An Efficient Algorithm for Mining Frequent Closed Itemsets

Gang Fang [1, 2], Yue Wu [1], Ming Li [1] and Jia Chen [1]
[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P. R. China
[2]School of Computer Science and Engineering, Chongqing Three Gorges University, Wanzhou, Chongqing, 404000, P. R. China
E-mail: gangfang07@sohu.com, ywu@uestc.edu.cn, ming.m.li@alcatel-lucent.com, jchen@uestc.edu.cn

Keywords: frequent closed itemsets, Galois connection, granular computing, association rules, data mining

Received: February 6, 2014

To avoid generating an undesirably large set of frequent itemsets for discovering all high confidence association rules, the problem of finding frequent closed itemsets in a formal mining context is proposed. In this paper, aiming to these shortcomings of typical algorithms for mining frequent closed itemsets, such as the algorithm A-close and CLOSET, we propose an efficient algorithm for mining frequent closed itemsets, which is based on Galois connection and granular computing. Firstly, we present the smallest frequent closed itemsets and its characters, contain some properties and theorems, then propose a novel notion, called the smallest frequent closed granule, which can help the algorithm save reading the database to reduce the costed I/O for discovering frequent closed itemsets. And then we propose a novel model for mining frequent closed itemsets based on the smallest frequent closed granules, and a connection function for generating the smallest frequent closed itemsets. The generator function create the power set of the smallest frequent closed itemsets in the enlarged frequent 1-item manner, which can efficiently avoid generating an undesirably large set of candidate smallest frequent closed itemsets to reduce the costed CPU and the occupied main memory for generating the smallest frequent closed granules. Finally, we describe the algorithm for the proposed model. On these different datasets, we report the performances of the algorithm and its trend of the performances to discover frequent closed itemsets, and further discuss how to solve the bottleneck of the algorithm. For mining frequent closed itemsets, all these experimental results indicate that the performances of the algorithm are better than the traditional and typical algorithms, and it also has a good scalability. It is suitable for mining dynamic transactions datasets.

Povzetek: Opisan je nov algoritem asociativnega učenja za pogoste entitete.

1 Introduction

Association rules mining is introduced in [1], Agrawal et al. firstly propose a classic algorithm for discovering association rules in [2], namely, the Apriori algorithm. However, it is also well known that mining frequent patterns often generates a very large number of frequent itemsets and association rules, which reduces not only efficiency but also effectiveness of mining since users have to sift through a large number of mined rules to discover useful ones. In order to avoid the shortcoming, Pasquier et al. introduce the problems of mining frequent closed itemsets in [3], and propose an efficient Apriori-based mining algorithm, called A-close. Subsequent, Zaki and Hsiao propose another mining algorithm in [4], called CHARM, which improves mining efficiency by exploring an item-based data structure. However, we find A-close and CHARM are still costly when mining long patterns or low minimum support thresholds in large database, especially, CHARM depends on the given data structure and need the overlarge memory. As a continued study on frequent patterns mining without candidate generation in [5], J. Pei et al. propose an efficient method for mining frequent closed itemsets without candidate generation in [6], called CLOSET. There are more study works for mining frequent closed itemsets in [7-13]. The familiar algorithms include MAFIA in [7], CLOSE+ in [8] and DCI-CLOSED in [9].

At present, for mining frequent closed itemsets, there are two types of main current methods as follows:

The first is the method of mining frequent closed itemsets with candidate based on the Apriori algorithm in [3 and 14]. The A-close algorithm in [3] is a well-known typical algorithm for the first method, which adopts the bottom-up search strategy as the Apriori-like in [2], and constructs the set of generators in a level-wise manner: $(i+1)-generators$ are created by joining $i-generators$. For the first method, the advantages are the less usage of memory, simple data structure, and easy implementing it and maintaining; its disadvantages are the more occupied CPU for matching candidate patterns, and the overlarge costed I/O for the repeatedly scanning the database to compute the support.

The second is the method of mining frequent closed itemsets without candidate based on the FP-tree structure in [6, 15 and 16]. The CLOSET algorithm in [6] is an extended study of the FP-Growth for mining frequent patterns in [5]. For the second method, the advantages are reducing the overlarge computing corresponding to the joined potential generators in the A-close algorithm, and saving the costed I/O of reading the database. But it has these disadvantages, such as complex data structure costs more memory, creating recursion FP-tree occupies more CPU, and implementing it is troublesome.

Rough set theory in [17] and formal concept analysis in [18 and 19] are two efficient methods for the representation and discovery of knowledge in [20 and 21]. Rough set theory and formal concept analysis are actually related and often complementary approaches to data analysis, but rough set models enable us to precisely define and analyse many notions of granular computing in [22 and 23].

Reference [22] develops a general framework for the study of granular computing and knowledge reduction in formal concept analysis. In formal concept analysis, granulation of the universe of discourse, description of granules, relationship between granules, and computing with granules are issues that need further scrutiny. Since the basic structure of a concept lattice induced from a formal context is the set of object concepts and every formal concept in the concept lattice can be represented as a join of some object concepts, each object concept can be viewed as an information granule in the concept lattice.

An important notion in formal concept analysis is thus a formal concept, which is a pair consisting of a set of objects (the extension) and a set of attributes (the intension) such that the intension consists of exactly those attributes that the objects in the extension have in common, and the extension contains exactly those objects that share all attributes in the intension in [22]. For the study of granular computing, the formal concept is defined as a granule, such as an information granule.

Based on the notions of granularity in [24] and abstraction in [25], the ideas of granular computing have been widely investigated in artificial intelligence in [26], such as, granular computing has been applied to association rules mining in [27 and 28], where a partition model of granular computing is applied to constructing information granule in [26], which depends on rough set theory in [29] and quotient space theory in [30].

In this paper, we propose a novel model based on granular computing, namely, an efficient algorithm for mining frequent closed itemsets, which constructs the set of generators in the enlarged frequent $1-item$ manner to reduce the costed CPU, and adopts granular computing to reduce the costed I/O.

The rest of the paper is organized as follows:

In Section 2, we present the related concepts with closed itemset and granular computing; In Section 3, we propose a novel model for mining frequent closed itemsets based on granular computing; In Section 4, we describe the efficient mining algorithm; Section 5 reports the performance comparison of our with A-close and CLOSET. In Section 6, we summarize study work and discuss some future research directions.

2 Related concepts

In this section, referring to the definitions and theorems in [3, 4, 6, and 22], we present the following *definitions*, *properties*, *theorems*, and *propositions* with closed itemsets and granular computing.

Definition 2.1 A formal context is a triplet $D=(U, A, R)$, where

$U=\{u_1,u_2,...,u_n\}$ $(n=\|U\|)$, called the universe of discourse, is a finite nonempty set of objects;

$A=\{a_1,a_2,...,a_m\}$ $(m=\|A\|)$, called the attributes set, is also a finite nonempty set of attributes;

$R\subseteq U\times A$, called the relations, is a binary relation between objects U and attributes A, where each couple $(u,a)\in R$ denotes the fact that the object u $(u\in U)$ is related to the attribute a $(a\in A)$.

Here, we make the following ratiocinations become concise, and then let the attribute $a(a\in A)$ be Boolean, where each attribute is regarded as an item, i.e. the attributes set A is a general itemset. In fact, these ratiocinations are also suitable for the quantitative attributes.

Definition 2.2 Galois connection, let $D=(U,A,R)$ be a formal context, for $O\subseteq U$ and $I\subseteq A$, we define:

$\omega(O):P(U)\to P(A)$, namely

$\omega(O)=\{i\in A\mid \forall o\in O,(o,i)\in R\}$, which denotes the maximal set of items shared by all objects o $(o\in O)$;

$\varphi(I):P(A)\to P(U)$, namely

$\varphi(I)=\{o\in U\mid \forall i\in I,(o,i)\in R\}$, which denotes the maximal set of objects that have all items i $(i\in I)$;

And the couple of applications (ω,φ) is defined as a Galois connection between the power set of U (i.e. $P(U)$) and the power set of A (i.e. $P(A)$).

