International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **10(B)**, October **2012**

 ISSN 1349-4198
pp. 7117–7132

MINING MOST GENERALIZATION ASSOCIATION RULES BASED ON FREQUENT CLOSED ITEMSET+

Bay Vo[1], Tzung-Pei Hong[2,3,*] and Bac Le[4]

[1]Department of Computer Science
Ho Chi Minh City University of Technology
144/24 Dien Bien Phu, Ho Chi Minh, Viet Nam
vdbay@hcmhutech.edu.vn

[2]Department of Computer Science and Information Engineering
National University of Kaohsiung
No. 700, Kaohsiung University Rd., Nan Tzu Dist., Kaohsiung 811, Taiwan
*Corresponding author: tphong@nuk.edu.tw

[3]Department of Computer Science and Engineering
National Sun Yat-sen University
No. 70, Lienhai Rd., Kaohsiung 80424, Taiwan

[4]Department of Computer Science
University of Science
227 Nguyen Van Cu, Ho Chi Minh, Viet Nam
lhbac@fit.hcmus.edu.vn

Received August 2011; revised January 2012

Abstract. *Association rule mining plays an important role in knowledge discovery and data mining. The rules obtained by some previous works based on support and confidence measures might be redundant to a certain degree. This paper thus proposes the concept of most generalization association rules (MGARs), which are more compact than the three previous rule types that include traditional association rules, non-redundant association rules and minimal non-redundant association rules. Some theorems relating to the properties of MGARs are derived as well, and an algorithm based on the theorems for effectively pruning unpromising rules early is then proposed. Hash tables are used to check whether the generated rules are redundant or not. Experimental results show that the number of MGARs generated from a database is much smaller than that of non-redundant association rules and that of minimal non-redundant association rules.*
Keywords: Association rules, Data mining, Frequent closed itemset, Most generalization association rule, Non-redundant association rules

1. **Introduction.** Association rules are the most common knowledge type in data mining [1]. They have wide applications, such as basket data analysis [1], evaluation of regional environments, and bioinformatics, among others. There are a variety of association rule types, such as traditional association rules [1], non-redundant association rules [20,21] and minimal non-redundant association rules [2,11,12]. A traditional association rule is an expression form $R : X \xrightarrow{s,c} Y$, where s is the support of R, and c is the confidence of R. An association rule $R_1 : X_1 \xrightarrow{s,c} Y_1$ is a minimal non-redundant association rule if and only if there does not exist an association rule with the same support value and confidence value as R_1, but with a more specific antecedent part and a more general consequent part. Although the approaches for mining association rules are different, their processing

+ This is a modified and expanded version of the paper "Mining the most generalization association rules", presented at the ACIIDS 2010, Viet Nam [21].

is nearly the same. Their mining processes are usually divided into the following two phases:

i) Mining frequent (FIs) or frequent closed itemsets (FCIs);

ii) Mining traditional or (minimal) non-redundant association rules from FIs/FCIs.

The number of traditional association rules mined out in real applications is usually large. Some approaches have thus been proposed to reduce the number of rules and increase rule usefulness for users [2,11,12,20,21]. Although these approaches may generate fewer rules than the previous methods, the number of rules is still large. For example, for the Chess database with $minSup = 70\%$ and $minConf = 0\%$, the number of rules generated by the traditional method [1] is 8,111,370; by Zaki's method [20] is 152,074; and by Bastide et al.'s method [2] is 3,373,625. In fact, in a mined knowledge set, some rules may be inferred from some other rules. For illustrating this problem, we consider an example database [22] as follows:

TABLE 1. An example database

TID	*Items Bought*
1	*A, C, T, W*
2	*C, D, W*
3	*A, C, T, W*
4	*A, C, D, W*
5	*A, C, D, T, W*
6	*C, D, T*

If minimal non-redundant rules are mined with $minSup = 50\%$ and $minConf = 80\%$, the results are shown in Table 2.

TABLE 2. Minimal non-redundant rules mined from the database in Table 1 with $minSup = 50\%$ and $minConf = 80\%$

Exact rules (conf = 100%)	Rules with 80% ≤ conf < 100%
	$C \xrightarrow{5,5/6} W$
$D \xrightarrow{4,1} C$	
$T \xrightarrow{4,1} C$	
$W \xrightarrow{5,1} C$	$W \xrightarrow{4,4/5} AC$
$A \xrightarrow{4,1} CW$	
$DW \xrightarrow{3,1} C$	
$AT \xrightarrow{3,1} W$	
$TW \xrightarrow{3,1} A$	

There are nine rules, and some of them are redundant according to their supports and confidences. For example, the rule $DW \xrightarrow{3,1} C$ is a weaker (more specific) rule because there exist some stronger (more general) rules such as $D \xrightarrow{4,1} C$ and $W \xrightarrow{5,1} C$. The rule $AT \xrightarrow{3,1} W$ is also a weaker rule because there is a stronger rule $A \xrightarrow{4,1} CW$. Similarly, the rule $W \xrightarrow{4,4/5} AC$ will not be kept since its more general rule $W \xrightarrow{5,1} C$ has been generated. The two previous methods (non-redundant association rules and minimal non-redundant

association rules) will not prune the above rules. Based on the consideration above, it is thus necessary to have a method for pruning these weak rules.

The main contribution of the paper includes the following three main points. Firstly, we propose a new concept, named *most generalization rules* (MGARs), which are the rules sets after weak rules are pruned and are thus more compact (in the number of generated rules) than the three previous rule types. Secondly, some theorems are presented and proved to guide the pruning of redundant rules in the mining process. Thirdly, the experimental results also show that the proposed algorithm has good improvement in terms of pruning unnecessary rules when compared with traditional association rules, non-redundant association rules and minimal non-redundant association rules.

A practical use of the results is for fast prediction. In rule-based prediction, the strongest matched rule in a rule set is often used to predict an event or a sequence of events. A smaller rule set represents less match time. Therefore, mining a smallest rule set can help reduce the prediction time.

The rest of this paper is organized as follows. Some studies related to mining association rules are reviewed in Section 2. Some formal definitions and derived theorems are stated in Section 3. An algorithm for generating most generalization association rules based on FCIs is proposed in Section 4. An example to illustrate the proposed algorithm is given in Section 5. Experimental results of the proposed algorithm are shown in Section 6. The conclusions are given in Section 7.

2. **Related Works.**

2.1. **Mining frequent closed itemsets.**

Frequent itemsets play an important role in the mining process. A frequent itemset can be formally defined as follows [1]: Let D be a transaction database and I be the set of items in D. The support $\sigma(X)$ of an itemset $X(X \subseteq I)$ is the number of transactions in D containing X. X is called a frequent itemset if $\sigma(X) \geq minSup$, where $minSup$ is a predefined minimum support threshold.

