Discovering Phenomena - Correlations among Association Rules

Yi-Hung Wu, Maggie Yu-Chieh Chang, Arbee L. P. Chen[*]
Department of Computer Science
National Tsing Hua University
[*]*Department of Computer Science*
National Chengchi University
Taiwan, R.O.C.
alpchen@cs.nccu.edu.tw

Abstract

With the growth of various data types, mining useful association rules from large databases has been an important research topic nowadays. Previous works focus on the attributes of data items to derive a variety of association rules. In this paper, we use the attributes of transactions to organize the data as a *multiple-attribute hierarchical tree* where the *multiple-attribute association rules* can be efficiently derived. Furthermore, we store the derived rules as a *frequent hierarchical tree* and allow users to specify various types of queries for finding interesting correlations named *phenomena* among the rules. We then make experiments to evaluate the performance of our approach.

Keywords: data mining, association rule, data warehousing, correlation, query.

1 Introduction

Recently, the explosive growth of marketing data has brought urgent needs for new mining techniques. One of the important techniques is to mine *association rules* from large transaction database [1] [3] [4] [9] [13] [14] [15] [17] [18] [19]. The following are some terminologies used in this field. In a transaction database, each transaction consists of a transaction date and a set of items purchased. A set of items is called an *itemset*. The *support* of an itemset is the percentage of transactions that contain this itemset. An itemset is *large* when its support is larger than a user-defined threshold (called *minimum support*). For each large itemset, the association rules that cover this set of items can be easily derived. An *association rule* describes the associations among the data, indicating the purchasing behaviors of most customers. For instance, a manager might be interested in the rule — if a customer buys bread, she will also probably buy milk.

Most of the researchers derive the association rules that do not consider the layered concepts on items [2][5], e.g., people who buy food often buy drink at the same time (both are more general concepts). Recent works view the attribute values as the layered concepts and set up proper values of minimum supports to find the associations among the concepts at the same or different levels. For example, when we buy bread, soft drinks would be probably bought together. Srikant and Agrawal [11] employ a constant value of minimum support to mine the association rules over all the layers. On the other hand, Han and Fu [7] utilize variable values of minimum supports for different layers. In that approach, the higher the layer is, the larger the minimum support will be. These works claim that if we extend the data domain to include the layered concepts on items, the derived rules will reveal more specific and concrete information to us.

In addition to the layered concepts on items, other interesting rules can be found by dividing the transactions into partitions based on item types they contain. For instance, in summer the ice cream is a hot seller and the database is overwhelmed by the transactions that contain this item type. No matter how the sales of the air conditioner increase, it cannot be large due to the large sales of ice cream.

In this paper, we combine the above two ideas and consider the attributes of transactions to discover more comprehensive knowledge from data. At first, we recursively divide the transactions into groups by their attributes to construct a *multiple-attribute hierarchical tree* (called *MAH-tree*). After that, we focus on the information kept on this tree to derive association rules (called *MA association rules*) for each group of transactions. The derived rules are organized as a *frequent hierarchical tree* (called *FH-tree*), which allows users to specify queries for finding *correlations* among the rules (called *phenomena*).

In the following, we illustrate the phenomena in the real world and their relationships with the MA association rules. From the MAH-tree, we may derive the MA association rules like "in the U.S., if a boy buys a BB gun, he will also probably buys BB shots" and "in the U.S., if a women buys a first-aid box, she will also probably buys

*Contact Author

bandages". In this circumstance, the FH-tree may contain the large itemsets {BB gun, BB shot} and {first-aid box, bandage} with certain attribute values, e.g., in summer. By analyzing the FH-tree, we may discover the correlations between the two large itemsets, such as "in the U.S., a boy and a woman often buy a BB gun and a first-aid box respectively in the same seasons". It is easy for users to get this kind of phenomena by specifying a query like "in the U. S., what kinds of purchasing behavior do a boy and a woman often have at the same time?" To sum up, a phenomenon indicates the significant correlation among large itemsets within certain attribute domains.

Grahne and et al. [6] consider the problem of finding *circumstances* (i.e., time and locations) where a pattern holds. The circumstances for a pattern are first discovered and then the itemsets that match the pattern under the circumstances are derived. For example, the time period when ice cream has a large sale is first found and then the other itemsets that also have large sales during the same period. On the other hand, the itemsets that match a pattern under a circumstance can also be found. After that, more circumstances under which these itemsets also match the pattern are derived. For instance, for each itemset bought frequently in fall, other seasons in which the itemset is also bought frequently are derived. Compared with this paper, we can find the same results from the FH-tree and further obtain their correlations.

Another approach named *data warehousing* [16] proposes the concept of a *data cube* to analyze multi-dimensional data. A data cube stores all the aggregated values of data in the form of a multi- dimensional cube, where each dimension has its own hierarchical skeleton [8] [20]. However, this approach essentially lacks the concept of transactions and therefore provides no sufficient information about the association among items.

The rest of this paper is organized as follows. Section 2 introduces the MAH-tree construction. The derivation of MA association rules is described in Section 3 and the approach for phenomena mining is presented in Section 4. In Section 5, the experimental results are shown. Finally, Section 6 concludes this paper.

2 MAH-Tree

From a variety of information associated with the transactions, we consider the typical attributes in our work on mining. In this section, we first introduce the basic types of attributes and then show how to construct a MAH-tree.

2.1 Basic Types of Attributes

In a transaction database, each transaction is usually associated with four typical attributes, i.e., when, who, where, and what to buy. The manager of a supermarket is always interested in the following question "at what time, what kinds of customers in which areas will buy what kinds of items?" In this paper, we consider them as the basic types of attributes.

