Efficient Algorithms for Mining Frequent Itemsets with Constraint

Conference Paper · October 2011

DOI: 10.1109/KSE.2011.12 · Source: DBLP

4 authors:

Anh Tran
University of Dalat

Hai Duong
University of Dalat, Vietnam

Tin C Truong
University of Dalat

Bac Le
Ho Chi Minh City University of Science

2011 The Third International Conference on Knowledge and Systems Engineering

Efficient Algorithms for Mining Frequent Itemsets with Constraint

Anh N. Tran
University of Dalat,
Dalat, Vietnam
anhtn@dlu.edu.vn

Hai V. Duong
University of Dalat,
Dalat, Vietnam
haidv@dlu.edu.vn

Tin C. Truong
University of Dalat,
Dalat, Vietnam
tintc@dlu.edu.vn

Bac H. Le
University of Natural Science
Ho Chi Minh, Vietnam
lhbac@fit.hcmus.edu.vn

***Abstract*—An important problem of interactive data mining is "to find frequent itemsets contained in a subset $\mathcal{C}$ of set of all items on a given database". Reducing the database on $\mathcal{C}$ or incorporating it into an algorithm for mining frequent itemsets (such as Charm-L, Eclat) and resolving the problem are very time consuming, especially when $\mathcal{C}$ is often changed. In this paper, we propose an efficient approach for mining them as follows. Firstly, it is necessary to mine only one time from database the class $\mathcal{LG}_{\mathcal{A}}$ containing the closed itemsets together their generators. After that, when $\mathcal{C}$ is changed, the class of all frequent closed itemsets and their generators on $\mathcal{C}$ is determined quickly from $\mathcal{LG}_{\mathcal{A}}$ by our algorithm MINE_CG_CONS. We obtain the algorithm MINE_FS_CONS to mine and classify efficiently all frequent itemsets with constraint from that class. Theoretical results and the experiments proved the efficiency of our approach.**

Closed itemsets, frequent itemsets, constraint, generators, eliminable itemsets

I. INTRODUCTION

Currently, Internet makes the real changes in the ways human thinks and does. People access to Internet to get useful information from it. Normally, the data of websites for users is obtained and is saved in the tables (or databases). The number of attributes (items) is often enormous. However, for a while, they only take care of a set of attributes (called the constraint). To show immediately to the users the knowledge mined from them such as the frequent itemsets or association rules is very important. In [3, 5, 8, 9, 13, 14] some authors researched on mining frequent itemsets and association rules from the standpoint of the user's interaction with the system. *They studied mining frequent itemsets with many different kinds of constraints*. Nguyen et al. [9] proposed an architect including domain, class and SQL-style aggregate constraints. Some categories of constraints such as anti-monotone, monotone, and succinct have been integrated into the mining process [9]. In [13], Pei et al. proposed the concept of convertible constraints and considered pushing them into the mining progress of the FP-growth algorithm. Srikant et al. [14] considered the problem of integrating constraints that are Boolean expressions over the presence or absence of items in the association rules. Bayardo et al. [3] restricted the problem of mining association rules in two constraints of the consequent and the minimum improvement. In [5], Cong and Liu proposed an technique based on the concept of tree boundary to utilize previous mining results for reducing the mining time. They considered tightening and relaxing constraints such as increasing and decreasing supports.

This paper concentrates on solving the problem of to find frequent itemsets contained in a subset $\mathcal{C}$ of the set of all items from a given database (called the problem of mining frequent itemsets with constraint $\mathcal{C}$). A simple approach is to reduce the database on $\mathcal{C}$ and after that to mine them by an algorithm used widely such as Apriori [1], *Eclat* [20], *Charm-L* [18], FP-growth [8], etc. This approach is not efficient because the constraints are often changed. A different one is to filter the output of those algorithms in a post-processing step to determine frequent itemsets with constraint. It is also not efficient because their outputs are usually enormous. In [14], Srikant et al. showed that we should incorporate $\mathcal{C}$ into the mining process. They modified the apriori candidate generation procedure to only count candidates that contain selected items. However, there are still many candidates generated. Furthermore, when $\mathcal{C}$ is changed, users must run the algorithm. It is very time consuming. In [8], Han et al. also suggested incorporating $\mathcal{C}$ into the mining FP-Tree. However, they did not propose the algorithm to do it. We also incorporate constraint $\mathcal{C}$ into Charm-L and Eclat (well-known algorithms for mining frequent closed itemsets and all frequent itemsets) to mine frequent itemsets with constraint. This approach will be compared to our approach.

Recently, we [15, 16] showed the structure of each class of equivalent frequent itemsets having the same closure based on the generators and eliminable subsets of that closure. Based on it, we approach to the problem of mining frequent itemsets with constraint as follows. Firstly, it is necessary to mine only one time from the database the class $\mathcal{LG}_{\mathcal{A}}$ containing the closed itemsets and their generators. After that, when $\mathcal{C}$ is changed, the class $\mathcal{FLG}_{\mathcal{C}}$ of frequent closed itemsets and their generators is determined quickly from $\mathcal{LG}_{\mathcal{A}}$. Using a unique representation of frequent itemset, we derive completely, directly, non-repeatedly from $\mathcal{FLG}_{\mathcal{C}}$ all frequent itemsets with constraint. The mining script is shown as follows:

1) *To mine only one time the class $\mathcal{LG}_{\mathcal{A}}$*

978-0-7695-4567-7/11 $26.00 © 2011 IEEE
DOI 10.1109/KSE.2011.12

2) *User selects the constraint $\mathcal{C}$ and the minimum support threshold s:*

