

Short Paper

Mining Quantitative Association Rules in a Large Database of Sales Transactions

PAURAY S. M. TSAI AND CHIEN-MING CHEN*

Department of Information Management

Ming Hsin Institute of Technology

Hsinchu, Taiwan 304, R.O.C.

E-mail: paaray@mis.mhit.edu.tw

**Silicon Integrated Systems Corp.*

Hsinchu, Taiwan 300, R.O.C.

E-mail: allen@sis.com.tw

Previous studies on mining association rules focus on discovering associations among items without considering the relationships between items and their purchased quantities. However, exploring associations among items associated with their purchased quantities may discover information useful to improve the quality of business decisions. In this paper, we investigate the issue of mining quantitative association rules in a large database of sales transactions. When purchased quantities are considered, the supports of items associated with their purchased quantities may decrease drastically, and the number of potentially interesting association rules discovered will 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 partitioned data. Experimental results show that by our approach, the total execution time can be reduced significantly. Moreover, the potentially interesting rule discovered from the partitioned data can be considered to be a kind of generalized association rule. The generalized association rule is useful in marketing, business management and decision making, especially when the information from the rules generated from the original data is limited.

Keywords: data mining, quantitative association rule, partition scheme, large q -itemset generation, performance study

1. INTRODUCTION

Data mining has attracted much attention in database communities because of its wide applicability [1, 2, 5, 6, 8, 10]. One major application area of data mining is mining association rules among items in a large database of sales transactions [3, 9, 14]. Specifically, given a set of transactions, where each transaction consists of a set of items, an

Received February 2, 2000; revised July 24, 2000; accepted September 26, 2000.
Communicated by Arbee L. P. Chen.

association rule is an expression denoted by $X \Rightarrow Y$, 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 *support* of an itemset is defined to be the ratio of the number of transactions containing this itemset to the total number of transactions in the database. The problem of mining association rules can be decomposed into two subproblems. First, all itemsets whose supports are not less than a user-specified *minimum support* are identified. Each such itemset is referred to as a *large itemset*. Second, the association rules whose confidences are not less than a user-specified *minimum confidence* are generated from these large itemsets. The *confidence* of a rule $X \Rightarrow Y$ is defined to be the ratio of the supports of itemset $X \cup Y$ to itemset X . Given a large itemset Z , all rules of the form $X \Rightarrow Y$ satisfying $X \cup Y = Z$, $X \cap Y = \phi$, 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 as the AIS algorithm, was first explored in [3]. The AIS algorithm requires repeated scanning of the database. It uses the large itemsets discovered in the previous pass as the basis for generating 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 to be not large itemsets.

In [4], the Apriori and AprioriTid algorithms were proposed for efficiently mining association rules. Unlike the AIS algorithm, these two algorithms generate the candidate itemsets 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 as well. By this heuristic, the Apriori and AprioriTid algorithms can generate a much smaller number of candidate itemsets than the AIS algorithm.

Another effective algorithm for candidate set generation, called DHP, was proposed in [12]. By utilizing the hash technique, DHP can efficiently generate all large itemsets and, so, effectively reduce the size of the transaction database. The performance study in [12] shows that the number of candidate 2-itemsets generated by DHP is smaller than that generated 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 [9]. 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. A similar idea of extracting generalized association rules using a taxonomy was presented in [15].

The issue of mining optimized association rules for numerical and categorical attributes was investigated in [7, 13]. 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 [7], only a single optimal interval for a single numerical attribute can be determined. In [13] the optimized association rules problem was generalized to contain a number of uninstantiated attributes.

The problem of mining association rules in large relational databases was introduced in [16]. 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 resulting 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 [11], which combines similar “adjacent” association rules to form general rules. [11] proposed a geometric-based algorithm to perform 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 rule is useful for improving marketing strategies.

When purchased quantities are considered, the supports of items associated with their purchased quantities may decrease drastically, and 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 so that the support for each interval is not less than the minimum support. We also propose a mechanism, called LqiTid, to discover all large itemsets in the partitioned data. By recording the identifiers of transactions containing a large itemset, we can discover all large itemsets by scanning the database only once. Experimental results show that the performance of LqiTid greatly outperforms the modified DHP algorithm. Moreover, the number of potentially interesting association rules discovered from the partitioned data is larger than that of rules discovered from the original data in all cases. Especially when the minimum support exceeds a critical value, almost no rules can be discovered from the original data, but there are still some potentially interesting association rules that can be discovered from the partitioned data.

