Dynamic Mining of Multi-supported Association Rules with Classification Ontology

MING-CHENG TSENG [1], *WEN-YANG LIN* [2,*], *RONG JENG* [3]
[1, 3] *Institute of Information Engineering,* [2] *Dept. of Comp. Sci. & Info. Eng.*
[1, 3] *I-Shou University,* [2] *National University of Kaohsiung*
TAIWAN, R.O.C.
[1] *clark.tseng@msa.hinet.net,* [2] *wylin@nuk.edu.tw,* [3] *rjeng@isu.edu.tw*
([*]corresponding author)

Abstract

One of the predominant techniques used in the area of data mining is association rule mining. In real world, data mining analysts usually are confronted with a dynamic environment; the database would be changed over time, and the analysts may need to set different support constraints to discover real informative rules. Efficiently updating the discovered association rules thus becomes a crucial issue. In this paper, we consider the problem of dynamic mining of association rules with classification ontology and with non-uniform multiple minimum supports constraint. We investigate how to efficiently update the discovered association rules when there is transaction update to the database and the analyst has refined the support constraint. A novel algorithm called DMA_CO is proposed. Experimental results show that our algorithm is 14% to 80% faster than applying generalized associations mining algorithms to the whole updated database.

***Keywords*:** Association rules, classification ontology, data mining, database update, support constraint refinement.

1 Introduction

Data mining is to discover implicit, unknown, and useful patterns from databases or data warehouses and enables managers to identify necessary information to support in the decision-making. One of the predominant techniques used in the area of data mining is association rule mining. The classical association rule is based on a list of transactions gathered in a database. An association rule can be expressed as

$$A \Rightarrow B \ (sup = 20\%, conf = 90\%),$$

where A and B are sets of items. This rule reveals that in the transaction database, 20% (support) of customers purchase both A and B together, and 90% (confidence) of customers who purchase A also purchase B. The problem of mining association rules is to discover all association rules that satisfy support and confidence constraints.

In many applications, there exists classification ontology over the items implicitly, so it is more useful to find association rules at different levels of the classification ontology than only at the primitive concept level [6][12]. For example, consider the classification ontology of items in Figure 1, where "MC" stands for "Mobile Computer", "ASUS P" a kind of PDA and "ScanMaker" a kind of scanner. It is likely that the association rule

Acer TM ⇒ Epson EPL (sup = 20%, $conf$ = 80%)

does not hold when the minimum support is set to 25%, but the following association rule is true

Notebook ⇒ Epson EPL (sup = 30%, $conf$ = 75%).

This kind of association rule with classification ontology is also called generalized association rule [12] or multi-level association rule [6].

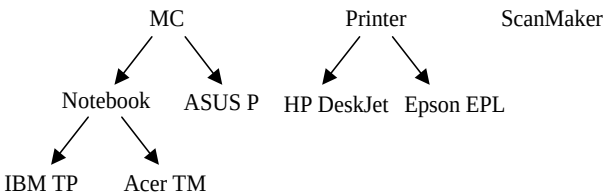


Figure 1. Example of item classification ontology.

In our previous work, we have investigated the problem of mining generalized association rules across different levels of classification ontology with non-uniform multiple minimum supports [14]. We proposed two efficient algorithms, MMS_Cumulate and MMS_Stratify, which not only can discover association rules that span different levels but also have high potential to produce rare but informative item rules. In real world, however, data mining practitioners usually are confronted with a dynamic environment. New transactions are continually added into the database as time passes, and because the work of discovering interesting association rules is an iterative process, the analysts need to repeatedly adjust the constraint of minimum support and/or minimum confidence to discover real informative rules. Under these circumstances, how to dynamically discover generalized association rules efficiently is a crucial issue.

One approach to dealing with this issue is to adopt the generalized association mining algorithms to the whole updated database. However, this way is not efficient since generating frequent itemsets is time-consuming, and discovered frequent itemsets are not re-used.

A better approach is to perform an initial association mining algorithm to generate and store the frequent itemsets, and when update to the original database and/or refinement to the support constraint do occur, employ an updating method to re-build the discovered rules. The challenge falls into deploying an efficient updating

algorithm to facilitate the whole mining process. This problem is nontrivial because update may invalidate some of the discovered association rules, and turn previous weak rules into strong ones.

In this paper, we examine this issue and propose an algorithm called DMA_CO (Dynamic Mining of multi-supported Association rules with Classification Ontology), which reuses discovered frequent itemsets and significantly reduces the number of candidate sets and database rescanning, and so can update the association rules efficiently. Experimental results show that our algorithm is faster than applying classifical generalized association mining algorithms to the whole updated database.

As founded in [1], the work of association rules mining can be decomposed into two steps: first, find all itemsets that are no less than the minimum support; second, from the frequent itemsets generate all association rules having confidence larger than or equal to the minimum confidence. Since the second step is less expensive, we focus only on the first step of finding all frequent itemsets.