   (2.1) to determine quickly the class $\mathcal{FLG}_{\mathcal{A}}^{s}$ of the frequent closed itemsets (and their generators) with respect to s from $\mathcal{LG}_{\mathcal{A}}$ in the first mining time; otherwise, from $\mathcal{FLG}_{\mathcal{A}}^{s_{max}}$ (was saved before), where s_{max} is maximum such that $s_{max} \leq s$,

   (2.2) from $\mathcal{FLG}_{\mathcal{A}}^{s}$, our algorithm MINE_CG_CONS (based on the propositions 2 and 3) is used to exploit directly the class $\mathcal{FLG}_{\mathcal{C}}$,

   (2.3) from $\mathcal{FLG}_{\mathcal{C}}$ the class $\mathcal{FS}_{\mathcal{C}}$ of all frequent itemsets with constraint $\mathcal{C}$ is mined quickly by our algorithm MINE_FS_CONS (using theorem 4).

3) *Return the step 2.*

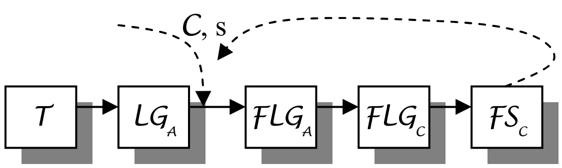

Figure 1. Mining frequent itemsets with constraint.

The method is suitable when $\mathcal{C}$ is usually changed. Indeed, the size of the class of all frequent closed itemsets and their generators is much smaller than the one of all frequent itemsets, especially on the dense databases. With the small values of minimum support thresold, this class can be still mined and saved in main memory by Charm-L and MinimalGenerators [18]. The response of the system in early times is often slow because the size of $\mathcal{LG}_{\mathcal{A}}$ is big. After the first period, the classes of $\mathcal{FLG}_{\mathcal{A}}^{s_1}$, $\mathcal{FLG}_{\mathcal{A}}^{s_2}$, .., $\mathcal{FLG}_{\mathcal{A}}^{s_n}$ are saved in the system (the values of s_j, j=1, .., n are distributed regularly on [0, 1], $0 \leq s_1 < s_2 < .. < s_n \leq 1$). Thus, we only need to select the frequent closed itemsets and generators that those supports exceed threshold s directly on $\mathcal{FLG}_{\mathcal{A}}^{s_{max}}$, where $s_{max} = \max \{s_j \mid s_j \leq s, j=1, .., n\}$. For the threshold s given by user, the corresponding threshold s_{max} is usually closed to it. Therefore, the time to exploit $\mathcal{FLG}_{\mathcal{A}}^{s}$ is often small. Using some simple operators on the itemsets in $\mathcal{FLG}_{\mathcal{A}}^{s}$, we can mine directly the class $\mathcal{FLG}_{\mathcal{C}}$ of the frequent closed itemsets and generators restricted on $\mathcal{C}$. The class $\mathcal{FS}_{\mathcal{C}}$ is partitioned into the disjoint equivalence classes. Each class contains frequent itemsets on $\mathcal{C}$ that have the same closure. So, they can be mined concurrently by parallel algorithms. A unique representation of frequent itemsets based on frequent closed itemset (represented to that class) and its generators is indicated to derive directly, non-repeatedly all frequent itemsets on $\mathcal{C}$ from $\mathcal{FLG}_{\mathcal{C}}$.

The rest of the paper is organized as follows. Section 2 recalls some primitive concepts and results. This section also proposes a unique representation of itemsets. The main results are in section 3. In it, we obtain the algorithm for determining quickly all frequent closed itemsets and their generators with constraint $\mathcal{C}$. The efficient algorithm to mine all frequent itemsets from them is also figured out. Sections 4 and 5 contain the experimental results and conclusions.

II. Primitive Concepts and Results

A. *Primitive concepts*

Given set $\mathcal{O}$ contained records or transactions of a database $\mathcal{T}$ and $\mathcal{A}$ contained attributes or items related to each of transaction $o \in \mathcal{O}$ and $\mathcal{R}$ is a binary relation in $\mathcal{O} \times \mathcal{A}$. Consider two operators: $\lambda: 2^{\mathcal{O}} \to 2^{\mathcal{A}}$, $\rho: 2^{\mathcal{A}} \to 2^{\mathcal{O}}$ determined as ($\rho(\varnothing) := \mathcal{O}$, $\lambda(\varnothing) := \mathcal{A}$):

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

Defining closure operator h in $2^{\mathcal{A}}$ [4] by: $h = \lambda \circ \rho$, we say that h(A) is the closure of itemset $A \subseteq \mathcal{A}$. If A = h(A), A is called *closed itemset*. The class of all closed itemsets is denoted as $\mathcal{CS}$. The *support* of itemset A is defined as the probability of the ocurrence of a transaction containing A on $\mathcal{O}$: $s(A) = |\rho(A)|/|\mathcal{O}|$. Denoted 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* [1]. Let $\mathcal{FS}$, $\mathcal{FCS}$ be the classes of all frequent itemsets and all frequent closed itemsets.

