Discovery of Quantitative Association Rules from a Large Database of Sales Transactions

Pauray S.M. Tsai Chien-Ming Chen

Department of Information Management
Ming Hsin Institute of Technology
Hsin-Feng, Hsinchu 304, Taiwan, R.O.C.
e-mail: pauray@mis.mhit.edu.tw

Abstract

In this paper, we examine the issue of mining quantitative association rules in a large database of sales transactions. When purchased quantities are considered, most of the supports for items associated with their purchased quantities may be low, and the number of potentially interesting association rules discovered may be few. In order to discover more potentially interesting rules, we present two partition algorithms to partition all the possible quantities into intervals for each item. We also propose an efficient mechanism to discover all the large itemsets from the partition result. Experimental results show that by our approach, the total execution time can be reduced significantly. Moreover, the number of potentially interesting association rules discovered from our partition result is larger than that of rules discovered from the original data, which demonstrates the significance of our work.

1 Introduction

Data mining has attracted much attention in database communities because of its wide applicability [1, 2, 5, 6, 7, 9, 11, 13, 15]. One major application area of data mining is mining association rules among items in a large database of salses transactions [3, 10, 17]. Specifically, given a set of transactions, where each transaction consists of a set of items, an association rule is an expression $X \Longrightarrow Y$, where X and Y are sets of items. A set of items is called an *itemset*. An example of such an association rule might be "80% of customers who buy itemset X also buy itemset Y". The percentage 80% is called the *confidence* of the rule.

The problem of mining association rules can be decomposed into two subproblems. Let the *support* of an itemset Z be the ratio of the number of transactions containing itemset Z and the total number of transactions in the database. First, all itemsets whose supports are no less than the user-specified *minimum support* are identified. Each such itemset is referred to as a *large itemset*. Second, the association rules whose confidences are no less than the user-specified *minimum confidence* are generated from these large itemsets. For example, let Z be a large itemset. The *confidence* of a rule $X \Longrightarrow Y$ is the ratio of the supports of itemset $X \cup Y$ and itemset X. All rules of the form $X \Longrightarrow Y$ satisfying $X \cup Y = Z$, $X \cap Y = \emptyset$, and the minimum confidence constraint are generated. Once all large itemsets are discovered, the desired association rules can be obtained in a straightforward manner.

An algorithm for finding all association rules, referred to as the AIS algorithm, was first explored in [3]. The AIS algorithm requires to repeatedly scan the database. It uses the large itemsets discovered in the previous pass as the basis to generate new potentially large itemsets, called *candidate* itemsets, and counts their supports during the pass over the data. Specifically, after reading a transaction, it is determined which of the large itemsets found in the previous pass are contained in the transaction. New candidate itemsets are generated by extending these large itemsets with other items in the transaction. However, the performance study in [4] shows that AIS is not efficient since it generates too many candidate itemsets that turn out not to be large itemsets.

In [4], the Apriori and AprioriTid algorithms were proposed for efficiently mining association rules. Different from the AIS algorithm, these two algorithms generate the candidate itemsets by using only the large itemsets found in the previous pass. For example, at the $(k-1)$th iteration, all large itemsets containing $k-1$ items, called large $(k-1)$-itemsets, are generated. In the next iteration, the candidate itemsets containing k items are generated by joining large $(k-1)$-itemsets. The heuristic is that any subset of a large itemset must be large. By the heuristic, the Apriori and AprioriTid algorithms can generate a much smaller number of candidate itemsets than the AIS algorithm.

Another effective algorithm for the candidate set generation, called DHP, was proposed in [14]. By utilizing a hash technique, DHP can efficiently generate all the large itemsets and effectively reduce the size of the transaction database. The performance study in [14] shows that the number of candidate 2-itemsets generated by DHP is smaller than that by the Apriori algorithm. Moreover, the transaction database size is trimmed at a much earlier stage of the iterations. As a result, the total execution time can be reduced significantly by DHP.

The notion of mining multiple-level association rules was introduced in [10]. In many applications, association rules discovered at multiple concept levels are useful. Usually, the association relationship expressed at a lower concept level provides more specific information than that expressed at a higher concept level. The approach is to first find large items at the top-most level, and then progressively deepen the mining process into their descendants at lower levels.

A similar idea of extracting generalized association rules using a taxonomy was presented in [18].

The issue of mining optimized association rules for numeric and categorical attributes was investigated in [8, 16]. An optimized association rule has the form $(X \in [v_1, v_2]) \wedge C_1 \rightarrow C_2$, where X is a numeric attribute, v_1 and v_2 are uninstantiated variables, and C_1 and C_2 are conditions containing only instantiated attributes. The problem is to determine values for variables v_1 and v_2 such that either the support or confidence of the rule is maximized. In [8], only a single optimal interval for a single numeric attribute can be determined. [16] generalized the optimized association rules problem to contain a number of uninstantiated attributes.

The problem of mining quantitative association rules in large relational databases was introduced in [19]. The attributes considered in a relation are quantitative or categorical. The values of the attribute are partitioned using an *equi-depth* approach (that is, each interval resulted from the partition contains roughly the same number of tuples), and then adjacent intervals are combined as necessary. A related problem is clustering association rules [12], which combines similar "adjacent" association rules to form a few general rules. [12] proposed a geometric-based algorithm to perform the clustering and applied the Minimum Description Length principle as a means of evaluating clusters.