FCIs are a variant of frequent itemsets that reduce the number of rules. Formally, an itemset X is called an FCI if it is frequent, and there is no other frequent itemset Y such that $X \subset Y$ and $\sigma(X) = \sigma(Y)$. Many methods have been proposed for mining FCIs from databases. They can be divided into the following four categories [8,19]:

i) Generate-and-test approaches: These approaches are mainly based on the Apriori algorithm and use the level-wise approach to discover FCIs. Examples include Close [12] and A-Close [11].
ii) Divide-and-conquer approaches: These approaches adopt the divide-and-conquer strategy and use compact data structures extended from the frequent-pattern (FP) tree to mine FCIs. Examples include Closet [13], Closet+ [16], and FPClose [3].
iii) Hybrid approaches: These approaches integrate the above two approaches to mine FCIs. They first transform the database to be processed into the item×Tidlist structure (the vertical data format), in which transactions with the same items are grouped together. These approaches then utilize some pruning properties to quickly prune non-closed itemsets. CHARM [22] and CloseMiner [14] are examples.
iv) Hybrid approaches without duplication: These approaches differ from the previous hybrid ones in that they do not use the subsumption-checking technique, such that FCIs need not be stored in the main memory. Unlike CHARM, they do not use the hash-table technique. DCI-Close [9], LCM [15], and PGMiner [10] belong to this category.

2.2. **Mining traditional association rules.** Mining traditional association rules was proposed by Agrawal et al. in 1993 [1]. An association rule has the expression form $X \xrightarrow{s,c} Y$, where X and Y are frequent itemsets, s is the support of the rule and is calculated by the count of X and Y (denoted by $\sigma(XY)$), and c is the confidence of the rule and is calculated by $\sigma(XY)/\sigma(X)$. The purpose of mining traditional association rules is to find all the rules with their supports greater than or equal to the *minSup* and their confidences greater than or equal to the *minConf*. After that, the Apriori algorithm was discussed [2]. Apriori was based on the downward-closure property to improve the time for mining frequent itemsets, and therefore, it improved the whole process of mining association rules.

2.3. **Mining (minimal) non-redundant association rules.** Mining minimal non-redundant association rules (MNARs) was proposed by Pasquier et al. in 1999 [11,12]. To better illustrate their approach, a minimal generator of a FCI X is first defined as follows: Let X be an FCI. An itemset $X' \neq \emptyset$ is called an generator of X if and only if $X' \subseteq X$ and $\sigma(X) = \sigma(X')$. Let $G(X)$ denote the set of all the generators of X. An itemset $X' \in G(X)$ is called a minimal generator of X if X' has no proper subset in $G(X)$. The set of all the minimal generators of X is denoted $mG(X)$.

An association rule $R_1 : X_1 \rightarrow Y_1$ is a minimal non-redundant association rule if and only if there does not exist an association rule $R_2 : X_2 \rightarrow Y_2$ with $\sigma(R_1) = \sigma(R_2)$, $conf(R_1) = conf(R_2)$, $X_2 \subseteq X_1$, and $Y_1 \subseteq Y_2$. Based on this definition, their approach first mines out all FCIs by computing the closure of each minimal generator, which is the largest itemset with the same support as the minimal generator. It then finds MNARs from the FCIs. There are two kinds of MNARs obtained:

i) Exact rules (their confidence = 100%): the rules have the form $X' \rightarrow X$, where X is an FCI, and $X' \in mG(X)$.
ii) Approximate rules (their confidence < 100%): the rules have the form $X' \rightarrow Y$, where X and Y are FCIs, and $X' \in mG(X)$, $X \subset Y$.

In 2000, Zaki proposed a method to mine non-redundant association rules (NARs) based on FCIs and their mG [20,21]. There are two kinds of candidate NARs:

i) NARs with their confidence = 100%: the following rules are generated from this kind of NARs.
   - Self-rules: the rules have the form $X' \rightarrow X$, where $X' \in mG(X)$ and X is an FCI. This kind of rules is the same as the exact rules of MNARs;
   - Down-rules: the rules have the form $Y' \rightarrow X$', where $Y' \in mG(Y)$, $X' \in mG(X)$, X and Y are FCIs, and $X \subset Y$.
ii) NARs with their confidence < 100%: the rules have the form $X' \rightarrow Y'$, where $X' \in mG(X)$, $Y' \in mG(Y)$, X and Y are FCIs, and $X \subset Y$.

3. **Characteristics of Most Generalization Association Rules.** In this section, we define the concept of most generalization association rules. Additionally, some of their characteristics are also derived.

Definition 3.1. – **General rule**. *Let R_1: $X_1 \rightarrow Y_1$ and R_2: $X_2 \rightarrow Y_2$. Rule R_1 is said to be more general than R_2, denoted as $R_1 \propto R_2$, if and only if $X_1 \subseteq X_2$ and $Y_2 \subseteq Y_1$.*

Definition 3.2. – **Rule has higher precedence**. *Let $R = \{R_1, R_2, \ldots, R_n\}$ be the set of rules that satisfy minSup and minConf; rule R_i is said to have a higher precedence than rule R_j, denoted as $R_i \succ R_j$, if $R_i \propto R_j$ $(i \neq j)$ and:*

i) The confidence of R_i is greater than that of R_j, or

ii) Their confidences are the same, but the support for R_i is greater than the support for R_j.

Definition 3.3. – **Most generalization association rules**. *Let $R = \{R_1, R_2, \ldots, R_n\}$ be the set of traditional association rules that satisfy minSup and minConf, and let $R_{MG} = R¥\{R_j | \exists R_i \in R : R_i \succ R_j\}$. R_{MG} is called the set of most generalization association rules (MGARs) of R.*

From Definition 3.3, we only generate association rules that have high precedence; i.e., we will prune rule R_j if in R there is the rule R_i such that $R_i \succ R_j$. In fact, we do not prune rules after generating all rules. We base on the following theorems to directly prune rules.

Theorem 3.1. *Let $R_1 : X_1 \to Y_1$, $R_2 : X_2 \to Y_2$, and $R_3 : X_3 \to Y_3$ are three rules belonging to R. If $R_1 \succ R_2$ and $R_2 \succ R_3$, then $R_1 \succ R_3$.*

Proof: The following two statements must be proven:

i) $R_1 \propto R_3$: Because $R_1 \succ R_2$ and $R_2 \succ R_3$, it is implied that $X_1 \subseteq X_2 \subseteq X_3$ and $Y_1 \supseteq Y_2 \supseteq Y_3$. Thus, $R_1 \propto R_3$.

ii) $\text{conf}(R_1) > \text{conf}(R_3)$ or $(\text{conf}(R_1) = \text{conf}(R_3)$ and $\sigma(R_1) \geq \sigma(R_3))$: $R_1 \succ R_2$ implies $\text{conf}(R_1) > \text{conf}(R_2)$ or $(\text{conf}(R_1) = \text{conf}(R_2)$ and $\sigma(R_1) \geq \sigma(R_2))$. There are three cases, as shown below:

a) $\text{conf}(R_1) > \text{conf}(R_2)$: Because $R_2 \succ R_3$, $\text{conf}(R_2) \geq \text{conf}(R_3)$, and $\text{conf}(R_1) > \text{conf}(R_2) \geq \text{conf}(R_3)$. This implies that $\text{conf}(R_1) > \text{conf}(R_3)$.

b) $\text{conf}(R_1) = \text{conf}(R_2)$, $\sigma(R_1) \geq \sigma(R_2)$, and $\text{conf}(R_2) > \text{conf}(R_3)$. It can be easily implied that $\text{conf}(R_1) > \text{conf}(R_3)$.

c) If $\text{conf}(R_1) = \text{conf}(R_2)$, $\sigma(R_1) \geq \sigma(R_2)$, $\text{conf}(R_2) = \text{conf}(R_3)$ and $\sigma(R_2) \geq \sigma(R_3)$. In this case, $\text{conf}(R_1) = \text{conf}(R_3)$ and $\sigma(R_1) \geq \sigma(R_3)$.