Property 2.1 For a formal context $D=(U,A,R)$, if $O,O_1,O_2\subseteq U$ and $I,I_1,I_2\subseteq A$, then we have:

(1) $I_1\subseteq I_2\Rightarrow \varphi(I_1)\supseteq\varphi(I_2)$;

(1*) $O_1\subseteq O_2\Rightarrow \omega(O_1)\supseteq\omega(O_2)$;

(2) $I\subseteq\omega(O)\Leftrightarrow O\subseteq\varphi(I)$.

Definition 2.3 Galois closure operators are defined as the operators $h=\omega\circ\varphi$ in $P(A)$ and $\hbar=\varphi\circ\omega$ in $P(U)$, where they are also expressed as the following notation: $h(I)=\omega\circ\varphi(I)=\omega(\varphi(I)), \hbar(O)=\varphi\circ\omega(O)=\varphi(\omega(O))$.

Property 2.2 For a formal context $D=(U,A,R)$, let (ω,φ) be the Galois connection. If $O,O_1,O_2\subseteq U$ and I, $I_1,I_2\subseteq A$, then we have:

Extension: (3) $I\subseteq h(I)$; (3*) $O\subseteq\hbar(O)$;

Idempotency: (4) $h(h(I))=h(I)$;

(4*) $\hbar(\hbar(O))=\hbar(O)$;

Monotonicity: (5) $I_1\subseteq I_2\Rightarrow h(I_1)\subseteq h(I_2)$;

(5*) $O_1\subseteq O_2\Rightarrow \hbar(O_1)\subseteq\hbar(O_2)$;

Definition 2.4 Closed itemsets, an itemsets $C \subseteq A$ from D is a closed itemset if and only if $h(C) = C$. The smallest (minimal) closed itemset containing an itemset I is obtained by applying h to I.

Here, we call $h(I)$ the closure of I.

Theorem 2.1 For a formal context $D = (U, A, R)$, let $I_1, I_2 \subseteq A$ be two itemsets. We have:

$h(I_1 \cup I_2) = h(h(I_1) \cup h(I_2))$.

Proof. Let $I_1, I_2 \subseteq A$ be two itemsets.

$\because I_1 \subseteq h(I_1), I_2 \subseteq h(I_2)$ (Extension)

$\therefore I_1 \cup I_2 \subseteq h(I_1) \cup h(I_2)$

$\therefore h(I_1 \cup I_2) \subseteq h(h(I_1) \cup h(I_2))$ (Monotonicity)

And $\because I_1 \subseteq I_1 \cup I_2, I_2 \subseteq I_1 \cup I_2$

$\therefore h(I_1) \subseteq h(I_1 \cup I_2), h(I_2) \subseteq h(I_1 \cup I_2)$

$\therefore h(h(I_1) \cup h(I_2)) \subseteq h(h(I_1 \cup I_2))$ (Monotonicity)

$\therefore h(h(I_1) \cup h(I_2)) \subseteq h(I_1 \cup I_2)$ (Idempotency)

$\therefore h(I_1 \cup I_2) = h(h(I_1) \cup h(I_2))$.

Proposition 2.1 For a formal context $D = (U, A, R)$, the closed itemset $h(I)$ corresponding to the closure by h of the itemset $I (I \subseteq A)$ is the intersection of all objects in U that contain I:

$h(I) = \bigcap_{o \in U} \{\omega(\{o\}) \mid I \subseteq \omega(\{o\})\}$.

Proof. Let $H = \bigcap_{o \in S} \omega(\{o\})$, where

$S = \{o \in U \mid I \subseteq \omega(\{o\})\}$. And we have

$h(I) = \omega(\varphi(I)) = \bigcap_{o \in \varphi(I)} \omega(\{o\}) = \bigcap_{o \in S^\circ} \omega(\{o\})$, where

$S^\circ = \{o \in U \mid o \in \varphi(I)\}$.

Let's show that $S^\circ = S$, i.e. $I \subseteq \omega(\{o\}) \Leftrightarrow o \in \varphi(I)$.

$\because \varphi(I) \supseteq \{o\}$; $\therefore \omega(\varphi(I)) \subseteq \omega(\{o\})$ (Property 2.1)

$\because I \subseteq \omega(\varphi(I))$ (Extension)

$\therefore o \in \varphi(I) \Leftrightarrow I \subseteq \omega(\varphi(I)) \subseteq \omega(\{o\})$

We have $S = S^\circ$, and also have $h(I) = H$.

Definition 2.5 Formal granule, for a formal context $D = (U, A, R)$, a two-tuple $G = < I, \varphi(I) >$ is defined as a formal granule of the context $D = (U, A, R)$, where

I, called the intension of formal granule, is an abstract description of common features or properties shared by objects in the extension, which is expressed as $I = \{i_1, i_2, ..., i_k\} (I \subseteq A, k = \| I \|)$.

$\varphi(I)$, called the extension of formal granule, is the maximal set of objects that have all items i $(i \in I)$, which is expressed as $\varphi(I) = \{o \in U \mid \forall i \in I, (o, i) \in R\}$.

Definition 2.6 Intersection operation of two formal granules is denoted by $\otimes$, which is described as follows:

There are two formal granules $G_\alpha = < I_\alpha, \varphi(I_\alpha) >$ and $G_\beta = < I_\beta, \varphi(I_\beta) >$, respectively; then we have:

$G = < I, \varphi(I) > = G_\alpha \otimes G_\beta = < I_\alpha \cup I_\beta, \varphi(I_\alpha) \cap \varphi(I_\beta) >$.

3 A novel mining model

Firstly, we present some *definitions*, *properties*, *theorems, and corollaries* from the Galois connection and granular computing. And propose a novel model for mining frequent closed itemsets based on granule computing.

3.1 Basic concepts

Definition 3.1 Itemset support, for a formal context $D = (U, A, R)$, the support of the itemset I is expressed as $support(I) = \| \varphi(I) \| / \| U \|$.

Definition 3.2 Frequent itemsets, the itemset I is said to be frequent if the support of I in D is at least the given $minsupport$. The set FI of frequent itemsets in D is defined as $FI = \{I \subseteq A \mid support(I) \geq minsupport\}$.

Property 3.1 All subsets of a frequent itemset are frequent; all supersets of an infrequent itemset are infrequent. (Intuitive in [2])

Definition 3.3 Frequent closed itemsets, the closed itemset C is said to be frequent if the support of C in D is at least the given $minsupport$. The set FCI of frequent closed itemsets in D is defined as follows:

$FCI = \{C \subseteq A \mid C = h(C) \wedge support(C) \geq minsupport\}$.

Property 3.2 Frequent closed itemsets FCI is the subset of frequent itemset FI, namely $FCI \subseteq FI$.

Definition 3.4 The smallest frequent closed itemsets, the frequent itemset I is said to be the smallest frequent closed itemset if $\forall I^\circ \subset I, support(I) < support(I^\circ)$. The set FC_{min} of the smallest frequent closed itemsets in D is $FC_{min} = \{I \in FI \mid \forall I^\circ \subset I \wedge support(I) < support(I^\circ)\}$.

Theorem 3.1 For a formal context $D = (U, A, R)$, if I be a frequent closed itemset, and there is the smallest frequent closed itemset $I'(\varphi(I) = \varphi(I'))$, i.e.

$\forall\ I \in FCI \Rightarrow \exists I' \in FC_{min} \wedge \varphi(I) = \varphi(I')$.

Proof. Let $\| I \| = k$, there are two cases as follows:

(1) If $\forall\ I_1 \subset I$ $(\| I_1 \| = k-1)$, and have $support(I) < support(I_1) \Rightarrow \forall\ I^\circ \subset I_1 \subset I \wedge support(I) < support(I^\circ)$. Since $I \in FCI \subseteq FI$, we have $I \in FC_{min}$. Let $I' = I$, and we have $I' \in FC_{min} \wedge \varphi(I) = \varphi(I')$.

(2) If $\exists\ I_1 \subset I$ $(\| I_1 \| = k-1)$, and have $support(I) = support(I_1) \Rightarrow \varphi(I) = \varphi(I_1)$.

(i) If $\forall\ I_2 \subset I_1 \subset I$ $(\| I_2 \| = k-2)$, and $support(I_1) < support(I_2) \Rightarrow \forall I^\circ \subset I_2 \subset I_1 \wedge support(I_1) < support(I^\circ)$. Since $I_1 \subset I \in FCI \subseteq FI$, we have $I_1 \in FC_{min}$. Let $I' = I_1$, and we have $I' \in FC_{min} \wedge \varphi(I) = \varphi(I_1) = \varphi(I')$.