In this paper, we examine the issue of mining quantitative association rules in a large database of sales transactions. A transaction in the database typically consists of the customer identifier, the items bought in the transaction, and the quantity associated with each purchased item. The previous approaches for mining association rules in the transaction database focus on discovering associations among items without considering the relationships between items and their purchased quantities. In real applications, it is essential to discover associations among items associated with their purchased quantities. For example, a quantitative association rule for a given transaction database might be "40% of customers who buy two loaves of bread also buy three bottles of milk". This kind of rules will be useful to improving marketing strategy.

When purchased quantities are considered, if most of the supports for items associated with their purchased quantities are low, the number of potentially interesting association rules discovered will be few. In order to discover more potentially interesting association rules, we present two partition algorithms to partition all the possible quantities into intervals for each item. The basic idea is to combine adjacent values into intervals such that the support for each single interval could exceed the minimum support. As a result, we could obtain a quantitative association rule such as "80% of customers who buy two or three loaves of bread also buy two or three bottles of milk". We also propose a mechanism, called LqiTid, to discover all the large itemsets from the partition result. By recording the identifiers of transactions containing the large itemset, we can scan the database only once. Experimental results show that the performance of LqiTid greatly outperforms that of the modified Apriori algorithm. Moreover, the number of potentially interesting association rules discovered from our partition result is larger than that of rules discovered from the original data in all cases. Especially when the minimum support exceeds a critical value, there is almost no rules discovered from the original data, but there are still some potentially interesting association rules discovered from the partition result. Our work is different from the work in [19]. [19] considered the problem of mining association rules in large relational tables containing quantitative attributes, in contrast to our algorithms designed for transaction-type items.

This paper is organized as follows. In Section 2, the problem description is given and two partition algorithms are introduced. The algorithm LqiTid proposed for efficient generation of all large itemsets is described in Section 3. In Section 4, the performance results are presented. Finally conclusions are given in Section 5.

2 Partition Algorithms

Let DB be a transaction database. A q_item, denoted as $< i, q >$, represents a purchased item i and a quantity q associated with this item. A transaction in the database consists of a transaction identifier (TID) and a set of q_items (q_itemset) purchased in the transaction. We assume that q_items in each transaction are sorted according to their items in the lexicographic order.

The support for a q_item is defined to be the value of dividing the number of transactions containing this q_item by the total number of transactions in database DB. Since q_items with the same item can have different quantities, the support for a q_item can be low. If most of q_items have low supports, the number of potentially interesting association rules discovered will be few. In order to discover more interesting information embedded in the transaction database, we partition the values of the quantity into intervals for each item and map each quantity to an integer which represents its corresponding interval.

Let $\{< i, q_1 >, < i, q_2 >, ..., < i, q_n >\}$ be the set of q_items in DB, which have the same item i, and $q_1 < q_2 < ... < q_n$. Assume that $c_1, c_2, ...,$ and c_n are the numbers of transactions containing $< i, q_1 >, < i, q_2 >, ...,$ and $< i, q_n >$, respectively. In the following, we consider two partition methods: the **PARTITION1** method, as shown in Figure 1, and the **PARTITION2** method, as shown in Figure 2. The **PARTITION1** scheme can be outlined as follows. Let T be the total number of transactions. First, we find a minimum integer k_1 such that $\frac{\sum_{m=1}^{k_1} c_m}{T}$ is greater than or equal to the minimum support, say s, and the interval $[q_1..q_{k_1}]$ is generated. Then, we find a minimum integer k_2 $(k_2 > k_1)$ such that $\frac{\sum_{m=k_1+1}^{k_2} c_m}{T} \geq s$, and the interval $[q_{(k_1+1)}..q_{k_2}]$ is generated. The rest is deduced by analogy. If $\frac{\sum_{m=k_j+1}^{n} c_m}{T} < s$, the last generated interval $[q_{(k_{(j-1)}+1)}..q_{k_j}]$ is combined with $[q_{(k_j+1)}..q_n]$ to form the interval $[q_{(k_{(j-1)}+1)}..q_n]$. The **PARTITION2** scheme can be described as follows. If $\frac{c_m}{T} \geq s$, where $1 \leq m \leq n$, then q_m is considered to be a separate interval represented as $[q_m]$. Note that $\frac{c_m}{T}$ denotes the support for the q_item $< i, q_m >$. Assume that the supports for q_items $< i, q_j >$ and $< i, q_k >$ are no less than s and those for q_items $< i, q_{(j+1)} >, < i, q_{(j+2)} >, ...,$ and $< i, q_{(k-1)} >$ are less than s. Then the interval $[q_{(j+1)}..q_{(k-1)}]$ is gener-

```
/* PARTITION1 */
    $j = 1$;
    $cnt = 0$;
    $V_j.first = q_1$;
    /* $V_j.first$ represents the first value
    in the interval $V_j$ */
        for ($k = 1; k \leq n$; $k$++) do
          $cnt = cnt + c_k$;
          if ($\frac{cnt}{T} \geq s$) then
          begin
             $V_j.last = q_k$;
             /* $V_j.last$ represents the last value
             in the interval $V_j$ */
             $j$++;
             $cnt = 0$;
             if ($k < n$) then $V_j.first = q_{(k+1)}$;
          end
          else if ($k = n$) then
                if ($j > 1$) then $V_{j-1}.last = q_n$;
                else $V_j.last = q_n$;
      end for
```