The rest of this paper is organized as follows. We formulate the problem in Section 2. In Section 3, we propose the DMA_CO algorithm and give an example in Section 4. In Section 5, we evaluate the performance of the proposed DMA_CO algorithm. We discuss previous work in Section 6. Finally, we conclude this work in Section 7.

2 Problem Formulation

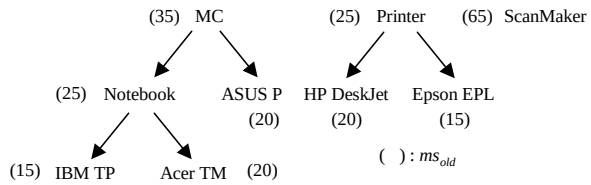
Let $I = \{i_1, i_2, \ldots, i_p\}$ be a set of items and $DB = \{t_1, t_2, \ldots, t_n\}$ be a set of transactions, where each transaction $t_i = \langle TID, A\rangle$ has a unique identifier *TID* and a set of items A ($A \subseteq I$). Assume that a set of classification ontology of items, T, is available and is denoted as a set of hierarchies (trees) on $I \cup J$, where $J = \{j_1, j_2, \ldots, j_q\}$ represents the set of generalized items derived from I. Following the concept in [9], we assume that the user can specify different minimum supports for different items in the classification ontology.

Definition 1. Let $ms(a)$ denote the minimum support of an item a in T. The minimum support of an itemset $A = \{a_1, a_2, \ldots, a_k\}$, where $a_i \in T$, equal to the lowest value of minimum support of items in A, i.e., $ms(A) = \min_{a_i \in A} ms(a_i)$.

Definition 2. An itemset A is *frequent* if its support is larger than or equal to the its minimum support constraint, i.e., $sup(A) \geq ms(A) = \min_{a_i \in A} ms(a_i)$.

Consider the situation when new transactions in *db* are added to *DB*, and the old minimum support constraint (ms_{old}) is changed into the new one (ms_{new}) with the same item classification ontology T. Let *ED* and *ed* denote the extended version of the original database *DB* and incremental database *db*, respectively, by adding the generalized items in T to each transaction. Further, let *UE* be the updated extended database containing *ED* and *ed*, i.e., $UE = ED + ed$. The problem of updating L^{ED} when new transactions in *db* are added to *DB*, and the old minimum support constraint (ms_{old}) is changed into the new one (ms_{new}) is equivalent to discovering the set of frequent itemsets in *UE* with respect to ms_{new}, denoted as L^{UE}.

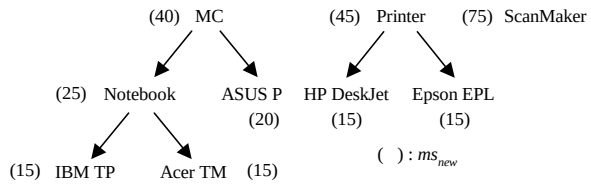
For illustration, consider the original database with item classification ontology having old item minimum supports (shown in parentheses) in Figure 2. Figure 3 shows an incremental database to be updated to the original database with the new minimum support specification under the same item classification ontology. The minimum support for item "MC" is changed from 35% to 40%, "Acer N" from 20% to 15%, "Printer" from 25% to 45%, "HP DeskJet" from 20% to 15%, and "ScanMaker" from 65% to 75%. Figure 4 shows the corresponding updated extended database *UE* after combing *ED* and *ed*.



Original Database (*DB*)

TID	Items Purchased
1	Acer TM, Epson EPL, ScanMaker
2	ASUS P, ScanMaker
3	Acer TM
4	IBM TP, ASUS P
5	Epson EPL, ScanMaker
6	ASUS P, ScanMaker
7	ScanMaker
8	Acer TM

Figure 2. Example of an original database with old minimum support specification.



Incremental Database (*db*)

TID	Items Purchased
9	HP DeskJet, ScanMaker
10	HP DeskJet, ScanMaker

Figure 3. Example of an incremental database with new minimum support specification.

3 The Proposed Approach

In this section, we first describe the basic concept behind the proposed approach and then detail the algorithm.

3.1 Algorithm Basics

The basic process of our mining multi-supported generalized association rules under incremental

Updated Extended database (*UE*)		
TID	Primitive Items	Generalized Items
1	Acer TM, Epson EPL, ScanMaker	Notebook, MC, Printer
2	ASUS P, ScanMaker	MC
3	Acer TM	Notebook, MC
4	IBM TP, ASUS P	Notebook, MC
5	Epson EPL, ScanMaker	Printer
6	ASUS P, ScanMaker	MC
7	ScanMaker	
8	Acer TM	Notebook, MC
9	HP Desk Jet, ScanMaker	Printer
10	HP Desk Jet, ScanMaker	Printer

Figure 4. The resulting updated extended database *UD* for the examples in Figures 2 and 3.

transaction and multiple support constraint refinement follows the level-wise approach used by famous Apriori-like algorithms [2].