For two non-empty itemsets G, A: $\varnothing \neq G \subseteq A \subseteq \mathcal{A}$, G is called a *generator* of A [11] iff: $h(G)=h(A)$ and $(\forall \varnothing \neq G' \subset G \Rightarrow h(G') \subset h(G))$. Let $\mathcal{G}$(A) be the class of all generators of A. Let $\mathcal{LG}_{\mathcal{A}}$ and $\mathcal{FLG}_{\mathcal{A}}$ [1] be the classes of all closed itemsets together their generators on $\mathcal{A}$ and all elements in $\mathcal{LG}_{\mathcal{A}}$ that are frequent with respect to s_0. In $2^{\mathcal{A}}$, an itemset R is called eliminable in S [15, 16] iff $R \subset S$ and $\rho(S) = \rho(S \setminus R)$. Let $\mathcal{N}$(S) denote the class of all *eliminable itemsets* in S, $\mathcal{N}^*(S) := \mathcal{N}(S) \setminus \{\varnothing\}$, we have [15]: $\mathcal{N}(S) = \{A: A \subseteq S \setminus G, G \in \mathcal{G}(S)\}$.

TABLE I. DATABASE 1

Trans ID	Items
1	aceg
2	acfh
3	adfh
4	bceg

Example 1. Let us consider database 1 in Table I, with minimum support $s_0 = 1/4$, used in all next examples of this paper. From the definitions of λ and ρ, we have: $\lambda(\{1, 4\})$ = ceg, ρ(ceg) = {1, 4} and then, h(ceg)=ceg. So ceg is a frequent closed itemset with the support $|\rho(\text{ceg})|/|\mathcal{O}| = 1/2$.

[1] For briefly, we write $\mathcal{FLG}_{\mathcal{A}}^{s}$ simply $\mathcal{FLG}_{\mathcal{A}}$.

978-0-7695-4567-7/11 $26.00 © 2011 IEEE
DOI 10.1109/KSE.2011.12

This itemset contains two generators e, g because h(e)=h(g)=h(ceg)=ceg.

B. Structure of Itemsets

In this part, the class of all itemsets is partitioned into the disjoint equivalence classes. The elements of an equivalence class have the same closure and can be derived from that closure and its generators. Thus, we only need to mine the class (with the small size) of frequent closed itemsets (and generators). When it is necessary, we can derive to users the frequent itemsets in a class that they are interested in.

Definition 1 [15] *(Equivalence relation $\sim_h$ over the class of all itemsets $2^{\mathcal{A}}$)*: $\forall$ A, B $\in 2^{\mathcal{A}}$:

$$A \sim_h B \Leftrightarrow h(A) = h(B).$$

Theorem 1 [15] *(A partition of $2^{\mathcal{A}}$)*: Relation $\sim_h$ partitions $2^{\mathcal{A}}$ into the disjoint equivalence classes. Each class contains itemsets that have the same closure. The equivalence class containing A is denoted as [A].

$$2^{\mathcal{A}} = \sum_{A\in\mathcal{CS}} [A] \text{ and } \mathcal{FS} = \sum_{A\in FCS} [A] .$$

Based on this partition, we can exploit independently each equivalence class. The elements in a class have the same support so we only compute and save it once.

Theorem 2 [15] *(Representation of itemset)*: For every itemset A such that $\emptyset \neq A \in \mathcal{CS}$:

$$X \in [A] \Leftrightarrow \exists G_0 \in \mathcal{G}(A), \exists X' \in \mathcal{N}(A): X = G_0 + X'.$$ [2]

Denoted $\mathcal{M}(S, G) := \{A: A \subseteq S\backslash G, G \in \mathcal{G}(S)\}$, it is obvious that: $\mathcal{N}(S) = \bigcup_{G\in\mathcal{G}(S)} \mathcal{M}(S,G)$. For G_1, $G_2 \in \mathcal{G}(S)$, $G_1 \neq G_2$, the intersection of $\mathcal{N}(S, G_1)$ and $\mathcal{N}(S, G_2)$ can be not empty. The above representation of a itemset in a class can be not unique because the representation of an eliminable itemset is not unique.

Example 2. Let us consider equivalence class [X], where X=aceg, $\mathcal{G}$(X) = {ae, ag}. We have: $\mathcal{N}$*(X, ae) = {cg, c, g}, $\mathcal{N}$*(X, ag)={ce, c, e}, $\mathcal{N}$*(X, ae)$\cap\mathcal{N}$*(X, ag) = {c} $\neq \emptyset$ and $\mathcal{N}$*(X) = $\mathcal{N}$*(X, ae)$\cup\mathcal{N}$*(X, ag) = {cg, c, g, e, ce}. Then, from theorem 2, [X] = {ae, aeg, aec, aecg, ag, agc}, in which, aeg can be represented by two ways: aeg = ae+g = ag+e.

C. Unique Representation of Itemset by Generator and Eliminable Itemset

The process of deriving of all itemsets of an equivalence class using theorem 2 can make the duplication because the representation of an itemset is not unique. Theorem 3 shows a unique representation of itemset, in other words, based on it, all itemsets in the same class can be derived non-repeatedly (as a result, quickly), completely.

$\forall X \in \mathcal{CS}$, let us call $X_U = \bigcup_{X_i\in\mathcal{G}(X)} X_i$, $X_{U,i}=X_U\backslash X_i$, $X\_=X\backslash X_U$, $\mathcal{IS}(X) := \{X'=X_i+X'_i+X^{\sim} \mid X_i\in\mathcal{G}(X), X^{\sim}\subseteq X\_, X'_i\subseteq X_{U,i}$, i=1 or (i>1: $X_k\not\subset X_i+X'_i$, $\forall$k: 1≤k<i)}.