Figure 1: The **PARTITION1** method.

```
/* PARTITION2 */
    $j = 1$;
    $V_j.first = q_1$;
    /* $V_j.first$ represents the first value
    in the interval $V_j$ */
    $flag = 0$;
    /* $flag = 0$ indicates that at present
    there is only a value in interval $V_j$ */
        for ($k = 1; k \leq n$; $k$++) do
          if ($\frac{c_k}{T} \geq s$) then
          begin
             if ($flag = 0$) then $V_j.last = q_k$;
             /* $V_j.last$ represents the last value
             in the interval $V_j$ */
             else
               begin
                  $V_j.last = q_{(k-1)}$;
                  $j$++;
                  $V_j.first = q_k$;
                  $V_j.last = q_k$;
               end
             $j$++;
             $V_j.first = q_{(k+1)}$;
             $flag = 0$;
          end
          else if ($k = n$) then $V_j.last = q_n$;
                 else if ($flag = 0$) then $flag = 1$;
      end for
```

Figure 2: The **PARTITION2** method.

ated. The generated intervals are mapped to consecutive integers such that the order of these intervals is preserved. The effects of these two methods on mining results will be examined in Section 4.

Example 1: Let $\{< i, 1 >, < i, 2 >, < i, 3 >, < i, 4 >, < i, 5 >\}$ be the set of q_items in database DB, which have the same item i, and 50, 30, 100, 20, and 40 be the numbers of transactions containing $< i, 1 >, < i, 2 >, < i, 3 >, < i, 4 >$, and $< i, 5 >$, respectively. Assume that the total number of transactions is 500 and the minimum support is 10%. By **PARTITION1**, we have three intervals for item i: [1], [2..3], and [4..5]. The intervals [1], [2..3], and [4..5] are mapped to integers 1, 2, and 3, respectively. Thus, q_item $< i, 3 >$ will be mapped to the generalized form $< i, 2 >$ (i.e., $< i,$ [2..3] $>$). By **PARTITION2**, we have four intervals for item i: [1], [2], [3], and [4..5]. The intervals [1], [2], [3], and [4..5] are mapped to integers 1, 2, 3, and 4, respectively. Thus, q_item $< i, 3 >$ will be mapped to the generalized form $< i, 3 >$ (i.e., $< i,$ [3] $>$).

After the partition process, each q_item in each transaction is mapped to a generalized form. Let DB' be the transaction database after performing partition process on database DB, and $X = \{x_1, x_2, ..., x_m\}$ the set of q_items in DB'. A *quantitative association rule* is of the form $Y \Longrightarrow Z$, where $Y \subset X, Z \subset X$, and $Y \cap Z = \emptyset$. The rule $Y \Longrightarrow Z$ holds in the transaction database DB' with *confidence c* if c% of the transactions in DB' that contain Y also contain Z. The rule $Y \Longrightarrow Z$ has *support s* in the transaction database DB' if s% of the transactions in DB' contain $Y \cup Z$. A transaction supports a q_itemset Y, if all the q_items in Y are contained in the transaction. The support for a q_itemset is determined by dividing the number of transactions supporting the q_itemset by the total number of transactions. A q_itemset is called a *large* q_itemset if its support is greater than or equal to the minimum support. A q_itemset of size k is called a $k-$q_itemset.

The problem of mining quantitative association rules mainly consists of two steps:

1. Partition the values of the quantity into intervals for each item. The partition methods have been introduced in this section.
2. Find all large q_itemsets. A new algorithm named **LqiTid** is presented in the next section.

After discovering large q_itemsets, the quantitative association rules can be extracted. Let X be a large q_itemset, $Y \cup Z = X$ and $Y \cap Z = \emptyset$. Then $Y \Longrightarrow Z$ is a quantitative association rule if its confidence is greater than or equal to the minimum confidence.

3 Large Q_itemset Generation

In this section, we propose a mechanism by which the database is scanned only once for discovering large q_itemsets. **Example 2** is used to illustrate our approach.

Example 2. Let Figure 3 be the original transaction database DB. Assume the minimum support is 20%. By **PARTITION1** (illustrated in Section 2), the mapping information and the resultant transaction database DB' after performing the partition on database DB are shown in Figure 4 and Figure 5, respectively.

TID	q_itemset
1	<B,1> <C,2> <F,1> <G,3>
2	<C,1> <D,1> <G,1>
3	<C,1> <F,3> <G,1>
4	<A,2> <B,1> <C,3> <G,2>
5	<A,1> <B,1>
6	<B,3> <C,2>
7	<B,3> <C,2> <D,1> <E,5> <F,1>
8	<B,1> <C,2> <G,3>
9	<B,4> <F,2>
10	<A,2> <B,2> <C,3> <F,1>
11	<B,2> <C,3> <F,1>
12	<A,1> <B,2> <G,2>
13	<B,2> <C,2> <G,4>
14	<A,1> <B,2> <C,3> <F,3> <G,5>
15	<A,3> <C,4> <G,3>

Figure 3: The transaction database *DB*.

