ARCA: An Algorithm for Mining Association Rules based Concept Lattice

Qing Xia  
Library  
Huaihai Institute of Technology  
Lianyungang, China, 222005  
Email:xiaqing1@163.com

Sujing Wang  
College of Computer Science and Technology.  
Jilin University, Changchun, China, 130012  
Email: sujingwang@hotmail.com

Zhen Chen  
College of Computer Science and Technology,  
Jilin University, Changchun, China, 130012  
Email: chenzhen@jlu.edu.cn

Tao Lv  
Weapon System Engineering Center  
Artillery Command Academy,XuanHua, China 075100  
Email: lvtom@sohu.com

Dongjing Wang  
Shizhuang Middle School, Lianyungang, China, 222200  
Email: rpcwangdongjing@163.com

Abstract—Association rule discovery is one of kernel tasks of data mining. Concept lattice, induced from a binary relation between objects and features, is a very useful formal analysis tool. It represents the unification of concept intension and extension. It reflects the association between objects and features, and the relationship of generalization and specialization among concepts. There is a one-to-one correspondence between concept intensions and closed frequent itemsets. This paper presents an efficient algorithm for mining association rules based concept lattice called Arca (Association Rule based Concept Lattice). Arca algorithm uses concept-matrix to build a part of concept lattice, in which the intension of every concept be put into one-to-one correspondence with a closed frequent itemset. Then all association rules are discovered by 4 operators which are defined in this paper performed on these concepts.

Index Terms—Concept lattice, rank of matrix, formal concept analysis.

I. INTRODUCTION

Association rule mining from a transaction database has been a very active research area since the publication of the Apriori algorithm [1]. Several improvements to the basic algorithm and many new approaches [2]–[10] have been proposed during the last decade. With the development of research, Association rule discovery is one of kernel tasks of data mining.

Formal Concept Analysis (FCA) was developed by Pro. Wille in 1982 [11]. Concept Lattice, the core data structure in Formal Concept Analysis, has been widely in machine learning, data mining and knowledge discovery, etc. Every node of concept lattice is a formal concept consisting of extent and intent. Concept lattice embodies the relations between extension and intension. Here is a one-to-one correspondence between concept intensions and closed frequent itemsets.

There are various algorithms [12]–[16] of association rule mining using concept lattice. However, These algorithms need to build a complete concept lattice. Based on CMCG algorithm [17], this paper presents an algorithm Arca of association rule mining using a part of concept lattice.

The paper is organized as follows. Section 2 recalls basic definitions of association rule and concept lattice. Section 3 discusses Arca algorithm and four operator. Section 4 gives an experimental evaluation on the time spent of Arca algorithm and Apriori algorithm. Section 5 concludes the paper.

II. THE DEFINES OF ASSOCIATION RULE AND CONCEPT LATTICE

Let \( \mathcal{I} = \{i_1, i_2, \ldots, i_m\} \) be a set of \( m \) items. Let \( (T) = \{t_1, t_2, \ldots, t_n\} \), the task-relevant data, be a set of database transactions where each transaction \( t \) is a set of items such that \( t \subseteq \mathcal{I} \). Each transaction is associated with an identifier, called TID. Each transaction \( t \) consists of a set of items \( I \) from \( \mathcal{I} \). If \( |I| = k \), then \( I \) is called a \( k \)-itemset. A transaction \( t \) is said to contain \( I \) if and only if \( I \subseteq t \). An association rule is an implication of ten form \( I_1 \Rightarrow I_2 \), where \( I_1, I_2 \subseteq \mathcal{I} \) and \( I_1 \cap I_2 = \emptyset \). The rule \( I_1 \Rightarrow I_2 \) holds in the transaction set \( \mathcal{T} \) with support \( s \), where \( s \) is the percentage of transactions in \( \mathcal{T} \) that contain \( I_1 \cup I_2 \) (i.e., both \( I_1 \) and \( I_2 \)). This is taken to be the probability, \( P(I_1 \cup I_2) \). The rule \( I_1 \Rightarrow I_2 \) has confidence \( c \) in the transaction set \( \mathcal{T} \) if \( c \) is the percentage of transactions in \( \mathcal{T} \) containing \( I_1 \) that also contain \( I_2 \). This is taken to be the conditional probability, \( P(I_2/I_1) \). That is,

\[
\text{support}(I_1 \Rightarrow I_2) = P(I_1 \cup I_2) \quad (1)
\]

\[
\text{confidence}(I_1 \Rightarrow I_2) = P(I_2/I_1) \quad (2)
\]