(ii) Otherwise $\exists\ I_2 \subset I_1 \subset I$ $(\| I_2 \| = k-2)$, and have $support(I_1) = support(I_2) \Rightarrow \varphi(I_1) = \varphi(I_2)$ ...

Go on doing until the k^{th} step, and $\exists\ support(I) = support(I_k) \wedge I_k \in FC_{min}$. Let $I' = I_k$, and we have $I' \in FC_{min} \wedge \varphi(I) = \varphi(I_1) = ... = \varphi(I_k) = \varphi(I')$.

Based on definition 2.4 and theorem 3.1, we have:

Corollary 3.1 Let I be the smallest frequent closed itemset, i.e. $I \in FC_{min}$. And the frequent closed itemset corresponding to I is $h(I) = \omega(\varphi(I))$.

Corollary 3.2 For a formal context $D = (U, A, R)$, the set FCI of frequent closed itemsets in D is expressed as $FCI = \{h(I) \mid I \in FC_{min}\}$.

Theorem 3.2 Let $I_\alpha \subseteq I_\beta \subseteq A$, where $support(I_\alpha) = support(I_\beta)$. Then we have $h(I_\alpha) = h(I_\beta)$ and $\forall I \subseteq A$, $h(I_\alpha \cup I) = h(I_\beta \cup I)$.

Proof. $\because I_\alpha \subset I_\beta \subseteq A \wedge support(I_\alpha) = support(I_\beta)$

$\therefore \ \| \varphi(I_\alpha) \| = \| \varphi(I_\beta) \|$

$\therefore \ \varphi(I_\alpha) = \varphi(I_\beta)$

$\therefore \omega(\varphi(I_\alpha)) = \omega(\varphi(I_\beta))$, i.e. $h(I_\alpha) = h(I_\beta)$

$\because I \subseteq A$

$\therefore h(I_\alpha \cup I) = h(h(I_\alpha) \cup h(I))$ (Theorem 2.1)

$\because h(I_\alpha) = h(I_\beta)$

$\therefore h(I_\alpha \cup I) = h(h(I_\beta) \cup h(I)) = h(I_\beta \cup I)$.

Theorem 3.3 $I \in FC_{min} \Rightarrow \forall\, I^\circ \subset I \wedge I^\circ \in FC_{min}$.

Proof. Suppose $I \in FC_{min} \Rightarrow \exists\, I_1 \subset I \wedge I_1 \notin FC_{min}$.

$\because \ I_1 \subset I \wedge I_1 \notin FC_{min}$

$\therefore \exists\, I_2 \subset I_1 \wedge support(I_1) = support(I_2)$

$\therefore \ \varphi(I_2) = \varphi(I_1)$

$\exists\, (I' = I - I_1) \wedge (I' \cup I_2 \subset I) \ \ (I_3 = I' \cup I_2)$

$\therefore \ \varphi(I') \supseteq \varphi(I_3) \supseteq \varphi(I)$

$\because I_3 \supset I_2$, i.e. $\varphi(I_3) \subseteq \varphi(I_2)$

$\therefore \ \varphi(I_3) \subseteq \varphi(I_1)$

$\therefore \varphi(I_1) \cap \varphi(I') \supseteq \varphi(I_3)$

$\because \varphi(I) = \varphi(I_1 \cup I') = \varphi(I_1) \cap \varphi(I')$ (Definition 2.6)

$\therefore \varphi(I_3) \subseteq \varphi(I)$

$\therefore \varphi(I_3) = \varphi(I)$, i.e. $support(I_3) = support(I)$

$\therefore \exists\, I_3 \subset I \wedge support(I_3) = support(I)$

$\therefore I \notin FC_{min}$. However, the itemset I is the smallest frequent closed itemset, namely $I \in FC_{min}$.

$\therefore I \in FC_{min} \not\Rightarrow \exists\, I_1 \subset I \wedge I_1 \notin FC_{min}$

$\therefore I \in FC_{min} \Rightarrow \forall\, I^\circ \subset I \wedge I^\circ \in FC_{min}$.

Corollary 3.3 $I \in FC_{min} \Rightarrow \forall\, I^\circ \subset I \wedge I^\circ \in FC_{min}$, $(\| I^\circ \| = \| I \| - 1)$

Definition 3.5 The smallest frequent closed granules set, the formal granule $G = \langle I, \varphi(I) \rangle$ is said to be the smallest frequent closed granule G_{min} if the intension I of G is the smallest frequent closed itemset. The set FG_{min} of the smallest frequent closed granules is defined as:

$$FG_{min} = \{G = \langle I, \varphi(I) \rangle \mid I \in FC_{min}\}$$

3.2 Frequent closed itemsets mining

In this section, we propose a novel model for mining frequent closed itemsets based on granule computing, a formal statement of which is described as follows.

For a $D = (U, A, R)$, discovering all frequent closed itemsets in D can be divided into two steps as follows:

(1)According to the minimal support given by user, mining the smallest frequent closed granules set in D. (Details in the steps from (1) to (18) from Section 4.2)

(2)Based on the smallest frequent closed granules set, discovering all frequent closed itemsets in D. (Details in the steps from (19) to (21) from Section 4.2)

Here the first step is based on definition 3.5, theorem 2.1, and theorem 3.2; the second step refers to Definition 2.4, Proposition 2.1, and Theorem 3.1(Corollary 3.1).

4 The efficient mining algorithm

In this section, we use an efficient mining algorithm to describe the novel model, which is denoted by EMFCI.

4.1 Generator function

Here, we propose a function for generating the intension of the smallest frequent closed granules.

Definition 4.1 Set vector operation $\odot$ for two sets is defined as follows:

Let $P = \{p_1, p_2, ..., p_m\}, Q = \{q_1, q_2, ..., q_n\}$ be two sets, and then the set vector operation is expressed as $P^T \odot Q$

$$= \begin{pmatrix} \{p_1\} \\ \{p_2\} \\ \cdots \\ \{p_m\} \end{pmatrix} \odot \begin{pmatrix} \emptyset & \{q_1\} & \{q_2\} & \cdots & \{q_n\} \end{pmatrix}$$

$$= \begin{pmatrix} \{p_1\} & \{p_1, q_1\} & \{p_1, q_2\} & \cdots & \{p_1, q_n\} \\ \{p_2\} & \{p_2, q_1\} & \{p_2, q_2\} & \cdots & \{p_2, q_n\} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \{p_m\} & \{p_m, q_1\} & \{p_m, q_2\} & \cdots & \{p_m, q_n\} \end{pmatrix}$$

$= \{\{p_1\}, \{p_1, q_1\}, \{p_1, q_2\}, ..., \{p_1, q_n\}, \{p_2\}, \{p_2, q_1\}, \{p_2, q_2\}, ..., \{p_2, q_n\}, ..., \{p_m\}, \{p_m, q_1\}, \{p_m, q_2\}, ..., \{p_m, q_n\}$ (Formal notation)

$= \{\{p_1\}, \{p_1 q_1\}, \{p_1 q_2\}, ..., \{p_1 q_n\}, \{p_2\}, \{p_2 q_1\}, \{p_2 q_2\}, ..., \{p_2 q_n\}, ..., \{p_m\}, \{p_m q_1\}, \{p_m q_2\}, ..., \{p_m q_n\}$.

(Simple notation)

The operation is the main idea of generator function, let P, Q be two sets, it is expressed as $f(P, Q) = P^T \odot Q$. The application of $f(P, Q)$ refers to Section 4.2.

