' \subseteq \mathcal{L}^{\sim}$. Since $L_k \cap R_i' = \varnothing$, $L_k \subseteq L_i+R_i$. Moreover, L_i, $L_k \in \mathcal{G}(L)$. Thus, $L_k \subset L_i+R_i$. This contradicts to the way to select R_i!

(b) "⊆": $\forall L' \in \mathcal{FS}_C(L)$, $\exists L_i \in \mathcal{G}(L)$, $R_i' \subseteq \mathcal{L}^{\sim} \subseteq L \backslash L_i$, $R_i \subseteq \mathcal{L}_{U,Li} \subseteq L \backslash L_i$: $L' = L_i+R_i+R_i'$. Thus, $R_i+R_i' \subseteq L \backslash L_i$. By proposition 2, $L' \in [L]_{\sim C}$.

"⊇": $\forall L' \in [L]_{\sim C}$, by proposition 2, let i be the minimum index such that $L' = L_i+U_i$, $L_i \in \mathcal{G}(L)$ and $U_i \subseteq L \backslash L_i$. Putting $R_i = U_i \cap \mathcal{L}_U$ and $R_i' = U_i \backslash \mathcal{L}_U$, it is easy to see that $R_i \subseteq \mathcal{L}_{U,Li}$, $R_i' \subseteq \mathcal{L}^{\sim}$. Then, $L' = L_i+R_i+R'_i$. Suppose that $\exists 1 \le k<i$: $L_k \subset L_i+R_i$, $L_k \in \mathcal{G}(L)$ then $L' = L_k+(L_i+R_i)\backslash L_k+R'_i = L_k+R_k+R'_k$, where $R_k := (L_i+R_i)\backslash L_k \subseteq L \backslash L_k$, $R'_k := R'_i \subseteq \mathcal{L}^{\sim} \subseteq L \backslash L_k$. Hence, $U_k = R_k+R'_k \subseteq L \backslash L_k$: it is absurd! Then, $L_k \not\subset L_i+R_i$, $\forall 1 \le k<i$, i.e., $L' \in \mathcal{FS}_C(L)$. The proof is completed.

Example 2. Consider $[aceg]_{\sim C}$. We have: $\mathcal{L}_U$ = aeg, $\mathcal{L}^{\sim}$ = c, $\mathcal{L}_{U,ae}$ = g, $\mathcal{L}_{U,ag}$ = e. According to generator ae, the following restricted frequent itemset ae+∅+∅, ae+∅+c, ae+g+∅, and ae+g+c are obtained. With the one, we get: ag+∅+∅, ag+∅+c. The itemsets ag+e+∅, ag+e+c do not appear once again because ae ⊂ ae+g.

Definition 2 (Sub restricted frequent itemsets). For every L, S ∈ $\mathcal{FCS}_C$, L⊂S, $L_i \in \mathcal{G}(L)$, define:

$$\mathcal{SFS}_C(S, L_i) \equiv \{\varnothing \ne R' \subseteq S, L_i+R' \in [S]_{\sim C}, L_i \cap R' = \varnothing\}$$
$$\equiv \{\varnothing \ne R' \subseteq S \backslash L_i \mid h_C(L_i+R') = S\}.$$

In other words, $\mathcal{SFS}_C(S, L_i)$ is the set of all restricted frequent itemsets that the closure of the union of each of it and L_i are S.

Proposition 4 (Generating non-repeatedly all sub restricted frequent itemsets): For every L, S ∈ $\mathcal{FCS}_C$ such that L ⊂S, $L_i \in \mathcal{G}(L)$, $R_{min}(S, L_i) := \{R^*_j := S_j \backslash L_i \mid S_j \in \mathcal{G}(S), S_j \backslash L_i \text{ is minimal}\}$.

