An Efficient Algorithm for Mining Frequent Closed Itemsets

Gang Fang 1,2, Yue Wu 1, Ming Li 1 and Jia Chen 1
1School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P. R. China
2School 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: \((i + 1) – \text{generators}\) are created by joining \(i – \text{generators}\). For the first method, the advantages are the less usage of memory, simple data structure, and easy implementing 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

$$U = \{u_1, u_2, ..., u_n\} \ (n = |U|), \text{ called the universe of discourse},$$

$$A = \{a_1, a_2, ..., a_m\} \ (m = |A|), \text{ called the attributes set},$$

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

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

$$\omega(O) : U \rightarrow P(A), \text{ namely}$$

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

$$\phi(I) : P(A) \rightarrow P(U), \text{ namely}$$

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

And the couple of applications $(\omega, \phi)$ 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 $O_1, O_2, O_3 \subseteq U$ and $I_1, I_2 \subseteq A$, then we have:

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

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

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

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

$h = \omega \circ \phi = \omega \phi = \omega h = \phi h = \omega h = \phi h = \omega h = \phi h.$

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

Extension: $$(3) I \subseteq h(I);$$

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

Monotonicity: $$(5) O \subseteq h(O) \Rightarrow h(O) \subseteq h(O).$$
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.

- $I_1 \subseteq h(I_1), I_2 \subseteq h(I_2)$ (Extension)
- $h(I_1) \cup h(I_2) \subseteq h(I_1 \cup I_2)$ (Monotonicity)
- $h(I_1 \cup I_2) \subseteq h(h(I_1) \cup h(I_2))$ (Monotonicity)

And:

- $I_1 \subseteq I_1 \cup I_2, I_2 \subseteq I_1 \cup I_2$
- $h(I_1) \subseteq h(I_1 \cup I_2), h(I_2) \subseteq h(I_1 \cup I_2)$
- $h(I_1) \cup h(I_2) \subseteq h(I_1 \cup I_2)$ (Idempotency)
- $h(I_1 \cup I_2) = h(h(I_1) \cup h(I_2))$.

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 $I(\subseteq A)$ is the intersection of all objects in $U$ that contain $I$:

$$h(I) = \bigcap_{o \in U} \{o \mid o \in h(I') \}$$

Proof. Let $H = \bigcap_{o \in U} \{o \mid o \in h(I') \}$, where $H \subseteq \bigcap_{o \in U} \{o \mid o \in h(I') \}$.

Let’s show that $S = \{o \mid o \in h(I') \}$, i.e. $I \subseteq h(I')$.

- $\bigcap_{o \in S} \{o \mid o \in h(I') \}$ (Property 2.1)
- $\bigcap_{o \in S} \{o \mid o \in h(I') \}$ (Extension)
- $o \in h(I') \leftrightarrow o \subseteq h(I')$

We have $S = S'$, and also have $h(I) = H$.

Definition 2.5 Formal granule, for a formal context $D = (U, A, R)$, a two-tuple $G = \langle I, \phi(I) \rangle$ 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 $I = \{i_1, i_2, ..., i_k \} \subseteq A, k = |I| \rangle$.

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

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

There are two formal granules $G_a = \langle I_a, \phi(I_a) \rangle$ and $G_b = \langle I_b, \phi(I_b) \rangle$, respectively; then we have:

$$G = G_a \otimes G_b = \langle I_a \cup I_b, \phi(I_a) \cap \phi(I_b) \rangle.$$
Corollary 3.1 Let $I$ be the smallest frequent closed itemset, i.e. $I \in FC_{\text{min}}$. And the frequent closed itemset corresponding to $I$ is $h(I) = \omega(I)$. 

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

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

Proof. $I_a \subseteq I_b \subseteq A \Rightarrow support(I_a) = support(I_b)$. Then we have $h(I_a) = h(I_b)$ and $\forall I \subseteq A$, $h(I_a \cup I) = h(I_b \cup I)$.

Theorem 3.3 $I \in FC_{\text{min}} \Rightarrow \forall I' \subseteq I \land I' \in FC_{\text{min}}$.