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 $\langle i, q \rangle$, 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 number of transactions containing this q_item divided 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 may be very low. If most q_items have low support, 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 quantities into intervals for each item and map each quantity to an integer which represents its corresponding interval.

Let $\{\langle i, q_1 \rangle, \langle i, q_2 \rangle, \dots, \langle i, q_n \rangle\}$ be the set of q_items in DB which have the same item i , and $q_1 < q_2 < \dots < q_n$. Assume that c_1, c_2, \dots , and c_n are the numbers of transactions containing $\langle i, q_1 \rangle, \langle i, q_2 \rangle, \dots$, and $\langle i, q_n \rangle$, respectively. In the following, we consider two partition methods: the **PARTITION1** method, shown in Fig. 1, and the **PARTITION2** method, shown in Fig. 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 generate the interval $[q_1..q_{k_1}]$. 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 generate the interval $[q_{(k_1+1)} .. q_{k_2}]$. The remaining intervals are similarly computed. 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 which we represent as $[q_m]$. Note that $\frac{c_m}{T}$ denotes the support for the $q_item \langle i, q_m \rangle$. Assume that the supports for $q_items \langle i, q_j \rangle$ and $\langle i, q_k \rangle$ are not less than s , and those for $q_items \langle i, q_{(j+1)} \rangle, \langle i, q_{(j+2)} \rangle, \dots$, and $\langle i, q_{(k-1)} \rangle$ are less than s . Then the interval $[q_{(j+1)} .. q_{(k-1)}]$ is generated. 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 $\{\langle i, 1 \rangle, \langle i, 2 \rangle, \langle i, 3 \rangle, \langle i, 4 \rangle, \langle i, 5 \rangle\}$ be the set of q_items in database DB , which have the same item i , and let 50, 30, 100, 20, and 40 be the numbers of transactions containing $\langle i, 1 \rangle, \langle i, 2 \rangle, \langle i, 3 \rangle, \langle i, 4 \rangle$, and $\langle i, 5 \rangle$, 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, $q_item \langle i, 3 \rangle$ will be mapped to the generalized form $\langle i, 2 \rangle$ (i.e., $\langle i, [2..3] \rangle$). 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, $q_item \langle i, 3 \rangle$ will be mapped to the generalized form $\langle i, 3 \rangle$ (i.e., $\langle i, [3] \rangle$).

```

/* PARTITION1 */
j = 1
cnt = 0
Vj.first = q1; /* Vj.first represents the first value in the interval Vj */
for (k = 1; k ≤ n; k++) do
    cnt = cnt + ck;
    if (  $\frac{cnt}{T} \geq s$  ) then
        begin
            Vj.last = qk; /* Vj.last represents the last value in the interval Vj */
            j++;
            cnt = 0;
            if (k < n) then Vj.first = q(k+1);
        end
    else if (k = n) then
        if (j > 1) then Vj-1.last = qn;
        else Vj.last = qn;
    end for

```

Fig. 1. The PARTITION1 method.

```

/* PARTITION2 */
j = 1
Vj.first = q1; /* Vj.first represents the first value in the interval Vj */
flag = 0; /* flag = 0 indicates that at present there is only a value in interval Vj */
for (k = 1; k ≤ n; k++) do
    if (  $\frac{c_k}{T} \geq s$  ) then
        begin
            if (flag = 0) then Vj.last = qk;
            /* Vj.last represents the last value in the interval Vj */
        else
            begin
                Vj.last = q(k-1);
                j++;
                Vj.first = qk;
                Vj.last = qk;
            end
        end
        j++;
        Vj.first = q(k+1);
        flag = 0;
    end
    else if (k = n) then Vj.last = qn;
        else if (flag = 0) then flag = 1;
    end for

```

Fig. 2. The PARTITION2 method.