Let $\mathcal{L}_{Li} = \bigcup_{R^*_j \in R_{min}(S,L_i)} R^*_j$, $\mathcal{L}_{Li,j} := \mathcal{L}_{Li} \backslash R^*_j$, $\mathcal{L}^{\sim}_{Li} := S \backslash (\mathcal{L}_{Li}+L_i)$,

$$\mathcal{FS}_C(S, L_i) \equiv \{R' = R^*_j+R'_j+R^{\sim}, R^*_j \in R_{min}(S, L_i), R^{\sim} \subseteq \mathcal{L}^{\sim}_{Li},$$
$$R'_j \subseteq \mathcal{L}_{Li,j}, j=1 \text{ or } (j>1: R^*_k \not\subset R^*_j+R'_j,$$
$$R^*_k \in R_{min}(S, L_i), \forall 1 \le k<j)\}.$$

The following statements are correct: (a) *all sets in* $\mathcal{FS}_C(S, L_i)$ *are non-repeatedly generated*, (b) $\mathcal{FS}_C(S, L_i) \equiv \mathcal{SFS}_C(S, L_i)$.

IV. MINING RESTRICTED ASSOCIATION RULES

First, the set of all restricted association rules (rule set) is partitioned into equivalence classes in order to easily mining it.

Definition 3 (Equivalence relation on the rule set): Let $\sim_{rC}$ be a binary relation in $\mathcal{ARS}_C$ determined as follows: ∀ L', S', Ls, Ss ∈ $\mathcal{FS}_C$, ∅≠L'⊂S', ∅≠Ls⊂Ss, r:L'→S'\L', s:Ls→Ss\Ls:

$$s \sim_{rC} r \;\; \text{iff:} \;\; Ls \in [L']_{\sim C} \text{ and } Ss \in [S']_{\sim C}.$$

Theorem 3 (Partition of the rule set): It is easy to see that $\sim_{rC}$ is an equivalence relation on $\mathcal{ARS}_C$. It partitions $\mathcal{ARS}_C$ into disjoint equivalence rule classes $\mathcal{AR}_C$(L, S). The representative of a class is a pair of two restricted frequent closed itemsets (L, S), L ⊆ S. The class contains rules of the same closures of left-handed side L and two-sided union S. Then, they all have the same confidence s(S)/s(L) and the same support s(S).

$$\mathcal{ARS}_C = \sum_{(L,S)} \mathcal{AR}_C(L, S).$$

Example 3. One have: $\mathcal{AR}_C$(ceg, aceg) = {ceg→a, cg→a, eg→a, ce→a, cg→ae, eg→ac, ce→ag, g→a, e→a, g→ac, g→ae, e→ag, e→ac, g→ace, e→acg} and $\mathcal{AR}_C$(ceg, ceg) = {e→g, e→c, e→cg, g→e, g→c, g→ec, eg→c, ec→g, gc→e}.

Thanks to this partition, we only need to consider mining all rules contained in each rule class $\mathcal{AR}_C$(L, S) with L ⊆ S, i.e., to consider mining based on the pairs of nested restricted frequent closed itemsets and their generators. Using *MineCGCons*, the class of all those itemsets is obtained. Then, the remaining task is to determine the set of arcs connected from parent itemsets to the child ones – the lattice $\mathcal{FLG}_c$. Due to this lattice, when fix the minimum confidence, we can mine all rules. The following algorithm is obtained to determine the lattice $\mathcal{FLG}_c$.

```
FLG_c  CreateLattice (FCS_c)
{  Init empty FLG_c;
   for each (C'∈ FCS_c)  push (FLG_c, C');
}
      push (FLG_c, C')
      {  L = |FLG_c|;
         for (i=L; i>=1; i--)
            if (FLG_c[i].Mark < L+1 and C'⊃FLG_c[i])
            {  Create the arc from C' to FLG_c[i];
               Mark all childs of FLG_c[i] by L+1;
            }
         C'.Mark = L+1;  FLG_c[L+1] = C';
      }
```

Example 4. By *CreateLattice* algorithm, from $\mathcal{FCS}_c$ in example 1, we have the following lattice $\mathcal{FLG}_c$.