From i) and ii), the theorem is proven.

Theorem 3.2. *Let X, Y $(X \neq Y)$ be two FCIs, and let $mG(X)$ be a set of minimal generators of X. If $X \not\subset Y$, then $\forall Z \in mG(X)$, $Z \not\subset Y$.*

Proof: The proof is done by contradiction. Assume that $\exists Z \in mG(X)$, and $Z \subset Y$. Since $Z \in mG(X)$, then $t(Z) = t(X)$, where $t(Z)$ is a mapping from an itemset Z to a set of transactions in the database which contain Z. Then, $t(Y) = t(Z \cup Y) = t(Z) \cap t(Y) \subseteq t(Z) = t(X)$. Thus, $t(Y) \subseteq t(X)$. This implies that $t(X \cup Y) = t(X) \cap t(Y) = t(Y)$. Therefore, Y is not closed, contradicting the hypothesis. Thus, $\forall Z \in mG(X)$, $Z \not\subset Y$.

From Theorem 3.2, if $X \not\subset Y$, and $Y \not\subset X$, it is not necessary to generate rules containing X and Y. If X and Y are FCIs, and $X \subseteq Y$, the following three kinds of rules need to be considered:

i) $X' \to Y$, where $X' \in mG(X)$;

ii) $X'' \to Y$, where X'' is a generator of X and $X'' \notin mG(X)$;

iii) $X'' \to Y''$, where X'' is a generator of X, $X'' \notin mG(X)$, and Y'' is a generator of Y.

The following three theorems prove that only the first kind of rule needs to be generated, and the other two kinds of rules are redundant.

Theorem 3.3. *Let X and Y be two FCIs $(X \subseteq Y)$. If $X' \in mG(X)$, and $X' \to Y$ is not an MGAR, then all the rules with the form $X'' \to Y''$ are not MGARs, where $X'' \in$ generators of X, and $Y'' \in$ generators of Y.*

Proof: If R: $X' \to Y$ is not an MGAR, then another rule R_1: $X_1 \to Y_1$ in R_{MG} must exist such that $R_1 \succ R$. Since $X'' \in$ generators of X, and $X' \in mG(X)$, we have

$X' \subseteq X''$. $Y'' \in$ generators of Y, such that $Y'' \subseteq Y$. Since $R_1 \succ R$, thus $X_1 \subseteq X'$. This implies that $X_1 \subseteq X''$. Similarly, it can be proven that $Y'' \subseteq Y_1$. Because $\sigma(X') = \sigma(X'')$ and $\sigma(Y) = \sigma(Y'')$, $\sigma(X' \to Y) = \sigma(X'' \to Y'')$, and $\text{conf}(X' \to Y) = \text{conf}(X'' \to Y'')$. This implies that $R_1 \succ X' \to Y \succ X'' \to Y''$. The rule $X'' \to Y$" cannot be an MGAR. This completes the proof.

Theorem 3.4. *Let X be an FCI, $X' \in mG(X)$, and $X' \neq X$. The rules with the form $X' \to X$ are the only candidates for MGARs with a confidence equal to 100%.*

Proof: Because $X' \in mG(X)$, $\sigma(X') = \sigma(X)$. Thus, the confidence of the rule $X' \to X$ is 100%. To prove the theorem, the following three statements need to be proven:

i) $\forall X'' \in$ generators of X, and $X'' \neq X$, any rule in the form $X' \to X''$ is not an MGAR. This can be proven as follows: Because $X'' \subset X$, the rule $X' \to X$ is more general than ($\propto$) the rule $X' \to X''$. This implies that $X' \to X''$ is not an MGAR according to Definition 3.3 and Theorem 3.3.

ii) $\forall X'' \in$ generators of X and $X'' \notin mG(X)$, any rule with a form of $X'' \to X$ is not an MGAR. This is proven as follows: Because $X'' \in$ generators of X, and $X'' \notin mG(X)$, there always exists $X' \in mG(X)$ such that $X' \subset X''$. This implies that the rule $X' \to X$ is more general than the rule $X'' \to X$. Thus, the rule $X'' \to X$ is not an MGAR from Theorem 3.3.

iii) Let X and Y be two FCIs, and $X \subset Y$. If X'' is a generator of X, and Y'' is a generator of Y, then there is no rule $X'' \to Y''$ with a confidence equal to 100%. This is proven as follows: Because X and Y are two FCIs and $X \subset Y$, $\sigma(X) > \sigma(Y)$. Since $X'' \in$ generators of X, thus $\sigma(X'') = \sigma(X)$. Similarly, $\sigma(Y'') = \sigma(Y)$. This implies that $\sigma(X'') > \sigma(Y'')$. Thus, the confidence of the rule $X'' \to Y''$ is $\sigma(Y'')/\sigma(X'')$, which is less than 100%.

Thus, the three kinds of rules mentioned above cannot be MGARs. The remaining rules, which have the form $X' \to X$, are possibly MGARs with a confidence equal to 100%.

Theorem 3.5. *Let X, Y be two FCIs with $X \subset Y$, and $X' \in mG(X)$. The rules with the form $X' \to Y$ are the only candidates for MGARs with a confidence $< 100\%$.*

Proof: To prove the theorem, the following two statements need to be proven:

i) $\text{conf}(X' \to Y) < 100\%$. This can be proved using the proof of Theorem 3.4.iii).

ii) $\forall X'' \in$ generators of X, and $\forall Y'' \in$ generators of $Y (Y'' \neq Y)$, so any rule in the form $X'' \to Y$" is not an MGAR. This is proven as follows: Because $X'' \in$ generators of X, and $Y'' \in$ generators of Y, there always exists $X' \in mG(X)$ such that $X' \subseteq X''$. This implies that the rule $X' \to Y$ is more general than the rule $X'' \to Y''$ (because $Y'' \subset Y$). Thus, the rule $X'' \to Y''$ is not an MGAR.

Based on the above theorems, the candidates for MGARs may be generated from $mG(X)$ to X, and from $mG(X)$ to Y, where X, $Y \in$ FCIs, and $X \subset Y$. All other rules generated without this form are redundant, and they need not be considered.

4. A Proposed Algorithm for Mining Most Generalization Association Rules.

In this section, we propose an algorithm for generating MGARs from the database based on the above theorems. Figure 1 shows the algorithm for generating MGARs from FCIs. In this algorithm, the main procedure, **GENERATE_MGARs,** generates all the MGARs from the given FCIs. It first finds MGARs with a confidence equal to 100% according to Theorem 3.4, and then derives MGARs with a confidence $< 100\%$ according to Theorem 3.5. Both actions invoke the procedure, **FIND_RULES** ($\boldsymbol{C}_i$, $\boldsymbol{C}_j$, ***RHS***),

which discovers MGARs from $mG(C_i)$ to C_j, where C_i and C_j are frequent closed itemsets, and $C_i \subseteq C_j$. A variable RHS is used to eliminate redundant items on the right-hand side. For example, assume that the two rules $A \to BC$, and $A \to CD$ exist. Then, the rule $A \to CD$ is modified to $A \to D$ because C is predicted by the first rule. Thus, C does not need to appear in the second rule.