After the partition process, each q_item in each transaction is mapped to a generalized form. Let DB' be the transaction database after performing the partition process on database DB , and let X be the set of q_items in DB' . A *quantitative association rule* is of the form $Y \Rightarrow Z$, where $Y \subset X$, $Z \subset X$, and $Y \cap Z = \emptyset$. The rule $Y \Rightarrow Z$ holds in the transaction database DB' with *confidence* c if $c\%$ of the transactions in DB' that contain Y also contain Z . The rule $Y \Rightarrow Z$ 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 not less than the minimum support. A q_itemset of size k is called a k -q_itemset.

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

1. Partition quantities into intervals for each item using the partition methods introduced in this section.
2. Find all large q_itemsets using the algorithm **LqiTid** presented in the next section.

After discovering all large q_itemsets, the quantitative association rules can be extracted in a straightforward manner. Given a large q_itemset X , all rules of the form $Y \Rightarrow Z$ satisfying $Y \cup Z = X$, $Y \cap Z = \emptyset$, and the minimum confidence constraint will be generated.

3. LARGE Q_ITEMSET GENERATION

In this section, we propose a mechanism that can discover all of the large q_itemsets by scanning the database only once. **Example 2** is used to illustrate our approach.

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

Fig. 3. Transaction database DB .

quantity	integer
1	1
2..3	2

quantity	integer
1	1
2	2
3..4	3

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

quantity	integer
1	1

quantity	integer
5	1

quantity	integer
1	1
2..3	2

quantity	Integer
1..2	1
3..5	2

Fig. 4. The mapping information.

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>

Fig. 5. Transaction database DB' .

3.1 Information for Discovering Large Q_itemsets

Let $TS(\{x\})$ be the set of TIDs of 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 of transactions containing the two q_items x_1 and x_2 , can be obtained by performing a 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 $x_1, x_2, \dots,$ and x_k are q_items. $TS(\{x_1, x_2, \dots, x_k\})$ is the set of TIDs of the transactions containing all of the q_items in the q_itemset $\{x_1, x_2, \dots, x_k\}$. $SP(\{x_1, x_2, \dots, x_k\})$, which represents the number of TIDs in $TS(\{x_1, x_2, \dots, x_k\})$, is defined as:

$$\begin{aligned}
 SP(\{x_1, x_2, \dots, x_k\}) &= \text{Card}(TS(\{x_1, x_2, \dots, x_k\})) \\
 &= \text{Card}(TS(\{x_1\}) \cap TS(\{x_2\}) \cap \dots \cap TS(\{x_k\}))
 \end{aligned}$$

where $\text{Card}(S)$ denotes the cardinality of set S .

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

Fig. 6. The information for discovering large q -itemsets.

3.2 Algorithm LqiTid

In this subsection, we introduce algorithm **LqiTid** for efficient generation of large q -itemsets. Let T be the total number of transactions in the transaction database, and define minsup as

$$\text{minsup} = \lceil T \times \text{minimum support} \rceil.$$

Lemma 1: If $SP(\{x_1, x_2, \dots, x_k\})$ is not less than minsup , then $(\{x_1, x_2, \dots, x_k\})$ is a large k - q -itemset.

Lemma 2: If the q -itemset $(\{x_1, x_2, \dots, x_k\})$ is a large k - q -itemset, $k \geq 2$, then any proper subset of the q -itemset is also a large q -itemset.

According to **Lemma 2**, candidate k - q -itemsets can be generated from the set of large $(k-1)$ - q -itemsets, L_{k-1} . This 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 $\text{item}(x[j])$ be the item value in q -item $x[j]$. For the q -itemset $\{x[1], x[2], \dots, x[k-1]\}$, we assume that $\text{item}(x[1]) < \text{item}(x[2]) < \dots < \text{item}(x[k-1])$.

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

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

The generation of candidate k -q_itemsets is similar to the generation of candidate k -itemsets in Apriori [4]. The TS value of candidate k -q_itemset $\{x_p[1], x_p[2], \dots, x_p[k-1], x_q[k-1]\}$ can be computed as

$$TS(\{x_p[1], x_p[2], \dots, x_p[k-1], x_q[k-1]\}) = TS(x_p) \cap TS(x_q)$$

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

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

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

```

/* 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 */
 $L_1 = \{x \mid x \text{ is a q\_item and } SP(\{x\}) \geq minsup\}$ 
/* Large  $k$ -q_itemset generation phase */
for ( $k = 2$ ;  $|L_{k-1}| > 1$ ;  $k++$ ) do begin
    According to Definition 2, generate  $C_k$  using  $L_{k-1}$ .
    for all 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

```

Fig. 7. Algorithm **LqiTid**.