Mapping Item A

quantity	integer
1	1
2..3	2

Mapping Item B

quantity	integer
1	1
2	2
3..4	3

Mapping Item C

quantity	integer
1..2	1
3..4	2

Mapping Item D

quantity	integer
1	1

Mapping Item E

quantity	integer
5	1

Mapping Item F

quantity	integer
1	1
2..3	2

Mapping Item G

quantity	integer
1..2	1
3..5	2

Figure 4: The mapping information.

3.1 Information for Discovering Large Q_itemsets

Let $TS(\{x\})$ be the set of TIDs for transactions containing q_item x. For example, in database DB', $TS(\{<A,1>\}) = \{5,12,14\}$ and $TS(\{<B,1>\}) = \{1,4,5,8\}$. The set $TS(\{x_1,x_2\})$, representing the set of TIDs for transactions containing the two q_items x_1 and x_2, can be obtained by performing the set intersection on $TS(\{x_1\})$ and $TS(\{x_2\})$:

$$TS(\{x_1,x_2\}) = TS(\{x_1\}) \cap TS(\{x_2\})$$

where the symbol "$\cap$" denotes the set intersection. For example, $TS(\{<A,1>,<B,1>\}) = TS(\{<A,1>\}) \cap TS(\{<B,1>\}) = \{5\}$.

Definition 1: Suppose $x_1, x_2, ...,$ and x_k are q_items. $TS(\{x_1,x_2,...,x_k\})$ is the set of TIDs for the transactions containing all the q_items in the q_itemset $\{x_1,x_2,...,x_k\}$. $SP(\{x_1,x_2,...,x_k\})$, which represents the number of TIDs in $TS(\{x_1,x_2,...,x_k\})$, is defined as:

$$SP(\{x_1,x_2,...,x_k\}) = Card(TS(\{x_1,x_2,...,x_k\}))$$
$$= Card(TS(\{x_1\}) \cap TS(\{x_2\}) \cap ... \cap TS(\{x_k\}))$$

TID	q_itemset
1	<B,1> <C,1> <F,1> <G,2>
2	<C,1> <D,1> <G,1>
3	<C,1> <F,2> <G,1>
4	<A,2> <B,1> <C,2> <G,1>
5	<A,1> <B,1>
6	<B,3> <C,1>
7	<B,3> <C,1> <D,1> <E,1> <F,1>
8	<B,1> <C,1> <G,2>
9	<B,3> <F,2>
10	<A,2> <B,2> <C,2> <F,1>
11	<B,2> <C,2> <F,1>
12	<A,1> <B,2> <G,1>
13	<B,2> <C,1> <G,2>
14	<A,1> <B,2> <C,2> <F,2> <G,2>
15	<A,2> <C,2> <G,2>

Figure 5: The transaction database DB'.

where $Card(S)$ denotes the cardinality of set S.

After scanning the transaction database DB', the information for discovering large q_itemsets is extracted as shown in Figure 6. $TS(\{x_1,x_2,...,x_k\})$ can be obtained according to the information.

q_item	*TS*	*SP*
<A,1>	{5,12,14}	3
<A,2>	{4,10,15}	3
<B,1>	{1,4,5,8}	4
<B,2>	{10,11,12,13,14}	5
<B,3>	{6,7,9}	3
<C,1>	{1,2,3,6,7,8,13}	7
<C,2>	{4,10,11,14,15}	5
<D,1>	{2,7}	2
<E,1>	{7}	1
<F,1>	{1,7,10,11}	4
<F,2>	{3,9,14}	3
<G,1>	{2,3,4,12}	4
<G,2>	{1,8,13,14,15}	5

Figure 6: The information for discovering large q_itemsets.

3.2 Algorithm LqiTid

In this subsection, we introduce the algorithm LqiTid for efficient generation of large q_itemsets. Let T be the total number of transactions in the transaction database. *minsup* is defined as

$$minsup = \lceil T \times minimum\ support \rceil .$$

Lemma 1: If $SP(\{x_1,x_2,...,x_k\})$ is greater than or equal to *minsup*, then $\{x_1,x_2,...,x_k\}$ is a large $k-$q_itemset.

Proof: According to the definition of $SP(\{x_1,x_2,...,x_k\})$, $SP(\{x_1,x_2,...,x_k\})$ is the number of transactions containing all the q_items in the q_itemset $\{x_1,x_2,...,x_k\}$. If $SP(\{x_1,x_2,...,x_k\}) \geq minsup$, then the support for the q_itemset $\{x_1,x_2,...,x_k\}$ is no less than the minimum support. Thus $\{x_1,x_2,...,x_k\}$ is a large $k-$q_itemset.

Lemma 2: If the q_itemset $\{x_1, x_2, ..., x_k\}$ is a large $k-$q_itemset, $k \geq 2$, then any proper subset of the q_itemset is also a large q_itemset.
Proof: Assume that the q_itemset $\{x_1, x_2, ..., x_k\}$ is a large $k-$q_itemset. Namely, $SP(\{x_1, x_2, ..., x_k\}) \geq minsup$. Let S be a proper subset of $\{x_1, x_2, ..., x_k\}$. Then, $SP(S) \geq SP(\{x_1, x_2, ..., x_k\})$. Therefore, $SP(S) \geq minsup$ and S is a large q_itemset by **Lemma 1**.