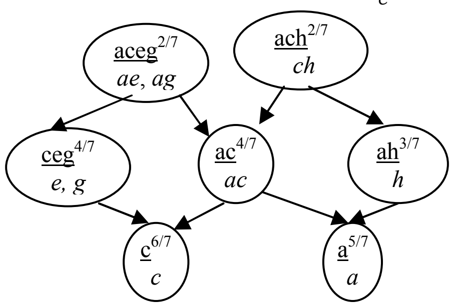


Figure 3. The lattice $\mathcal{FLG}_c$ according to database T: $s_0 = 2/7$, $\mathcal{C} = acegh$.

Next, an order relation over a rule class splits it into two disjoint subsets. The basic one contains minimal elements (according to that relation) whose the left-handed sides are minimal and the right-handed sides are maximal. Then, it is useful for users. Having the small size (compared to the one of the rule class), saving and using the basis are also easy. The remaining one includes in remaining rules that can be non-repeatedly derived by using the operators that we propose. It is called the consequence set.

Definition 4 (Order relation over a rule class): Consider a partial relation $\prec$ over $\mathcal{AR}_c$(L, S), defined as follows: $\forall r_j$: $L_j \to R_j \in \mathcal{AR}_c(L, S)$, $S_j = L_j+R_j$, j=1,2,

$$r_1 \prec r_2 \text{ iff: } L_1 \subseteq L_2 \text{ and } R_1 \supseteq R_2.$$

Definition 5 (The structures of rule subsets): a) Min-max basic rule set: $\mathcal{B}_c(L, S) = \{r_b: L_i \to S\backslash L_i \mid L_i \in \mathcal{G}(L)\}$,

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

b) Consequence rule set: $\mathcal{C}_c(L, S) = \mathcal{AR}_c(L,S) \setminus \mathcal{B}_c(L,S)$.

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

a) $L \subset S$: $\mathcal{B}_{-R+L}(L, S) \equiv \{r_c: L_i+R'' \to R'\backslash R'' \mid L_i \in \mathcal{G}(L), R' \in \mathcal{FS}_c(S, L_i), \varnothing \neq R'' \subseteq R' \cap L, R'' \subset R', i=1 \text{ or } (i>1: L_k \not\subset L_i+R'', \forall k: 1 \leq k<i)\}$,

$\mathcal{B}_{-R}(L, S) \equiv \{r_c: L_i \to R' \mid R' \in \mathcal{FS}_c(S, L_i)\backslash\{S\backslash L_i\}, L_i \in \mathcal{G}(L)\}$.

b) L ≡ S: $L \notin \mathcal{G}(L)$: $\mathcal{C}_c(L, L) = \{r_c: L_i+L'' \to R' \mid L_i \in \mathcal{G}(L), L_i+L''+R' \in \mathcal{FS}_c(L), R' \neq \varnothing \text{ and } (L'' \neq \varnothing \text{ or } R' \subset L\backslash L_i)\}$.

The following theorem ensures that all rules can be non-repeatedly mined.

Theorem 4: For $L \subset S$: (a) *all rules in* $\mathcal{B}_{-R}(L, S)$, $\mathcal{B}_{-R+L}(L, S)$ *are non-repeatedly generated*, (b) *all rules in* $\mathcal{B}_{-R}(L, S)$, $\mathcal{B}_{-R+L}(L, S)$, $\mathcal{B}_c(L, S)$ *are quite different*, (c) $\mathcal{C}_c(L, S) = \mathcal{B}_{-R}(L, S) + \mathcal{B}_{-R+L}(L, S)$. Thus, $\mathcal{AR}_c(L, S) = \mathcal{B}_c(L, S) + \mathcal{B}_{-R}(L, S) + \mathcal{B}_{-R+L}(L, S)$.