In the following we use the transaction database DB' shown in Fig. 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 Fig. 6. According to **Lemma 1**, L_1 is the set of q_items satisfying $SP \geq 2$, as shown in Fig. 8. Then the set of large 2-q_itemsets, L_2 , is determined using the information in Fig. 8. For example, $\{ \langle A, 1 \rangle, \langle B, 1 \rangle \}$ is a candidate 2-q_itemset. However, it is not a large 2-q_itemset since $TS(\{ \langle A, 1 \rangle, \langle B, 1 \rangle \}) = \{5\}$ and $SP(\{ \langle A, 1 \rangle, \langle B, 1 \rangle \}) = 1$, which is less than $minsup$. $\{ \langle A, 2 \rangle, \langle C, 2 \rangle \}$ is a large 2-q_itemset since $TS(\{ \langle A, 2 \rangle, \langle C, 2 \rangle \}) = \{4, 10, 15\}$ and $SP(\{ \langle A, 2 \rangle, \langle C, 2 \rangle \}) = 3$, which is greater than $minsup$. Fig. 9 shows the information for

L_2 , from which L_3 is next determined. For example, $\{\langle B, 2 \rangle, \langle C, 2 \rangle, \langle G, 2 \rangle\}$ is a candidate 3-q_itemset generated from large 2-q_itemsets $\{\langle B, 2 \rangle, \langle C, 2 \rangle\}$ and $\{\langle B, 2 \rangle, \langle G, 2 \rangle\}$. However, it is not a large 3-q_itemset since $TS(\{\langle B, 2 \rangle, \langle C, 2 \rangle, \langle G, 2 \rangle\}) = TS(\{\langle B, 2 \rangle, \langle C, 2 \rangle\}) \cap TS(\{\langle B, 2 \rangle, \langle G, 2 \rangle\}) = \{14\}$ and $SP(\{\langle B, 2 \rangle, \langle C, 2 \rangle, \langle G, 2 \rangle\}) = 1$, which is less than $minsup$. $\{\langle B, 2 \rangle, \langle C, 2 \rangle, \langle F, 1 \rangle\}$ is a large 3-q_itemset since $TS(\{\langle B, 2 \rangle, \langle C, 2 \rangle, \langle F, 1 \rangle\}) = TS(\{\langle B, 2 \rangle, \langle C, 2 \rangle\}) \cap TS(\{\langle B, 2 \rangle, \langle F, 1 \rangle\}) = \{10, 11\}$ and $SP(\{\langle B, 2 \rangle, \langle C, 2 \rangle, \langle F, 1 \rangle\}) = 2$, which is equal to $minsup$. Fig. 10 shows the information for L_3 . Since $C_4 = \emptyset$, $L_4 = \emptyset$ and the mining process terminates.

q_item	TS	SP
$\langle A, 1 \rangle$	{5, 12, 14}	3
$\langle A, 2 \rangle$	{4, 10, 15}	3
$\langle B, 1 \rangle$	{1, 4, 5, 8}	4
$\langle B, 2 \rangle$	{10, 11, 12, 13, 14}	5
$\langle B, 3 \rangle$	{6, 7, 9}	3
$\langle C, 1 \rangle$	{1, 2, 3, 6, 7, 8, 13}	7
$\langle C, 2 \rangle$	{4, 10, 11, 14, 15}	5
$\langle D, 1 \rangle$	{2, 7}	2
$\langle F, 1 \rangle$	{1, 7, 10, 11}	4
$\langle F, 2 \rangle$	{3, 9, 14}	3
$\langle G, 1 \rangle$	{2, 3, 4, 12}	4
$\langle G, 2 \rangle$	{1, 8, 13, 14, 15}	5

Fig. 8. The information for large 1-q_itemsets.