According to **Lemma 2**, candidate $k-$q_itemsets are generated from the set of large $(k-1)-$q_itemsets, L_{k-1}. The idea is similar to Apriori [4]. We use the notation $x[1], x[2], ..., x[k-1]$ to represent the $k-1$ q_items in the $(k-1)$-q_itemset x. Let $item(x[j])$ be the item value in q_item $x[j]$. For the q_itemset $\{x[1], x[2], ..., x[k-1]\}$, we assume that $item(x[1]) < item(x[2]) < ... < item(x[k-1])$.

Definition 2: The set of candidate $k-$q_itemsets $(k \geq 2)$, C_k, is defined as

$$\begin{aligned} C_k = & \{\{x_p[1], x_p[2], ..., x_p[k-1], x_q[k-1]\} \mid \\ & x_p \in L_{k-1} \text{and } x_q \in L_{k-1} \text{ and } x_p[1] = x_q[1], \\ & x_p[2] = x_q[2], ..., \text{and } x_p[k-2] = x_q[k-2], \\ & \text{and } item(x_p[k-1]) < item(x_q[k-1])\} \end{aligned}$$

The TS value of candidate $k-$q_itemset $\{x_p[1], x_p[2], ..., x_p[k-1], x_q[k-1]\}$ can be computed as
$TS(\{x_p[1], x_p[2], ..., x_p[k-1], x_q[k-1]\})$
$= TS(x_p) \cap TS(x_q)$

Different from Apriori, we need not scan the database anymore. Once a candidate q_itemset is generated, we can immediately determine whether it is a large q_itemset by computing its TS and SP values. The set of large $k-$q_itemsets is defined as:

$$L_k = \{x \mid x \in C_k \text{ and } SP(x) \geq minsup\}$$

The algorithm **LqiTid** consists of three phases: information extraction phase, large 1-q_itemset generation phase, and large k-q_itemset generation phase $(k \geq 2)$. The algorithm is as shown in figure 7.

In the following, we use the transaction database DB' shown in Figure 5 to illustrate our approach. Assume that the minimum support is 10% (i.e., *minsup* is 2). First, the set of large 1-q_itemsets, L_1, is determined using the information in Figure 6. According to **Lemma 1**, L_1 is the set of q_items satisfying $SP \geq 2$, as shown in Figure 8. Then the set of large 2-q_itemsets, L_2, is determined using the information in Figure 8. For example, {<A,1>, <B,1>} is a candidate 2-q_itemset. However, it is not a large 2-q_itemset since TS({<A,1>, <B,1>})= {5} and SP({<A,1>, <B,1>})=1 which is less than *minsup*. {<A,2>, <C,2>} is a large 2-q_itemset since TS({<A,2>, <C,2>})= $\{4, 10, 15\}$ and SP({<A,2>, <C,2>})=3 which is greater than *minsup*. Figure 9 shows the information for L_2. Next, L_3, is determined using the information in Figure 9. For example, {<B,2>,<C,2>,<G,2>} is a candidate 3-q_itemset generated from large 2-q_itemsets {<B,2>,<C,2>} and {<B,2>,<G,2>}. However, it is not a large 3-q_itemset since TS({<B,2>, <C,2>,<G,2>})= TS({<B,2>, <C,2>})$\cap TS$({<B,2>,<G,2>})= {14}

```
/* Information extraction phase */
Scan the transaction database once.
For each q_item $x$, compute $TS(\{x\})$ and $SP(\{x\})$.
/* Large 1-q_itemset generation phase */
$L_1 = \{x \mid x$ is a q_item and $SP(\{x\}) \geq minsup\}$
/* Large $k$-q_itemset generation phase */
for ($k$=2; $\mid L_{k-1} \mid > 1$; $k$++) do begin
    According to Definition 2, generate $C_k$
    using $L_{k-1}$.
    forall candidates $c \in C_k$ do begin
        Assume that $c$ is generated from large
        $(k-1)$-q_itemsets $S_1$ and $S_2$.
        $TS(c) = TS(S_1) \cap TS(S_2)$;
        $SP(c) = Card(TS(c))$;
        If $SP(c) \geq minsup$ then
            $L_k = L_k \cup \{c\}$;
    end for
end for
```

Figure 7: The algorithm **LqiTid**.

and SP({<B,2>, <C,2>,<G,2>})=1 which is less than *minsup*. {<B,2>, <C,2>, <F,1>} is a large 3-q_itemset since TS({<B,2>, <C,2>,<F,1>})= TS({<B,2>,<C,2>})$\cap TS$({<B,2>,<F,1>})= {10, 11} and SP({<B,2>, <C,2>,<F,1>})=2 which is equal to *minsup*. Figure 10 shows the information for L_3. Since $C_4 = \emptyset$, $L_4 = \emptyset$ and the mining process terminates.

large 1-q_itemset	TS	SP
{<A,1>}	{5,12,14}	3
{<A,2>}	{4,10,15}	3
{<B,1>}	{1,4,5,8}	4
{<B,2>}	{10,11,12,13,14}	5
{<B,3>}	{6,7,9}	3
{<C,1>}	{1,2,3,6,7,8,13}	7
{<C,2>}	{4,10,11,14,15}	5
{<D,1>}	{2,7}	2
{<F,1>}	{1,7,10,11}	4
{<F,2>}	{3,9,14}	3
{<G,1>}	{2,3,4,12}	4
{<G,2>}	{1,8,13,14,15}	5

Figure 8: The information for large 1-q_itemsets.