For example, for a $D = (U, A, R)$, let A be a general itemset $\{a, b, c\}$, and then we use the set vector operation to generate $P(A)$ $(\forall p \in P(A) \wedge p \neq \emptyset)$ as follows:

(1) $P(A) = \emptyset$;

(2) $I_x = \{a\} \Rightarrow P(A) = P(A) \cup (I_x^T \odot P(A))$
$= (\{a\}) \odot (\emptyset) = \{\{a\}\}$;

(3) $I_x = \{b\} \Rightarrow P(A) = P(A) \cup (I_x^T \odot P(A))$
$= \{\{a\}\} \cup ((\{b\}) \odot (\emptyset \quad \{a\}))$
$= \{\{a\}, \{b\}, \{ab\}\}$;

(4) $I_x = \{c\} \Rightarrow P(A) = P(A) \cup (I_x^T \odot P(A))$

$= \{\{a\},\{b\},\{ab\}\} \cup \left(\left(\{c\}\right) \odot \left(\emptyset \quad \{a\} \quad \{b\} \quad \{ab\}\right)\right)$

$= \{\{a\},\{b\},\{ab\},\{c\},\{ac\},\{bc\},\{abc\}\}$.

For a formal context $D=(U,A,R)$, if A is a general itemsets, namely, it is a set of Boolean attributes, $P(A)$ is general the power set where $\|P(A)\| = 2^{\|A\|}-1$. But if A is a set of quantitative attributes, where $P(A)$ is called the extended power set of A, and $\|P(A)\|$ is expressed as:

$\|P(A)\| = \prod_{a\in A}(\|V_a\|+1)-1$, here V_a is a reprocessed discrete range of attribute $a \in A$.

4.2 An algorithm for mining frequent closed itemsets

Here, we describe the efficient algorithm based on the novel model in Section 3 via the following pseudo code.

Algorithm: EMFCI

Input: a formal context $D=(U,A,R)$, the minimal support $minsupport$.

Output: frequent closed itemsets FCI.

(1)Read D;

(2)Construct $FG=\{FG_a \mid a\in A \wedge \forall\ G=<I,\varphi(I)>\in FG_a \wedge I\subset V_a \wedge \|I\|=1 \wedge \|\varphi(I)\| \geq minsupport\}$;

(3) $F=\{F_a \subset V_a \mid \forall v\in F_a \wedge G=<\{v\},\varphi(\{v\})>\in FG_a \wedge FG_a \in FG \wedge a\in A\}$; // V_a is the range of attribute $a\in A$.

(4) $FC_{min}=\emptyset$;

(5)For $(\forall \alpha \in F)$ do begin

(6) $S_c = \alpha \odot FC_{min}$; //Generate the candidate

(7) For $(\forall s\in S_c)$ do begin

(8) If $((\forall t_1 \in N_{FI} \wedge t_1 \not\subset s) \wedge (\forall t_2 \in N_{FCmin} \wedge t_2 \not\subset s))$ then

(9) Construct $G=<s,\varphi(s)>$;

(10) If $(\|\varphi(s)\| \geq minsupport)$ then

(11) If $(\forall t\subset s \wedge \|\varphi(s))\| < \|\varphi(t))\|)$ then

(12) Write $G=<s,\varphi(s)>$ to FG_{min};

(13) Write s to FC_{min};

(14) else

(15) Write s to N_{FCmin};

(16) else

(17) Write s to N_{FI};

(18)End

(19)For $(\forall G=<I,\varphi(I)>\in FG_{min})$ do begin

(20) Write $h(I)=\omega(\varphi(I))$ to FCI;

(21)End

(22)Answer FCI;

These steps from (1) to (18) in the algorithm extract the smallest frequent closed granules set. And these steps from (19) to (21) generate all frequent closed itemsets.

4.3 Example and analysis

Here, we firstly provide an example for the algorithm, and then analyse the pruning strategies in the algorithm.

For a formal context $D=(U,A,R)$, where $A=\{a,b,c,d,e\}, U=\{u_1,u_2,u_3,u_4,u_5\}, u_1=\{acd\}, u_2=\{bc\}, u_3=\{abe\}, u_4=\{be\}, u_5=\{ace\}$; and $minsupport=40\%$. The course of discovering frequent closed itemsets is described as table 1.

No.	Operation
1	$FG=\{<\{a\},\{1,3,5\}>, <\{b\},\{2,3,4\}>,$ $<\{c\},\{1,2,5\}>, <\{e\},\{3,4,5\}>\}$ (Pruning $\{d\}$ by property 3.1and definition 3.3)
2	$F=\{\{a\},\{b\},\{c\},\{e\}\}$
3	$\alpha=\{a\} \Rightarrow S_c=\{\{a\}\}$ $FG_{min}=\{<\{a\},\{1,3,5\}>\}$, $FC_{min}=\{\{a\}\}$
4	$\alpha=\{b\} \Rightarrow S_c=\{\{b\},\{ab\}\}$ $FG_{min}=\{<\{a\},\{1,3,5\}>, <\{b\},\{2,3,4\}>\}$ $FC_{min}=\{\{a\},\{b\}\}$ (Pruning $\{ab\}$ by property 3.1and definition 3.3)
5	$\alpha=\{c\} \Rightarrow S_c=\{\{c\},\{ac\},\{bc\}\}$ $FG_{min}=\{<\{a\},\{1,3,5\}>, <\{b\},\{2,3,4\}>\}$ $<\{c\},\{1,2,5\}>, <\{ac\},\{1,5\}>\}$ $FC_{min}=\{\{a\},\{b\},\{c\},\{ac\}\}$ (Pruning $\{bc\}$ by property 3.1and definition 3.3)
6	$\alpha=\{e\} \Rightarrow S_c=\{\{e\},\{ae\},\{be\},\{ce\},\{ace\}\}$ $FG_{min}=\{<\{a\},\{1,3,5\}>, <\{b\},\{2,3,4\}>\}$ $<\{c\},\{1,2,5\}>, <\{ac\},\{1,5\}>, <\{e\},\{3,4,5\}>,$ $<\{ae\},\{3,5\}>, <\{be\},\{3,4\}>\}$ $FC_{min}=\{\{a\},\{b\},\{c\},\{ac\},\{e\},\{ae\},\{be\}\}$ (Pruning $\{ce,ace\}$ by property 3.1and definition 3.3)` Note: the search course is ended, discovering all the smallest frequent closed granules FC_{min}
7	$h(\{a\})=\{u_1\cap u_3\cap u_5\}=\{a\}$ $h(\{b\})=\{u_2\cap u_3\cap u_4\}=\{b\}$ $h(\{c\})=\{u_1\cap u_2\cap u_5\}=\{c\}$ $h(\{ac\})=\{u_1\cap u_5\}=\{ac\}$ $h(\{e\})=\{u_3\cap u_4\cap u_5\}=\{e\}$ $h(\{ae\})=\{u_3\cap u_5\}=\{ae\}$ $h(\{be\})=\{u_3\cap u_4\}=\{be\}$ Note: based on the smallest frequent closed granules set FC_{min}, getting all frequent closed itemsets
8	Answer $FCI=\{\{a\},\{b\},\{c\},\{ac\},\{e\},\{ae\},\{be\}\}$

Table 1: Frequent closed itemsets mining for $minsupport=40\%$.

For mining frequent closed itemsets, the algorithm adopts some pruning strategies as follows, property 3.1, definition 3.3 and 3.4, and theorem 3.3. They can help the algorithm efficiently reduce the search space for mining frequent closed itemsets.

5 Performance and scalability study

In this section, we design the following experiments on these different datasets:

Firstly, we report the performances of the algorithm EMFCI with A-Close and CLOSET on the six different datasets.

Secondly, we report the relationships between some parameters of the datasets and the performances of the algorithm EMFCI for mining frequent closed itemsets.

Finally, for the bottleneck of the algorithm EMFCI, we improve it to get the algorithm IEMFCI, and report its performances on the extended high dimension dataset to show the scalability of the algorithm EMFCI.

There are two original datasets as follows:

The first is the Food Mart 2000 retail dataset, which comes from SQL Server 2000. It contains 164558 records in 1998. By the same customer at the same time as a basket, we take items purchased from these records. Because the supports of the bottom items are small, we generalize the bottom items to **the product department**. Finally, we obtain 34015 transactions with time-stamps. It is a dataset with the **Boolean** attributes.

The second is from a Web log data, which is a real data that expresses some behaviour of students browsing, where the attributes set is made of $login\ time, duration, network\ flow, IDtype, and\ sex$. The dataset with the **discrete quantitative** attributes has 296031 transactions.

Now, we generalize attributes, and replicate some attributes or transactions to create the following extended datasets described as table 2, where each dataset can be defined as a formal mining context $D = (U, A, R)$.