large 2-q_itemset	TS	SP
$\{\langle A, 1 \rangle, \langle B, 2 \rangle\}$	{12, 14}	2
$\{\langle A, 2 \rangle, \langle C, 2 \rangle\}$	{4, 10, 15}	3
$\{\langle B, 1 \rangle, \langle C, 1 \rangle\}$	{1, 8}	2
$\{\langle B, 1 \rangle, \langle G, 2 \rangle\}$	{1, 8}	2
$\{\langle B, 2 \rangle, \langle C, 2 \rangle\}$	{10, 11, 14}	3
$\{\langle B, 2 \rangle, \langle F, 1 \rangle\}$	{10, 11}	2
$\{\langle B, 2 \rangle, \langle G, 2 \rangle\}$	{13, 14}	2
$\{\langle B, 3 \rangle, \langle C, 1 \rangle\}$	{6, 7}	2
$\{\langle C, 1 \rangle, \langle D, 1 \rangle\}$	{2, 7}	2
$\{\langle C, 1 \rangle, \langle F, 1 \rangle\}$	{1, 7}	2
$\{\langle C, 1 \rangle, \langle G, 1 \rangle\}$	{2, 3}	2
$\{\langle C, 1 \rangle, \langle G, 2 \rangle\}$	{1, 8, 13}	3
$\{\langle C, 2 \rangle, \langle F, 1 \rangle\}$	{10, 11}	2
$\{\langle C, 2 \rangle, \langle G, 2 \rangle\}$	{14, 15}	2

Fig. 9. The information for large 2-q_itemsets.

large 3-q_itemset	TS	SP
$\{\langle B, 2 \rangle, \langle C, 2 \rangle, \langle F, 1 \rangle\}$	{10, 11}	2
$\{\langle B, 1 \rangle, \langle C, 1 \rangle, \langle G, 2 \rangle\}$	{1, 8}	2

Fig. 10. The information for large 3-q_itemsets.

Given a large q -itemset X , $Y \Rightarrow Z$ is a quantitative association rule if $Y \cup Z = X$, $Y \cap Z = \emptyset$, and the minimum confidence constraint is satisfied. The confidence of $Y \Rightarrow Z$ is determined by $\frac{SR(X)}{SR(Y)}$. Continuing with the above example, assume that the minimum confidence is 65%. We can obtain the following quantitative association rules, using the mapping information in Fig. 4:

$\{ \langle A, 1 \rangle \} \Rightarrow \{ \langle B, 2 \rangle \}$ (67%)
 $\{ \langle A, [2..3] \rangle \} \Rightarrow \{ \langle C, [3..4] \rangle \}$ (100%)
 $\{ \langle B, [3..4] \rangle \} \Rightarrow \{ \langle C, [1..2] \rangle \}$ (67%)
 $\{ \langle D, 1 \rangle \} \Rightarrow \{ \langle C, [1..2] \rangle \}$ (100%)
 $\{ \langle B, 2 \rangle, \langle C, [3..4] \rangle \} \Rightarrow \{ \langle F, 1 \rangle \}$ (67%)
 $\{ \langle B, 2 \rangle, \langle F, 1 \rangle \} \Rightarrow \{ \langle C, [3..4] \rangle \}$ (100%)
 $\{ \langle C, [3..4] \rangle, \langle F, 1 \rangle \} \Rightarrow \{ \langle B, 2 \rangle \}$ (100%)
 $\{ \langle B, 1 \rangle, \langle C, [1..2] \rangle \} \Rightarrow \{ \langle G, [3..5] \rangle \}$ (100%)
 $\{ \langle B, 1 \rangle, \langle G, [3..5] \rangle \} \Rightarrow \{ \langle C, [1..2] \rangle \}$ (100%)
 $\{ \langle C, [1..2] \rangle, \langle G, [3..5] \rangle \} \Rightarrow \{ \langle B, 1 \rangle \}$ (67%)

4. EXPERIMENTAL RESULTS

To assess the performance of LqiTid, we conducted several experiments on a 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 DHP [12]. 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.

Table 1. Parameters

$ D $	Number of transactions
$ T $	Average size of the transactions
$ I $	Average size of the maximal potentially large q -itemsets
$ L $	Number of maximal potentially large q -itemsets
N	N Number of items
MQ	Maximum value of the quantities

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 $|I|$. q -itemsets in L are generated as follows: Items in the first q -itemset are randomly chosen 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 $q_item \langle x, q \rangle$ 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 q is also retained. 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 $|T|$. Each transaction is generated as follows. First, a potentially large $q_itemset$ is randomly chosen from L , whose size is less than or equal to the transaction size. To model the fact 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.