Let X be a large q_itemset, $Y \cup Z = X$, and $Y \cap Z = \emptyset$. $Y \Longrightarrow Z$ is a quantitative association rule if its confidence is greater than or equal to the minimum confidence. The confidence of $Y \Longrightarrow Z$ is determined by $\frac{SP(X)}{SP(Y)}$. Let us continue from the above example. Assume that the minimum confidence is 65%. We can obtain the following quantitative association rules, referencing the mapping information in Figure 4:

{<A, 1>} $\Longrightarrow$ {<B, 2>} (67%)
{<A, [2..3]>} $\Longrightarrow$ {<C, [3..4]>} (100%)
{<B, [3..4]>} $\Longrightarrow$ {<C, [1..2]>} (67%)
{<D, 1>} $\Longrightarrow$ {<C, [1..2]>} (100%)
{<B, 2>,<C, [3..4]>} $\Longrightarrow$ {<F, 1>} (67%)

large 2-q_itemset	*TS*	*SP*
{<A,1>,<B,2>}	{12,14}	2
{<A,2>,<C,2>}	{4,10,15}	3
{<B,1>,<C,1>}	{1,8}	2
{<B,1>,<G,2>}	{1,8}	2
{<B,2>,<C,2>}	{10,11,14}	3
{<B,2>,<F,1>}	{10,11}	2
{<B,2>,<G,2>}	{13,14}	2
{<B,3>,<C,1>}	{6,7}	2
{<C,1>,<D,1>}	{2,7}	2
{<C,1>,<F,1>}	{1,7}	2
{<C,1>,<G,1>}	{2,3}	2
{<C,1>,<G,2>}	{1,8,13}	3
{<C,2>,<F,1>}	{10,11}	2
{<C,2>,<G,2>}	{14,15}	2

Figure 9: The information for large 2-q_itemsets.

large 3-q_itemset	*TS*	*SP*
{<B,2>,<C,2>,<F,1>}	{10,11}	2
{<B,1>,<C,1>,<G,2>}	{1,8}	2

Figure 10: The information for large 3-q_itemsets.

{<B, 2>,<F, 1>} $\Longrightarrow$ {<C, [3..4]>} (100%)
{<C, [3..4]>,<F, 1>} $\Longrightarrow$ {<B, 2>} (100%)
{<B, 1>,<C, [1..2]>} $\Longrightarrow$ {<G, [3..5]>} (100%)
{<B, 1>,<G, [3..5]>} $\Longrightarrow$ {<C, [1..2]>} (100%)
{<C, [1..2]>,<G, [3..5]>} $\Longrightarrow$ {<B, 1>} (67%)

4 Experimental Results

To assess the performance of LqiTid, we conduct several experiments on Sun SPARC/20 workstation. We first describe the generation of synthetic data used in the experiments. Then, we compare the performance of LqiTid and the modified version of Apriori [4]. Finally, the effects of the proposed partition algorithms on the number of rules discovered are evaluated.

4.1 Generation of Synthetic Data

The method used to generate synthetic transactions is similar to the one used in [4]. Table 1 summarizes the parameters used in our experiments.

$\mid D \mid$	Number of transactions
$\mid T \mid$	Average size of the transactions
$\mid I \mid$	Average size of the maximal potentially large q_itemsets
$\mid L \mid$	Number of maximal potentially large q_itemsets
N	Number of items
MQ	Maximum value of the quantities

Table 1: Parameters.

We first generate a set of potentially large q_itemsets L. The size of each potentially large q_itemset in L is determined from a Poisson distribution with mean equal to $\mid I \mid$. Q_itemsets in L are generated as follows. Items in the first q_itemset are chosen randomly from N items and quantities are chosen randomly from the range $[1..MQ]$. In order to have common q_items in subsequent q_itemsets, some fraction of q_items in a q_itemset are chosen from the previous q_itemset generated. For each q_item $< x, q >$ in the previous q_itemset, we flip a coin to determine whether the item x will be contained in the current large q_itemset. If item x is retained in the current q_itemset, a coin is flipped to determine whether the associated quantity is q. If the answer is "no", we randomly choose a value from $[1..MQ]$ as the associated quantity for item x. The remaining q_items are picked at random.

Then we generate transactions in the database. The size of each transaction is determined from a Poisson distribution with mean equal to $\mid T \mid$. Each transaction is generated as follows. First a potentially large q_itemset is chosen randomly from L, whose size is less than or equal to the transaction size. To model the phenomenon that all the q_items in a large q_itemset are not always bought together, we flip a coin to determine whether the q_itemset will be contained in the transaction. If the answer is "yes", the remaining q_items are picked at random. Otherwise, for each q_item in the chosen q_itemset, we flip a coin to determine whether the q_item will be retained in the current transaction. Similarly, the remaining q_items are picked at random.

```
$L_1$ = {large 1-q_itemsets};
for ($k = 2; L_{k-1} \neq \emptyset; k++$) do
begin
    According to Definition 2 described in
    Section 3.2, generate $C_k$ using $L_{k-1}$.
    forall transactions $t \in$ the database do begin
      $C_t$ = subset($C_k, t$);
      /* the set of candidate q_itemsets
      contained in $t$ */
      forall candidates $d \in C_t$ do
         $d$.count++;
    end
    $L_k = \{d \in C_k \mid d.\text{count} \geq minsup\}$
end
```

Figure 11: Algorithm M_Apriori.