However, the well-known apriori pruning technique based on the concept of *downward closure* [2] does not work for multiple support specification. For example, consider four items a, b, c, and d that have minimum supports specified as $ms(a) = 15\%$, $ms(b) = 20\%$, $ms(c) = 4\%$, and $ms(d) = 6\%$. Clearly, a 2-itemset $\{a, b\}$ with 10% support is discarded for $10\% < \min\{ms(a), ms(b)\}$. According to the downward closure, the 3-itemsets $\{a, b, c\}$ and $\{a, b, d\}$ will be pruned even though their supports may be larger than $ms(c)$ and $ms(d)$, respectively. To solve this problem, we have adopted the *sorted closure property* [8] described as follows. Hereafter, to distinguish from the traditional itemset, a sorted k-itemset denoted as $\langle a_1, a_2, \ldots, a_k\rangle$ is used.

Lemma 1. If a sorted k-itemset $\langle a_1, a_2, \ldots, a_k\rangle$, for $k \geq 2$ and $ms(a_1) \leq ms(a_2) \leq \ldots \leq ms(a_k)$, is frequent, then all of its sorted subsets with k–1 items are frequent, except the subset $\langle a_2, a_3, \ldots, a_k\rangle$.

Lemma 2. For $k = 2$, the procedure apriori-gen(L_1) [2] fails to generate all candidate 2-itemsets in C_2.

For example, consider a sorted candidate 2-itemset $\langle a, b\rangle$. It is easy to find if we want to generate this itemset from L_1, both items a and b should be included in L_1; that is, each one should be occurring more frequently than the corresponding minimum support $ms(a)$ and $ms(b)$. Clearly, the case $ms(a) \leq sup(b) < ms(b)$ fails to generate $\langle a, b\rangle$ in C_2 even $sup(\langle a, b\rangle) \geq ms(a)$.

For the above reason, all items within an itemset are sorted in the increasing order of their minimum supports, and a sorted itemset, called *frontier set*, is facilitated to generate the set of candidate 2-itemsets C_2 [8]. Please refer to [8] for more details.

Lemma 3. For $k \geq 3$, any k-itemset $A = \langle a_1, a_2, \ldots, a_k\rangle$ generated by procedure apriori-gen(L_{k-1}) [2] can be pruned if there exists one (k–1) subset of A, say $\langle a_{i_1}, a_{i_2}, \ldots, a_{i_{k-1}}\rangle$, such that $\langle a_{i_1}, a_{i_2}, \ldots, a_{i_{k-1}}\rangle \notin L_{k-1}$ and $a_{i_1} = a_1$ or $ms(a_{i_1}) = ms(a_{i_2})$.

For more details about how to refine the classical apriori candidate generation in adopting the sorted closure property, please refer to [8] or [14].

Now let us focus the kernel issue: how to efficiently rediscover qualified frequent itemsets when the database has been updated and the analyst has adjusted the support constraint. A straightforward method would be to run any of the algorithms for finding generalized frequent itemsets, such as our previously proposed MMS_Cumulate and MMS_Stratify [14], on the updated extended database *UE* with respect to the new support constraint ms_{new}. This simple way, however, ignores two pieces of useful information: the discovered frequent itemsets and the relationship between ms_{old} and ms_{new}. Previous work [3][4][5][15] has demonstrated that making use of the discovered frequent itemsets can avoid unnecessary computation as well as database scan in the course of incremental update, though the situation might become more complicated when taking into account ms_{old} and ms_{new}. The intuition is that a frequent itemset A remains frequent if $ms_{\text{new}}(A) \leq ms_{\text{old}}(A)$, and an infrequent itemset A keeps infrequent if $ms_{\text{new}}(A) > ms_{\text{old}}(A)$. In view of this, a better approach is to, within the set of discovered frequent itemsets L^{ED}, differentiate the frequent itemsets from the others and utilize them to avoid unnecessary computation as well as database scan.

Lemma 4. The support count, $count_{ED}(A)$, of an itemset A in *ED* is kept unchanged under the same item classification ontology.
Proof. This is straightforward since the original extended database is kept unchanged. □

This lemma implies that $count_{ED}(A)$ is immediately available if A is a discovered frequent itemset in *ED* with respect to ms_{old}, i.e., $A \in L^{ED}$, for L^{ED} being the set of frequent itemsets in *ED* with respect to ms_{old}.

Lemma 5. If an itemset A is frequent in *ED* with respect to $ms_{\text{old}}(A)$ and $ms_{\text{new}}(A) \leq ms_{\text{old}}(A)$, then A is frequent in *ED* with respect to $ms_{\text{new}}(A)$.
Proof. The lemma follows immediately from the fact that $sup_{ED}(A) \geq ms_{\text{old}}(A) \geq ms_{\text{new}}(A)$. □

Lemma 6. If an itemset A is infrequent in *ED* with respect to ms_{old} and $ms_{\text{new}}(A) \geq ms_{\text{old}}(A)$, then A is infrequent in *ED* with respect to ms_{new}.
Proof. Since A is infrequent in *ED* and $ms_{\text{new}}(A) \geq ms_{\text{old}}(A)$, we have $sup_{ED}(A) < ms_{\text{old}}(A) \leq ms_{\text{new}}(A)$. □

After clarifying what will become a candidate itemset in the original databse *ED* that undergoes the support constraint refinement, we will advance to the more complicated situation that embraces the incremental database *ed*.