Proof. Suppose $I \in FC_{\text{min}} \Rightarrow \exists I_1 \subseteq I \land I_1 \in FC_{\text{min}}$.

Corollary 3.3 $I \subseteq FC_{\text{min}} \Rightarrow \forall I' \subseteq I \land I' \in FC_{\text{min}}$.

Definition 3.5 The smallest frequent closed granules set, the formal granule $G = h(I)$ is said to be the smallest frequent closed granule $G_{\text{min}}$ if the intension $I$ of $G$ is the smallest frequent closed itemset. The set $FCG_{\text{min}}$ of the smallest frequent closed granules is defined as: $FCG_{\text{min}} = \{G = h(I) | I \in FC_{\text{min}}\}$.

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 $P = \{p_1, p_2, \ldots, p_n\}, Q = \{q_1, q_2, \ldots, q_n\}$ be two sets, and then the set vector operation is expressed as $P \odot Q$

$$\begin{align*}
&= \{p_1\} \\
&\quad \odot (\emptyset, \{q_1\}, \{q_2\}, \ldots, \{q_n\}) \\
&= \{p_2\} \\
&\quad \odot (\emptyset, \{q_1\}, \{q_2\}, \ldots, \{q_n\}) \\
&\quad \vdots \\
&= \{p_n\} \\
&\quad \odot (\emptyset, \{q_1\}, \{q_2\}, \ldots, \{q_n\})
\end{align*}$$

(Simple notation)

The operation is the main idea of generator function, let $P, Q$ be two sets, it is expressed as $f(P, Q) = P \odot Q$.

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 $P(A)$ ($\forall p \in P(A) \land p \neq \emptyset$) as follows:

1) $P(A) = \emptyset$

2) $I_a = \{a\} \Rightarrow P(A) = P(A) \cup (I_a \odot P(A))$

3) $I_b = \{b\} \Rightarrow P(A) = P(A) \cup (I_b \odot P(A))$

4) $I_c = \{c\} \Rightarrow P(A) = P(A) \cup (I_c \odot P(A))$
The formal context is a general power set where \( P(A) \) is the range of attribute \( a \). Let \( D \) be the minimum support, then \( \alpha = \min(s) \). For a formal context \( D = (U, A, R) \), if \( A \) is a general set of Boolean attributes, \( P(A) \) is 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 \( \minsup \).

Output: frequent closed itemsets \( FCI \).

1. Read \( D \);
2. Construct \( FG = \{ FG_a \} \) where \( a \in A \), \( \forall G \Rightarrow (I, \phi(I)) \rightarrow FG_a \), and \( I \subset V_a \) \( \| I \| = 1 \), \( \| (\phi(I)) \| \geq \minsup \); \( F = \{ F_a \} \) where \( \forall F_a \in F \), \( \forall V_a \in V_g \), \( F_a \subset F \), \( \phi(V_a) \rightarrow FG_a \); \( F_a \subset FG \) and \( a \in A \); //\( V_a \) is the range of attribute \( a \in A \).
3. \( FC_{min} = \emptyset \);
4. For (\( \forall a \in F \)) do begin;
5. \( S_a = a \cap FC_{min} \); //Generate the candidate
6. For (\( \forall s \in S_a \)) do begin;
7. If (\( \forall t_i \in F_{s+} \land t_i \subset s \) and \( \forall t_j \in F_{s+} \land t_j \subset s \)) then;
8. Construct \( G = s < s, \phi(s) > \);
9. If (\( \| \phi(s) \| \geq \minsup \)) then;
10. \( \alpha = \{ a \} \Rightarrow S_a = \{ \} \); \( FC_{min} = \{ \} \);
11. Write \( G = s < s, \phi(s) > \) to \( FG_{min} \);
12. Write \( s \) to \( FC_{min} \);
13. else;
14. Write \( s \) to \( N_{FC_{min}} \);
15. else;
16. Write \( s \) to \( N_{s+} \);
17. End for (\( \forall G = < I, \phi(I) > \) in \( FG_{min} \)) do begin;
18. Write \( h(I) = \alpha \phi(I) \) to \( FCI \);
19. End for (\( \forall G = < I, \phi(I) > \) in \( FG_{min} \)) do begin;
20. Write \( h(I) = \alpha \phi(I) \) to \( FCI \);
21. End
22. Answer \( FCI \);

