Available online at www.sciencedirect.com
ScienceDirect
Decision Support Systems 45 (2008) 150–163
Decision Support Systems
www.elsevier.com/locate/dss

Context-based market basket analysis in a multiple-store environment

Kwei Tang [a,b,*], Yen-Liang Chen [c], Hsiao-Wei Hu [c]

[a] Krannert School of Management, Purdue University, West Lafayette, IN 47907, USA
[b] TiasNimbas Business School, Tilburg, The Netherlands
[c] Department of Information Management, National Central University, Chung-Li 320, Taiwan, ROC

Received 28 September 2006; received in revised form 4 December 2007; accepted 16 December 2007
Available online 3 January 2008

Abstract

We propose a new approach to performing market basket analysis in a multiple-store and multiple-period environment. In using the method, the user first defines a time concept hierarchy and a place (location) hierarchy, according to his or her application and needs. A set of contexts is systematically derived from the two hierarchies by combining the concept levels of the two hierarchies. We developed an efficient algorithm for extracting the association rules, which meet the support and confidence requirements for all the contexts. Using the approach, a decision maker can analyze purchasing patterns at very detailed concept levels of time and place, such as a combination of days and stores, at more general levels, such as a combination of quarters and states, and combinations of detailed levels of one with general level of the other, such as a combination of days and regions. In addition to this flexibility, the association rules are well organized, because they are generated according to the contexts derived from the time and place hierarchies. A numerical evaluation shows that the algorithm is efficient in running time and may generate more specific and richer information than the store-chain rules and the traditional rules.

Keywords: Association rule; Data mining; Store chain; Algorithm

1. Introduction

Market basket analysis (also known as association rule mining) is a data mining method that examines a large transactional database to determine which items are most frequently purchased jointly. Since its introduction by Agrawal, Imielinski, and Swami [1], it has drawn much research interest and has quickly developed into a major research area. Brief literature reviews have been given by Chen, Han, and Yu [5] and Han and Kamber [8]. Numerous applications have been identified, including, for example, cross-selling [4,10], Web site analysis [3,20,16], decision support [11,13,22], credit evaluation [7], privacy issue [17,19], criminal event prediction [23], customer behavior analysis [9,12,15,18] and fraud detection [14,21].

The basic form of association rules is based upon the assumption that if a customer purchases a certain set of items from a store, he or she is more (or less) likely to purchase another set of items. Consider two non-overlapping subsets of product items, X and Y, an association rule in form of $X \Rightarrow Y$ indicates a purchase pattern that if a customer purchases X then he or she also purchases Y. Two measures, *support* and *confidence*, are commonly used in selecting association rules. Support is a measure

* Corresponding author. Krannert School of Management, Purdue University, West Lafayette, IN 47907, USA. Tel.: +1 765 494 4464; fax: +1 765 494 9658.
E-mail address: ktang@purdue.edu (K. Tang).

0167-9236/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2007.12.016

of how often the transactional records containing both X and Y in the database, and confidence is a measure of the accuracy of the rule, defined as the ratio of the number of transactional records with both X and Y to the number of transactional records with X only. The main challenge in practice is to develop an efficient algorithm for extracting association rules which meet the selection criteria from large databases consisting of large numbers of product items and transaction records. The Apriori algorithm [1] has been shown as an efficient method for the basic association rules described above.

A common assumption used in association rules is that product items under consideration are on shelf all the time across all stores (if multiple stores and multiple time periods are considered). This assumption posts a serious limitation because, in today's competitive business environment, most major companies have subsidiaries, branches, or stores in many geographical locations. Furthermore, these companies use different product-mix strategies in different stores (locations) and change them dynamically over time. As a result, important purchasing patterns may not be correctly identified by the traditional methods.

To overcome this problem, Chen, Kang, Shen, and Hu [6] proposed a method, called the store-chain association (SCA) rule, which adds "contexts" to the traditional rules to indicate combinations of the time and place, where the rules are applicable. Consider a rule, for example, {notebook computer, extended warranty} ⇒ {laser printer} with contexts: (March 1, store 50), (March 2, store 50), …, (May 30, store 50), (March 1, store 51), …, (May 29, store 51), …, and (May 31, store 73). This rule suggests that a noticeable portion of the customers who purchased a notebook computer with an extended warranty also purchased a laser printer in the listed combinations of days and stores.