Given the user defined minimum support \( \text{minsupp} \) and minimum confidence \( \text{minconf} \) thresholds. If the support of \( I \subseteq t \) itemset \( I \) be greater or equal to \( \text{minsupp} \), \( I \) is called a frequent itemset.

Example 2.1: For \( \mathcal{I} = \{A, B, C, D, E\}, \mathcal{T} = \{A, B, C, D, E\} \), Table I represents a transaction database.

Definition 2.1: A data mining context is a triple: \( (\mathcal{I}, \mathcal{J}, \mathcal{R}) \), where \( \mathcal{I} \) and \( \mathcal{J} \) are two sets, and \( \mathcal{R} \) is a relation between \( \mathcal{I} \) and \( \mathcal{J} \). \( \mathcal{J} = \{t_1, t_2, \ldots, t_n\} \), each \( t_i (i \leq n) \) is...
TABLE I
A TRANSACTION DATABASE

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A CD</td>
</tr>
<tr>
<td>2</td>
<td>BC E</td>
</tr>
<tr>
<td>3</td>
<td>ABC E</td>
</tr>
<tr>
<td>4</td>
<td>B E</td>
</tr>
<tr>
<td>5</td>
<td>ABC E</td>
</tr>
</tbody>
</table>

TABLE II
A TRANSACTION DATABASE

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 1. A context-matrix of the data mining context showed in Table II

called an object. $I = \{i_1, i_2, \ldots, i_m\}$, each $i_j (j \leq m)$ is called an attribute.

In a data mining context $D = (I, J, R)$, if $(t, i) \in R$, we say that the attribute $i$ is an attribute of the object $t$, or that $t$ verifies $i$. In this paper, $(t, i) \in R$ is denoted by 1, and $(t, i) \notin R$ is denoted by 0. Thus, a data mining context can be represented by a matrix only with 0 and 1. We say that the matrix is the context-matrix of $D$.

Example 2.2: Table II represents a data mining context corresponding with the transaction database showed in Table I.

Example 2.3: Fig 1 a context-matrix of the data mining context showed in Table II.

Definition 2.2: Let $D = (I, J, R)$ be a data mining context. We define a function $f(T)$ that produces the set of their common attributes for every set $T \subseteq I$ of objects to know which attributes from $J$ are common to these entire objects: $f(T) = \{i \in J \mid \forall t \in T, (t, i) \in R\}$.

Dually, we define $Y$ for subset of attributes $I \subseteq J$, $g(I)$ denotes the set consisting of those objects in $T$ that have all the attributes from $I$: $g(I) = \{t \in T \mid \forall i \in I, (t, i) \in R\}$. Let $h(I) = f(g(I))$.

These two functions are used to determine a formal concept.

Definition 2.3: Let $D = (I, J, R)$ be a data mining context. A pair $(T, I)$ is called a formal concept of $D$, for short, a concept, if and only if $T \subseteq I$, $I \subseteq J$, $f(T) = I$ and $g(I) = T$. $T$ is called extent, $I$ is called intent.

Definition 2.4: Let $D = (I, J, R)$ be a data mining context. The set of all concepts of $D$ is denoted by $B(D)$. $C_1 = (T_1, I_1)$ and $C_2 = (T_2, I_2)$ are two concepts in $B(D)$. An partial ordering relation $\succ$ is defined on $B(D)$ by: $C_1 \succ C_2 \iff T_1 \subset T_2$ or $C_1 \prec I_2 \subset I_1$.

We say that $C_2$ is called a superconcept of $C_1$ and $C_1$ is called a subconcept of $C_2$. $B(D)$ and the partial ordering relation $\succ$ form a complete lattice called the concept lattice of $D$ and denoted by $L(D)$.

Example 2.4: Fig 2 a concept lattice for the context of Table II.