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

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( FG = { &lt; a, { 1, 3, 5 } &gt; &lt; b, { 2, 3, 4 } &gt; ), ( &lt; c, { 1, 2, 5 } &gt; &lt; e, { 3, 4, 5 } &gt; )</td>
</tr>
<tr>
<td>2</td>
<td>( \alpha = { a } \Rightarrow S_a = { } ); ( FC_{min} = { } )</td>
</tr>
<tr>
<td>3</td>
<td>( \alpha = { b } \Rightarrow S_a = { } ); ( FC_{min} = { } )</td>
</tr>
<tr>
<td>4</td>
<td>( \alpha = { c } \Rightarrow S_a = { } ); ( FC_{min} = { } )</td>
</tr>
</tbody>
</table>

Table 1: Frequent closed itemsets mining for \( \minsup = 40\% \).

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

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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Descriptions</th>
<th>| P(A)|/| U|</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>The first original dataset</td>
<td>2^{16} – 1; 34015</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>Replicating dataset 1 three attributes</td>
<td>2^{25} – 1; 34015</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>Replicating dataset 1 four times</td>
<td>2^{32} – 1; 34015</td>
</tr>
<tr>
<td>Dataset 4</td>
<td>The second original dataset</td>
<td>5*4*4*1*3*3*1; 296031</td>
</tr>
<tr>
<td>Dataset 5</td>
<td>Replicating dataset 1 one attribute</td>
<td>5*4*4*1*3*3*5*1; 296031</td>
</tr>
<tr>
<td>Dataset 6</td>
<td>Replicating dataset 4 one time</td>
<td>5*4*4*1*4*3*3*1; 2*296031</td>
</tr>
<tr>
<td>Dataset 7</td>
<td>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</td>
<td>2^{102} – 1; 34015</td>
</tr>
</tbody>
</table>

Table 2: The datasets used in the experiments.
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.

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:

\[
p = 1 - \frac{t(EMFCI)}{t(CLOSET)}.
\]

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 figure 17, we know that the runtime of the algorithm is added as the search space or the number of objects is increasing.

Based on figure 18, 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.
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 adding, but it becomes 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 quantitative 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 becomes 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 $I(\subseteq A)$ 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 $I = \{a_1^i, a_2^i, ..., a_n^i\}(I \subseteq A)$, and then we have the following $\| P(I) \| = \prod_{m=1}^{n}(\| V_{a_m^i} \| + 1) - 1$, namely,

$$\| P(I) \| + 1 = \prod_{m=1}^{n}(\| V_{a_m^i} \| + 1) = \prod_{m=1}^{n}(\| V_{a_{m+1}^i} \| + 1) \cdot \prod_{m=1}^{n}(\| V_{a_{m+2}^i} \| + 1) \cdot \prod_{m=1}^{n}(\| V_{a_{m+3}^i} \| + 1) \cdot ... \cdot \prod_{m=1}^{n}(\| V_{a_{m+n}^i} \| + 1)$$

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

Obviously, we also have $\| P(I_v) \| + 1 = (\| P(I_m) \| + 1) \cdot (\| P(I_{m_1}) \| + 1) \cdot ... \cdot (\| P(I_{m_k}) \| + 1)$,

Where $I_m = \{a_1^i, a_2^i, ..., a_n^i\}$,

$\{a_1^{m_1}, a_2^{m_1}, ..., a_{n_1}^{m_2}\}$, ...

$\{a_{m-1}^{m_k}, ..., a_{n-1}^{m_k}, a_{n_k}\}$.

In this paper, we let $\| P(I_v) \| < \lambda = 2^{19}$. If $\lambda$ is too big, the method also has the same bottleneck; if $\lambda$ 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. $F_G = \{< a_1>, [1,3,5], < b_1>, [2,3,4],< c_1>, [1,2,5], < e_1>, [3,4,5] >\}$.  