The SCA rules have addressed several important problems associated with the traditional association rules in the multiple-store and multiple-period environment. In particular, they take into account the on-shelf time of a product. However, as illustrated in the above example, contexts associated with rules may be fragmented and unorganized, especially when stores have very different product mixes and change them very frequently. As a result, it may be difficult to use the rules to form business strategies. For reference, we call the SCA rules are rule-based because the output of the method is a set of rules with attached contexts.

In this paper, we propose a context-based approach to extract association rules based on a pre-determined structure of contexts. In contrast to the SCA rules, we first form a time and a place concept hierarchies, by defining a sequence of mapping from a set of low-level concepts to higher-level, more general concepts [8]. Consider the time and place (location) hierarchies in Fig. 1. These time and place hierarchies are defined by users and arranged in increasing granularity from top to bottom. The time hierarchy consists of year, quarter, month, and day. At each hierarchy level, there are several higher granularity levels. For example, spring, summer, fall, and winter are the higher granularity levels of the hierarchy level, year. Similarly, country, region, city, and store comprise the place hierarchy.

We develop an efficient algorithm to extract association rules for all combinations of time and place hierarchy levels. The results would help a decision maker analyze purchasing patterns at very detailed concept levels of both time and place, such as a combination of days and stores, at more general levels, such as a combination of quarters and states, and combinations of detailed levels of one with a general level of the other, such as a combination of days and regions. In addition to this flexibility, the association rules are well organized, because they are generated according to the contexts systematically derived from the time and place hierarchies.

This paper is organized as follows. We formally define the problem statement in Section 2 and, in Section 3, we develop an algorithm for extracting association rules for all the contexts derived from given concept hierarchies of time and place. In Section 4, we perform two experiments to evaluate the proposed algorithm. Finally, the conclusion is given in Section 5.

2. Problem definition

We consider a market basket database D, containing transaction records from multiple stores over multiple time periods. The records are first sorted by a store identifier and then by time. For ease of presentation, the cardinality of a set, say $\sum$, is denoted by $|\sum|$. Let $I=\{i_1, i_2, \ldots, i_n\}$ be the set of product items included in D, where i_k represents the kth item. Let X be a subset of items in I. We refer to X as a k-itemset if it contains k items or $|X|=k$. Furthermore, each transaction record in D is a subset of I and is attached with a timestamp and store identifier to indicate the time and store that the transaction occurred.

Let $T=\{T_1, T_2, \ldots, T_a\}$ and $P=\{P_1, P_2, \ldots, P_b\}$ denote the time and place (location) hierarchies, and T_i and P_j denote the i-th granularity level of T and the j-th granularity level of P, respectively. We let T_1 be the set of the smallest time units considered, and T_2, T_3, …, T_a be arranged in decreasing granularity. A typical example is $T=\{$days, months, quarters, years$\}$. Similarly, we let P_1 be the set of individual stores, and P may be {stores, states, regions,