Definition 2.5: Let $D = (I, J, R)$ be a data mining context. $C_1$ and $C_2$ are two concepts in $B(D)$. If $C_1 \subset C_2$ and there is no concept $C_3$ in $B(D)$ fulfilling $C_1 \subset C_3 \subset C_2$, $C_1$ is called a lower neighbor of $C_2$, denoted by $C_1 \prec C_2$, and $C_2$ is called an upper neighbor of $C_1$.

The set of all lower neighbors of a given concept is a subset of the set consisting of all subconcepts of it.

Definition 2.6: Let $I \subseteq J$ be a set of items from $D = (I, J, R)$. The support count of the itemset $I$ in $D$ is:

$$\text{support}(I) = \frac{|g(I)|}{|J|}$$

(3)

Definition 2.7: Let $I \subseteq J$ be a set of items from $D = (I, J, R)$. If $\text{support}(I) \geq \text{minsupp}$, $I$ is called a frequent itemset.

Definition 2.8: Let $D = (I, J, R)$ be a data mining context. For given a item $i \in I$, if the count of 1s in the corresponding column of the item $i$ in the concept-matrix of $D$ is $n$ we say that the rank of item $i$ in concept-matrix of $D$ is $n$, denoted by $r(i) = n$. If $m = \max\{r(i) \mid i \in I\}$, we say that the rank of the data mining context $D$ is $m$.

Example 2.5: Fig 3 represents the concept-matrix of concept $(135, AC')$. 

Fig. 2. A Concept lattice for the context of Table II

Fig. 3 represents the concept-matrix of context.
Definition 2.10: Let \( \mathcal{D} = (\mathcal{I}, \mathcal{J}, \mathcal{R}) \) be a data mining context. \( C = (T, I) \) is a concept in \( B(\mathcal{D}) \). If the count of \( T \) in the corresponding column of the item \( i \) in the concept-matrix of \( C \) is \( n \), we say that the rank of item \( i \) in concept-matrix of concept \( C \) is \( n \), denoted by \( R_C(i) = n \). If \( m = \max\{R_C(i) \mid i \in \mathcal{J}, i \notin I\} \), we say that the rank of concept \( C \) is \( m \).

Property 2.1: The count of objects of subconcept of concept \( C \) is equal or lesser than \( m \).

Definition 2.11: Let \( \mathcal{D} = (\mathcal{I}, \mathcal{J}, \mathcal{R}) \) be a data mining context. \( C = (T, I) \) is a concept in \( B(\mathcal{D}) \). Given subset \( I_1 \subseteq \mathcal{J} \), \( gc(I_1) \) denotes the set consisting of those transactions in \( T \) that have all the itemsets from \( \mathcal{J} \): \( gc(I_1) = \{ t \in T \mid \forall i \in I_1 \} \).

Definition 2.12: Let \( \mathcal{D} = (\mathcal{I}, \mathcal{J}, \mathcal{R}) \) be a data mining context. \( C = (T, I) \) is a concept in \( B(\mathcal{D}) \). If \( |T| \geq |\mathcal{J}| \times \minsupp \), \( C \) is called a frequent concept.

Property 2.2: If \( C = (T, I) \) is a concept, \( I \) is a frequent itemset.

Proof: \( |T| \geq |\mathcal{J}| \times \minsupp \), so \( \frac{|T|}{|\mathcal{J}|} \geq \minsupp \). And \( T = g(I) \), so \( \frac{|T|}{|I|} \geq \minsupp \). Then \( I \) is a frequent itemset.

Theorem 2.1: Let \( \mathcal{D} = (\mathcal{I}, \mathcal{J}, \mathcal{R}) \) be a data mining context. \( C = (T, I) \) is a concept in \( B(\mathcal{D}) \). The rank of concept \( C \) is \( m \). For \( \forall i \in \mathcal{J} \), \( gc(i) \) is a lower neighbor of \( C \).

Proof: By Definition 2.10 and Definition 2.11, we have \( |gc(i)| = m \). Suppose there exist \( C_3 = (T_3, I_3) \), where \( C_1 \prec C_2 \). We can obtain \( m = |gc(i)| < |T_2| < |T| \). By Property 2.1, we have \( |T_2| \leq m \). This result contradicts with \( m < |T_2| \).