- *Time Attribute* (TA) specifies when the transactions occur, e.g., seasons.
- *Customer Attribute* (CA) specifies who creates the transactions. In this paper, we combine gender and age of each customer to be CA, i.e., male or female at the age below 20, from 20 to 40, from 40 to 60, or above 60.
- *Location Attribute* (LA) specifies where the transactions occur. In this paper, we apply the concept of *MX-Quadtree* [12] to generate LA.
- *Item Attribute* (IA) specifies what the transactions contain. In this paper, we denote each product as a distinct symbol (i.e., item) and regard the itemsets as IA.

In the following, we briefly describe how to generate LA. At first, the positions where the transactions occur are classified into four disjoint regions with equal area. After that, the *entropy* of each region, indicating the degree of disorder in the region, is computed by the following formula [10], where A_j is one of the n positions in the region and $P(A_j)$ is the probability that a transaction occur in A_j.

$$Entropy\ (A) = -\sum_{j=1}^{n} P(A_j) \log P(A_j) \qquad (1)$$

Based on the entropy of each region, we can identify the non-uniform regions and continue to classify them into smaller regions. When there is no more non-uniform region, this process terminates and the resultant regions are numbered as LA. Take Figure 1 as an example. Only the region at the top left corner of Figure 1(a) is further classified into four smaller regions. At last, seven regions as shown in Figure 1(b) are numbered as LA.

In the following, we illustrate how to transform a database into the one with the above four attributes. For a transaction database in Table 1, we first transform it into a database with four attributes. As Table 2 shows, we use alphabets to denote IA, e.g., "a" stands for "drink", "b" means "food", etc. In the other columns, we use numbers

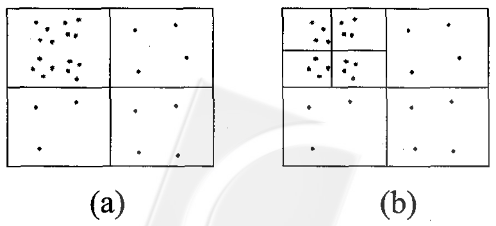


Figure 1 An example for generating LA

Table 1 A transaction database

Date	Age	Gender	Location	Products
99/4/1	25	male	McDonald in the east area	cola, French fries, ice cream
99/5/3	52	female	beach in the north area	French fries
99/10/30	31	male	movie theater in the east area	French fries, ticket, ice cream, juice, popcorn
99/12/5	13	male	movie theater in the south area	cola, French fries, ticket
…	…	…	…	…
02/6/8	65	male	MTV in downtown	cola, ticket

to denote TA (e.g., "1" means "spring"), CA, and LA, respectively.

2.2 Tree Construction

According to the four types of attributes, there are six kinds of structures for the MAH-tree, as shown in Figure 2. At first, we consider the position of three attributes in the MAH-tree. We make initial experiments to examine whether the tree structure influences the performance of tree construction and large itemset generation. Based on the results (to be detailed in Section 5.2), we take the order TA→CA→LA from top to bottom as the structure of the MAH-tree.

A MAH-tree contains four layers, where each corresponds to one of the four types of attributes. To build a MAH-tree, we recursively divide all the transactions in the database into several groups according to the attribute values. Finally, the information about transactions is stored in the leaf nodes of the MAH-tree.

Firstly, at the top layer, we divide the transactions into four groups (called *partition*) by TA and name each of them *spring tree*, *summer tree*, *fall tree*, and *winter tree*, respectively. Figure 3 is the MAH-tree of Table 2. A spring tree consists of all the transactions whose TA is spring and the other trees are built in the same way. Consequently, the database is fully represented by these trees. Each internal node on the MAH-tree keeps a vector of three fields, which refer to the values of the three attributes TA, CA and LA

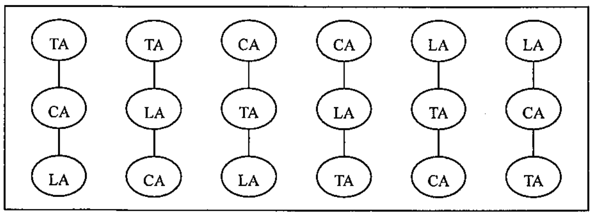


Figure 2 Six kinds of tree structures

Table 2 The transformed database of Table 1

T-ID	TA	CA	LA	IA
1	1	3	2	a b d
2	1	6	4	b
3	3	3	2	b c d e f
4	4	1	7	a b c
…	…	…	…	…
50	1	7	1	a c

associated with the transactions under this node. For example, the vector on the root of the spring tree indicates that the TA values of the transactions under this tree are spring (1), and the values of CA and LA are un-designated (00). Moreover, we also associated the number of transactions for each partition (called the *partition size*) with the corresponding internal node.

The transactions in each partition are then divided recursively by CA and LA into smaller partitions. Similarly, the vector 110 indicates that the TA is spring, the CA is a boy and the LA is un-designated. For each partition at the bottom layer, we collect the information about each item, including the number of transactions that it appears (called the *count*) and the set of the corresponding T-ID's (called the *T-ID set*), to build the leaf nodes. In this way, the count can be used to decide whether the item is large, and the T-ID set can be used to compute all the large itemsets.

3 MA Association Rule Mining

Based on the MAH-tree, we can derive the MA association rules like "in summer, if the youth buy ice skates in a shopping mall, they also buy protecting fittings," and store the large itemsets as an FH-tree for efficient phenomena mining.