For L≡S: $L \notin \mathcal{G}(L)$: $\mathcal{C}_c(L, L) = \{r_c: L_i+L'' \to R' \mid L_i \in \mathcal{G}(L), L_i+L''+R' \in \mathcal{FS}_c(L), R' \neq \varnothing \text{ and } (L'' \neq \varnothing \text{ or } R' \subset L\backslash L_i)\}$.

Proof: $L \subset S$: (a) By proposition 4.a, all rules in $\mathcal{B}_{-R}(L, S)$ are non-repeatedly generated. Assume that there exist two identical rules r_{cj}: $L_{ij}+R''_j \to R'_j\backslash R''_j \in \mathcal{B}_{-R+L}(L, S)$, j=1,2, $1 \leq i_1 < i_2$, then $L_{i1}+R''_1 = L_{i2}+R''_2$. Since $R''_1 \neq \varnothing$, $R''_2 \neq \varnothing$ and L_{i1}, L_{i2} are two different generators of L, $L_{i1} \subset L_{i2}+R''_2$: it contradicts to the selection of the index i_2!

(b) All rules in $\mathcal{B}_c(L, S)$, $\mathcal{B}_{-R}(L, S)$ and $\mathcal{B}_{-R+L}(L, S)$ are quite different because either their left-handed sides or right-handed sides are different.

(c) "$\supseteq$": For every r_c:$L_i \to R' \in \mathcal{B}_{-R}(L, S)$, where $R' \in \mathcal{FS}_c(S, L_i)$, $R' \subset S\backslash L_i$, $L_i \in \mathcal{G}(L)$, then $h_C(L_i) = L$ and by proposition 4.b, $h_C(L_i+R')=S$, so $r_c \in \mathcal{C}_c(L, S)$. For every r'_c:$L_i+R'' \to R'\backslash R'' \in \mathcal{B}_{R+L}(L, S)$, where $L_i \in \mathcal{G}(L)$, $R' \in \mathcal{FS}_c(S, L_i)$, $\varnothing \neq R'' \subset R'$, $R'' \subseteq R \cap L$, then $h_C(L_i+R'') = L$ and by proposition 4.b, $h_C(L_i+R''+(R'\backslash R'')) = h_C(L_i+R') = S$, so $r'_c \in \mathcal{C}_c(L, S)$.

"$\subseteq$": For every r_c:$L' \to R' \in \mathcal{C}(L, S)$: $h_C(L'+R')=S$, $h_C(L')=L$ and ($L' \neq L_i$ or $R' \neq S\backslash L_i$, $L_i \in \mathcal{G}(L)$). Let i be the minimum index such that $L'=L_i+L'' \in \mathcal{FS}_c(L)$, where $L_i \in \mathcal{G}(L)$, $L'' \in \mathcal{N}(L)$, $R' \subseteq S\backslash L'$. Since $R' \subseteq S\backslash L_i$ and $L'+R' \subseteq h_C(L_i+R') \subseteq h_C(L'+R')=S$, then $h_C(L_i+R')=S$, so $R'= R_k+R_k'+R^{\sim} \in \mathcal{FS}_c(S, L_i)$. If $R'=S\backslash L_i$, $L' \subseteq S\backslash R'=L_i$. So $L'=L_i$: it is absurd! Thus, $R' \subset S\backslash L_i$, i.e., $R' \in \mathcal{FS}_c(S, L_i)\backslash\{S\backslash L_i\}$. If $L''=\varnothing$, r_c:$L_i \to R' \in \mathcal{B}_{-R}(L, S)$. If $L'' \neq \varnothing$, $R:=R'+L'' \subseteq S\backslash L_i$, $h_C(L_i+R)=S$, so $R \in \mathcal{FS}_c(S, L_i)$, $R'=R\backslash L''$, $\varnothing \neq L'' \subseteq R \cap L$ and $L'' \subset R$; otherwise, the hypothesis $L_k \subset L_i+L''$, $\forall k$:$1 \leq k<i$ is similarly proved to the proof of a). Hence, r_c:$L_i+L'' \to R\backslash L'' \in \mathcal{B}_{-R+L}(L, S)$.