Comparison of LqiTid and the modified DHP algorithms

The DHP algorithm [12] is modified to the version called M_DHP , which can process q_items . We replace only items in DHP by q_items in M_DHP . Fig. 11 shows the relative execution times for LqiTid and M_DHP over various minimum supports. In the experiment, the hash table $|H_2|$ is chosen to be a quarter of C_2^N , where N is the number of q_items . This value for $|H_2|$ was shown to have good overall performance in [12]. From Fig. 11, it can be seen that LqiTid has better performance than M_DHP . This is because M_DHP needs to scan the database to count supports for candidate k - $q_itemsets$ in each pass and trim the database to generate a reduced database.

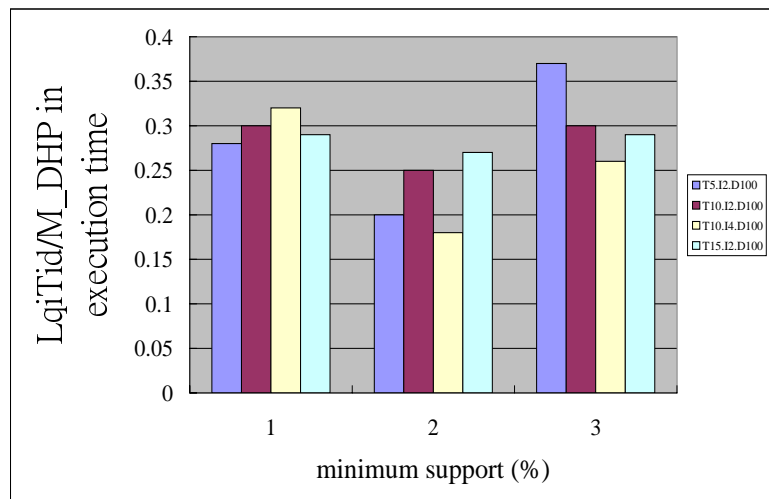


Fig. 11. Comparison of execution time for LqiTid and M_DHP .

Effects of partitions

It is the responsibility of the user to determine whether a rule is useful or not. In general, the more rules are generated, the more potentially interesting rules will result. In the following experiments, the idea of **equi-depth** [16] is used as the basis of another scheme called "PARTITION3". Rather than the approach of [16] for relational tables, we consider the minimum support constraint in the transaction database. Fig. 12 shows the numbers of rules generated with confidence not less than 80%, where "original" indicates that the test data is the original transaction data, and "partition1," "partition2" and "partition3" represent the respective schemes. From these experiments, we find that the numbers of rules generated from the partitioned data using the **PARTITION1**, **PARTITION2** or **PARTITION3** 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 low, whereas many rules are still generated from the partitioned data using the **PARTITION1**, **PARTITION2** or **PARTITION3** schemes. The threshold values in Fig. 12 is 2%, 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%. We also find that the **PARTITION1** or **PARTITION2** scheme performs better than the **PARTITION3** scheme in all cases. Since the numbers of transactions with different quantities for the same item may vary greatly, and the intervals resulting from partitioning should have supports greater than the minimum support, the numbers of transactions in the partitioned intervals may differ greatly.

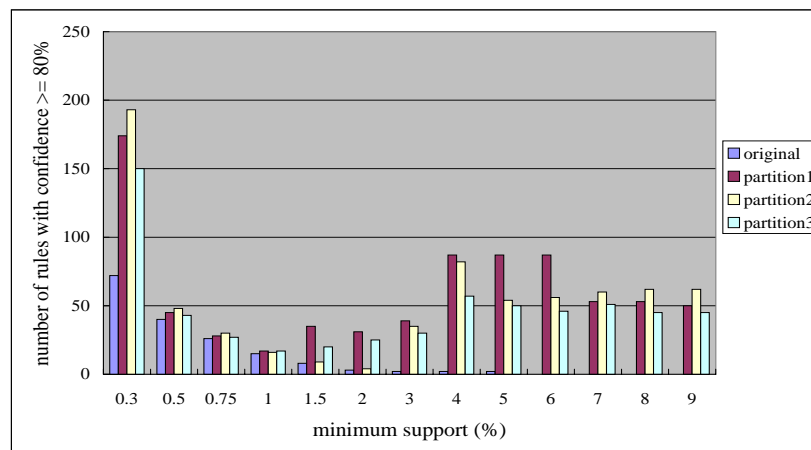


Fig. 12. Number of rules generated with confidence $\geq 80\%$ (TS. IZ. D100).