Due to the various sizes of partitions on the MAH-tree, at the beginning we compute the ratio of each partition to the entire database and filter out the partitions with small ratios. After that, we respectively derive the large itemsets in each partition from the bottom to the top of the MAH-tree. For efficiency, we regard the large itemsets derived from the lower layer as the candidate itemsets for the higher layer.

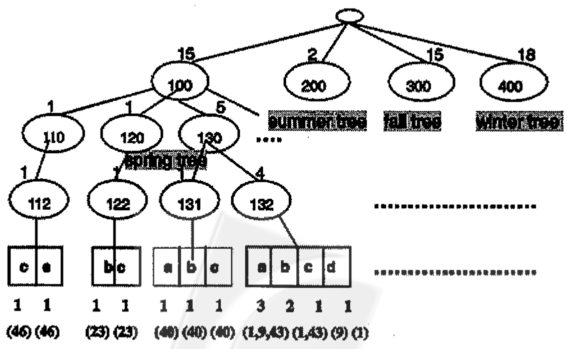


Figure 3 A MAH-tree

3.1 Size Filtering

For each layer, we define the inverse of the domain size for the corresponding attribute as the *expected ratio* and compute a threshold for size filtering (called *SF threshold*) by multiplying its expected ratio with the ones of all the layers above it. In Table 2, the expected ratio of TA, CA, and LA are 1/4, 1/8, and 1/7 respectively. The SF thresholds of three layers are 1/4, 1/4*1/8, and 1/4*1/8*1/7 respectively. The mining task on a partition can be skipped only if the following inequality fails.

$$\frac{partition\ size}{database\ size} \geq SF\ threshold \qquad (2)$$

For example, the partition under node 100 at the TA layer has 15 transactions, which meets inequality (2) (15/50 > 1/4). Therefore, we have to derive the large itemsets from this partition. On the other hand, the partition under node 200 at the same layer has 2 transactions, which contradicts inequality (2). Therefore, we do not need to derive the large itemsets in node 200. In addition to the SF threshold, we also allow users to specify a value, called the *criterion of SF threshold*, between 0.1 and 1.0 to adjust the magnitude of the SF threshold.

3.2 Mining Algorithm

After the filtering process, we start from the bottom layer of the MAH-tree to derive the large itemsets for all the qualified partitions. For each partition at the bottom layer, we use the partition size and the count kept in each leaf node to derive large-1 itemsets. After that, we generate large-k itemsets by intersecting the T-ID sets of any k large-1 itemsets. Finally, we store the following information in each internal node of the MAH-tree.

- The T-ID set and the count for each 1-itemset.
- The count of each large k-itemset.

Based on the stored information of the partitions at the bottom layer, we continue to derive the large itemsets for the partitions at the higher layer. For each partition (denoted by P), we sum up the sizes of the partitions under P. At this point, we have the following three cases to consider:

Case 1

For each 1-itemset, we sum up its counts from each partition under P and join the T-ID sets. Moreover, we compute its support by the following formula to decide whether it is large in P or not, where PS_i denotes the size of the i_{th} partition and C_i is its count in the i_{th} partition.

$$\frac{\sum C_i}{\sum PS_i} \qquad (3)$$

Case 2

For each large-k itemset that appears in all the partitions under P (k > 1), we also compute its support by formula (3) to decide whether it is large in P or not.

Case 3

For each large-k itemset X in some but not all the partitions under P, we use the derived large-1 itemsets that are subsets of X to decide whether X is large in P.

Finally, we follow the structure of the MAH-tree to build the corresponding FH-tree in a bottom-up fashion, where each node keeps the derived large itemsets. An example FH-tree is shown in Figure 4.

To illustrate, take the MAH-tree in Figure 3 as an example. Let the minimum support be set to 50%. Firstly, we consider node 131 and use the partition size and the count of each item to derive the large-1 itemsets in it, i.e., a, b, and c. Then, we intersect the T-ID sets of the three large-1 itemsets to generate the large-2 and the large-3 itemsets, i.e., ab, ac, bc and abc. Finally, we store the large itemsets in the corresponding node on the FH-tree. In this way, we can also derive the large itemsets of node 132.

Secondly, we generate the large itemsets of node 130 for the three cases, respectively. In case 1, we apply formula (3) to get the large-1 itemsets, which are a, b, and c. In case 2, the common large-k itemset in the partitions under node 130 is ab, so we apply formula (3) to generate the large-2 itemset (ab) for node 130. In case 3, the large-k itemsets that appear in one of the partitions under node 130 are bc, ac and abc. We use the large-1 itemsets of node 130 (i.e., a, b, c) to generate the large-k itemsets by applying the *Apriori* property. Consequently, the derived large-k itemsets for node 130 are ab and ac. Finally, the derived large itemsets are stored into the FH-tree as shown in Figure 4.

4 Phenomena Mining By User Queries

Before mining the phenomena, the users have to spec-

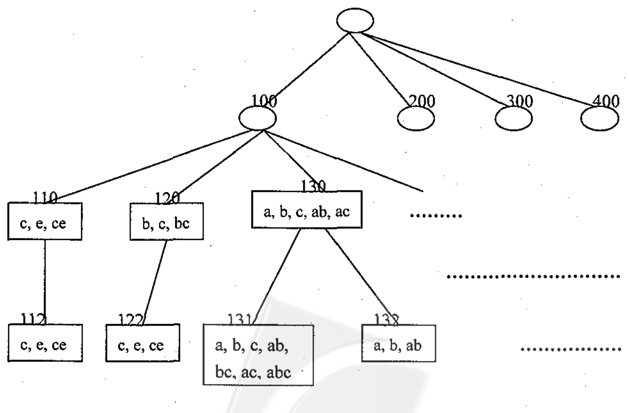


Figure 4 The FH-tree derived from Figure 3

ify queries (called *phenomena queries*) for the phenomena they want to find. Consider the query "in spring, what items do young women and children often purchase in the same areas?" In this query, the query conditions include TA and CA, while the support counts of phenomena depend on LA. We call the former attributes *specified attributes* (abbreviated as SA), while the latter *count-by attributes* (CBA in short).