Name	Descriptions	$\|\|P(A)\|\|;\|\|U\|\|$
Dataset 1	The first original dataset	$2^{22}-1$; 34015
Dataset 2	Replicating dataset 1 three attributes	$2^{25}-1$; 34015
Dataset 3	Replicating dataset 1 four times	$2^{22}-1$; 5*34015
Dataset 4	The second original dataset	5*4*4*14*3-1; 296031
Dataset 5	Replicating dataset 1 one attribute	5*4*4*14*3*5-1; 296031
Dataset 6	Replicating dataset 4 one time	5*4*4*14*3-1; 2*296031
Dataset 7	For the Food Mart 2000, we regard the same customer at the same time as a basket and generalize the bottom items to **the product subcategory**	$2^{102}-1$; 34015

Table 2: The datasets used in the experiments.

All the experiments are performed on an Intel (R) Core (TM)2 Duo CPU (T6570 @) 2.10 GHz 1.19GHz) PC with 1.99 GB main memory, running on Microsoft Window XP Professional. All the programs are written in C# with Microsoft Visual Studio 2008. The algorithm A-close and CLOSET are implemented as described in [3] and [6].

5.1 The experiments of performance comparison

In this section, for discovering frequent closed itemsets on these different datasets, we compare the algorithm EMFCI with the algorithm A-close and CLOSET from the following two aspects, namely, one is comparing the performances among them as the minimal support is added; the other is comparing them as the number of frequent closed itemsets is added.

1. Testing on the original datasets

For the two original datasets, we firstly compare the algorithm EMFCI with the A-close and CLOSET based on the varying minimal support and the number of frequent closed itemsets. These experimental results are described as figure 1, 2, 3, and 4, respectively.

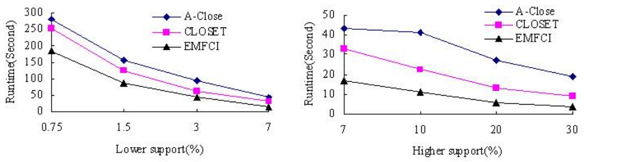

Figure 1: Performance comparison with the support on dataset 1.

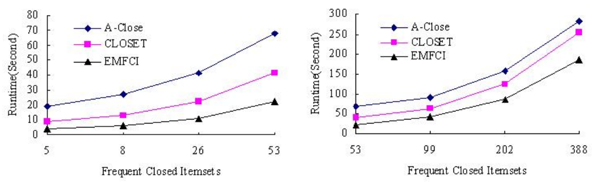

Figure 2: Performance comparison with the number of frequent closed itemsets on dataset 1.

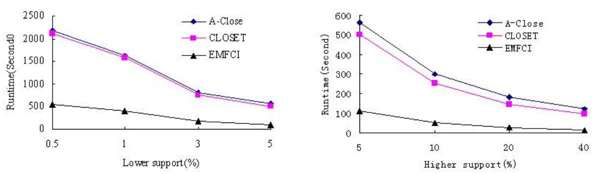

Figure 3: Performance comparison with the support on dataset 4.

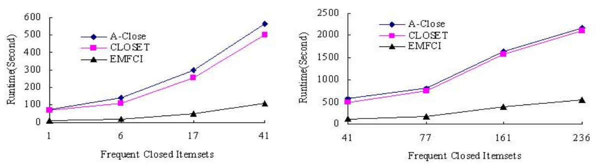

Figure 4: Performance comparison with the number of frequent closed itemsets on dataset 4.

Based on the comparison results from figure 1, 2, 3, and 4, we know that the performances of the algorithm EMFCI are better than the A-close and CLOSET.

Obviously, the algorithm CLOSET is also superior to the A-close. Hence, we don't compare the EMFCI with the A-close in the following experiments.

2. Testing on the extended datasets

We further report the performances of the algorithm EMFCI on the extended datasets. Based on the different minimal support and the number of frequent closed itemsets, we compare the EMFCI with the CLOSET, the experimental results are described as figure 5 to 12.

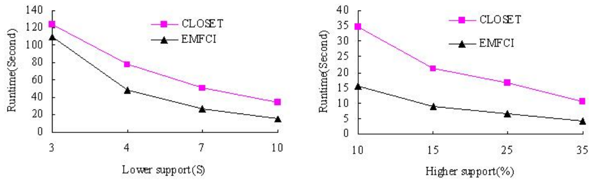

Figure 5: Performance comparison with the support on dataset 2.

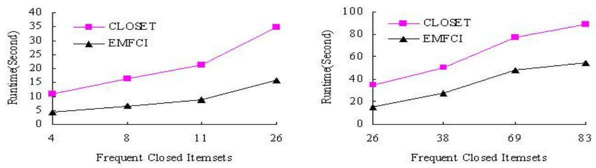

Figure 6: Performance comparison with the number of frequent closed itemsets on dataset 2.

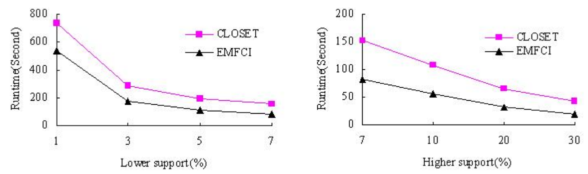

Figure 7: Performance comparison with the support on dataset 3.

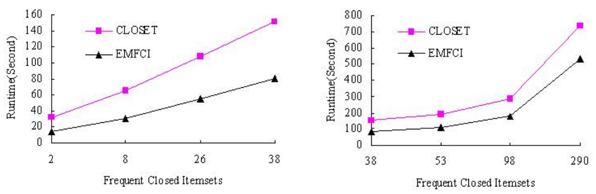

Figure 8: Performance comparison with the number of frequent closed itemsets on dataset 3.

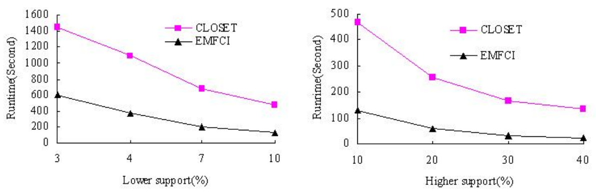

Figure 9: Performance comparison with the support on dataset 5.

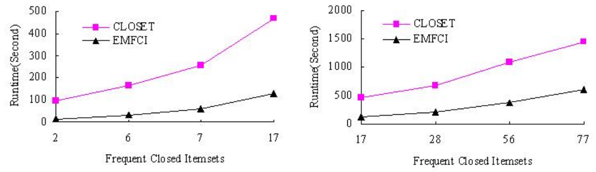

Figure 10: Performance comparison with the number of frequent closed itemsets on dataset 5.

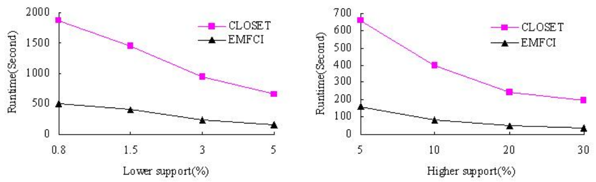

Figure 11: Performance comparison with the support on dataset 6.

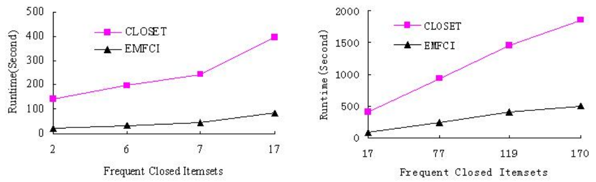

Figure 12: Performance comparison with the number of frequent closed itemsets on dataset 6.

Based on the comparison results from figure 5 to 12, we know that the performances of the algorithm EMFCI are also better than the CLOSET on the datasets with the Boolean or quantitative attributes.

5.2 The relationships between these parameters and performances

In this part, we mainly discuss the relationships between the performances and the following parameters:

$\| U \|$, is the number of objects in the formal mining context $D = (U, A, R)$, in other word, it is the number of transactions in the mining database.