In the algorithm below (Figure 1), the FCIs are first sorted in ascending order according to their length (Line 1). Then, for each FCI C_i, the algorithm initially sets $RHS = \emptyset$ (Line 5), and generates MGARs with a confidence equal to 100% from $mG(C_i)$ to C_i by calling the procedure **FIND_RULES** (Line 6). It then finds the superset of C_i to generate rules with a confidence $< 100\%$ (Lines 7 and 8). The FCIs in the superset are then sorted in descending order according to supports (line 10). This generates rules from high confidence to low confidence and from $\boldsymbol{C}_i$ to all $\boldsymbol{C}_j \in$ supersets.

Given two closed itemsets $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$, where $\boldsymbol{C}_i \subseteq \boldsymbol{C}_j$, the procedure **FIND_RULES** generates MGARs from $X \in \boldsymbol{mG}(\boldsymbol{C}_i)$ to $\boldsymbol{C}_j$ (by Theorem 3.4 and Theorem 3.5). It considers a rule r with the left-hand side being X, and the right-hand side being $Z = \boldsymbol{C}_j \backslash (X \cup RHS)$. Its support is $\boldsymbol{C}_j.sup$, and its confidence is $\boldsymbol{C}_j.sup/\boldsymbol{C}_i.sup$. If $Z \neq \emptyset$, then the rule r is checked by the function **CHECK_REDUNDANT**(r) to determine whether it is an MGAR. It is checked by determining whether the right-hand side of r' in R_{MG} contains at least an item of the right-hand side in r. If r is an MGAR, then it is added into R_{MG}, and Z is added into RHS (Lines 8 and 14 to 20). The function **CHECK_REDUNDANT** uses a hash table to increase efficiency.

Input: A set of FCIs and the parameter *minConf*.
Output: The set of most generalization rules R_{MG} that satisfy *minConf*.
Algorithm:

```
GENERATE_MGARs( )
 1.  Sort (FCIs)   // In ascending order according to lengths
 2.  RMG = ∅       // Initialize RMG as empty
 3.  for each Ci ∈ FCIs do
 4.      superset = ∅
 5.      RHS = ∅   // Initialize the right-hand side of the MGAR rules generated by Ci.
 6.      FIND_RULES(Ci, Ci, RHS) // Generate MGARs with conf = 100% according to
     Theorem 3.4
 7.      for each Cj ∈ FCIs with j > i do
 8.          if Ci ⊂ Cj and Cj.sup/Ci.sup ≥ minConf, then // by Theorem 3.2
 9.              superset = superset ∪ Cj   // Find Ci's superset in FCIs.
10.      Sort(superset)    // in descending order according to supports
11.      ENUMERATE_MGARs(Ci, superset, RHS) // Generate MGARs with conf <
                                            //100% according to Theorem 3.5
ENUMERATE_MGARs( C, S, RHS)
12.  for each Cj ∈ S do
13.      FIND_RULES( C, Cj, RHS)
FIND_RULES(Ci, Cj, RHS)    //Find MGARs from mG(Ci) to Cj, Ci ⊆ Cj
14.  for each X ∈ mG(Ci) do
15.      Z = Cj \ (X ∪ RHS) // To eliminate redundant items on the RHS of rules
16.      if  Z ≠ ∅ then     // The right hand-side of rule is not empty
17.          r = X → Z with sup = Cj.sup and conf = (Cj.sup/Ci.sup)   // generate a rule
18.          if CHECK_REDUNDANT(r) = False then // r is an MGAR.
19.              RMG = RMG ∪ r      // Add r into RMG
20.              RHS = RHS ∪ Z
```

FIGURE 1. Algorithm for generating MGARs from FCIs

The complexity of the proposed algorithm can be analyzed below as $O(n^2)$, where n is the number of frequent closed itemsets. Considering the algorithm in Figure 1, the complexity of the proposed algorithm depends on Lines 3, 7 and 14. In Line 3, the algorithm must traverse each C_i one time. Therefore, its cost is thus n. In Line 7, the algorithm must traverse all C_j following C_i, and the complexity is thus $n - i$. Line 14 traverses all minimal generators of an FCI to generate rules, and the complexity is thus m, where m is the number of minimal generators. Often, m is very small when compared with |FCIs|. Therefore, the complexity of the proposed algorithm is:

$$\sum_{i=1}^{n}(n-i) = \sum_{i=0}^{n-1} i = \frac{(n-1)n}{2}.$$

This implies that the complexity of the proposed algorithm is $O(n^2)$.

In fact, the complexity of the proposed algorithm also depends on the function **CHECK_REDUNDANT** in Line 18. However, because of using a hash table, the complexity of **CHECK_REDUNDANT** is often O(1).

5. **An Example.** An example is given in this section to illustrate the above algorithm. Consider the database in Table 1. Seven FCIs and their minimum generators are generated by MG-CHARM [17] with *minSup* = 50%, as shown in Table 3.

Table 4 shows the execution process of the algorithm for generating MGARs when the threshold of the minimum confidence (*minConf*) is set at 60%. The seven FCIs are processed one by one.

Table 3. Seven FCIs and their *mG* generated from the database in Table 1

FCIs	σ (count)	*mG*
C	6	C
CD	4	D
CT	4	T
CW	5	W
ACW	4	A
CDW	3	DW
ACTW	3	AT, TW

Table 4. Process of generating MGARs from the FCIs in Table 3

FCIs	σ	***mG***	superset	*RHS*	MGARs satisfy *minConf* = 60%
C	6	C	CW, CD, CT, ACW	WDTA	$C \xrightarrow{5,5/6} W$, $C \xrightarrow{4,4/6} D$ $C \xrightarrow{4,4/6} T$, $C \xrightarrow{4,4/6} A$
CD	4	D	CDW	CW	$D \xrightarrow{4,1} C$, $D \xrightarrow{3,3/4} W$
CT	4	T	ACTW	CAW	$T \xrightarrow{4,1} C$, $T \xrightarrow{3,3/4} AW$
CW	5	W	ACW, CDW, ACTW	CADT	$W \xrightarrow{5,1} C$, $W \xrightarrow{4,4/5} A$ $W \xrightarrow{3,3/5} D$, $W \xrightarrow{3,3/5} T$
ACW	4	A	ACTW	CWT	$A \xrightarrow{4,1} CW$, $A \xrightarrow{3,3/4} T$
CDW	3	DW			
ACTW	3	AT, TW			$TW \xrightarrow{3,1} A$

In Table 4, the FCI C is first processed. The variable RHS is initially set to $\emptyset$. Since $mG(C) = \{C\}$, no MGAR with a confidence equal to 100% is generated, according to Theorem 3.4. The superset of $\{C\}$ in FCIs includes $\{CW, CD, CT, ACW\}$ after being sorted in descending order according to supports. The superset $\{CW\}$ is then processed. The MGAR rule $C \xrightarrow{5,5/6} W$ is then derived, with $RHS = W$. The other three MGARs generated from the supersets of $\{C\}$ are:

For CD: $C \xrightarrow{4,4/6} D$ with $RHS = RHS \cup \{D\} = WD$

For CT: $C \xrightarrow{4,4/6} T$ with $RHS = RHS \cup \{T\} = WDT$

For ACW: $C \xrightarrow{4,4/6} A$ with $RHS = RHS \cup \{A\} = WDTA$

Note that when we process the rule generation from C to ACW, the rule $C \to AW$ will be generated. Because $C \to W$ has been generated, only the rule $C \to A$ needs to be checked. The same process is then performed on the other FCIs. After the first six FCIs are processed, the derived MGARs set $R_{MG} = \Big\{C \xrightarrow{5,5/6} W, C \xrightarrow{4,4/6} D, C \xrightarrow{4,4/6} T, C \xrightarrow{4,4/6} A, D \xrightarrow{4,1} C, D \xrightarrow{3,3/4} W, T \xrightarrow{4,1} C, T \xrightarrow{3,3/4} AW, W \xrightarrow{5,1} C, W \xrightarrow{4,4/5} A, W \xrightarrow{3,3/5} T, A \xrightarrow{4,1} CW,$ and $A \xrightarrow{3,3/4} T\Big\}$. Finally, the closed itemset $ACTW$ is processed. Two candidate MGAR rules are generated from the itemset by **FIND_RULES***(ACTW, ACTW, RHS)*. The results are $AT \xrightarrow{3,1} CW$ and $TW \xrightarrow{3,1} A$. Because the rule $A \xrightarrow{4,1} CW$ is in R_{MG}, the rule $AT \xrightarrow{3,1} CW$ is thus not the most generalization association rule. The rule $TW \xrightarrow{3,1} A$ is an MGAR.

6. **Experimental Results.** Experiments were conducted to show the performance of the proposed algorithm. They were implemented on a Centrino Core 2 Duo (2×2.53 GHz) PC with 4 GB of RAM running Windows 7. The algorithms were coded in C# 2008. Seven databases from [7] were used for the experiments; their features are shown in Table 5.

Table 5. Features of the adopted databases

Database	# of Trans	# of Items
Chess	3196	76
Mushroom	8124	120
Pumsb*	49046	7117
Pumsb	49046	7117
Connect	67557	130
Retail	88162	16469
Accidents	340183	468

Experiments were first made to show the compact effect of the most generalization association rules (MGARs). The numbers of MGARs obtained from the above databases are shown in Table 6 and are compared with those obtained by traditional association rules [1], non-redundant association rules (NARs) [20] and minimal non-redundant association rules (MNARs) [2]. Table 6 shows that the number of MGARs is always smaller than those of NARs, MNARs, and traditional association rules. For example, consider the database Chess with $minSup = 80\%$ and $minConf = 0\%$. The number of MGARs is 951, whereas the number of NARs is 27711, the percentage of #MGARs per #NARs is 3.4%; the number of MNARs is 316057, and the percentage is 0.3%; the number of traditional association rules is 552564, and the percentage is 0.17%.

Experiments were then made to compare the numbers of MGARs and MNARs for various *minSup* values with the same *minConf* being 0%. The results for the seven databases are shown in Figures 2 to 8.

TABLE 6. Comparison of numbers of MGARs, NARs [20], MNARs [2] and traditional ARs [1] with $minConf = 0\%$

Database	*minSup* (%)	# of rules				Ratio (1)/(2) %	Ratio (1)/(3) %	Ratio (1)/(4) %
		MGARs (1)	NARs (2)	MNARs (3)	Traditional (4)			
Chess	80	951	27711	316057	552564	3.43	0.3	0.17
Connect	97	101	1116	4600	8092	9.05	2.2	1.25
Mushroom	40	283	475	1168	7020	59.58	24.23	4.03
Pumsb*	60	126	192	611	12840	65.63	20.62	0.98
Pumsb	95	101	267	690	1170	37.83	14.64	8.63