Given a value pair of two SA's, we can collect the corresponding large itemsets from the FH-tree. A value pair of two SA's may lead to any number of large itemsets, including zero. After that, we compute the frequencies of these large itemsets over the CBA values to find significant ones called *behaviors*. Finally, we analyze the correlation among the behaviors to discover phenomena. In this section, we first present the classification of phenomena queries and then the method for mining phenomena.

4.1 Types of Phenomena Queries

At first, the syntax of a phenomena query is presented as follows.

SA <(X,{V_{x1}, V_{x2}...V_{xi}}), (Y,{V_{y1}, V_{y2}...V_{yj}})>
CBA (Z)
CONDITION <n>

Here X, Y, and Z are selected from the basic types of attributes, and the V_{xi}'s and V_{yj}'s are the values of X and Y respectively. The criterion n in the condition limits the number of the same behaviors in the phenomena to three values, i.e., ZERO, ONE, and ALL. ZERO and ALL mean no and all behaviors are the same, respectively, while ONE means that one of the SA's has the same behavior for each value.

Example 1

The phenomena query "in spring, what items do young women and children often purchase in the same areas, where the items purchased by young women and children are completely different?" can be stated as follows:

SA <(TA,{spring}), (CA,{young women, children})>
CBA (LA)
CONDITION <ZERO>

CBA Classification

According to the position of CBA on the FH-tree, we can classify the phenomena queries into three types (called *CBA classification*), corresponding to the three ways to

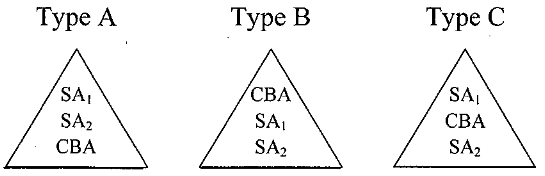


Figure 5 Three types of phenomena queries with CBA classification

Table 3 An example of medical database

T-ID	Time	Patient	Circumstance	Disease
1	1	3	2	3
2	2	5	6	7
...	...	...	...	...

collect large itemsets for phenomena queries. As Figure 5 shows, the CBA of types A, B, and C are located at the bottom, the top, and the middle layers, respectively.

Suppose there is only one value for each SA and the domain sizes of both SA's and CBA are n_1, n_2 and n_3 respectively. During traversing the FH-tree to collect the large itemsets for the query, we have to compare the specified SA values with each value on the corresponding layers. For type A, the comparisons need n_1+n_2 times, while for types B and C, the comparisons need $n_3(n_1+n_2)$ and $n_1+n_2n_3$ times respectively. Therefore, the queries of type A require the least comparisons, while type B needs the most.

Given more than one value in each SA, we may derive more results by collecting more large itemsets. Suppose the numbers of SA values are δ_1 and δ_2, respectively. For type A, the comparisons need $(n_1+n_2\delta_2)\delta_1$ times, while for type B and C, the comparisons need $(n_1+n_2\delta_2)n_3\delta_1$ and $(n_1+n_2n_3\delta_2)\delta_1$ times respectively. Similarly, we conclude that type A achieves the best efficiency in query processing while type B is the worse one.

Phenomena Classification

In addition to the CBA classification, we also classify the phenomena queries into three types (called phenomena classification) based on the number of behaviors that are the same in the phenomena. The three types of phenomena queries are illustrated in Figure 6. We assume that there are two values specified in each SA, i.e., Vx_1, Vx_2, Vy_1 and Vy_2. Let A, B, C and D be the behaviors. We introduce them as follows:

- Type 1: All behaviors in the phenomena are different.
- Type 2: One of the SA's has the same behavior for each value.
- Type 3: All behaviors in the phenomena are the same.

Example 2

We show the three types of phenomena by the example in Table 3.

Type 1 query

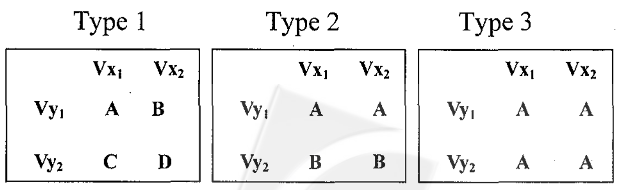


Figure 6 Three types of phenomena queries based on phenomena classification

SA <(TA,{summer,fall}), (LA,{muggy,dark})>
CBA (CA)
CONDITION <ZERO>

A possible result is stated as follows, "in summer and fall, people who live in a muggy circumstance tend to suffer from measles and asthma, and people who live in a dark circumstance tend to suffer from photophobia and psychology problem; this phenomenon often appears to the patients with the same age." According to this phenomenon , we can take actions to prevent the diseases in advance when people live under these circumstances.

Type 2 query
SA <(TA,{spring, summer}), (CA,{group A, group B})>
CBA (LA)
CONDITION <ONE>

A possible result is stated as follows, "in spring, the patients of group A and group B tend to suffer from allergy; in summer, the patients of group A and group B tend to suffer from dermatitis; this phenomenon often appears in the same circumstances." It implies that the patients suffering from allergy will suffer from dermatitis after a season.