$\| P(I) \|$, is the number of nonempty power sets for attribute values, called the **search space** of the algorithm, where I is the smallest frequent closed itemsets from the attribute set A, $P(I)$ is defined as the power set of I. (Refer to section 4.1)

Here, the representation of the performances has two kinds of parameters as follows:

$t(x)$: is the runtime of algorithm x, which is from input to output for mining frequent closed itemsets.

p, is defined as the improved ratio of the runtime between the algorithm EMFCI and CLOSET, which is denoted by the following equation:

$p = 1 - t(EMFCI) / t(CLOSET)$.

1. The relationships between the performances and the search space

(1)Reporting the relationships on the extended dataset of the first original dataset

For the first original dataset, namely, dataset 1, we test the trend of the performances as the search space is increasing on dataset 2, which is the extended dataset with replicating three attributes of the first dataset. As the search space is varying, the trend of the runtime for the algorithm EMFCI is expressed as figure 13, the trend of the improved ratio between the algorithm EMFCI and CLOSET is expressed as figure 14.

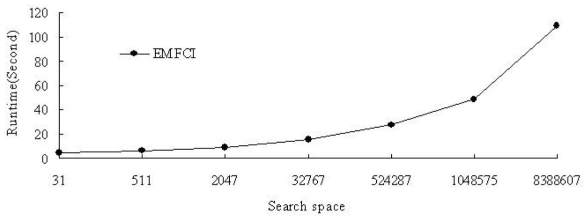

Figure 13: The trend of the runtime on dataset 2

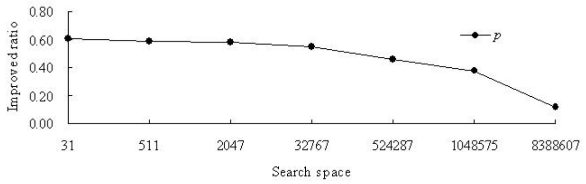

Figure 14: The trend of the improved ratio on dataset 2

Based on figure 13, we know that the runtime is added as the search space is increasing. Based on figure 14, we find that the improved ratio is reduced as the search space is increasing.

(2)Reporting the relationships on the extended dataset of the second original dataset

For the second original dataset, namely, dataset 4, we extend an attribute to get dataset 5, and test the trend of the performances on the dataset. The experimental results are expressed as figure 15 and 16, respectively.

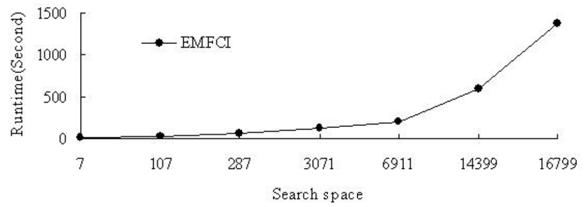

Figure 15: The trend of the runtime on dataset 5

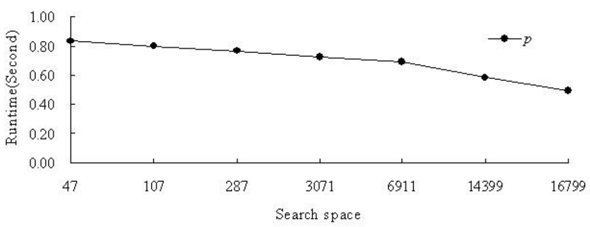

Figure 16: The trend of the improved ratio on dataset 5

According to figure 15 and 16, we get the similar comparisons results as above. Hence, we can draw the following conclusions:

The runtime of the algorithm EMFCI is added as the search space is increasing; on the contrary, the improved ratio is reduced. Namely, if the search space is increasing, the performances of the algorithm EMFCI will become worse and worse. In other word, the algorithm is not suitable for mining the dataset with too many smallest frequent closed itemsets.

2. The relationships among the performances, the search space and the number of objects

(1)Reporting the relationships on the first original dataset and its extended dataset

For the first original dataset (dataset 1), and its extended dataset, dataset 3 with replicating its objects four times, we test the trend of the performances as the search space is increasing on the two datasets. As the search space is varying, the trend of the runtime for the algorithm EMFCI is expressed as figure 17, the trend of the improved ratio between the algorithm EMFCI and CLOSET is expressed as figure 18.

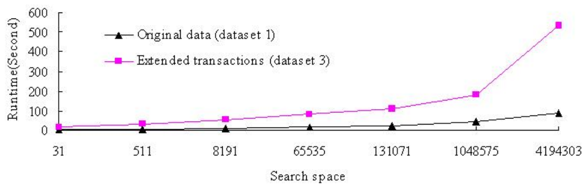

Figure 17: The trend of the runtime on dataset 1 and 3

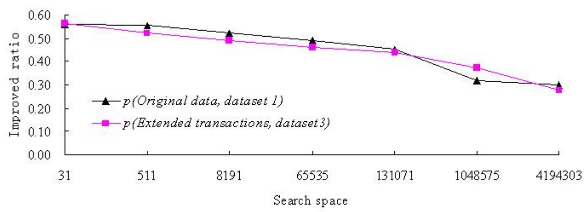

Figure 18: The trend of the improved ratio on dataset 1 and 3

Based on figure17, we know that the runtime of the algorithm is added as the search space or the number of objects is increasing.

Based on figure18, we find that the improved ratio of the algorithm is reduced as the search space is increasing, but it become relatively stable as the number of objects is increasing.

(2)Reporting the relationships on the second original dataset and its extended dataset

For the second original dataset, namely, dataset 4, we replicate its objects one time to get dataset 6, and test the trend of the performances on the dataset 4 and 6. The experimental results are expressed as figure 19 and 20, respectively.

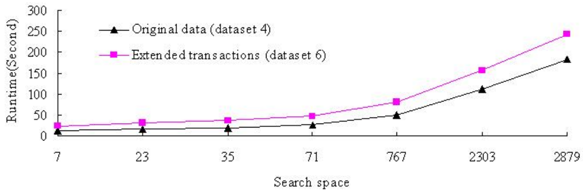

Figure 19: The trend of the runtime on dataset 4 and 6

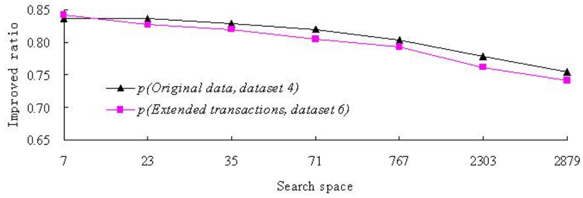

Figure 20: The trend of the improved ratio on dataset 4 and 6

According to figure 19 and 20, we draw the same conclusions as follows:

The runtime of the algorithm EMFCI is added as the search space or the number of objects is increasing, the improved ratio of the algorithm is reduced as the search space is increasing, but it become relatively stable as the number of objects is adding. Namely, the performances of the algorithm EMFCI will become relatively stable as the number of objects is increasing. Hence, it is suitable for mining dynamic transactions datasets.

According to all these experimental results, we can **draw the following conclusions**:

(1) The performances of the algorithm EMFCI are better than the traditional typical algorithms for mining frequent closed itemsets on the datasets with the Boolean attributes or the 1uantitative attributes.

(2) The runtime of the algorithm EMFCI is added as the search space. If the search space is too large, its performances will become worse and worse. This is the bottleneck of the algorithm.

(3) The runtime of the EMFCI is also added as the number of objects is increasing.

(4) For the algorithm CLOSET, the improved ratio of the algorithm is reduced as the search space is adding, but it become relatively stable as the number of objects is increasing. Namely, the performances of the EMFCI will become relatively stable as the number of objects is increasing. It is suitable for mining dynamic transactions datasets.

5.3 A further discussion for solving the bottleneck of the algorithm

Based on these conclusions in section 5.2, for the formal mining context $D=(U,A,R)$, if the search space $\|P(I)\|$ is overlarge, where $I(I\subseteq A)$ is the smallest frequent closed itemsets, $P(I)$ is defined as the power set of I, the performance of EMFCI will become worse and worse.