Lemma 7. If an itemset A is frequent in *ED* with respect to ms_{old} and is frequent in *ed* with respect to ms_{new}, and $ms_{\text{new}}(A) \leq ms_{\text{old}}(A)$, then A is also frequent in the updated extended database *UE* with respect to ms_{new}.
Proof. Since A is frequent in *ED* and $ms_{\text{new}}(A) \leq ms_{\text{old}}(A)$, according to Lemma 5, we have $sup_{ED}(A) \geq ms_{\text{new}}(A) \Rightarrow$

$count_{ED}(A) \geq ms_{\text{new}}(A) \times |ED|$. Further, A is frequent in ed, so $sup_{ed}(A) \geq ms_{\text{new}}(A) \Rightarrow count_{ed}(A) \geq ms_{new}(A) \times |ed|$. Then $count_{ED}(A) + count_{ed}(A) \geq ms_{\text{new}}(A) \times (|ED| + |ed|) \Rightarrow count_{UE}(A) \geq ms_{\text{new}}(A) \times |UE| \Rightarrow sup_{UE}(A) \geq ms_{\text{new}}(A)$, which proves the lemma. □

Lemma 8. If A is infrequent in ED with respect to ms_{old} and is infrequent in ed with respect to ms_{new}, and $ms_{\text{new}}(A) \geq ms_{\text{old}}(A)$, then A is also infrequent in UE with respect to ms_{new}.

Proof. Since A is infrequent in ED and $ms_{\text{new}}(A) > ms_{\text{old}}(A)$, according to Lemma 6, $sup_{ED}(A) < ms_{\text{new}}(A) \Rightarrow count_{ED}(A) < ms_{\text{new}}(A) \times |ED|$. Further, A is infrequent in ed, so $sup_{ed}(A) < ms_{\text{new}}(A) \Rightarrow count_{ed}(A) < ms_{\text{new}}(A) \times |ed|$. Then $count_{ED}(A) + count_{ed}(A) < ms_{\text{new}}(A) \times (|ED| + |ed|) \Rightarrow count_{UE}(A) < ms_{\text{new}}(A) \times |UE| \Rightarrow sup_{UE}(A) < ms_{\text{new}}(A)$, which proves the lemma. □

Now, we will show how to employ the above properties to infer whether frequent or not a candidate itemset A is in the whole updated database UE with respect to $ms_{\text{new}}(A)$. We conclude that there are six different cases summarized in Table 2.

(1) If A is a frequent itemset in ED and ed, and $ms_{\text{new}}(A) \leq ms_{\text{old}}(A)$, then it is also frequent in the updated extended database UE.
(2) If A is not a frequent itemset in ED and ed, and $ms_{\text{new}}(A) \geq ms_{\text{old}}(A)$, then it is also infrequent in UE.
(3) If A is a frequent itemset in ED and ed, and $ms_{\text{new}}(A) > ms_{\text{old}}(A)$, then a simple calculation can determine whether A is frequent or not in UE.
(4) If A is frequent in ED but is not frequent in ed, then no matter $ms_{\text{new}}(A) \leq ms_{\text{old}}(A)$ or $ms_{\text{new}}(A) > ms_{\text{old}}(A)$, a simple calculation can determine whether A is frequent or not in UE.
(5) If A is not a frequent itemset in ED but is frequent in ed, then no matter $ms_{\text{new}}(A) \leq ms_{\text{old}}(A)$ or $ms_{\text{new}}(A) > ms_{\text{old}}(A)$, it is an undetermined itemset in UE, i.e., it may be either frequent or infrequent.
(6) If A is not a frequent itemset in ED and ed, and $ms_{\text{new}}(A) < ms_{\text{old}}(A)$, then it is an undetermined itemset in UE.

Table 2. Six cases arising in determining the status of a candidate itemset under incremental database and multiple minimum support update.

Conditions			Results		
L^{ED} (ms_{old})	L^{ed} (ms_{new})	$ms_{\text{new}}(A)$ to $ms_{\text{old}}(A)$	UE (ms_{new})	Action	Case
$\in$	$\in$	$\leq$	freq.	no	1
		$>$	?	compare $sup_{UE}(A)$ with $ms_{\text{new}}(A)$	3
	$\notin$	$\leq, >$	?	compare $sup_{UE}(A)$ with $ms_{\text{new}}(A)$	4
$\notin$	$\in$	$\leq, >$	?	scan ED	5
	$\notin$	$<$	?	scan ED	6
		$\geq$	infreq.	no	2

Cases 1 and 2 correspond to the situations expressed in Lemmas 7 and 8, respectively. So there is no need of extra action to determine what A will become. On the other hands, Cases 3 to 6 represent the situations with uncertainty. In Cases 3 and 4, since the support counts of A in ED and ed are available, a simple comparison of $sup_{UE}(A)$ with $ms_{\text{new}}(A)$ suffices for the purpose. But in Cases 5 and 6, the support count of A in ED is unknown; so an additional scan of ED is required. In summary, there is no need to invoke a further scan of ED for Cases 1 to 4 to determine the support count of itemset A. That is, we have utilized the information of the discovered frequent itemsets to avoid such a database scan. Furthermore, the identification of itemsets satisfying Cases 2 to 4 provides another opportunity for candidate pruning.