Type 3 query
SA <(CA,{group A, group B}), (LA,{moist, muggy})>
CBA (TA)
CONDITION <ALL>

A possible result is stated as follows, "the patients of group A and group B who live in the moist and muggy circumstances often suffer from diarrhea in the same seasons." From this phenomenon, we can take precaution against this disease for the patients of the two groups living in the moist and muggy circumstances.

4.2 Phenomena Mining

In the following, we present our approach to phenomena mining, which takes into account the phenomena classification. The first steps for all the query types are the same. For each combination of the SA values in the query, we collect the corresponding large itemsets from the FH-tree, store them into a table, and use a cube to organize them, as shown in Figure 7(a).

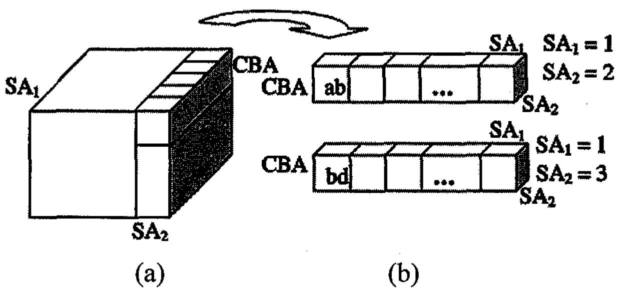


Figure 7 The structure for collecting the large itemsets from the FH-tree

In this cube, each dimension refers to one of the three attributes, while each cell keeps the large itemsets according to certain attribute values. Given a value pair of two SA's, we can collect the cells to form a *unit array*, where its indices refer to the CBA values and the contents keep the large itemsets. For example, in Figure 7(b), the set of unit arrays is a part of the cube in Figure 7(a).

In the second step, for each large itemset, we count the number of its occurrences (called the *frequency*) in every unit array. Based on a user-specified threshold (called the *behavior threshold*), we then select the large itemsets that appear in most of the cells as the behaviors and keep the set of the corresponding cells' indices (called the *index set*).

$$\frac{\textit{frequency}}{\text{size of unit array}} \geq \textit{behavior threshold} \tag{4}$$

At last, based on the other user-specified threshold (called the *phenomenon threshold*), we estimate the correlation among two or more behaviors in different unit arrays to identify the phenomena as follows:

$$\frac{|I|}{|U|} \geq \textit{phenomenon threshold} \tag{5}$$

The notation $|I|$ is the number of indices in the intersection of two or more index sets, while $|U|$ is the number of indices in the union of two or more index sets. In the following, the ways to process the three types of phenomena queries are described respectively.

Type 1 query processing

Inequality (5) is applied to examine every combination of the distinct behaviors collected from each unit array and the qualified ones are then selected as the phenomena.

Type 2 query processing

We iteratively choose an SA and divide the unit arrays into groups by the values of the chosen SA. Moreover, inequality (5) is applied to the behaviors that appear in each unit array of a group. After that, we apply inequality (5) again to every combination of the qualified behaviors collected from each group and regard the qualified combinations as the phenomena.

Type 3 query processing

Inequality (5) is applied to the behaviors that appear in each unit array and the qualified ones are regarded as the phenomenon.

Example 3

Let the behavior threshold and the phenomenon threshold be set to 40% and 30%, respectively. The follow-

Table 4 Collected behaviors by the SA values

Summer	LA1	LA2	LA3	LA4	LA5	LA6	LA7
Young Women	abc, d	ac	acd	ef	ace	b	ac
Children	ae	ac, b	ae	c, d	ace	cd, b	ae

(a)

Winter	LA1	LA2	LA3	LA4	LA5	LA6	LA7
Young Women	acf	bcf, ad	acf	b, ad	bcf	be	acf
Children	abe, de	ac	ade	bcf	bce	bdf, def	ade

(b)

ing is a query example.

SA <(TA,{summer, winter}), (CA, {young women, children})>
CBA (LA)
CONDITION <ZERO/ONE/ALL>

At the first step, we collect the large itemsets from the FH-tree based on the SA values. Table 4 shows an example, where (a) and (b) stand for two unit arrays, respectively. Since the behavior threshold is set to 40%, the frequency of a behavior must be larger than 3.

For each unit array, inequality (4) is then used to find out the behaviors and the results are shown in Figure 8. For type 1 queries, every combination of the distinct behaviors collected from each unit array is examined by inequality (5) and the resultant phenomenon is shown in Figure 9. For type 2 and type 3 queries, there is no phenomenon because each behavior appears in exactly one unit array.

5 Experimental Results

In this section, we describe how to generate the synthetic data and demonstrate the experimental results on the effect of tree construction, large itemset generation and phenomena discovery, respectively. In addition, we also implement an array-based data cube to compare it with our tree structure in various aspects.

5.1 Synthetic Data Generation

Synthetic data are used to simulate a customer purchasing in a retail environment. We generate the synthetic data by a simple version method. Table 5 summarizes the parameters used. The transaction is repeatedly assigned items from a set of potentially large itemsets, until the length of a transaction reaches a predefined maximum.

summer / young women	ac (LA number: 1 2 3 5 7)
summer / children	ae (LA number: 1 3 5 7) c (LA number: 2 4 5 6)
winter / young women	acf (LA number: 1 3 7) b (LA number: 2 4 5 6)
winter / children	de (LA number: 1 3 7)

Figure 8 Behaviors of the young women and children in summer and winter

Table 5 Parameters used in the experiment

Parameter	Meaning
|D|	Number of transactions
|MT|	Maximum size of the transactions
|L|	Number of potentially large itemsets

As for the other three types of attributes (TA, CA, LA), after randomly generating the buying times, the customers (with different ages and genders) and the locations (in coordinate format), we classify the attributes into different types and associate them with each transaction. Basically, the time attribute is classified into four types, the customer is classified into eight types and the location is classified into seven types. We generate the data by |L| = 2000 and |MT| = 20. The number of transactions is set to be in the range from 1K to 100K.