Theorem 3 (Unique representation of itemset by generator and eliminable itemset): We have:

a. *[X] = $\mathcal{IS}$(X).*

b. *All itemsets of $\mathcal{IS}$(X) are derived non-repeatedly.*

Proof:

(a) ."$\subseteq$": If X'$\in$[X], assume that i is the minimum index such that $X_i\in\mathcal{G}(X)$, $X''_i\subseteq X\backslash X_i$ and $X' = X_i+X''_i$. Let $X'_i = X''_i\cap X_U$, $X^{\sim} = X''_i\backslash X_U$, then $X'_i\subseteq X_{U,i}$, $X^{\sim} = X'\backslash X_U \subseteq X\_$ and $X' = X_i+X'_i+X^{\sim}$. Assume that there exists the index k such that 1≤k<i, $X_k\in\mathcal{G}(X)$, $X_k\subset X_i+X'_i$ then $X'=X_k+X''_k$, where $X''_k=X'_k+X^{\sim}$ and $X'_k=(X_i+X'_i)\backslash X_k\subseteq X\backslash X_k$, $X^{\sim}\subseteq X\backslash X_k$. Thus, $X''_k\subseteq X\backslash X_k$: it is absurd!

."$\supseteq$": If $X'\in\mathcal{IS}(X)$, there exists $X_i\in\mathcal{G}(X)$, $X^{\sim}\subseteq X\_\subseteq X\backslash X_i$, $X'_i\subseteq X_{U,i}\subseteq X\backslash X_i$: $X'=X_i+X'_i+X^{\sim}$. Let $X''= X'_i+X^{\sim}\in\mathcal{N}(X)$, then $X'=X_i+X''$, thus X'$\in$[X] by theorem 2.

(b) Assume that there exists i, k such that i>k≥1 and $X_i+X'_i+X^{\sim}_i \equiv X_k+X'_k+X^{\sim}_k$, where: X_i, $X_k \in\mathcal{G}(X)$; $X^{\sim}_i$, $X^{\sim}_k\subseteq X\_$; $X'_i\subseteq X_{U,i}$, $X'_k\subseteq X_{U,k}$. Since $X_k\cap X^{\sim}_i=\emptyset$, so $X_k\subset X_i+X'_i$ (the equality do not occur because X_i and X_k are two different generators of X). It contradicts to the selection of index i! □

Example 3. Let us consider class [X] where X=aceg, $\mathcal{G}$(X) = {X_1=ae, X_2=ag}, we have: X_U=aeg, $X_{U,1}$=g, $X_{U,2}$=e, X_=c. By theorem 3, itemset X'=aceg$\in\mathcal{IS}$(X) is generated uniquely as follows: $X'=X_1+X'_1+X^{\sim}$ where $X'_1=g \subseteq X_{U,1}$, $X^{\sim}$=c $\subseteq$X_. By theorem 2, X' has two duplicate representations: X'=ae+cg=ag+ce. If the condition "i>1: $X_k\not\subset X_i+X'_i$, $\forall$k: 1≤k<i" is absent, then duplicate X' is generated once again: $X'=X_2+X'_2+X^{\sim}$, where $X'_2=e \subseteq X_{U,2}$ and $X_1\subset X_2+X'_2$. Similarly, all itemsets of [X] = $\mathcal{IS}$(X) = {ae, aeg, aegc, aec, ag, agc} are derived non-repeatedly.

From theorem 3, the algorithm ***GEN_ITEMSETS*** is obtained (see Fig. 2) to generate *non-repeatedly all itemsets in each equivalence class* [X], X$\in\mathcal{CS}$.

III. MINING FREQUENT ITEMSETS WITH CONSTRAINT

As the discussion in introduction, to mine frequent itemsets on $\mathcal{C}$ with minimum support threshold s_0, firstly, without the general, we need to determine the class $\mathcal{FLG}_{\mathcal{A}}$ of all frequent (with respect to s_0) closed itemsets and their generators from $\mathcal{LG}_{\mathcal{A}}$. After that, it is quickly to mine the class $\mathcal{FLG}_{\mathcal{C}}$ of all frequent closed itemsets and generators restricted on $\mathcal{C}$ from $\mathcal{FLG}_{\mathcal{A}}$. That bases on some relations between closed itemsets and generators of $\mathcal{FLG}_{\mathcal{C}}$ and the corresponding ones of $\mathcal{FLG}_{\mathcal{A}}$. Finally, the partition of the

[2] The symbol + is denoted as the union of two disjoint sets.

978-0-7695-4567-7/11 $26.00 © 2011 IEEE
DOI 10.1109/KSE.2011.12

IEEE Computer Society

class of all frequent itemsets with constraint $\mathcal{C}$ allows us to use the algorithm *GEN_ITEMSETS* for deriving quickly them.

```
<IS(X), s(X)>  GEN_ITEMSETS (X, s(X), G(X)):
IS(X) = ∅; X_U = ⋃_{X_i∈G(X)} X_i ; X_ = X\X_U;
for each (i=1; X_i∈G(X); i++) do {
    X_{U,i} = X_U\X_i;
    for each (X'_i ⊆ X_{U,i}) do {
        IsDuplicate = false;
        for (k=1; k<i; k++) do
            if (X_k ⊂ X_i+X'_i) then {
                IsDuplicate = true; break;
            }
        if (not(IsDuplicate)) then
            for each (X~ ⊆ X_) do
                IS(X) = IS(X) + {X_i+X'_i+X~};
    }
}
return <IS(X), s(X)>;
```

Figure 2. GEN_ITEMSETS, the algorithm to generate non-repeatedly all itemsets in class [X].