Comparison of LqiTid and the modified Apriori

In order to process q_items, the algorithm Apriori is modified to the version shown in Figure 11. The modified version is called M_Apriori. In the following experiments, the dataset is generated by setting $N = 1,000$, $\mid L \mid = 2,000$, and $MQ = 10$. We use $Ta.Ib.Dc$ to represent that $\mid T \mid = a$, $\mid I \mid = b$ and $\mid D \mid = c \times 1,000$.

Figure 12 shows the relative execution times for LqiTid and M_Apriori over various minimum supports. It indicates that LqiTid constantly takes much less time than M_Apriori. The reason is that LqiTid only scans the database once, whereas M_Apriori needs scan the database repeatedly. Figure 13 shows

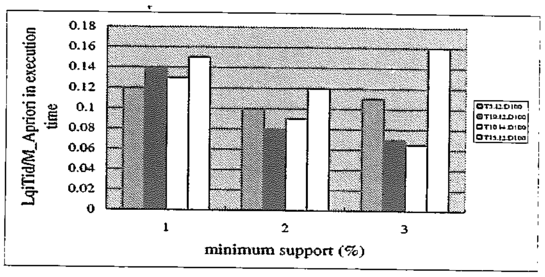

Figure 12: Execution time comparison between LqiTid and M_Apriori.

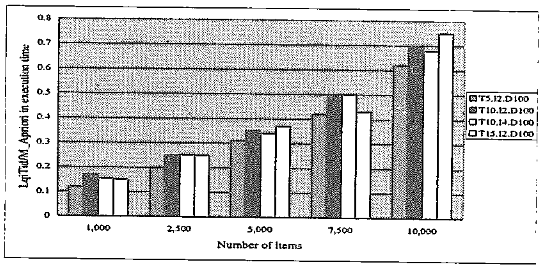

Figure 13: Relative execution time (LqiTid/M_Apriori) when the number of items increases.

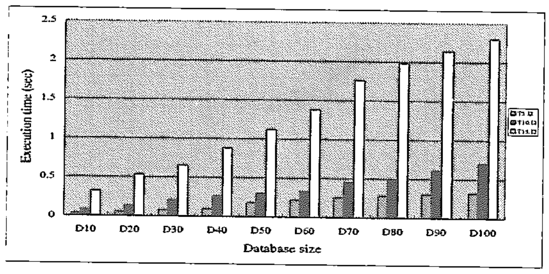

Figure 14: Performance of LqiTid when the database size increases.

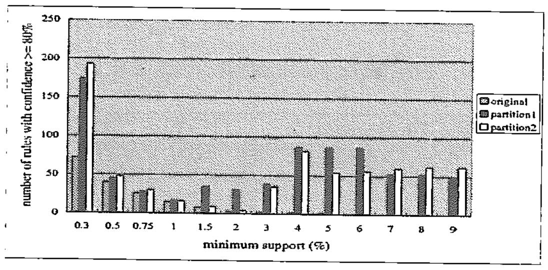

Figure 15: Number of rules generated with confidence $\geq$ 80% (T5.I2.D100).

the relative execution times when the number of items increases from 1,000 to 10,000. The minimum support is set to 1% for this experiment. It can be observed that as the number of items increases, the relative execution time increases gradually. This is because as we increase the number of items, the number of database scans required in M_Apriori decreases gradually. In the experiment for Figure 14, the minimum support is set to 1%. It shows that the execution time of LqiTid increases gradually as the database size increases. The reason is that the execution time of LqiTid is dominated by the database size.

Effects of partitions

It is the responsibility of the user to determine whether a rule is useful or not. In general, the more the number of rules generated is, the more the number of potentially interesting rules will be. Figures 15, 16 and 17 show the numbers of rules generated with confidence no less than 80%, where "original" indicates that no partition is applied on the transaction data, and "partition1" and "partition2" represent that the **PARTITION1** scheme and the **PARTITION2** scheme are used, respectively. From these experiments, we find that the numbers of rules generated form the data after partition using the **PARTITION1** or **PARTITION2** schemes are larger than that generated from the original data in all cases. Especially when the minimum support exceeds a threshold value, the number of rules generated from the original data is very few, whereas there is still many rules generated from the data after partition using the **PARTITION1** or **PARTITION2** schemes. The threshold values in Figures 15, 16 and 17 are 2%, 8% and 4%, respectively. It is interesting to note that the **PARTITION2** scheme performs better than the **PARTITION1** scheme when the minimum supports are between 1.5% and 6% in Figure 15, between 5% and 9% in Figure 16, and between 5% and 7% in Figure 17.