5.2 Effects of Tree Construction

Since the positions of the three attributes in the MAH-tree and FH-tree are significant to our approach, in the first experiment we examine whether different tree structures influence the performance of tree construction. Because the data cube has a structure similar to ours, the second experiment evaluates the storage cost of our approach via a comparison.

Domain Sizes of Attributes

We set the number of transaction to be 1000, the domain size of TA and CA 4, and increase the domain size of LA from 4 to 200. Under this setting, there are three kinds of tree structure, that is, LA is in the top (denoted by L**), middle (*L*) and bottom (**L) layer.

From Figure 10, we see that the increase of domain size leads to the increase of tree construction time for any kind of tree structure. When an attribute with a larger domain size is put in the higher layer, the tree construction costs more. When the domain sizes for TA, CA and LA are set to 4, 8 and 7 respectively, the influence of tree structure is small. We take the first kind of tree structure (TA-CA-LA) as our settings in the subsequent experiments.

Storage Costs

We increase the number of transactions from 1K to

	young women	**children**
summer	ac	ae
winter	acf	de

Figure 9 An example phenomenon returned from Type 1 queries

100K. In Figure 11, we compare the storage cost of the transformed database in the disk with the one of MAH-tree in the memory by calculating the ratio of storage space of MAH-tree to the transformed database. We find that the storage cost of MAH-tree is much less than the transformed database. The MAH-tree can still be stored in the memory even when the number of transactions is 100K.

Figure 12 indicates FH-tree takes up less space than the data cube, especially when the number of transaction becomes larger. This is because we only have to store large itemsets in FH-tree for discovering phenomena, while the data cube needs to store the counts of all the items.

5.3 Effects of Large Itemset Generation

In Section 3.2, we propose a method for mining MA association rules with three cases. In this section, we propose a brute-force method for comparison. This method combines the itemsets computed in the lower layer to derive the large itemsets in the higher layer. This experiment evaluates the efficiency of the large itemset generation as either the number of transactions or the criterion of the SF threshold increases.

Number of Transactions

The number of transactions is from 1K to 100K, the criterion of the SF threshold is fixed to 0.5, and the minimum support is set to 0.1%. The execution time of both methods increases when more transactions are examined. Our method costs less than the brute-force method, as shown in Figure 13, especially when the number of transactions reaches 100K.

Criterion of the SF Threshold

Next, we investigate how the criterion of the SF threshold affects the performance of our approach. The criterion of the SF threshold varies from 0.1 to 1.0. The minimum support is set to 1%. There are three sets of transactions with numbers 1000 (denoted as D1000), 10000 (D10000) and 50000 (D50000), respectively.

The results are shown in Figure 14. When the criterion of the SF threshold increases, the number of qualified partitions will decrease. In addition, it is interesting to note that when the criterion of the SF threshold increases as 0.5, the number of qualified partitions descends to the minimum value and this condition is the same for the three sets. After checking the size of each qualified partition, we find that when the criterion of the SF threshold is greater than 0.5, the average partition size remains large.