Then \( C_3 \) is a lower neighbor of \( C \).

Theorem 2.2: Let \( \mathcal{D} = (\mathcal{I}, \mathcal{J}, \mathcal{R}) \) be a data mining context. \( C = (T, I) \) is a concept in \( B(\mathcal{D}) \). The rank of concept \( C \) is \( m \), \( C_1 = (gc(i_1), f(gc(i_1))) \) is a subconcept of \( C \), where \( i_1 \in \mathcal{J} \) and \( \frac{|gc(i_1)|}{|T|} > \frac{|gc(i)|}{|T|} \). On the other hand, \( C_1 \prec C_2 \). \( C_1 \prec C_2 \) implies that \( gc(i_1) \subset I_3 \). This result contradicts with \( gc(i_1) \subset I_3 \).

Proof: Suppose there exist \( C_3 = (T_3, I_3) \), where \( C_1 < C_3 < C \). We have \( m_1 = |g(i_1) \cap T| < |T_3| |T| \), it implies that \( C_3 \) is a concept in \( C_2 \), \( |C_2| < |C| \) and \( m_1 = |C_2| < |T_2| \), and NOT \( gc(i_1) \subset I_3 \). On the other hand, \( C_3 \prec C_3 \) implies that \( gc(i_1) \subset I_3 \). This result contradicts with \( gc(i_1) \subset I_3 \).

Definition 2.13: An association rule is an implication between itemsets of the form \( r : I_1 \rightarrow I_2 \), where \( I_1, I_2 \subseteq \mathcal{J} \) and \( I_1 \cap I_2 = \emptyset \). \( I_1 \) is called the antecedent of \( r \) and \( I_2 \) is called the consequent of \( r \). Below, we define the support and confidence of an association rule:

\[
\text{support}(r) = \frac{|g(I_1 \cup I_2)|}{|\mathcal{J}|}
\]

\[
\text{confidence}(r) = \frac{\text{support}(I_1 \cup I_2)}{\text{support}(I_1)} = \frac{|g(I_1 \cup I_2)|}{|g(I_1)|}
\]

Mining association rules is to find all rules \( r \), where \( \text{support}(r) \geq \minsupp \) and \( \text{confidence}(r) \geq \minconf \).

III. ARCA ALGORITHM

When a concept lattice is built, each concept \( C = (T, I) \) holds \( |T| \geq \minsupp \).

Example 3.1: Fig. 4 represents a concept lattice while \( \minsupp = 0.4 \).

Table III represents basic association rules from Fig. 4 with \( \minconf = 0.5 \).

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
basic association rule & \multicolumn{1}{c|}{minimum support} \\
\hline \( \emptyset \rightarrow BE \) & 4/5 \\
\( \emptyset \rightarrow C \) & 4/5 \\
BE \rightarrow C & 3/4 \\
C \rightarrow BE & 3/4 \\
C \rightarrow A & 3/4 \\
BCE \rightarrow A & 2/3 \\
AC \rightarrow BE & 2/3 \\
\hline
\end{tabular}
\end{table}

Definition 3.1: Let \( \mathcal{D} = (\mathcal{I}, \mathcal{J}, \mathcal{R}) \) be a data mining context. \( C_1 = (T_1, I_1) \) and \( C_2 = (T_2, I_2) \) are two concepts in \( B(\mathcal{D}) \) and \( C_1 \prec C_2 \). If \( \frac{|T_1|}{|T_2|} \geq \minconf \), the rule \( r : I_2 \Rightarrow I_1 - I_2 \) is called a basic association rules. The minimum support of \( r \) is defined as \( \frac{|T_1|}{|T_2|} \).

Basic association rules can be mined from concept lattice.

Example 3.2: Table III represents basic association rules from Fig 4 with \( \minconf = 0.5 \).