A. *Mining frequent closed itemsets and their generators with constraint*

We will define again the operators λ, ρ and h over constraint $\mathcal{C}$ and figure out the relation between them with the corresponding ones over $\mathcal{A}$. From that, the algorithm for mining quickly $\mathcal{FLG}_{\mathcal{C}}$ from $\mathcal{LG}_{\mathcal{A}}$ is indicated.

Definition 2 (The Galois connection operators over constraint $\mathcal{C}$): For every $\mathcal{C}\in 2^{\mathcal{A}}\backslash\{\varnothing\}$, let us consider operators: $\rho_{\mathcal{C}}: 2^{\mathcal{C}}\to 2^{\mathcal{O}}$, $\lambda_{\mathcal{C}}: 2^{\mathcal{O}}\to 2^{\mathcal{C}}$ and $h_{\mathcal{C}}: 2^{\mathcal{C}}\to 2^{\mathcal{C}}$ defined as follows: $\forall\varnothing\neq C'\subseteq\mathcal{C}$, $\varnothing\neq O\subseteq\mathcal{O}$,

$$\rho_{\mathcal{C}}(C') = \{o\in\mathcal{O}: (o, a)\in\mathcal{R},\ \forall a\in C'\},\ \rho_{\mathcal{C}}(\varnothing) := \mathcal{O},$$
$$\lambda_{\mathcal{C}}(O) = \{a\in\mathcal{C}: (o, a)\in\mathcal{R},\ \forall o\in O\},\ \lambda_{\mathcal{C}}(\varnothing) := \mathcal{C},$$
$$h_{\mathcal{C}} = \lambda_{\mathcal{C}}\ o\ \rho_{\mathcal{C}}.$$

An itemset $C'\subseteq\mathcal{C}$ is called closed itemset on $\mathcal{C}$ iff $h_{\mathcal{C}}(C') = C'$. The class of all frequent itemsets on $\mathcal{C}$ is denoted as $\mathcal{FS}_{\mathcal{C}}$. The class $\mathcal{FCS}(\mathcal{C})$ contains all frequent closed itemsets on $\mathcal{C}$.

Proposition 1: For every $\mathcal{C}\in 2^{\mathcal{A}}\backslash\{\varnothing\}$, $\varnothing\neq C'\subseteq\mathcal{C}$, $O\subseteq\mathcal{O}$, the following statements are true:

a. $\rho_{\mathcal{C}}(C') = \rho(C')$, so $s(C') = |\rho(C')| = |\rho_{\mathcal{C}}(C')|$,

b. $\lambda_{\mathcal{C}}(O) = \lambda(O)\cap\mathcal{C}$,

c. $h_{\mathcal{C}}(C') = h(C')\cap\mathcal{C}$.

Proof: Obviously from definitions of ρ, $\rho_{\mathcal{C}}$, λ, $\lambda_{\mathcal{C}}$, h and $h_{\mathcal{C}}$. □

Proposition 1.c enables us to determine frequent closed itemset on $\mathcal{C}$ by intersecting $\mathcal{C}$ with each frequent closed itemset (on $\mathcal{A}$) of $\mathcal{FLG}_{\mathcal{A}}$. This way can make the duplications. In other words, a frequent closed itemset on $\mathcal{C}$ can be derived many times.

Example 4. Let us consider database 1, we have: $\mathcal{FLG}_{\mathcal{A}}$={acfh, aceg, adfh, bceg, *afh*, ceg, *ac*, c, *a*}. Then, by proposition 1.c, with $\mathcal{C}$=abde, $\mathcal{FLG}_{\mathcal{C}}$={ae, ad, be, e, <u>a</u>}. Some of its elements are derived many times, for example: <u>a</u> $= acfh\cap\mathcal{C} = afh\cap\mathcal{C} = ac\cap\mathcal{C} = a\cap\mathcal{C}$.

Based on a condition over generators, proposition 2 is obtained to eliminate the duplication in generating frequent closed itemsets restricted on $\mathcal{C}$.

Proposition 2 (Generating non-repeatedly all frequent closed itemsets with constraint $\mathcal{C}$*):* Let us call $\mathcal{FCS}_{\mathcal{C}}$:= $\{C'=L\cap\mathcal{C} \mid L\in\mathcal{FCS}, \exists L_i\in\mathcal{G}(L): L_i\subseteq\mathcal{C}\}$, we have:

a. $\mathcal{FCS}_{\mathcal{C}} = \mathcal{FCS}(\mathcal{C})$.

b. *All elements of* $\mathcal{FCS}_{\mathcal{C}}$ *are generated non-repeatedly.*

Proof: a. **"⊆"**: $\forall\varnothing\neq C'\in\mathcal{FCS}_{\mathcal{C}}$: $C' = L\cap\mathcal{C}\subseteq\mathcal{C}$, $h(C')\subseteq h(L) = L$. Then, $C'\subseteq h_{\mathcal{C}}(C') = h(C')\cap\mathcal{C}\subseteq L\cap\mathcal{C} = C'$. Therefore $C' = h_{\mathcal{C}}(C')$, i.e., $C'\in\mathcal{FCS}(\mathcal{C})$.