5.4 Effects of Phenomena Discovery

The last experiment evaluates the performance of

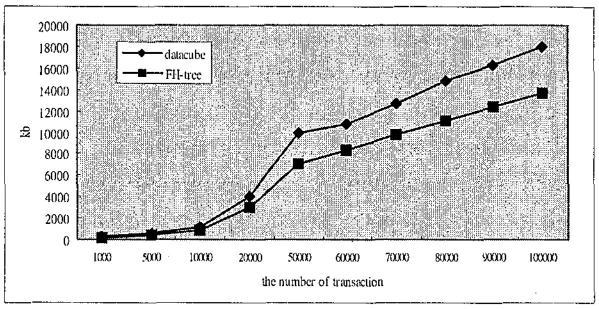


Figure 12 FH-tree vs. Data cube

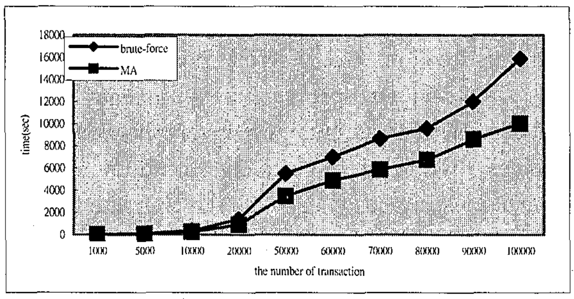


Figure 13 Number of transactions vs. Execution time for large itemset generation

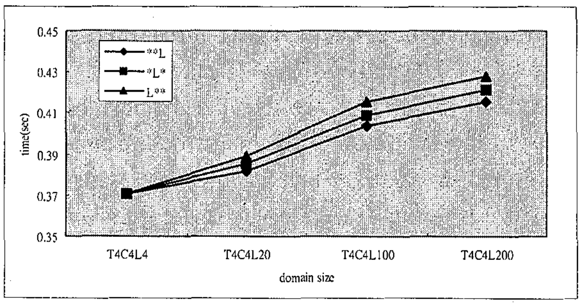


Figure 10 Domain size vs. Execution time for tree construction

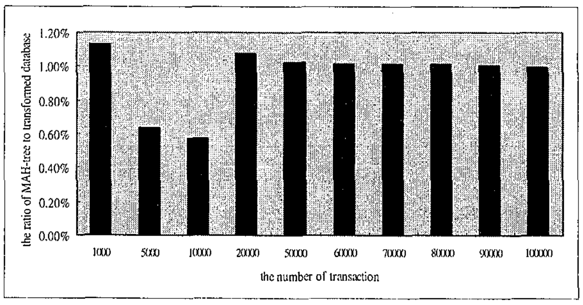


Figure 11 MAH-tree vs. Transformed database

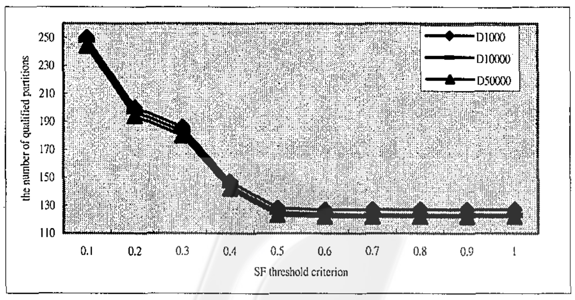


Figure 14 Criterion of the SF threshold vs. Number of qualified partitions

phenomena mining for various query types, the number of large itemsets, the number of the SA values, and the domain size of the CBA.

Query Types

In this experiment, the minimum support and |D| are set to 1% and 1000, respectively. Moreover, both the behavior threshold and the phenomenon threshold are 0.5, and the number of the SA values is 2. As Figure 15 shows, the execution time of query type A in the CBA classification is the least among the three query types in the phenomena classification. We take query type A in the CBA classification as the basic query type in the subsequent experiments.

Number of Large Itemsets

To observe the influence of the number of large itemsets on the phenomena mining, we vary the number of large itemsets in the range from 300 to 1800. Both the behavior threshold and the phenomenon threshold are 0.5. Besides, the number of the SA values is 2. As shown in Figure 16, the execution time of each query type in the phenomena classification increases with the growth of the number of large itemsets.

Number of the SA Values

In Figure 17, the number of the SA values varies from 2 to 6. Moreover, the minimum support and |D| are 1% and 1000, respectively. Besides, both the behavior threshold and the phenomenon threshold are 0.5. The execution time for each query type in the phenomena classification increases when the number of the SA values increases.

Domain Size of the CBA

In Figure 18, the domain size of the CBA increases from 10 to 200. The minimum support and |D| are 1% and 1000. Both the behavior threshold and the phenomenon threshold are 0.5. Besides, the number of the SA values is 2. The execution time for each query type in the phenomena classification increases when the domain size of the CBA increases.

6 Conclusion

In this paper, we propose a novel approach for mining association rules with multiple attributes and deriving interesting phenomena from the large itemsets. A multiple attribute hierarchical tree is built for efficient association rule mining and then a frequent hierarchical tree is built for phenomena mining. The experimental results indicate that our method for phenomena mining is insensitive to the number of large itemsets, the number of the SA values, and the domain size of the CBA. In the future, we will study the various kinds of phenomena in the real-world applications.

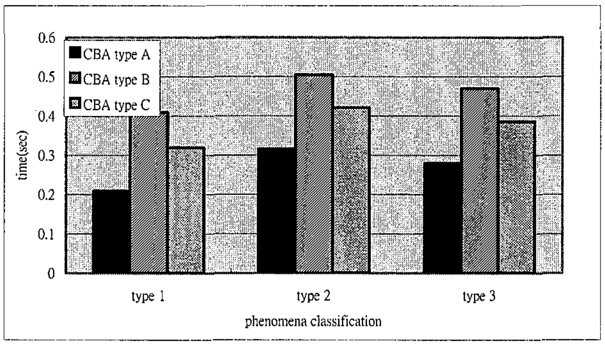


Figure 15 Classification of phenomena queries vs. Execution time for phenomena mining

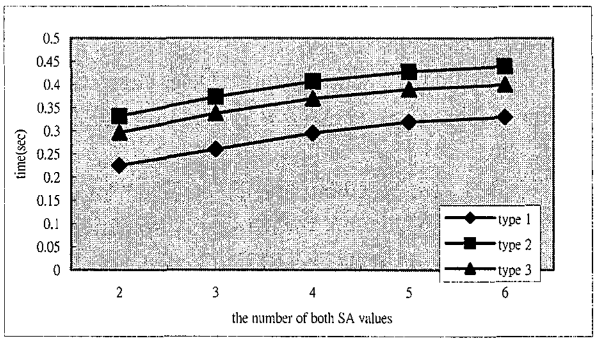


Figure 17 Number of the SA values vs. Execution time for phenomena mining

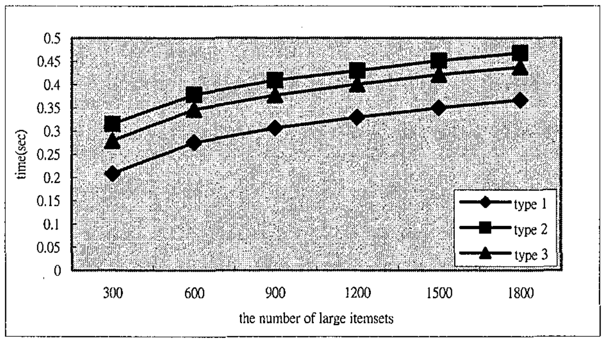


Figure 16 Number of large itemsets vs. Execution time for phenomena mining

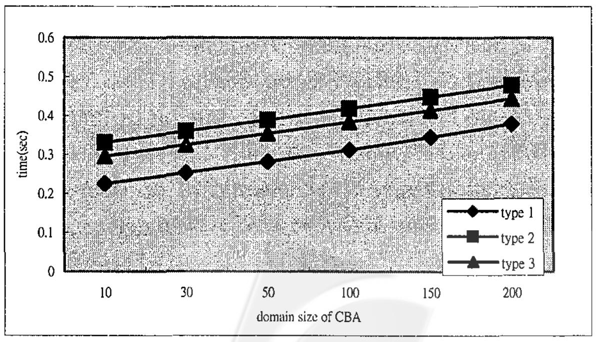


Figure 18 Domain size of the CBA vs. Execution time for phenomena mining

References

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proceedings of ACM SIGMOD Conference*, Washington, D.C., May 1993, pp. 207-216.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in *Proceedings of VLDB Conference*, September 1994, pp. 487-499.