Based on definitions 5, 6 and theorem 4, the following algorithms are obtained in order to mine min-max restricted rules and to generate non-repeatedly all remaining consequence ones together with their confidences.

```
B_c(L, S)  MineBasis (L, S): L, S ∈ FLG_c: L ⊆ S.
{  B_c(L, S) = ∅;
   for each (L_i∈ G(L)) do  B_c(L, S) = B_c(L, S) + {r_b:L_i→S\L_i};
   return  B_c(L, S);
}
```

```
C_c(L, S)  MineCSet1 (L, S): L, S ∈ FLG_c: L ⊂ S.
{  B_-R(L, S) = ∅;   B_-R+L(L, S) = ∅;
   for each (L_i ∈ G(L)) do
      for each (R ∈ FS_c(S, L_i)\{S\L_i}) do
         B_-R(L, S) = B_-R(L, S) + {r_c:L_i→R};
```

```
    for each (i=1; Li ∈ G(L); i++) do
      for each (R ∈ FS_C(S, Li)) do
        for each (∅≠R' ⊆ L∩R and R'⊂ R) do
        { Duplicate = false;
          for each (k=1; k<i; k++) do
            if (Lk ⊂ Li+R') then { Duplicate = true; break; }
          if (not(Duplicate)) then
            B-R+L(L, S) = B-R+L(L, S) + {rc:Li+R'→R\R'};
        }
    return C_C(L, S) = B-R(L, S) + B-R+L(L, S);
}
```

```
C_C(L, L) MineCSet2 (L): L∈ FLG_C, L∉ G(L)
{   C_C(L, L) = ∅;
    for each (L' ∈ FS_C(L)) do
        for each (∅≠R' ⊆ L\L') do
            if (L' ∉ G(L) or L\R' ∉ G(L)) then
                C_C(L, L) = C_C(L, L) + {rc:L'→R'};
    return C_C(L, L);
}
```

Example 5. It is easy to see that $\mathcal{B}_{\mathcal{C}}$(ceg, aceg) = {e→acg, g→ace} and $\mathcal{B}_{\mathcal{C}}$(ceg, ceg) = {e→cg, g→ce}. Since $\mathcal{L}_e$= R*$_1$=a and $\tilde{\mathcal{L}}_e$=cg, $\mathcal{FS}_{\mathcal{C}}$(aceg, e) = {a+∅, a+c, a+g, a+cg}. Similarly, $\mathcal{FS}_{\mathcal{C}}$(aceg, g) = {a+∅, a+c, a+e, a+ce}. Hence, $\mathcal{B}_{\text{-R}}$(ceg, aceg) = {e→a, e→ac, e→ag} + {g→a, g→ac, g→ae} and $\mathcal{B}_{\text{R+L}}$(ceg, aceg) = {e+c→a, e+g→a, e+c→ag, e+g→ac, e+cg→a} + {g+c→a, g+c→ae}. The rules *g+e*→a, *g+e*→ac, *g+ce*→a are not generated repeatedly because their left-handed sides all contain generator e. Clearly, $\mathcal{B}_{\mathcal{C}}$(ceg, aceg) + $\mathcal{B}_{\text{-R}}$(ceg, aceg) + $\mathcal{B}_{\text{-R+L}}$(ceg, aceg) = $\mathcal{AR}_{\mathcal{C}}$(ceg, ceg). Moreover, $\mathcal{FS}_{\mathcal{C}}$(ceg) = {e, ec, eg, egc, g, gc}. Then, $\mathcal{C}_{\mathcal{C}}$(ceg, ceg) = {e→c, e→g} + {ec→g} + {eg→c}+ {g→e, g→c} + {gc→e}.

V. EXPERIMENTAL RESULTS

The following experiments were performed on a 2.93 GHz Pentium(R) Dual-Core CPU E6500 with 1.94GB of RAM, running Linux, Cygwin. Algorithms were coded in C++ (referenced to [19]). Four benchmark databases in [18] were used during these experiments. Table I shows their characteristics.