"⊇": $\forall\varnothing\neq C'\in\mathcal{FCS}(\mathcal{C})$: $C' = h_{\mathcal{C}}(C') = h(C')\cap\mathcal{C} = L\cap\mathcal{C}$, where $L := h(C')\in\mathcal{FCS}$. Let $C_i\in\mathcal{G}(C')$ (there always exists), then: $h(C_i)=h(C')=L=h(L)$ and $\forall C_0\subset C_i$ then $h(C_0)\subset h(C_i) = L$. Thus, $C_i\in\mathcal{G}(L)$ and $C_i\subseteq C'\subseteq\mathcal{C}$, i.e., $C'\in\mathcal{FCS}_{\mathcal{C}}$. We conclude that $C'\in\mathcal{FCS}_{\mathcal{C}}$.

b. Assume that, with k=1, 2, $C'_k = L_k\cap\mathcal{C}\in\mathcal{FCS}_{\mathcal{C}}$, let $L_k\in\mathcal{FCS}$, $L_{k,0}\in\mathcal{G}(L_k)$, $L_{k,0}\subseteq\mathcal{C}$ such that $C'_1\equiv C'_2$ and $L_1\neq L_2$. We have $L_{k,0}\subseteq L_k\cap\mathcal{C}$, $L_k = h(L_{k,0})\subseteq h(L_k\cap\mathcal{C})\subseteq h(L_k) = L_k$. Therefore, $h(C'_k) = L_k$, $\forall k=1, 2$ and $L_1 = L_2$: it is absurd! □

In the next step, we will show how to determine the generators of frequent closed itemsets on $\mathcal{C}$.

Definition 3 (The generators of C' restricted on $\mathcal{C}$): For every G, C': $\varnothing\neq G\subseteq C'\subseteq\mathcal{C}$, G is called a generator of C' on $\mathcal{C}$ iff:

$$h_{\mathcal{C}}(G)=h_{\mathcal{C}}(C') \text{ and } (\forall\varnothing\neq G'\subset G\Rightarrow h_{\mathcal{C}}(G')\subset h_{\mathcal{C}}(G)).$$

The set of all generators of C' on $\mathcal{C}$ is denoted as $\mathcal{G}_{\mathcal{C}}(C')$.

Proposition 3 (Determining the generators with constraint $\mathcal{C}$*):* $\forall C'=L\cap\mathcal{C}\in\mathcal{FCS}_{\mathcal{C}}$:

$$\mathcal{G}_{\mathcal{C}}(C') = \mathcal{G}(C') = \{L_i\in\mathcal{G}(L): L_i\subseteq C'\}.$$

Proof: It is obvious because $\rho_{\mathcal{C}}(C') = \rho(C')$. □

From propositions 2 and 3, the algorithm ***MINE_CG_CONS*** is indicated to *mine quickly the class*

$$\mathcal{FLG}_{\mathcal{C}} := \{<C', s(C'), \mathcal{G}_{\mathcal{C}}(C')> \mid C'\in\mathcal{FCS}_{\mathcal{C}}\}$$

from $\mathcal{LG}_{\mathcal{A}}$.

978-0-7695-4567-7/11 $26.00 © 2011 IEEE
DOI 10.1109/KSE.2011.12

```
FLG_C  MINE_CG_CONS (LG_A, C, s0):
FLG_C = ∅;
for each (<L, s(L), G(L)> ∈ LG_A) do
  if (s(L) ≥ s0) then    // L ∈ FLG_A
    if (∃ Li ∈ G(L) and C ⊇ Li) then {
      // not to generate repeatedly
      C'      = L ∩ C;
      G_C(C') =  {Li ∈ G(L) | Li ⊆ C'};
      FLG_C = FLG_C + <C', s(L), G_C(C')>;
    }
return FLG_C;
```

Figure 3. *MINE_CG_CONS*, the algorithm to generate non-repeatedly all frequent closed itemsets and their generators with constraint *C*.

Example 5. The process of mining frequent closed itemsets and generators on $\mathcal{C}$=*abde* from $\mathcal{LG}_A$={<L, s(L), $\mathcal{G}$(L)>} is shown in Table II (C'=L∩$\mathcal{C}$).

TABLE II. AN EXAMPLE OF MINING FREQUENT CLOSED ITEMSETS AND THEIR GENERATORS WITH CONSTRAINT

L	$L_i \in \mathcal{G}(L)$	s(L)	$\mathcal{C} \supseteq L_i$	C'	$\mathcal{G}_C$(C')	s(C')
acfh	cf, ch	¼				
aceg	ae, *ag*	¼	ae	ae	ae	¼
adfh	d	¼	d	ad	d	¼
bceg	b	¼	b	be	b	¼
afh	f, h	½				
ceg	e, *g*	½	e	e	e	½
ac	ac	½				
c	c	¾				
a	a	¾	a	a	a	¾

B. *Mining all frequent itemsets with constraint*

Here, we will partition the class $\mathcal{FS}_C$ of all frequent itemsets restricted on $\mathcal{C}$ into the disjoint equivalence classes. Each class contains the itemsets having the same closure with the frequent closed itemset represented to that class. Thus, it is correct to use the efficient algorithm *GEN_ITEMSETS* for mining quickly $\mathcal{FS}_C$ from $\mathcal{FLG}_C$.