3.2 Algorithm DMA_CO

Based on the aforementioned concepts, we developed an algorithm, called DMA_CO, to accomplish the task. The main procedure of DMA_CO algorithm is as follows.

First, generate the set of candidate k-itemsets. If $k = 1$, then set C_1 to be the set of items in the item classification ontology T, i.e., all items in T are candidate 1-itemsets; if $k = 2$, generate candidate 2-itemsets C_2 from the frontier set F while for $k \geq 3$ invoke procedure apriori-gen to generate candidate k-itemsets C_k from L_{k-1}^{UE}.

Next, load the original frequent k-itemsets L_k^{ED} and divide C_k into two independent subsets: C_X, and C_Y, where C_X consists of k-itemsets in L_k^{ED}, and C_Y is composed of k-itemsets not in L_k^{ED}. Then, scan ed to get the support counts of itemsets in C_k and generate L_k^{ed}. For each k-itemset in C_X, its support count in ED is available while for those in C_Y, we need to scan ED to get their counts. Besides, for all candidate k-itemsets in C_Y and $k \geq 2$, delete those that are infrequent in ed and $ms_{\text{new}}(A) \geq ms_{\text{old}}(A)$, and then rescan ED to get the support counts of the remaining itemsets in C_Y.

Finally, accumulate the support count for each k-itemset in ed and ED, compare their supports with $ms_{\text{new}}(A)$, and get frequent k-itemsets for L_1^{UE}.

The DMA_CO algorithm is shown in Figure 5.

4 Example of Algorithm DMA_CO

Consider the example in Figures 2 and 3. To simplify the illustration, we use item "A" to stand for "MC", "B" for "Notebook", "C" for "IBM TP", "D" for "Acer TM", "E" for "ASUS P", "F" for "Printer", "G" for "HP DeskJet", "H" for "Epson EPL", and "I" for "ScanMaker" in the item classification ontology. The resulting figure is shown in Figure 6. The original frequent itemsets from Figure 2 are shown in Table 3.

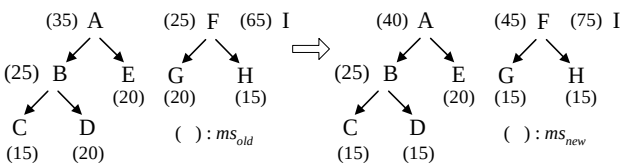
First, let candidate 1-itemsets C_1 be the set of items in the item classification ontology T, i.e., all items in T are candidate 1-itemsets. Second, load the original frequent 1-itemsets L_1^{ED} and divide C_1 into two subsets: C_X and C_Y, where $C_X = \{A, B, D, E, F, H\}$ and $C_Y = \{C, G, I\}$. Next, scan ed for C_1 and scan ED for C_Y. Calculate the support count of each 1-itemset in C_1, and generate the frontier set

F = {D, G, H, E, B, A, F, I}. After comparing the supports of each 1-itemset with the corresponding *ms*, the set of new frequent 1-itemsets L_1^{UE} is {A, B, D, E, G, H}.

Input: (1)*DB*: original database; (2)*T*: item classification ontology; (3)*db*: incremental database; (4)ms_{old}: old minimum support setting; (5)ms_{new}: new minimum support setting; (6)L^{ED}: set of original frequent itemsets.
Output: L^{UE}: set of new frequent itemsets in *UE* with respect to ms_{new}.
Steps:

1. $k = 1$;
2. **repeat**
3. **if** $k = 1$ **then** generate C_1 from T;
4. **else if** $k = 2$ **then** generate C_2 from the frontier set F;
5. **else** C_k = apriori-gen(L_{k-1}^{UE});
6. Delete any candidate in C_k that consists of an item and its ancestor;
7. Load original frequent *k*-itemsets L_k^{ED} ;
8. Divide C_k into two subsets: C_X and C_Y;
9. **for** each $A \in C_X$ **do** /* Cases 1, 3 & 4 */
10. Get $count_{ED}(A)$ from L_k^{ED} ;
11. Scan *ed* to compute $count_{ed}(A)$ for each itemset A in C_k;
12. $L_k^{ed} = \{A \mid A \in C_k \text{ and } sup_{ed}(A) \geq ms_{\text{new}}(A)\}$;
13. **if** $k = 1$ **then** scan *ED* for C_Y and generate the frontier set F;
14. **else if** $k \geq 2$ **then**
15. Delete any candidate A from C_Y if $A \notin L_k^{ed}$ and $ms_{\text{new}}(A) \geq ms_{\text{old}}(A)$; /* Case 2 */
16. Scan *ED* to count $count_{ED}(A)$ for each itemset A in C_Y; /* Cases 5& 6 */
17. **end if**
18. Accumulate $count_{UE}(A) = count_{ED}(A) + count_{ed}(A)$ for each itemset A in C_k;
19. $L_k^{UE} = \{A \mid A \in C_k \text{ and } sup_{UE}(A) \geq ms_{\text{new}}(A)\}$;
20. **until** $L_k^{UE} = \varnothing$
21. $L^{UE} = \cup_k L_k^{UE}$;