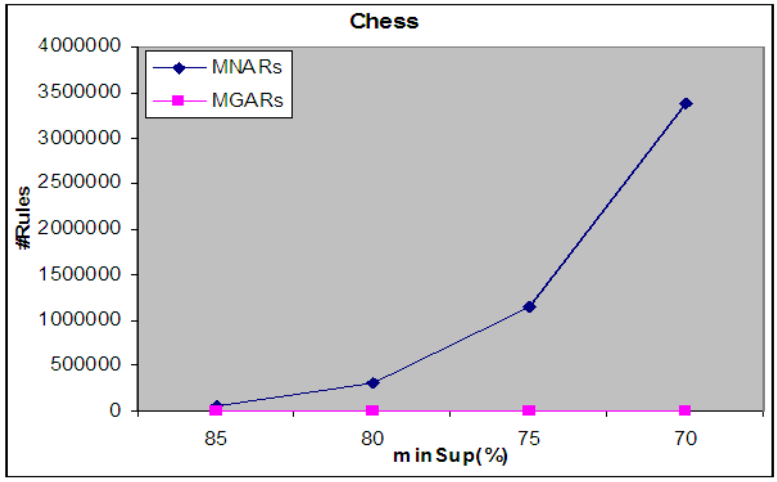

FIGURE 2. Numbers of MGARs and MNARs in Chess for various *minSup* values

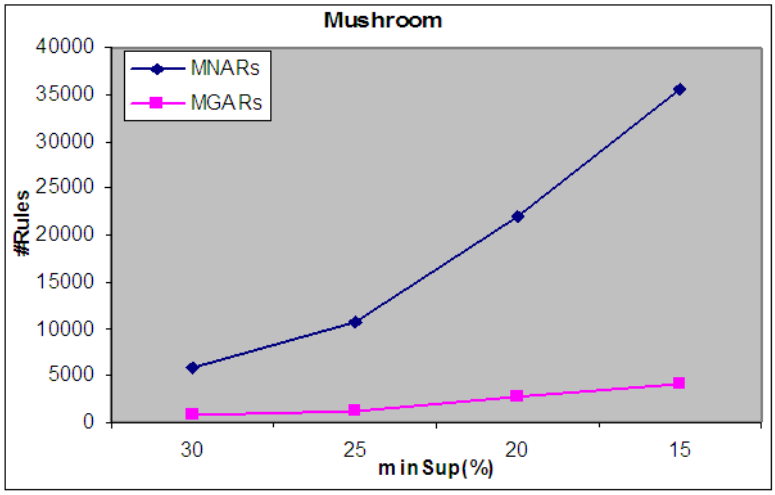

FIGURE 3. Numbers of MGARs and MNARs in Mushroom for various *minSup* values

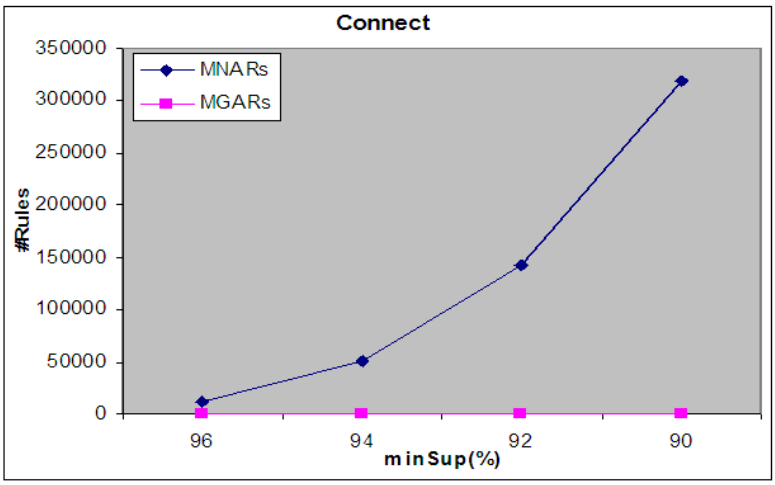

FIGURE 4. Numbers of MGARs and MNARs in Connect for various *minSup* values

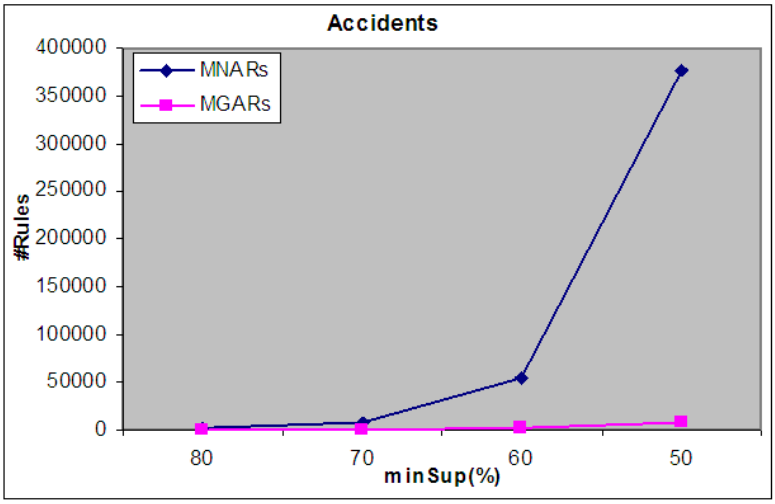

FIGURE 5. Numbers of MGARs and MNARs in Accidents for various *minSup* values

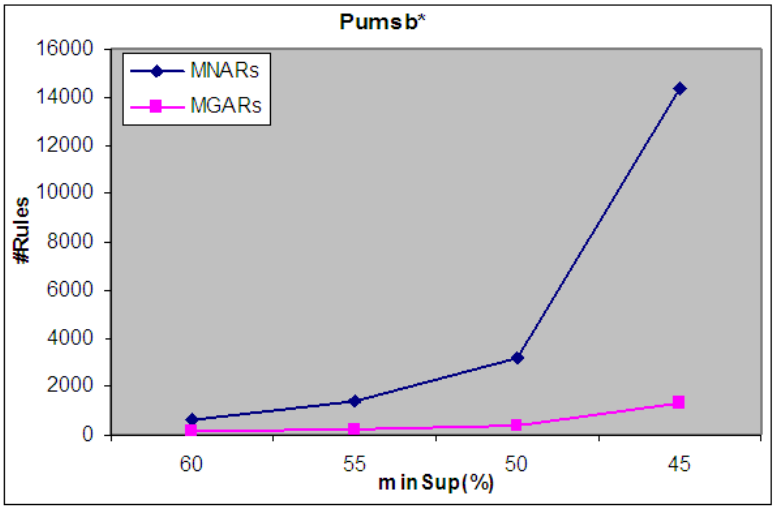

FIGURE 6. Numbers of MGARs and MNARs in Pumsb* for various *minSup* values

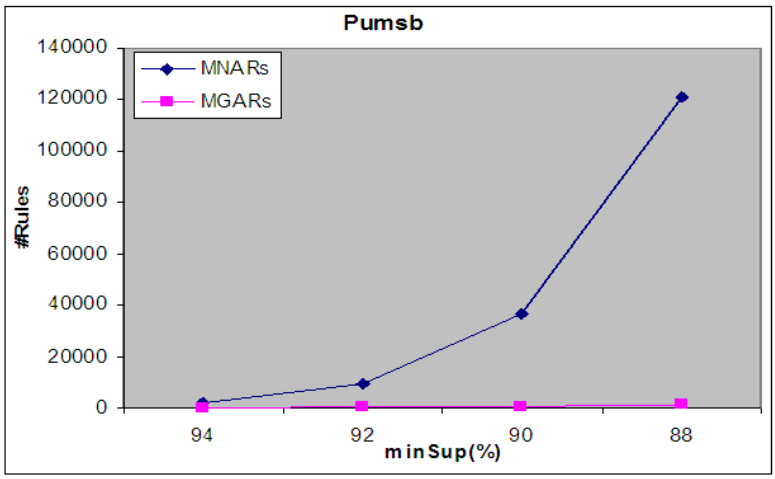

FIGURE 7. Numbers of MGARs and MNARs in Pumsb for various *minSup* values

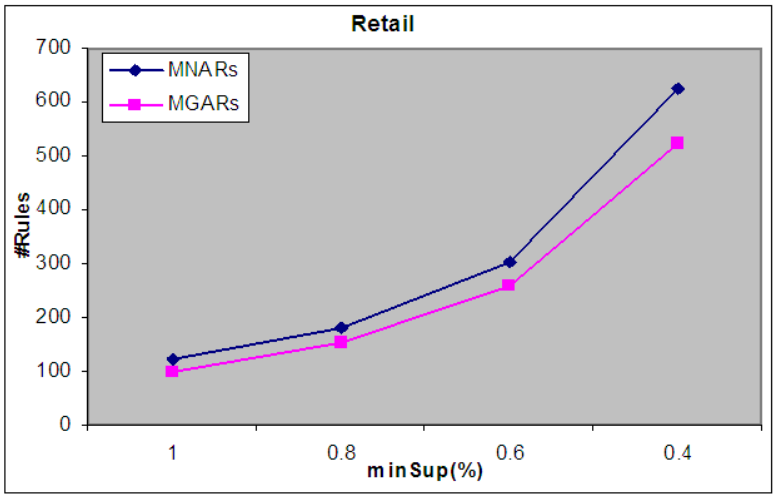

FIGURE 8. Numbers of MGARs and MNARs in Retail for various *minSup* values

The results from the above figures show that the number of MGARs is always smaller than that of MNARs for various *minSup* values. For example, consider the Chess database with a *minSup* = 70%. The number of MGARs is 3,175, and the number of MNARs is 3,373,625. The size ratio is only 0.094%.

Experiments were then made to show the execution time of the algorithm. The results for the seven databases for various *minSup* values are shown in Figures 9 to 15.

The mining time of MGARs in Figures 9 to 15 depends on the size ratio of the number of generated rules between MGARs and MNARs. When the scale is large, the mining time is large. For example, considering the Chess database (*minSup* = 70%) in which the size ratio is 0.094%, the mining time is about 259(s). This is because the proposed algorithm must check a lot of redundant rules along with the generated rules. However, when we consider the Mushroom database (*minSup* = 15%) where the size ratio is 11.6%, the mining time is about 3(s).