Definition 4 (Equivalence relation over $2^{\mathcal{C}}$): $\forall$ A, B $\in 2^{\mathcal{C}}$:

$$A \sim_C B \Leftrightarrow h_C(A) = h_C(B).$$

Theorem 4 (Partition and representation of $\mathcal{FS}_C$): The equivalence relation $\sim_C$ partitions $\mathcal{FS}_C$ into the disjoint equivalence classes. Each class contains the frequent itemsets having the same closure:

$$\mathcal{FS} = \sum_{C' \in \mathcal{FCS}(\mathcal{C})} [C']_{\sim_C} = \sum_{C' \in \mathcal{FCS}_C} \mathcal{IS}(C').$$

Proof: This theorem is consequence of theorems 1, 3 and the proposition 2. □

The algorithm ***MINE_FS_CONS*** *mines and classifies quickly all frequent itemsets on* $\mathcal{C}$ from $\mathcal{LG}_A$.

```
FS_C   MINE_FS_CONS (LG_A, C, s0):
FS_C  = ∅;
FLG_C = MINE_CG_CONS (LG_A, C, s0);
for each (<C', s(C'), G_C(C')> ∈ FLG_C) do {
  <IS(C'), s(C')> =
          GEN_ITEMSETS (C', s(C'), G_C(C'));
  FS_C = FS_C + {<IS(C'), s(C')>};
               // classify FS_C
}
return FS_C;
```

Figure 4. *MINE_FS_CONS*, the algorithm to generate non-repeatedly all frequent itemsets on *C*.

Example 6. The processes of mining from $\mathcal{LG}_A$ = {X=<L, s(L), $\mathcal{G}$(L)>} all frequent itemsets restricted on $\mathcal{C}_1$=abde and $\mathcal{C}_2$=abceg are figured out in Tables III and IV, where $\mathcal{FLG}_C$ = {Y = <C', s(C'), $\mathcal{G}_C$(C')>}.

TABLE III. MINING ALL FREQUENT ITEMSETS ON $\mathcal{C}_1$

X ∈ $\mathcal{LG}_A$	Y ∈ $\mathcal{FLG}_C$	<$\mathcal{IS}$(C'), s(C')>
acfh, ¼, {cf, ch}		
aceg, ¼, {ae, ag}	ae, ¼, {ae}	{ae}, ¼
adfh, ¼, {d}	ad, ¼, {d}	{d, da}, ¼
bceg, ¼, {b}	be, ¼, {b}	{b, be}, ¼
afh, ½, {f, h}		
ceg, ½, {e, g}	e, ½, {e}	{e}, ½
ac, ½, {ac}		
c, ¾, {c}		
a, ¾, {a}	a, ¾, {a}	a, ¾

TABLE IV. MINING ALL FREQUENT ITEMSETS ON $\mathcal{C}_2$

X ∈ $\mathcal{LG}_A$	Y ∈ $\mathcal{FLG}_C$	<$\mathcal{IS}$(C'), s(C') >
acfh, ¼, {cf, ch}		
aceg, ¼, {ae, ag}	aceg, ¼, {ae, ag}	{ae, aec, aeg, aegc, ag, agc}, ¼
adfh, ¼, {d}		
bceg, ¼, {b}	bceg, ¼, {b}	{b, bc, be, bg, bce, bcg, beg, bceg}, ¼
afh, ½, {f, h}		
ceg, ½, {e, g}	ceg, ½, {e, g}	{e, ec, eg, egc, g, gc}, ¼
ac, ½, {ac}	ac, ½, {ac}	{ac}, ½
c, ¾, {c}	c, ¾, {c}	{c}, ¾
a, ¾, {a}	a, ¾, {a}	{a}, ¾

IV. EXPERIMENTAL RESULTS

The following experiments were performed on a 2.93 GHz Pentium(R) Dual-Core CPU E6500 with 1.94GB of

978-0-7695-4567-7/11 $26.00 © 2011 IEEE
DOI 10.1109/KSE.2011.12

IEEE Computer Society

RAM, running Linux, Cygwin. Algorithms were coded in C^{++}. The code of Zaki [22] is used to run Charm-L, MinimalGenerators and Eclat. Four databases in [21] are used during these experiments. They have been used as benchmark for testing mining algorithms. Table V shows their characteristics.

TABLE V. DATABASE CHARACTERISTICS

Database (DB)	# Records	# Items	Average size
Mushroom (M)	8124	119	23
Connect (Co)	67557	129	43
Chess (Ch)	3196	75	37
Pumsb (P)	49046	7117	50

As the discussion in the introduction, Srikant et al. incorporated $\mathcal{C}$ into the mining process by modified the apriori candidate generation procedure. In this experiment, to compare with our approach in mining frequent itemsets with constraint from $\mathcal{LG}_{\mathcal{A}'}$, we incorporate $\mathcal{C}$ into the Charm-L and Eclat (well-known algorithms for mining frequent itemsets). This is done easily by choosing only frequent items that are in $\mathcal{C}$ to work in the next steps of those algorithms. Those new versions are called $\mathcal{C}$–Charm, and $\mathcal{C}$–Eclat.

The items of the constraints are selected from the set $\mathcal{A}^F$ of all frequent (corresponding to the minimum support MS) items of $\mathcal{A}$ with the ratios of ¼, ½ and ¾. We have the constraints with the sizes of l_1=¼*$|\mathcal{A}^F|$, l_2=½*$|\mathcal{A}^F|$ and l_3=¾*$|\mathcal{A}^F|$. For each l_i, $\mathcal{C}$ is constructed from two subsets: $\mathcal{C} = \mathcal{C}_1+\mathcal{C}_2$. In the reality, the constraints that users are interested in usually contain the high-support items. Thus, we will sort all items by the order of their supports. To determine a constraint $\mathcal{C}$ with the size l_i, firstly, we construct the first subset $\mathcal{C}_1$ of $\mathcal{C}$ containing $[p*l_i]$ items randomly selected from the set of high-support items in $\mathcal{A}^F$, where $p\in[0; 1]$; the remained part $\mathcal{C}_2$ of $\mathcal{C}$ contains $[(1-p)*l_i]$ randomly selected items from $\mathcal{A}^F\backslash\mathcal{C}_1$. For experiments at here, we set p = 0.5 and consider two constraints for each l_i.