Figure 5. Algorithm DMA_CO.



Original Extended Database (*ED*)

TID	Primitive Items	Generalized Items
1	D, H, I	A, B, F
2	E, I	A
3	D	A, B
4	C, E	A, B
5	H, I	F
6	E, I	A
7	I	
8	D	A, B

Original Extended Incremental Database (*ed*)

TID	Primitive Items	Generalized Items
9	G, I	F
10	G, I	F

Figure 6. Example of an extended database with old and new multiple minimum supports.

After generating L_1^{UE} , we use the frontier set F to generate candidate 2-itemsets C_2, obtaining C_2 = {DG, DH, DE, DF, DI, GH, GE, GB, GA, GI, HE, HB, HA, HI, EB, EF, EI, BF, BI, AF, AI}. After loading the original frequent 2-itemsets L_2^{ED} , divide C_2 into two subsets: C_X = {HI, EI, AI} and C_Y = {DG, DH, DE, DF, DI, GH, GE, GB, GA, GI, HE, HB, HA, EB, EF, BF, BI, AF}. Note that {DB}, {DA}, {GF}, {HF}, {EA} and {BA} are pruned because of the existence of item-ancestor relationships. Next, scan *ed* for C_2 and determine the frequent 2-itemsets in *ed*, i.e., L_2^{ed} . According to Case 2, {DG}, {DH}, {DE}, {DF}, {DI}, {GH}, {GE}, {GB}, {GA}, {HE}, {HB}, {HA}, {EB}, {EF}, {BF}, {BI} and {AF} are discarded from C_Y since they are infrequent in *ed*. Then scan *ED* for the remaining candidates in C_Y. Calculate the support count for each 2-itemset in C_2 and generate L_2^{UE} = {GI, HI, EI}. Finally, we use L_2^{UE} to generate candidate 3-itemsets C_3 and apply the same procedure to generate L_3^{UE} , resulting in an empty set.

The frequent itemsets of the incremental database from Figure 3 are shown in Table 4. The resulting frequent itemsets from Figure 6 are shown in Table 5.

Table 3. Frequent itemsets generated from the database in Figure 2.

L_1^{ED}	*ms* (%)	*sup* (%)	L_2^{ED}	*ms* (%)	*sup* (%)
H	15	25	H, I	15	25
E	20	37.5	E, I	20	25
D	20	37.5	F, I	25	25
F	25	25	A, I	35	37.5
B	25	50			
A	35	75			

Table 4. Frequent itemsets generated from the incremental database in Figure 3.

L_1^{ed}	*ms* (%)	*sup* (%)	L_2^{ed}	*ms* (%)	*sup* (%)
G	15	100	G, I	15	100
F	45	100	F, I	45	100
I	75	100			

Table 5. Resulting fequent itemsets from the updated database in Figure 6.

L_1^{UE}	*ms* (%)	*sup* (%)	L_2^{UE}	*ms* (%)	*sup* (%)
D	15	30	G, I	15	20
G	15	20	H, I	15	20
H	15	20	E, I	20	20
E	20	30			
B	25	40			
A	40	60			

5 Experiments

In order to examine the performance of DMA_CO, we conduct experiments to compare its performance with that

of applying our proposed multi-supported generalized association mining algorithms, including MMS_Cumulate and MMS_Stratify [14], to the whole updated database using refined support constraint. A synthetic dataset generated by the IBM data generator [2] was used in the experiments. The parameter settings for synthetic data are shown in Table 6.

Table 6. Parameter setting.

Parameter		Default value
$\|DB\|$	Number of original transactions	200000
$\|db\|$	Number of incremental transactions	80000
$\|t\|$	Average sizes of transactions	16
N	Number of items	231
R	Number of groups	30
L	Number of levels	3
F	Fanout	5

The comparisons were evaluated from two aspects: the effect of varying minimum supports and that of incremental transaction sizes. Each aspect was then examined under two extreme cases of support refinement that set bounds to the performance behavior of our DMA_CO algorithm: 1) all items having $ms_{new} > ms_{old}$; and 2) all items having $ms_{new} < ms_{old}$. Besides, we adopted the ordinary case that the minimum support of an item is no larger than any of its ancestors. The vertical intersection counting strategy [10][17] were used in the implementation of each algorithm. All experiments were performed on an Intel Pentium-IV 2.80GHz with 2GB RAM, running on Windows 2000.