Step2. $F = \{a_1, b_1, c_1, e_1\}$, $P(F) = 15 > \lambda = 4$.  

Step3. Partitioning the search space, get two search spaces $F_1 = \{a_1, b_1\}, F_2 = \{c_1, e_1\}$, where $P(F_1) < 4$.  

Step4. For the first search space $F_1 = \{a_1, b_1\}$, have

$1) a_1 \Rightarrow S_1 = \{a_1\}$

$FG_{min}^1 = \{< a_1>, [1,3,5] >, F_{G1} = \{a_1\}$

$2) b_1 \Rightarrow S_1 = \{b_1, \}$

$FG_{min}^1 = \{< a_1>, [1,3,5] >, < b_1>, [2,3,4] >\}$

$FC_{min}^1 = \{a_1, b_1\}$.  

For the second search space $F_2 = \{c_1, e_1\}$, have

$1) c_1 \Rightarrow S_2 = \{c_1\}$

$FG_{min}^2 = \{< c_1>, [1,2,5] >, FC_{min}^2 = \{c_1\}$

$2) e_1 \Rightarrow S_2 = \{e_1\}$

$FG_{min}^2 = \{< e_1>, [1,2,5] >, < e_1>, [3,4,5] >\}$

\[ FC_{\text{min}}^2 = \{ \{ c \}, \{ e \} \} . \]

Step 5. \[ F = \{ FC_{\text{min}}^1, FC_{\text{min}}^2 \} , \] repeating the step 2, 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.

\[ \alpha = FC_{\text{min}}^1 \Rightarrow S_\alpha = \{ \{ a \}, \{ b \} \} ; \]

\[ FC_{\text{min}}^1 = \{ \{ a \}, \{ b \} \} ; \quad FG_{\text{min}} = \langle \langle a, \{ 1, 3, 5 \} \rangle, \langle b, \{ 2, 3, 4 \} \rangle \rangle \]

\[ \alpha = FC_{\text{min}}^2 \Rightarrow S_\alpha = \left\{ \left\langle \{ c \}, \{ e \} \right\rangle \left\{ \emptyset, \{ a \}, \{ b \} \right\rangle = \right\} \]

\[ \{ \{ c \}, \{ a \}, \{ b \}, \{ e \}, \{ a \}, \{ b \} \} ; \]

\[ FC_{\text{min}}^2 = \{ \{ a \}, \{ b \} \} ; \quad FG_{\text{min}} = \langle \langle a \rangle, \{ 1, 3, 5 \} \rangle, \langle b \rangle, \{ 2, 3, 4 \} \rangle \rangle \]

\[ < \{ c \}, \{ 1, 2, 5 \} \rangle, \langle \{ a \}, \{ 1, 5 \} \rangle, \langle \{ e \}, \{ 3, 4, 5 \} \rangle \rangle, \]

\[ < \{ a \}, \{ 3, 5 \} \rangle, < \{ b \}, \{ 3, 4 \} \rangle \rangle \]

\[ FC_{\text{min}} = \{ \{ a \}, \{ b \}, \{ c \}, \{ a \}, \{ e \}, \{ a \}, \{ b \} \} \]

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.

<table>
<thead>
<tr>
<th>Items</th>
<th>Time complexity</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-close</td>
<td>( O(</td>
<td></td>
</tr>
<tr>
<td>CLOSET</td>
<td>( O(</td>
<td></td>
</tr>
<tr>
<td>IEMFCI</td>
<td>( O(L /k+1) \cdot</td>
<td></td>
</tr>
</tbody>
</table>

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](image)

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

Then, for the improved algorithm IEMFCI, we adopt different parameters \( \lambda \) to test its trend of performance, where \( \lambda = 2^2, 2^9 \) and \( \lambda = 2^{12} \). 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 22: Performance comparison with the higher support on dataset 7](image)

Figure 22: Performance comparison with the higher support 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

References