We do not guarantee that all the rules generated from the partitioned data are useful for all users. However, the potentially interesting rule generated from the partitioned data can be considered to be a kind of generalized association rule [9, 15]. The generalized association rules discovered from the partitioned data are useful in marketing, business

management and decision making. Discovering generalized association rules from partitioned data is significant, especially when we cannot obtain enough information from the rules generated from the original data.

5. CONCLUSIONS

In this paper, we investigate the issue of mining quantitative association rules in a large database of sales transactions. When purchased quantities are considered, the supports of items associated with their purchased quantities may decrease drastically, and the number of potentially interesting association rules discovered will be few. In order to discover more potentially interesting rules, we present two algorithms to partition all of the possible quantities into intervals for each item. By our approach, we could obtain quantitative association rules 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 of the large q -itemsets from the partitioned data. Experimental results show that the performance of Lq_iTid significantly outperforms that of the modified DHP. Moreover, the generalized association rules are useful in marketing, business management and decision making, especially when we cannot obtain enough information from the rules generated from the original data.

REFERENCES

1. R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami, “An interval classifier for database mining applications,” in *Proceedings of the VLDB Conference*, 1992, pp. 560-573.
2. R. Agrawal, T. Imielinski, and A. Swami, “Database mining: A performance perspective,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 6, 1993, pp. 914-925.
3. R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases,” in *Proceedings of ACM SIGMOD*, 1993, pp. 207-216.
4. R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” in *Proceedings of the VLDB Conference*, 1994, pp. 487-499.
5. Y. Cai, N. Cercone, and J. Han, “An attribute-oriented approach for learning classification rules from relational databases,” in *Proceedings of the IEEE International Conference on Data Engineering*, 1990, pp. 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*, Vol. 8, No. 6, 1996, pp. 866-883.
7. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, “Mining optimized association rules for numeric attributes,” in *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1996, pp. 182-191.
8. J. Han, Y. Cai, and N. Cercone, “Knowledge discovery in databases: an attribute-oriented approach,” in *Proceedings of the VLDB Conference*, 1992, pp. 547-559.
9. J. Han and Y. Fu, “Discovery of multiple-level association rules from large databases,” in *Proceedings of the VLDB Conference*, 1995, pp. 420-431.

10. X. Hu and N. Cercone, "Mining knowledge rules from databases: a rough set approach," in *Proceedings of the IEEE International Conference on Data Engineering*, 1996, pp. 96-106.
11. B. Lent, A. Swami, and J. Widom, "Clustering association rules," in *Proceedings of the IEEE International Conference on Data Engineering*, 1997, pp. 220-231.
12. 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*, Vol. 9, No. 5, 1997, pp. 813-825.
13. R. Rastogi and K. Shim, "Mining optimized association rules with categorical and numeric attributes," in *Proceedings of the IEEE International Conference on Data Engineering*, 1998, pp. 503-512.
14. A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases," in *Proceedings of the VLDB Conference*, 1995, pp. 432-444.
15. R. Srikant and R. Agrawal, "Mining generalized association rules," in *Proceedings of the VLDB Conference*, 1995, pp. 407-419.
16. R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," in *Proceedings of the ACM SIGMOD*, 1996, pp. 1-12.

Pauray S. M. Tsai (蔡秀滿) received the B.S. and M.S. degrees in Computer Science and Information Engineering from National Chiao Tung University, Taiwan, in 1990 and 1992, respectively, and the Ph.D. degree in Computer Science from National Tsing Hua University, Taiwan, in 1996. Currently, she is an associate professor in the Department of Information Management, Ming Hsin Institute of Technology, Taiwan. Her research interests include multidatabase systems, object-oriented database systems, and data mining.

Chien-Ming Chen (陳健銘) received the B.S. degree in Computer Science and Information Engineering from National Chiao Tung University, Taiwan, in 1990, and the Ph.D. degree in Computer Science from National Tsing Hua University, Taiwan, in 1996. Currently, he is a senior engineering in the Silicon Integrated Systems Corp., Taiwan. His research interests include computer architecture, parallel processing, and data mining.