[3] R. J. Bayardo, "Efficiently Mining Long Patterns from Databases," *Proceedings of ACM SIGMOD Conference*, 1998, pp. 85-93.

[4] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data," *Proceedings of ACM SIGMOD Conference*, 1997, pp. 255-264.

[5] 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, 5:926-938.

[6] G. Grahne, L. V. S. Laskshmanan, X. Wang, and M. H. Xie, "On Dual Mining: From Patterns to Circumstances, and Back," *Proceedings of IEEE Conference on Data Engineering*, 2001, pp. 195-204.

[7] J. W. Han and Y. Fu, "Mining Multiple-level Association Rules in Large Databases," *IEEE Transactions on Knowledge and Data Engineering*, 1999, 11(5): 798-804.

[8] T. Palpanas, "Knowledge Discovery in Data Warehouses," *ACM SIGMOD Record*, 2000, 29(3): 88-100.

[9] J. S. Park, M. S. Chen, and P. S. Yu, "An Effective Hash-based Algorithm for Mining Association Rules," *Proceedings of ACM SIGMOD Conference*, San Jose, May 1995, pp. 175-186.

[10] K. Sayood, *Introduction to Data Compression*, 1996, pp. 15-17.

[11] R. Srikant and R. Agrawal, "Mining Generalized Association Rules," *Proceedings of VLDB Conference*, 1995, pp. 407-419.

[12] V. S. Subrahmanian, *Principles of Multimedia Database Systems*, 1998, pp. 83-88.

[13] J. Tang, "Using Incremental Pruning to Increase the Efficiency of Dynamic Itemset Counting for Mining Association Rules," *Proceedings of ACM Conference on Information and Knowledge Management*, 1998, pp. 273-280.

[14] H. Toivonen, "Sampling Large Databases for Association Rules," *Proceedings of VLDB Conference*, 1996, pp. 134-145.

[15] K. Wang, Y. He, and J. Han, "Mining Frequent Itemsets Using Support Constraints," *Proceedings of VLDB Conference*, 2000, pp. 43-52.

[16] J. Widom, "Research Problems in Data Warehousing," *Proceedings of ACM Conference on Information and Knowledge Management*, 1995, pp.25-30.

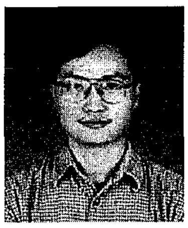
[17] S. J. Yen and A. L. P. Chen, "The Analysis of Relationships in Databases for Rule Derivation," *Journal of Intelligent Information Systems*, 1996, 7: 1-24.

[18] S. J. Yen and A. L. P. Chen, "A Graph-Based Approach for Discovering Various Types of Association Rules," *IEEE Transactions on Knowledge and Data Engineering*, September/October 2001, 13(5): 839-845.

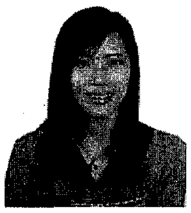
[19] C. L. Yip, K. K. Loo, B. Kau, D. W. Cheung, and C. K. Cheng, "LGen--A Lattice-Based Candidate Set Generation Algorithm for I/O efficient Association Rule Mining," *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1999, pp. 54-63.

[20] Y. Zhao, P. Deshpande, and J. F. Naughton, "An Array-based Algorithm for Simultaneous Multidimensional Aggregates," *Proceedings of ACM SIGMOD Conference*, 1997, pp. 159-170.

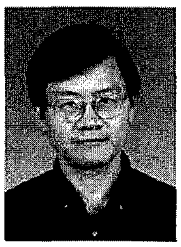
Biographies



Yi-Hung Wu received the BS degree in computer science from National Chiao Tung University, Taiwan, in 1993, and the PhD degree in computer science from National Tsing Hua University, Taiwan, in 2001. He is now with the Computer & Communication Research Center at National Tsing Hua University, Taiwan, as a postdoctoral fellow. His current research interests are in the area of data mining, with emphasis on data stream analysis and applications. He also works on the problems related to Web data mining, multimedia data mining, and privacy-preserving data mining.



Yu-Chieh Chang received the BS degree in computer science from National Tsing Hua University, Taiwan, in 1999, and the MS degree in computer science from National Tsing Hua University, Taiwan, in 2001. She works in Telecommunication Laboratories, Chunghwa Telecom Corporation, and joins the Broadband Service Operation Support Technology Project.



Arbee L.P. Chen received a PhD in computer engineering from University of Southern California in 1984. He worked at System Development Corporation and Bellcore before he joined National Tsing Hua University in Taiwan in 1990. He was the chairman of the Department of Computer Science and Information Engineering at National Dong Hwa University from 2001 to 2004. Since 2004, he becomes a Chengchi University Chair Professor at National Chengchi University. Dr. Chen organized IEEE Data Engineering Conference in Taiwan in 1995. He has been a visiting scholar at Beijing Tsinghua University, Kyoto University, Stanford University, and King's College of the University of London. He has been a recipient of the Outstanding Research Award from National Science Council since 1996.