Definition 3.2: Let \( r_1 \equiv I_1 \rightarrow I_2 \), \( r_2 \equiv I_3 \rightarrow I_4 \), the minimum supports of \( r_1 \) and \( r_2 \) are \( \text{conf}_{f_1} \) and \( \text{conf}_{f_2} \), respectively. If \( I_1 \cup I_2 = I_3 \), The operator \( \cdot \) can be implemented on \( r_1 \) and \( r_2 \). \( r_1 + r_2 \equiv I_1 \rightarrow I_4 \). The minimum support of \( I_1 \rightarrow I_4 \) is defined as \( \text{conf}_{f_1} \cdot \text{conf}_{f_2} \). Let \( r_3 \equiv I_4 \rightarrow I_5 \) and its minimum supports is \( \text{conf}_{f_3} \). If \( I_1 \cup I_2 = I_3 \) and \( I_3 \cup I_4 = I_5 \), the operator \( \cdot \) can be implemented on \( r_1 \), \( r_2 \) and \( r_3 \). \( r_1 + r_2 + r_3 \equiv I_1 \rightarrow I_6 \) and its minimum support
TABLE IV

<table>
<thead>
<tr>
<th>Basic rules</th>
<th>The result of ‘+’</th>
<th>The result of ‘⊕’</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬→ C</td>
<td>¬→ BCE</td>
<td></td>
</tr>
<tr>
<td>BE→ A</td>
<td>BE→ AC</td>
<td></td>
</tr>
<tr>
<td>¬→ B</td>
<td>¬→ ABCE</td>
<td></td>
</tr>
<tr>
<td>C→ A</td>
<td>C→ ABCE</td>
<td></td>
</tr>
<tr>
<td>¬→ B</td>
<td>¬→ AC</td>
<td></td>
</tr>
<tr>
<td>C→ B</td>
<td>C→ ABCE</td>
<td></td>
</tr>
<tr>
<td>¬→ B</td>
<td>¬→ ABCE</td>
<td></td>
</tr>
</tbody>
</table>

is \(\text{conf}_{123} = \text{conf}_1 \times \text{conf}_2 \times \text{conf}_3\). The operator '+' can be defined among \(n\) rules by same way.

\textbf{Definition 3.3:} Let \(r_1 \equiv I_1 \rightarrow I_2\), \(r_2 \equiv I_3 \rightarrow I_4\), the minimum supports of \(r_1\) and \(r_2\) are \(\text{conf}_1\) and \(\text{conf}_2\), respectively. If \(I_1 \cup I_2 = I_3\), the operator '⊕' can be implemented on \(r_1\) and \(r_2\), \(r_1 \oplus r_2 \equiv I_1 \rightarrow I_2 \cup I_4\). The minimum support of \(I_1 \rightarrow I_2 \cup I_4\) is defined as \(\text{conf}_{123} = \text{conf}_1 \times \text{conf}_2\).

\(r_3 \equiv I_4 \rightarrow I_5\) and its minimum supports is \(\text{conf}_3\). If \(I_1 \cup I_2 = I_3\) and \(I_3 \cup I_4 = I_5\), the operator '⊕' can be implemented on \(r_1\), \(r_2\) and \(r_3\), \(r_1 \oplus r_2 \oplus r_3 \equiv \bigcap I_1 \rightarrow I_2 \cup I_4 \cup I_6\) and its minimum support is \(\text{conf}_{123} = \text{conf}_1 \times \text{conf}_2 \times \text{conf}_3\).

The operator '⊕' can be defined among \(n\) rules by same way.

\textbf{Example 3.3:} Since BE, the union of the antecedent and consequent of rule , be not equal to the antecedent of rule , the operators '+' and '⊕' cannot be implemented on rule and rule . The antecedent of rule be equal to BE. Therefore, the operators '+' and '⊕' cannot be implemented on rule and rule .

\[
+ \equiv \emptyset \rightarrow C \quad (6) \\
⊕ \equiv \emptyset \rightarrow BCE \quad (7)
\]

Every basic association rule after rule is check whether the operators '+' and '⊕' can be implemented on rule and itself.

Table IV represents the result of performing '+' and '⊕' on basic association rules in Table III.