TABLE I. DATABASE CHARACTERISTICS

Database (DB)	# Transaction	# Items	Average size
Pumsb (Pu)	49046	7117	74
Mushroom (Mu)	8124	119	23
Connect (Co)	67557	129	43
Chess (Ch)	3196	75	37

We compare two approaches for mining restricted association rule:

- *Traditional approach*: (a) using algorithms of $\mathcal{C}$-*Charm* [1] and *MinimalGenerators* to mine from database the lattice $\mathcal{FLG}_{\mathcal{C}}$ of restricted frequent closed itemsets whenever $\mathcal{C}$ changes, then (b) mining restricted association rules by the algorithm of Pasquier et al. (that was corrected in [14]) based on $\mathcal{FLG}_{\mathcal{C}}$.
- *Our approach*: (a) using *MineCGCons* and *CreateLattice* to mine $\mathcal{FLG}_{\mathcal{C}}$ from $\mathcal{LG}_{\mathcal{A}}$ when $\mathcal{C}$ changes, (b) mining restricted association rules by applying *MineBasis*, *MineCSet1* and *MineCSet2*.

The items of the constraints are selected from the set $\mathcal{A}^{\mathcal{F}}$ of all frequent (according to the minimum support s_0) items of $\mathcal{A}$ with the ratios of ¼, ½ and ¾. The constraints have the sizes of l_1=¼*|$\mathcal{A}^{\mathcal{F}}$|, l_2=½*|$\mathcal{A}^{\mathcal{F}}$| and l_3=¾*|$\mathcal{A}^{\mathcal{F}}$|. In fact, the users are interested in the high-support items. Thus, we will sort all items by the ascending order of their supports. The constraint $\mathcal{C}$ with the size l_i is constructed from two subsets: $\mathcal{C} = \mathcal{C}_1+\mathcal{C}_2$. The first one contains [p*l_i] items randomly selected from the set of high-support items whose indices range from h to |$\mathcal{A}^{\mathcal{F}}$|, where: p ∈ [0; 1], and h ∈ {1, 2, .., |$\mathcal{A}^{\mathcal{F}}$|}. All [(1-p)*$l_i$] items of the remaining one are randomly selected from $\mathcal{A}^{\mathcal{F}}\backslash\mathcal{C}_1$. For the present experiments, we set p = 4/5, h = ⌊1/3*|$\mathcal{A}^{\mathcal{F}}$|⌋ and select two constraints for each l_i. Then, for each experiment, we consider six different constraints.

The following experiments compare the average (on six constraints) time for mining all restricted association rules by the traditional approach with the one by our approach. The used minimum supports (MS) and confidences (MC) according to each database are given in Fig. 4.a. Fig. 4.b shows that *our approach gets reductions in the mining time ranging from a factor of 40 to more than 1000 times!*