Minimum Supports: We use the following formula [8] to generate multiple minimum supports for each item a:

$$ms(a) = \begin{cases} \alpha \times sup_{DB}(a), & \text{if } \alpha \times sup_{DB}(a) \geq k \\ k, & \text{otherwise} \end{cases},$$

where $k = 0.4$.

We first compared the performance of these three algorithms with varying support specification and 80,000 incremental transactions. The experimental results are shown in Figure 7. For differentiation, we used DMA_CO1 to denote DMA_CO running under Case 1 ($ms_{new} > ms_{old}$) and DMA_CO2 for DMA_CO under Case 2 ($ms_{new} < ms_{old}$). Not surprisingly, DMA_CO1 is faster than DMA_CO2, mainly due to no new itemset in *ED* being generated in Case 1. As shown in the figure, DMA_CO outperforms MMS_Cumulate and MMS_Stratify in either case, with performance gain ranging from 14% to 80%. For $\alpha \geq 0.4$, the performance gain is tiny because of the significant decrease in the number of frequent itemsets.

Transaction Sizes: We then compared the three algorithms under varying transaction sizes. In this experiment, α was set to 0.3 for all item *ms* setting. As the results shown in Figure 8, the running time of all algorithms increase in proportional to the incremental size. And the performance gain of DMA_CO over MMS_Cumulate or MMS_Stratify remains nearly constant, independent of the transaction size.

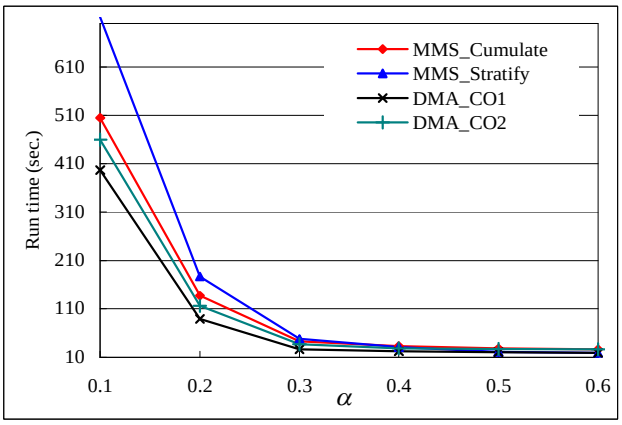


Figure 7. Performance comparison with varying support settings.

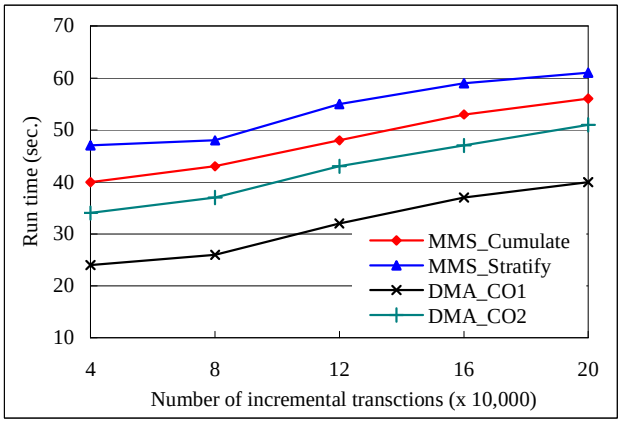


Figure 8. Performance comparison with different incremental sizes.

6 Related Work

The problem of updating association rules incrementally was first addressed by Cheung *et al.* [3], whose work was later be extended to incorporate the situation of deletion and modification [5]. Since then, a number of techniques have been proposed to improve the efficiency of incremental mining algorithms [7][9][11][13]. But all of them were confined to mining associations among primitive items. The problem of mining association rules in the presence of classification ontology information was first introduced in [6] and [12], independently. Cheung *et al.* [4] were the first to consider the problem of maintaining multi-level association rules.

The problem of mining association rules with multiple minimum supports was investigated by Liu *et al.* [8]. Their method allowed users to specify different minimum supports to different items and could find rules involving both frequent and rare items. However, their model considers no classification ontology at all, and hence fails to find generalized association rules. We have extended this problem model to that of incorporating the classification ontology of items [14], further advanced the maintenance issue of frequent itemsets with respect to

database update [15], and dealt with the situation for adjusting minimum support constraint to carry on iterative association mining [16]. In this context, this paper is a continuation of our previous work, taking into account both the database update and support constraint refinement to facilitate more real-world applications of associations mining.

7 Conclusion

In this paper, we have investigated the problem of dynamically mining association rules with classification ontology under multiple minimum support update when new transactions are continually added into the original database over time. We also have presented a novel algorithm, DMA_CO, for updating multi-supported frequent itemsets. Experimental results on a synthetic data show that the DMA_CO algorithm is 14% to 80% faster than applying multi-supported generalized associations mining algorithms to the whole updated database. In the future, we will extend the problem of updating generalized association rule to a more general model that the item classification and/or composition ontology can be evolving along with incremental transactions update.