7. **Conclusions and Future Work.** In this paper, we proposed a method for mining most generalization association rules from transaction databases, which generates more compact results than three previous rule types, such as traditional association rules, NARs

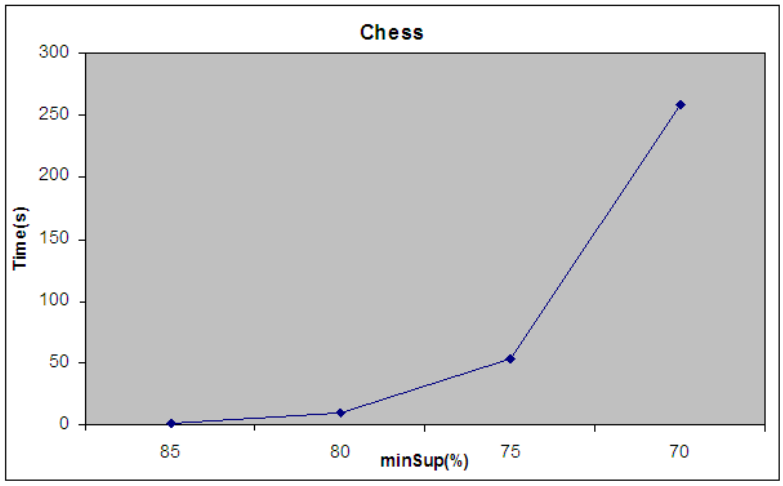

FIGURE 9. Execution time of the proposed algorithm in Chess for various *minSup* values

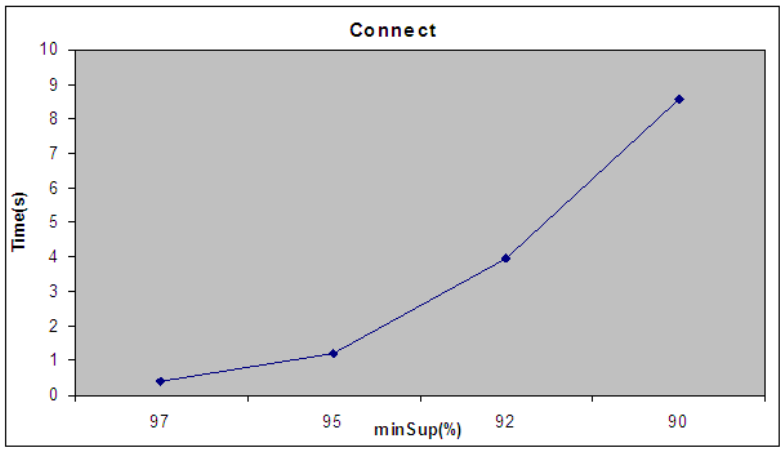

FIGURE 10. Execution time of the proposed algorithm in Connect for various *minSup* values

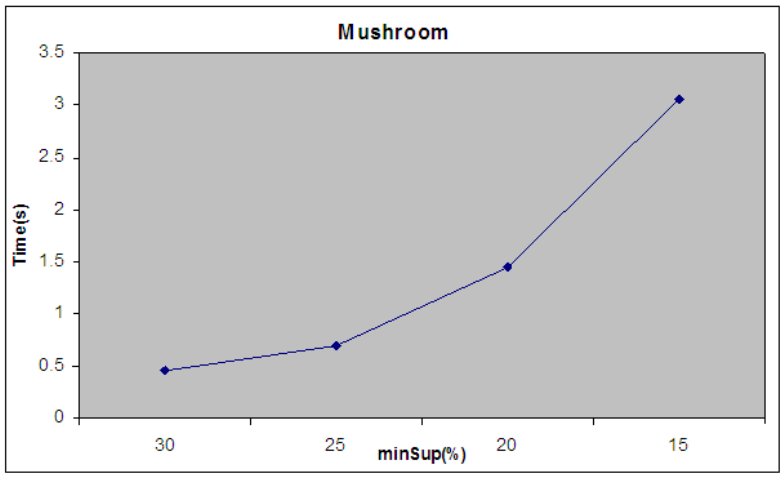

FIGURE 11. Execution time of the proposed algorithm in Mushroom for various *minSup* values

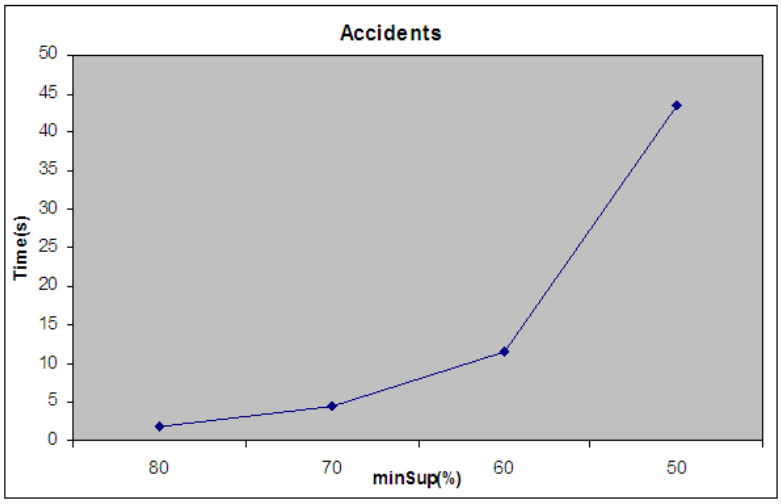

FIGURE 12. Execution time of the proposed algorithm in Accidents for various *minSup* values

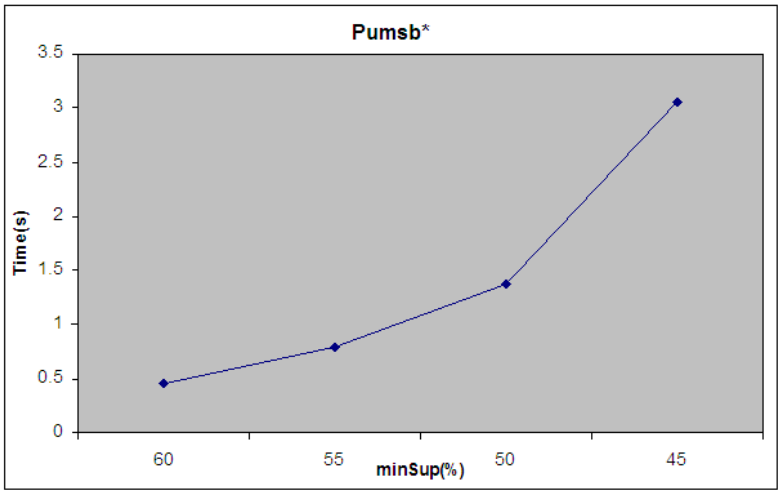

FIGURE 13. Execution time of the proposed algorithm in Pumsb* for various *minSup* values