In this section, we adopt **a partitioning method** to avoid the bottleneck. In other word, the overlarge search space is divided into some smaller search spaces. The theoretical basis can be described as follows:

Let $I=\{a^{t_1},a^{t_2},...,a^{t_m}\}(I\subseteq A)$, and then we have the following $\|P(I)\|=\prod_{i=1}^{m}(\|V_{a^{t_i}}\|+1)-1$, namely,

$$\|P(I)\|+1=\prod_{i=1}^{m}(\|V_{a^{t_i}}\|+1)=$$

$$\underbrace{(\|V_{a^{t_1}}\|+1)\cdot(\|V_{a^{t_2}}\|+1)\cdot...\cdot(\|V_{a^{t_{m_1}}}\|+1)\cdot}_{m_1}$$

$$\underbrace{(\|V_{a^{t_{m_1+1}}}\|+1)\cdot(\|V_{a^{t_{m_1+2}}}\|+1)\cdot...\cdot(\|V_{a^{t_{m_1+m_2}}}\|+1)\cdot...\cdot}_{m_2}$$

$$\underbrace{(\|V_{a^{t_{m_1+m_2+...+m_{(k-1)}+1}}}\|+1)\cdot...\cdot(\|V_{a^{t_{m_1+m_2+...+m_{(k-1)}+m_k}}}\|+1)}_{m_k};$$

$(m_1+m_2+...+m_k=m)$.

Obviously, we also have $\|P(I)\|+1=$

$(\|P(I_{m_1})\|+1)\cdot(\|P(I_{m_2})\|+1)\cdot...\cdot(\|P(I_{m_k})\|+1)$;

Where $I_{m_1}=\{a^{t_1},a^{t_2},...,a^{t_{m_1}}\}$,

$I_{m_2}=\{a^{t_{m_1+1}},a^{t_{m_1+2}},...,a^{t_{m_1+m_2}}\},...,$

$I_{m_k}=\{a^{t_{m_1+m_2+...+m_{(k-1)}+1}},...,a^{t_{m_1+m_2+...+m_{(k-1)}+m_k}}\}$.

In this paper, we let $\|P(I_{m_i})\|<\lambda=2^{19}$. If λ is too big, the method also has the same bottleneck; if λ is too small, the cost of partitioning search space is expensive. For these two cases, their performances are expressed as figure 23.

The partitioning method is used in the algorithm EMFCI, which is called improved EMFCI, i.e. IEMFCI.

5.3.1 Example

For the example in section 4.3, we use the algorithm IEMFCI to discover frequent closed itemsets, the course of which is described as follows, where $\lambda=4$.

(**Note:** $\lambda=4$ used in the example, $\lambda=2^{19}$ used in the following experiments)

Step1. $FG=\{<\{a\},\{1,3,5\}>,<\{b\},\{2,3,4\}>,$
$<\{c\},\{1,2,5\}>,<\{e\},\{3,4,5\}>\}$.

Step2. $F=\{\{a\},\{b\},\{c\},\{e\}\},\|P(F)\|=15>\lambda=4$.

Step3. **Partitioning** the search space, get two search spaces $F_1=\{\{a\},\{b\}\},F_2=\{\{c\},\{e\}\}$, where $\|P(F_i)\|<4$.

Step4. For the first search space $F_1=\{\{a\},\{b\}\}$, have

① $\alpha=\{a\}\Rightarrow S_c=\{\{a\}\}$

$FG^1_{min}=\{<\{a\},\{1,3,5\}>\}$, $FC^1_{min}=\{\{a\}\}$;

② $\alpha=\{b\}\Rightarrow S_c=\{\{b\},\{ab\}\}$

$FG^1_{min}=\{<\{a\},\{1,3,5\}>,<\{b\},\{2,3,4\}>\}$,

$FC^1_{min}=\{\{a\},\{b\}\}$.

For the second search space $F_2=\{\{c\},\{e\}\}$, have

① $\alpha=\{c\}\Rightarrow S_c=\{\{c\}\}$

$FG^2_{min}=\{<\{c\},\{1,2,5\}>\}$, $FC^2_{min}=\{\{c\}\}$;

② $\alpha=\{e\}\Rightarrow S_c=\{\{e\},\{ce\}\}$

$FG^2_{min}=\{<\{c\},\{1,2,5\}>,<\{e\},\{3,4,5\}>\}$,

$FC_{min}^{2} = \{\{c\},\{e\}\}$.

Step5. $F = \{FC_{min}^{1}, FC_{min}^{2}\}$, repeating the step2, where $\| P(F) \| = 15 > 4$, but $\| F \| = 2$, the partitioning operation must be ended; otherwise, the algorithm need to continue to partition the search space.

$\alpha = FC_{min}^{1} \Rightarrow S_c = \{\{a\},\{b\}\}$,

$FG_{min} = \{<\{a\},\{1,3,5\}>, <\{b\},\{2,3,4\}>\}$,

$FC_{min} = \{\{a\},\{b\}\}$;

$$\alpha = FC_{min}^{2} \Rightarrow S_c = \begin{pmatrix} \{c\} \\ \{e\} \end{pmatrix} \begin{pmatrix} \emptyset & \{a\} & \{b\} \end{pmatrix} =$$

$\{\{c\},\{ac\},\{bc\},\{e\},\{ae\},\{be\}\}$;

$FG_{min} = \{<\{a\},\{1,3,5\}>, <\{b\},\{2,3,4\}>$

$<\{c\},\{1,2,5\}>, <\{ac\},\{1,5\}>, <\{e\},\{3,4,5\}>,$

$<\{ae\},\{3,5\}>, <\{be\},\{3,4\}>\}$

$FC_{min} = \{\{a\},\{b\},\{c\},\{ac\},\{e\},\{ae\},\{be\}\}$

The rest of steps are the same as the example in section 4.3. The algorithm IEMFCI reduces the checking of itemset $\{ace\}$, but adds the task of partitioning. As the number of transactions is lesser, the example does not show its advantage, please see the experiments in section 5.3.3. Here, the example only describes the execution course of IEMFCI.

5.3.2 Comparisons of the time and space complexity

For $D = (U, A, R)$, let C be a set of frequent closed itemsets, and let L be the average length of frequent closed itemsets, $k \geq 2$ is a parameter with partitioning the search space. The comparisons are expressed as table 3.

Items	Time complexity	Space complexity
A-close	$O(\|C\|^{L})$	$O(\|C\| / \|A\|)$
CLOSET	$O(\|C\|^{2})$	$O(\|C\|)$
IEMFCI	$O((L/k+1) \cdot \|C\|)$	$O(\|C\| / k \cdot \|A\|)$

Table 3: Comparisons of the time and space complexity.

5.3.3 Test on the high dimension datasets

In this section, to show the scalability of the algorithm EMFCI, firstly, we compare the improved algorithm IEMFCI with EMFCI, A-close and CLOSET on the high dimension dataset (dataset 7 as table 1), which is an extended dataset based on the first original dataset. The comparison results are expressed as figure 21 and 22, where the parameter $p(2,m) = 2^{m}$ on the abscissa shows the search space $\| P(I) \|$ of the given support.

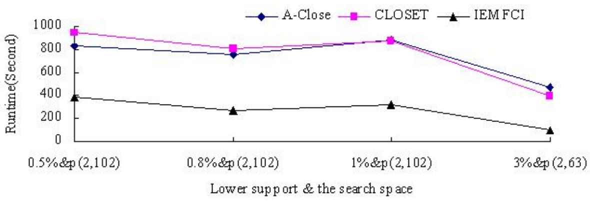

Figure 21: Performance comparison with the lower support on dataset 7

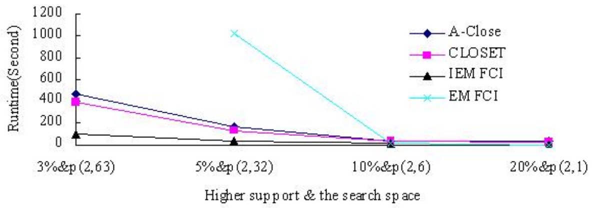

Figure 22: Performance comparison with the higher support on dataset 7

Then, for the improved algorithm IEMFCI, we adopt different parameters λ to test its trend of performance, where $\lambda = 2^{5}$, $\lambda = 2^{19}$ and $\lambda = 2^{22}$. The comparison result is expressed as figure 23, where IEMFCI ($\lambda = p(2,n)$) is the improved algorithm IEMFCI when the parameter of partitioning the search space is $\lambda = p(2,n) = 2^{n}$.