References

[1] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," in *Proc. 1993 ACM-SIGMOD Intl. Conf. Management of Data*, 1993, pp. 207-216.

[2] R. Agrawal, and R. Srikant, "Fast Algorithms for Mining Association Rules," in *Proc. 20th Intl. Conf. Very Large Data Bases*, 1994, pp. 487-499.

[3] D.W. Cheung, J. Han, V.T. Ng, and C.Y. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Update Technique," in *Proc. 1996 Int. Conf. Data Engineering*, 1996, pp. 106-114.

[4] D.W. Cheung, V.T. Ng and B.W. Tam, "Maintenance of Discovered Knowledge: A case in Multi-level Association Rules," in *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining*, 1996, pp. 307-310.

[5] D.W. Cheung, S.D. Lee and B. Kao, "A General Incremental Technique for Maintaining Discovered Association Rules," in *Proc. DASFAA'97*, 1997, pp. 185-194.

[6] J. Han and Y. Fu, "Discovery of Multiple-level Association Rules from Large Databases," in *Proc. 21st Intl. Conf. Very Large Data Bases, Zurich, Switzerland*, 1995, pp. 420-431.

[7] T.P. Hong, C.Y. Wang and Y.H. Tao, "Incremental Data Mining Based on Two Support Thresholds," in *Proc. 4th Int. Conf. Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, 2000, pp. 436-439.

[8] B. Liu, W. Hsu and Y. Ma, "Mining Association Rules with Multiple Minimum Supports," in *Proc. 5th Intl. Conf. Knowledge Discovery and Data Mining*, 1999, pp. 337-341.

[9] K.K. Ng and W. Lam, "Updating of Association Rules Dynamically," in *Proc. 1999 Int. Symp. Database Applications in Non-Traditional Environments*, 2000, pp. 84-91.

[10] A. Savasere, E. Omiecinski and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," in *Proc. 21st Intl. Conf. Very Large Data Bases*, 1995, pp. 432-444.

[11] N.L. Sarda and N.V. Srinivas, "An Adaptive Algorithm for Incremental Mining of Association Rules," in *Proc. 9th Int. Workshop on Database and Expert Systems Applications*, 1998, pp. 240-245.

[12] R. Srikant and R. Agrawal, "Mining Generalized Association Rules," in *Proc. 21st Int. Conf. Very Large Data Bases*, 1995, pp. 407-419.

[13] S. Thomas, S. Bodagala, K. lsabti and S. Ranka, "An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases," in *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining*, 1997, pp. 263-266.

[14] M.C. Tseng and W.Y. Lin, "Mining Generalized Association Rules with Multiple Minimum Support," in *Proc. Int. Conf. Data Warehousing and Knowledge Discovery, Lecture Notes in Computer Science*, Vol. 2114, 2001, pp. 11-20.

[15] M.C. Tseng and W.Y. Lin, "Maintenance of Generalized Association Rules with Multiple Minimum Supports," *Intelligent Data Analysis*, Vol. 8, No. 4, 2004, pp. 417-436.

[16] M.C. Tseng, W.Y. Lin and R. Jeng, "Efficient Remining of Generalized Association Rules under Multiple Minimum Support Refinement," in *Proc. 9th Int. Conf. Knowledge-Based Intelligent Information & Engineering Systems, Lecture Notes in Computer Science*, Vol. 3683, 2005, pp. 1338-1344.

[17] M.J. Zaki, "Scalable Algorithms for Association Mining," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 2, 2000, pp. 372-390.

Author Biographies

Ming-Cheng Tseng is a PhD student in the Department of Information Engineering at the I-Shou University, Taiwan. He received the BS and MS degrees in Information Management from the I-Shou University, Taiwan, in 1999 and 2002, respectively. His research interests include data mining, data warehousing, database systems, fuzzy theory and information systems.

Wen-Yang Lin is a Professor and the Chair of Department of Computer Science and Information Engineering in National University of Kaohsiung. He received his B. S. and M. S. both in Computer Science and Information Engineering from National Chiao-Tung University in 1988 and 1990, respectively. He then received his Ph.D. in Computer Science and Information Engineering from National Taiwan University in 1994. Dr. Lin has published more than 30 journal publications and 70 conference publications in the area of data warehousing, data mining, evolutionary computation, sparse matrix technology and large-scale supercomputing. Dr. Lin is a member of IEEE, the Taiwanese AI Association and the Institute of Information and Computing Machinery.

Rong Jeng is an Associate Professor in the Department of Information Management at the I-Shou University, Taiwan. He received the BS degree in Control Engineering from the National Chiao Tung University, Taiwan, and the MS degree in Electrical Engineering from the Memphis State University, and PhD degree in Electrical and Computer Engineering from the University of Cincinnati, in 1989. His research interests include information education, application software, statistical analysis, operation research and system simulation.