DB	MS = MC (%) 1	2	3
Co	96	95	94
Ch	87	86	85
Pu	91	90	89
Mu	30	28	26

(a) *Used MS and MC thresholds*

(b) *The reduction in the mining time by our approach*

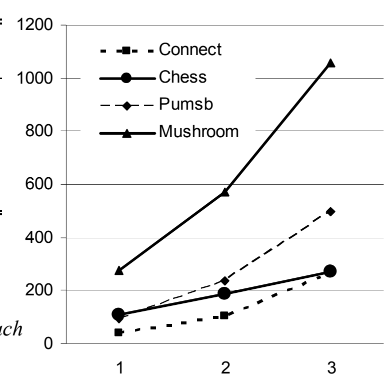


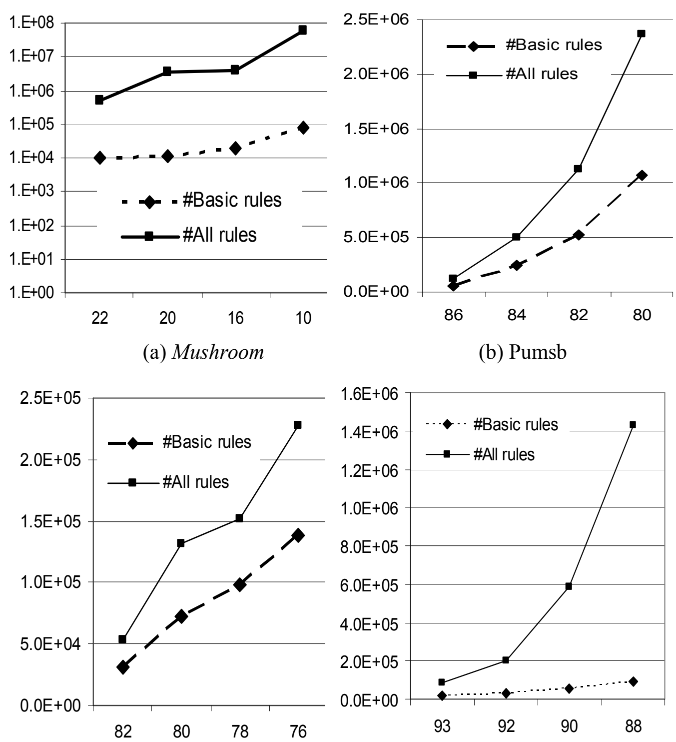
Figure 4. The reduction in the time for mining all restricted association rules by our approach compared to the traditional one.

Remind that one can select to mine the rule class that they are interested in. Using *MineBasis*, we can mine its basis. The basis contains min-max basic rules, which are useful. When necessary, the set of all remaining consequence ones can be quickly generated from the basic ones, using *MineCSet1* or *MineCSet2*. This set also splits into the different subsets. Then, our approach is very efficient and close to user! For low minimum supports and confidences, users can still get the rules that they take care of. For example, *we can mine all rules in the rule class determined by them as well as only basic rules*. This may not be possible in the traditional approach.

In the remaining experiments, we will prove that: *spending only a little of time, we can mine min-max basis with the small* *size* from lattice $\mathcal{FLG}_C$. Table II shows that the average time (in seconds) for mining min-max basic rules T_B is much less than the one for mining all rules T_A, where: MS is equal to MC and they are written in column M, the percent ratio of T_B to T_A is shown in column R_T. If only mining the basis, we can *save* the amounts of the *mining time* (in percents) ranging from 93.0% to 99.9%. Furthermore, Fig. 5 shows that the cardinality (the average number on constraints) of the min-max basis is also less than the one of the rule set. For database Mushroom (MS=MC=10), we can get *the reduction in the number of rules* ranging up to more than *500 times*!

TABLE II. MINING TIMES: BASIC RULES VS. ALL RULES

DB	M	T_A (s)	T_B	R_T (%)	DB	M	T_A	T_B	R_T
Mu	22	5.1	0.03	0.6	Co	93	1.6	0.1	4.6
	20	30.1	0.04	0.1		92	3.4	0.1	3.7
	16	36.9	0.08	0.2		90	8.8	0.3	2.9
	10	592.1	0.35	0.1		88	19.6	0.4	2.2
Pu	86	3.9	0.2	5.5	Ch	82	1.8	0.1	6.7
	84	17.2	0.9	5.5		80	4.3	0.3	6.8
	82	38.3	2.1	5.5		78	5.5	0.4	7.0
	80	79.7	4.4	5.5		76	8.0	0.6	7.0



(a) *Mushroom* (b) Pumsb

(c) Chess (d) *Connect*

Figure 5. The number of rules: basic vs. all.

VI. CONCLUSION

Thanks to the model of mining frequent closed itemsets with constraints proposed in [1], we consider mining association rules restricted on a given itemset *C*. First, this paper proposes the algorithm to create the lattice structure on the class of frequent closed itemsets restricted on *C*. Due to this lattice and recent works [14, 15] of mining association rules based on the lattice of frequent closed itemset, we suggest the efficient solution for the problem. The restricted association rule set is partitioned into disjoint equivalence classes. Each class splits into the subsets of basic and consequence. Spending only a little of time, we can mine min-max basic set with the small size that is useful for users. When necessary, all remaining consequence ones together with their confidences can be quickly generated from that basis.