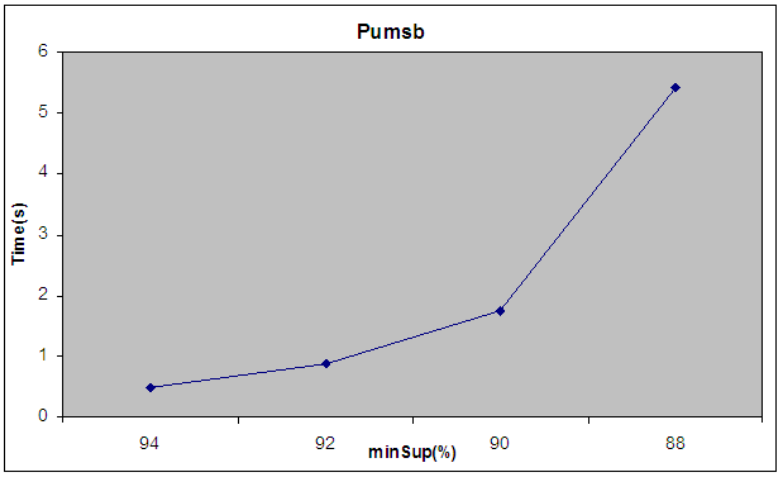

FIGURE 14. Execution time of the proposed algorithm in Pumsb for various *minSup* values

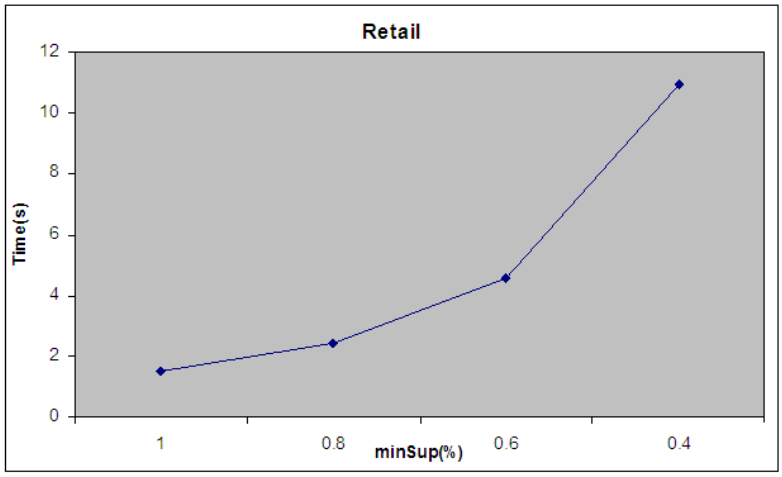

FIGURE 15. Execution time of the proposed algorithm in Retail for various *minSup* values

and MNARs, based on support and confidence [1,2,11,12,20,21]. An efficient algorithm for mining MGARs was also proposed. It was based on FCIs for mining MGARs. In addition, some theorems were derived to prune redundant rules quickly. There were a lot of rules that could be generated. However, based on these theorems, we only generated the candidate rules from $mG(X)$ to Y, with X, Y being frequent closed itemsets, and $X \subseteq Y$. Hash tables were used for checking some other redundant rules quickly as well.

In the future, we will attempt to use other data structures, such as lattices, to mine MGARs faster. Besides, mining association rules in incremental databases has been developed in recent years [4-6]. Such approaches can save a lot of maintenance time when transactions are inserted or deleted. Therefore, we will also study how to maintain MGARs when the database is changed.

Acknowledgment. This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2010.02. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, *Proc. of VLDB*, Santiago de Chile, Chile, pp.487-499, 1994.

[2] Y. Bastide, N. Pasquier, R. Taouil, G. Stumme and L. Lakhal, Mining minimal non-redundant association rules using frequent closed itemsets, *Proc. of the 1st International Conference on Computational Logic*, London, UK, pp.972-986, 2000.

[3] G. Grahne and J. Zhu, Fast algorithms for frequent itemset mining using FP-trees, *IEEE Transactions on Knowledge and Data Enineering*, vol.17, no.10, pp.1347-1362, 2005.

[4] T.-P. Hong, C.-W. Lin and Y.-L. Wu, An efficient FUFP-tree maintenance algorithm for record modification, *International Journal of Innovative Computing, Information and Control*, vol.4, no.11, pp.2875-2887, 2008.

[5] T. P. Hong, C. W. Lin and Y. L. Wu, Maintenance of fast updated frequent pattern trees for record deletion, *Computational Statistics and Data Analysis*, vol.53, no.7, pp.2485-2499, 2009.

[6] T. P. Hong and C. J. Wang, An efficient and effective association-rule maintenance algorithm for record modification, *Expert Systems with Applications*, vol.37, no.1, pp.618-626, 2010.

[7] *http://fimi.cs.helsinki.fi/data/*, 2005.

[8] A. J. T. Lee, C. S. Wang, W. Y. Weng, J. A. Chen and H. W. Wu, An efficient algorithm for mining closed inter-transaction itemsets, *Data & Knowledge Engineering*, vol.66, no.1, pp.68-91, 2008.

[9] C. Lucchese, S. Orlando and R. Perego, Fast and memory efficient mining of frequent closed itemsets, *IEEE Transactions on Knowledge and Data Engineering*, vol.18, no.1, pp.21-36, 2006.
[10] H. D. K. Moonestinghe, S. Fodeh and P. N. Tan, Frequent closed itemsets mining using prefix graphs with an efficient flow-based pruning strategy, *Proc. of the 6th ICDM*, Hong Kong, pp.426-435, 2006.
[11] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, Discovering frequent closed itemsets for association rules, *Proc. of the 5th International Conference on Database Theory*, Jerusalem, Israel, pp.398-416, 1999.
[12] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, Efficient mining of association rules using closed itemset lattices, *Information Systems*, vol.24, no.1, pp.25-46, 1999.
[13] J. Pei, J. Han and R. Mao, CLOSET: An efficient algorithm for mining frequent closed itemsets, *Proc. of the 5th ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Dallas, TN, USA, pp.11-20, 2000.
[14] N. G. Singh, S. R. Singh and A. K. Mahanta, CloseMiner: Discovering frequent closed itemsets using frequent closed tidsets, *Proc. of the 5th ICDM*, Washington DC, USA, pp.633-636, 2005.
[15] T. Uno, T. Asai, Y. Uchida and H. Arimura, An efficient algorithm for enumerating closed patterns in transaction databases, *Proc. of the 7th International Conference on Discovery Science*, Padova, Italy, pp.16-31, 2004.
[16] J. Wang, J. Han and J. Pei, CLOSET+: Searching for the best strategies for mining frequent closed itemsets, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington DC, USA, pp.236-245, 2003.
[17] B. Vo and B. Le, Fast algorithm for mining minimal generators of frequent closed itemsets and their applications, *Proc. of the 39th International Conference on Computers & Industrial Engineering*, Troyes, France, pp.1407-1411, 2009.
[18] B. Vo and B. Le, Mining the most generalization association rules, *Proc. of ACIIDS*, Hue, Viet Nam, pp.207-216, 2010.
[19] S. B. Yahia, T. Hamrouni and E. M. Nguifo, Frequent closed itemset based algorithms: A thorough structural and analytical survey, *ACM SIGKDD Explorations Newsletter*, vol.8, no.1, pp.93-104, 2006.
[20] M. J. Zaki, Generating non-redundant association rules, *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, USA, pp.34-43, 2000.
[21] M. J. Zaki, Mining non-redundant association rules, *Data Mining and Knowledge Discovery*, vol.9, no.3, pp.223-248, 2004.
[22] M. J. Zaki and C. J. Hsiao, Efficient algorithms for mining closed itemsets and their lattice structure, *IEEE Transactions on Knowledge and Data Engineering*, vol.17, no.4, pp.462-478, 2005.