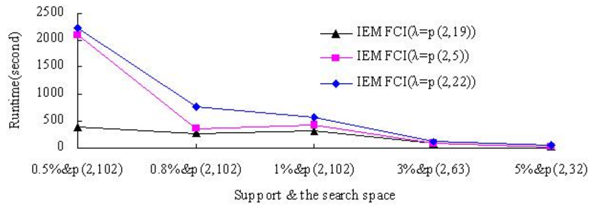

Figure 23: The trend of performance with the different parameter on dataset 7

Based on these comparisons, we draw the following conclusions:

Firstly, the improved algorithm IEMFCI is better than the algorithms EMFCI, A-close and CLOSET.

Secondly, the improved algorithm IEMFCI gets rid of the bottleneck in the algorithms EMFCI, especially, when the search space $\| P(I) \|$ is overlarge, the advantage of IEMFCI is very distinct.

Finally, for the improved algorithm IEMFCI, the parameter of partitioning the search space is not too big, but it is not too small.

6 Conclusion

In this paper, for the shortcomings of typical algorithms for mining frequent closed itemsets, we propose an efficient algorithm for mining frequent closed itemsets, which is based on Galois connection and granular computing. We present the notion of smallest frequent closed granule to reduce the costed I/O for discovering frequent closed itemsets. And we propose a connection function for generating the smallest frequent closed itemsets in the enlarged frequent 1-item manner to reduce the costed CPU and the occupied main memory. But the number of the smallest frequent closed itemsets is too many, the performances of the algorithm become worse and worse, so we further discuss how to solve the bottleneck, namely, propose its improved algorithm on high dimension dataset. The algorithm is also suitable for mining dynamic transaction datasets.

Acknowledgement

The authors would like to thank the anonymous reviewers for the constructive comment. This work was a project supported by Chongqing Cutting-edge and

Applied Foundation Research Program (Grant No. cstc2014jcyjA40035). And it was also supported by Scientific and Technological Research Program of Chongqing Three Gorges University (Grant No.13ZD20).

References

[1] R. Agrawal, T. Imielinski, and A. Swami (1993). Mining association rules between sets of items in large databases. *In Proceedings of the 1993 ACM SIGMOD Int'l Conference on Management of Data*, Washington DC, USA, pp. 207–216.

[2] R. Agrawal and R. Srikant (1994). Fast algorithms for mining association rules. *In Proceedings of the 20th Int'l Conference on Very large Data Bases*, Santiago, Chile, pp. 487–499.

[3] N. Pasquier, Y. Bastide and R. Taouil et al. (1999). Discovering frequent closed itemsets for association rules. *In Proceedings of the 7th Int'l Conference on Database Theory*, Jerusalem, Israel, January, pp. 398–416.

[4] Mohammed J. Zaki, Ching-Jui Hsiao (1999). Charm: An efficient algorithm for closed association rule mining. *Technical Report 99-10, Computer Science*, Rensselaer Polytechnic Institute.

[5] J. Han, J. Pei, and Y. Yin (2000). Mining frequent patterns without candidate generation. *In Proceedings of the 2000 ACM SIGMOD Int'l Conference on Management of Data*, New York, USA, pp. 1–12.

[6] J. Pei, J. Han, and R. Mao (2000). CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. *In Proceedings of the 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. Dallas, Texas, USA, pp. 21–30.

[7] D. Burdick, M. Calimlim, and J. Gehrke (2001). MAFIA: A maximal frequent item set algorithm for transactional databases. *In Proceedings of the 17th Int'l Conference on Data Engineering*. Heidelberg, pp. 443-452.

[8] J. Y. Wang, J. Han, and J. Pei (2003). CLOSET+: Searching for the best strategies for mining frequent closed itemsets. *In Proceedings of the 9th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, Washington, DC, pp. 236 - 245.

[9] C. Lucchese, S. Orlando, and R. Perego (2006). Fast and memory efficient mining of frequent closed itemsets. *IEEE Trans on Knowledge and Dada Engineering*, vol. 18, no. 1, pp. 21- 36.

[10] R. Singh, T. Johnsten, and V. Raghavan et al (2010). Efficient Algorithm for Discovering Potential Interesting Patterns with Closed Itemsets. *In Proceedings of the 2010 IEEE Int'l Conference on Granular Computing*, pp. 414 - 419.

[11] F. Nori, M. Deypir, and M. Hadi et al. (2011). A new sliding window based algorithm for frequent closed itemset mining over data streams. *In Proceedings of the 1st Int'l Conference on Computer and Knowledge Engineering*, IEEE Press, pp. 249-253.

[12] Guang-Peng Chen, Yu-Bin Yang, and Yao Zhang (2012). MapReduce-Based Balanced Mining for Closed Frequent Itemset. *In Proceedings of the 2012 IEEE 19th Int'l Conference on Web Services*, IEEE Press, pp. 652 - 653.

[13] M. Sreedevi, Reddy L.S.S. (2013). Mining regular closed patterns in transactional databases. *In Proceedings of the 2013 7th Int'l Conference on Intelligent Systems and Control*, IEEE Press, pp. 380 - 383.

[14] Yu-quan Z. and Yu-qing S. (2007). Research on an Algorithm for Mining Frequent Closed Itemsets. *Journal of Computer Research and Development*, vol. 44, no. 7, pp. 1177-1183.

[15] Shengwei L., Lingsheng L., and Chong H. (2009). Mining closed frequent itemset based on FP-Tree. *In Proceedings of the IEEE Int'l Conference on Granular Computing*, IEEE Press, pp. 354 - 357.

[16] Wachiramethin J., Werapun J. (2009). BPA: A Bitmap-Prefix-tree Array data structure for frequent closed pattern mining. *In Proceedings of the 2009 Int'l Conference on Machine Learning and Cybernetics*, IEEE Press, vol .1, pp. 154 - 160.

[17] Z. Pawlak (1982). Rough sets. *Journal of computing and information science in Engineering*, no.11, pp. 341–356.

[18] R. Wille (1982). Restructuring lattice theory: an approach based on hierarchies of concepts. *In: I. Rival (Ed.), Ordered Sets,* Reidel, Dordrecht-Boston, pp. 445–470.

[19] B. Ganter, R. Wille (1999). Formal Concept Analysis, Mathematic Foundations. *Springer*, Berlin.

[20] J. Poelmans, D. I. Ignatov, and S. O. Kuznetsov et al (2013). Formal concept analysis in knowledge processing: A survey on applications. *Expert Systems with Applications*, vol.40, no. 16, pp. 6538–6560.

[21] M. W. Shao, Y. Leung (2014). Relations between granular reduct and dominance reduct in formal contexts. *Knowledge-Based Systems*, vol.65, pp. 1–11.

[22] Wei-Zhi W., Yee Leung, Ju-Sheng M. (2009). Granular Computing and Knowledge Reduction in Formal Contexts. *IEEE Transactions on Knowledge and Data Engineering*, vol.21, no.10, pp. 1461-1474.

[23] R. Belohlavek, B. D. Baets, J. Konecny (2014). Granularity of attributes in formal concept analysis. *Information Sciences*, vol.260, pp.149–170.

[24] Hobbs J. R. (1985). Granularity. *In Proceedings of the 9th International Joint Conference on Artificial Intelligence*, San Francisco, USA, pp. 432-435.

[25] Giunchglia F., Walsh T. (1992). A theory of abstraction. *Artificial Intelligence*, vol. 57, no. 2-3. pp. 323-389.

[26] Yao Y.Y. (2004). A partition model of granular computing. *Lecture Notes in Computer Science Transactions on Rough Sets*, vol. 3100, pp.232–253.

[27] T. R. Qiu, X. Q. Chen, and Q. Liu et al. (2010). Granular Computing Approach to Finding Associa-

tion Rules in Relational Database. *International Journal of intelligent systems*, no. 25, pp. 165–179.

[28] G. Fang, Y. Wu (2013). Frequent Spatiotemporal Association Patterns Mining Based on Granular Computing. *Informatica*, vol.37, no.4, pp.443-453.

[29] Pawlak Z. (1998). Granularity of knowledge, indiscernibility and rough sets. *In Proceedings of IEEE Int Conf on Fuzzy Systems*, IEEE Press, Anchorage, AK, pp.106–110.

[30] Zhang L., Zhang B. (2003). The quotient space theory of problem solving. *Lecture Notes in Computer Science*, vol. 2639, pp. 11–15.