\textbf{Definition 3.4:} Let \(r_1 \equiv I_1 \rightarrow I_2\). The result of performing operator \(right move\) is a set of association rules \(\{I'_1 \rightarrow I_2 \cup I'_1 \cap I'_2 = \emptyset, I'_1 \cap I'_2 = I_1, I'_1 \neq \emptyset, I'_2 \neq \emptyset, g(I'_1) \leq g(I'_2)\}\)

\textbf{Definition 3.5:} Let \(r_1 \equiv I_1 \rightarrow I_2\). The result of performing operator decompose is a set of association rule \(\{I_1 \rightarrow I'_2, I'_2 \neq\emptyset, I'_2 \subset I_2\}\)

All association rules which confidence be less than 1 are mining out by performing four operators on basic association rules. The pseudo codes are given by Algorithm 1.

\textbf{Algorithm 1 GenBasicRules(\mathcal{P}, \text{minsupp}, \text{minconf})}

\textbf{Require:} Input: minimum support \(\text{minsupp}\), minimum confidence \(\text{minconf}\) and data mining context \(\mathcal{P} = (\mathcal{T}, \mathcal{A}, \mathcal{R})\)

\textbf{Ensure:} Output: a set of basic association rules \(\text{BasicRules}\)

\begin{algorithmic}
\STATE Initialize a queue \(\text{Queue}\)
\STATE \text{BasicRules} \leftarrow \emptyset
\WHILE {\text{Queue} \neq \emptyset}
\STATE \(C = \text{Queue}.\text{DeQueue}\)
\FORALL {\(C_1\) such that \(C_1 \in \text{SUBNODES} * (C)\)}
\IF {(|\text{Extent}(C_1)|/|\text{Extent}(C)|) \geq \text{minconf}}
\STATE \text{Queue}.\text{EnQueue}(\langle C_1 \rangle)
\ENDIF
\ENDFOR
\ENDWHILE
\STATE \text{return BasicRules}
\end{algorithmic}

\textbf{Algorithm 2 SUBNODES*(C, \mathcal{P}, \text{minsupp})}

\textbf{Require:} Input: given a concept \(C = (T, I)\), minimum support \(\text{minsupp}\), and data mining context \(\mathcal{P} = (\mathcal{T}, \mathcal{A}, \mathcal{R})\)

\textbf{Ensure:} Output: a set \(\text{subnodes}\) of concepts which extent’s count be greater or equal to \(\text{minsupp}\)

\begin{algorithmic}
\STATE \text{subnodes} \leftarrow \emptyset
\STATE \(M\) \leftarrow \text{the concept matrix of concept } C
\STATE \(m\) \leftarrow \text{the rank of concept } C
\WHILE {\(m \geq \text{minsupp}\)}
\STATE \(S\) \leftarrow \text{the set of attributes which ranks equal to } m
\WHILE {\(S \neq \emptyset\)}
\STATE \(I_1\) \leftarrow \text{the set consisting of a attribute } a \text{ from } S \text{ and these attributes from } S \text{ which corresponding columns are same as column of } a \text{ in } M
\STATE \(S\leftarrow S - I_1\)
\STATE \(T_1\) \leftarrow g_C(I_1)
\STATE \(I_1\leftarrow I \cup I_1\)
\IF {\(\forall C_2 = (T_2, I_2) \in \text{subnodes}\)}
\STATE \text{subnodes} \leftarrow \text{subnodes} \cup \langle T_1, I_1 \rangle
\ENDIF
\ENDWHILE
\STATE \(m\leftarrow m - 1\)
\ENDWHILE
\STATE \text{return subnodes}
\end{algorithmic}

Algorithm 1 is to generate basic association rules by build a concept lattice called function \text{SUBNODES*}. The pseudo codes of this function are given by Algorithm 2.
IV. EVALUATION

In order to evaluate, we implement Algorithm Arca and Algorithm Aprior by Visual C++ and STL. The data set, generated randomly by IBM dataset generator, have 1000 items and 10000 transactions. The result shows that the performance of Arca is as four times higher as Aprior on average. Fig 5 represents Running time of algorithms versus lattice size with $minconf = 0.01$.

V. CONCLUSION

Now, there are many algorithms of mining association rules. There is a one-to-one correspondence between concept intensions and closed frequent itemsets. Concept lattice is a good tool for mining association rules.

REFERENCES