We will do two comparisons. Firstly, in Table VI, we compare the average time for mining frequent closed itemsets and their generators with constraint $\mathcal{C}$ by our algorithm *MINE_CG_CONS* (shown in column T_C^O) to the one by *$\mathcal{C}$–CharmGen* (column T_C^C) upon P, M, Co, and Ch. *$\mathcal{C}$–CharmGen* is the combination of $\mathcal{C}$–Charm (for mining frequent closed itemsets with constraint) and MinimalGenerators [19] (for determining their generators). The reduction in the mining time is shown in column RT_C^O ($RT_C^O = T_C^C/T_C^O$). With the different minimum support thresholds, we see that it is drastic, ranging from a factor of *60* to ***316*** times! That reduction plays an important role in mining quickly association rule with constraints because, as the discussion on [15, 6], all association rules can be mined quickly from frequent closed itemsets and their generators.

TABLE VI. MINING FREQUENT CLOSED ITEMSETS AND GENERATORS WITH CONSTRAINT: *MINE_CG_CONS* VS *C-CHARMGEN*.

DB	MS (%)	T_C^C	T_C^O	RT_C^O
Co	75	1.07	0.005	*213*
	65	1.29	0.013	*99*
	55	3.49	0.026	*134*
Ch	65	1.21	0.011	*115*
	55	6.08	0.055	*111*
	50	10.93	0.111	*98*
P	80	1.50	0.010	*145*
	70	13.74	0.115	*120*
	65	39.22	0.243	*161*
M	15	0.16	0.003	*60*
	5	0.84	0.003	***316***
	1	4.80	0.018	*264*

In the second, the time for mining all frequent closed itemsets restricted on $\mathcal{C}$ by our algorithm *MINE_FS_CONS* is compared to the one by *$\mathcal{C}$–Eclat*. We did experiments on on three databases (that have many items) Co, P, M. The reduction in the mining time by our approach is drastic. It ranges from a factor of 14 to *1063* times and is shown in Fig. 5.

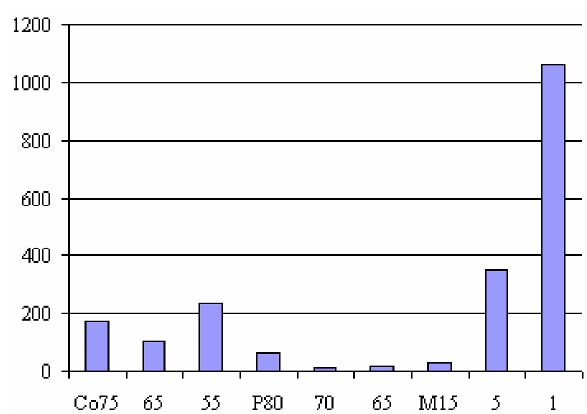

Figure 5. The reduction in mining time all frequent itemsets with constraint: *MINE_FS_CONS* vs *$\mathcal{C}$–Eclat*.

For database P, the reductions are small because the average size of transactions is small compared with the number of all items. Then, practically (users are usually interested in high-support items), we should consider the constraints containing many high-support items. Indeed, the bigger of the number of high-support items (corresponding with the big values of p) are, the bigger reductions become. Table VII showed that. In addition, the output of our algorithm *MINE_FS_CONS* is classified into disjoint classes. All frequent itemsets with constraint in a class have

978-0-7695-4567-7/11 $26.00 © 2011 IEEE
DOI 10.1109/KSE.2011.12

IEEE Computer Society

the same closure and support. Thus, when it is necessary, we only access without the need to compute them.

TABLE VII. THE REDUCTIONS IN MINING TIME FREQUENT ITEMSETS WITH CONSTRAINT: *DATABASE P*

p \ MS	80	70	65
½	61	13	15
¾	*51*	*26*	*36*
4/5	*70*	*37*	*65*

V. CONCLUSIONS

This paper proposed an approach to mine and classify efficiently frequent itemsets with constraint C on a given database, especially, when C is often changed. The correctness and efficiency of the approach were ensured by the theoretical results. The corresponding algorithms were obtained and were tested on benchmark databases. In future, based on this approach, we will research on the problem of mining association rules with constraint.

REFERENCES

[1] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules", Proceeding of the 20th International Conference on Very Large Data Bases, 1994, pp. 478-499.

[2] R.J. Bayardo, "Efficiently Mining Long Patterns from Databases," Proceedings of the SIGMOD Conference, 1998, pp. 85–93.

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

[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, 2002, pp. 107-114.

[6] B. Ganter, and R. Wille, Formal Concept Analysis: Mathematical Foundations, Springer-Verlag, 1999.

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

[8] 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, 2004, pp. 53-87.

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

[10] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient mining of association rules using closed item set lattices," Information systems, vol. 24, no. 1, 1999, pp. 25-46.

[11] 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, 2005, pp. 29-60.

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

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

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

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

[16] T.C. Tin, T.N. Anh, and T. Thong, "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.

[17] R. Wille, "Concept lattices and conceptual knowledge systems," Computers and Math. with App., no. 23, 1992, pp. 493-515.

[18] 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, 2005, pp. 462-478.

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

[20] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97), pp. 283–296.

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

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

978-0-7695-4567-7/11 $26.00 © 2011 IEEE
DOI 10.1109/KSE.2011.12

IEEE Computer Society