REFERENCES

[1] T.N. Anh, D.V. Hai, T.C. Tin, and L.H. Bac, "Efficient Algorithms for Mining Frequent Itemsets with Constraint," in Proceedings of the third international conference on knowledge and systems engineering, 2011 (to appear, http://fit.hanu.edu.vn/kse2011/acceptedpapers.html).

[2] R. Agrawal, T. Imielinski, and N. Swami, "Mining association rules between sets of items in large databases," in Proceedings of the ACM SIGMOID, pp. 207-216, 1993.

[3] R.J. Bayardo, R. Agrawal, and D. Gunopulos, "Constraint-Based Rule Mining in Large, Dense Databases," Data Mining and Knowledge Discovery, Kluwer Academic Pub., vol. 4, No. 2/3, pp. 217–240, 2000.

[4] H.T. Bao, "An approach to concept formation based on formal concept analysis," IEICE Trans. Infor. and systems, E78-D, no. 5, 1995.

[5] G. Cong and B. Liu, "Speed-up Iterative Frequent Itemset Mining with Constraint Changes," ICDM, pp. 107-114, 2002.

[6] J. Han, J. Pei, and J. Yin, "Mining frequent itemsets without candidate generation," in Proceedings of SIGMOID'00, pp. 1-12, 2000.

[7] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," Data mining and knowledge discovery, no 8, pp. 53-87, 2004.

[8] A.J. Lee, W.C. Lin, and C.S.Wang, "Mining Association rule with multi-dimensional constraints," Journal of Systems and Software, no. 79, pp. 79-92, 2006.

[9] R.T. Nguyen, V.S. Lakshmanan, J. Han, and A. Pang, "Exploratory Mining and Pruning Optimizations of Constrained Association Rules," in Proceedings of the 1998 ACM-SIG-MOD Int'l Conf. on the Management of Data, pp. 13-24, 1998.

[10] N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, and L. Lakhal, "Generating a condensed representation for association rules," J. of Intelligent Information Systems, vol. 24, no. 1, pp. 29-60, 2005.

[11] J. Pei, J. Han, and R. Mao, "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets," in Proceedings of the DMKDWorkshop on Research Issues in Data Mining and Knowledge Discovery, pp. 21–30, 2000.

[12] J. Pei, J. Han, and V.S. Lakshmanan, "Pushing Convertible Constraints in Frequent Itemset Mining," Data Mining and Knowledge Discovery, no 8, pp. 227–252, 2004.

[13] R. Srikant, Q. Vu, and R. Agrawal, "Mining association rules with item constraints," in Proceeding KDD'97, pp. 67-73, 1997.

[14] T.C. Tin and T.N. Anh, "Structure of set of association rules based on concept lattice," SCI 283, Adv. in Intelligent Infor. and Database Systems, Springer-Verlag, Berlin Heidelberg, pp. 217-227, 2010.

[15] T.C. Tin, T.N. Anh, and T. Thong, "Structure of Association Rule Set based on Min-Min Basic Rules," in Proceedings of the 2010 IEEE-RIVF International Conference on Computing and Communication Technologies, pp. 83-88, 2010.

[16] M.J. Zaki and C.J. Hsiao, "Efficient algorithms for mining closed itemsets and their lattice structure," IEEE Trans. Knowledge and data engineering, vol. 17, no. 4, pp. 462-478, 2005.

[17] M.J. Zaki, "Mining non-redundant association rules," Data mining and knowledge discovery, no. 9, pp. 223-248, 2004.

[18] Frequent Itemset Mining Dataset Repository (FIMDR), http://fimi.cs.helsinki.fi/data/, acessed 2009.

[19] http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software#patutils, acessed 2010.