5 Conclusions

In this paper, we examine the issue of mining quantitative association rules in a large database of sales transactions. When purchased quantities are considered, most of the supports for items associated with their purchased quantities may be low, and the number of potentially interesting association rules discovered may be few. In order to discover more potentially interesting rules, we present two partition algorithms to partition all the possible quantities into intervals for each item. As a result, we could obtain a quantitative association rule such as "80% of customers who buy two or three loaves of bread also buy two or three bottles of milk". We also propose an efficient mechanism to discover all the large q_itemsets from the partition result. Experimental results show that the performance of LqiTid greatly outperforms that of the modified Apriori. Moreover, the number of potentially interesting association rules discovered from our partition result is larger than that of rules discovered from the original data in all cases, which demonstrates the significance of our work.

Acknowledgement

This work was partially supported by the Republic of China National Science Council under Contract No. NSC 89-2213-E-159-002.

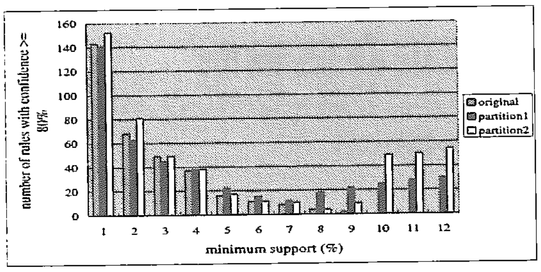

Figure 16: Number of rules generated with confidence $\geq$ 80% (T10.I2.D100).

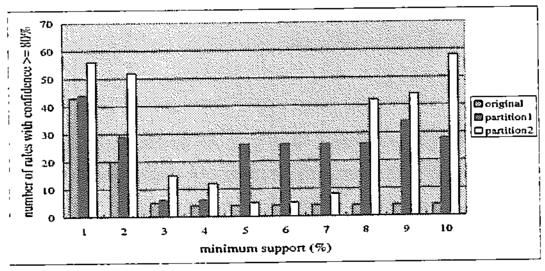

Figure 17: Number of rules generated with confidence $\geq$ 80% (T15.I2.D100).

References

[1] R. Agrawal, S. Ghosh, T Imielinski, B. Iyer, and A. Swami, An Interval Classifier for Database Mining Applications, *Proceedings of the VLDB Conference,* (1992) 560-573.

[2] R. Agrawal, T Imielinski, and A. Swami, Database Mining: A Performance Perspective, *IEEE Transactions on Knowledge and Data Engineering*, (1993) 914-925.

[3] R. Agrawal, T Imielinski, and A. Swami, Mining Association Rules between Sets of Items in Large Databases, *Proceedings of ACM SIGMOD*, (1993) 207-216.

[4] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, *Proceedings of the VLDB Conference,* (1994) 487-499.

[5] Y. Cai, N. Cercone, and J. Han, An attribute-Oriented Approach for Learning Classification Rules from Relational Databases, *Proceedings of the IEEE International Conference on Data Engineering,* (1990) 281-288.

[6] M.S. Chen, J. Han, and P.S. Yu, Data Mining: An Overview from a Database Perspective, *IEEE Transactions on Knowledge and Data Engineering,* (1996) 866-883.

[7] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining,* AAA I/MIT Press, (1996).

[8] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, Mining Optimized Association Rules for Numeric Attributes, *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems,* (1996).

[9] J. Han, Y. Cai, and N. Cercone, Data-Driven Discovery of Quantitative Rules in Relational Databases, *IEEE Transactions on Knowledge and Data Engineering,* (1993) 29-40.

[10] J. Han and Y. Fu, Discovery of Multiple-Level Association Rules from Large Databases, *Proceedings of the VLDB Conference,* (1995) 420-431.

[11] X. Hu and N. Cercone, Mining Knowledge Rules from Databases: A Rough Set Approach, *Proceedings of the IEEE International Conference on Data Engineering,* (1996) 96-106.

[12] B. Lent, A. Swami, and J. Widom, Clustering Association Rules, *Proceedings of the IEEE International Conference on Data Engineering,* (1997) 220-231.

[13] R. Ng and J. Han, Efficient and Effective Clustering Method for Spatial Data Mining, *Proceedings of the VLDB Conference,* (1994) 144-155.

[14] J.S. Park, M.S. Chen, and P.S. Yu, Using a Hash-Based Method with Transaction Trimming for Mining Association Rules, *IEEE Transactions on Knowledge and Data Engineering*, 9(5), (1997) 813-825.

[15] G. Piatetsky-Shapiro and W.J. Frawley, *Knowledge Discovery in Databases,* AAAI/MIT Press, (1991).

[16] R. Rastogi and K. Shim, Mining Optimized Association Rules with Categorical and Numeric Attributes, *Proceedings of the IEEE International Conference on Data Engineering,* (1998) 503-512.

[17] A. Savasere, E. Omiecinski, and S. Navathe, An Efficient Algorithm for Mining Association Rules in Large Databases, *Proceedings of the VLDB Conference,* (1995) 432-444.

[18] R. Srikant and R. Agrawal, Mining Generalized Association Rules, *Proceedings of the VLDB Conference,* (1995) 407-419.

[19] R. Srikant and R. Agrawal, Mining Quantitative Association Rules in Large Relational Tables, *Proceedings of the ACM SIGMOD,* (1996) 1-12.