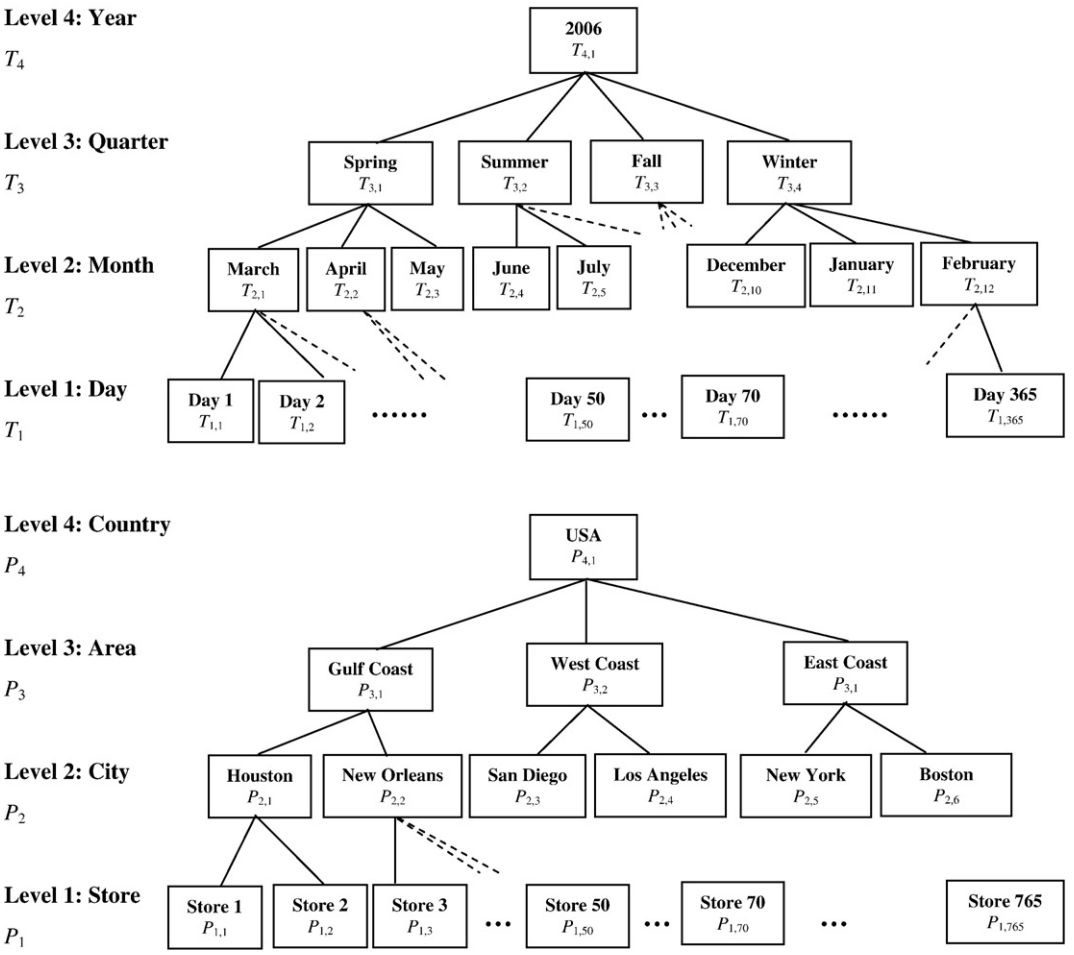


Fig. 1. Place and time hierarchies.

countries}. Let $T_i = \{T_{(i,1)}, T_{(i,2)}, \ldots, T_{(i,x)}, \ldots, T_{(i,\,n_i)}\}$ and $P_j = \{P_{(j,1)}, P_{(j,2)}, \ldots, P_{(j,y)}, \ldots, P_{(j,\,m_j)}\}$, where n_i is the number of time periods at level i of T and m_j is the number of places at level j of P.

Without loss of generality, let $T_{t'}$ and $P_{p'}$, $t' \geq 1$ and $p' \geq 1$, be the highest time and place granularity levels, respectively, that the user is interested in generating association rules. For given t' and p', we need to consider totally $(a-t'+1)\times(b-p'+1)$ combinations of time and place granularity levels. To simplify our presentation, we use a lattice structure to represent the set of these combinations with each node in the lattice corresponding to the combination a time granularity level and a place granularity level. Accordingly, we define the *TP* lattice, which satisfies the following three conditions:

(1) node (T_i, P_j) exists in *TP* for $t' \leq i \leq a$ and $p' \leq j \leq b$
(2) arc from (T_i, P_j) to (T_{i+1}, P_j) exists in *TP* for $t' \leq i < a$ and $p' \leq j \leq b$
(3) arc from (T_i, P_j) to (T_i, P_{j+1}) exists in *TP* for $i = t'$ and $p' \leq j < b$.

Consider a time hierarchy T and a place hierarchy P, each with four levels, and $t' = 2$ and $p' = 2$. The *TP* lattice with nine nodes is shown in Fig. 2.

2.1. Context and section

The term *Section*, denoted by $S^k_{i,j}$, refers to the subset of transactions in database D, which occurred in time $T_{i,x}$ and store $P_{j,y}$, where $k = (y-1)\times n_i + x$. For example, consider two concept hierarchy levels in Fig. 1: area ($j=3$) and month ($i=2$). There are thirty-six combinations of areas and months. According to the definition, $S^1_{2,3}$ is the subset of transactions in Gulf Coast ($P_{3,1}$) and March ($T_{2,1}$), $S^2_{2,3}$ is that in Gulf Coast and April, …, and $S^{36}_{2,3}$ is that in East Coast and February.

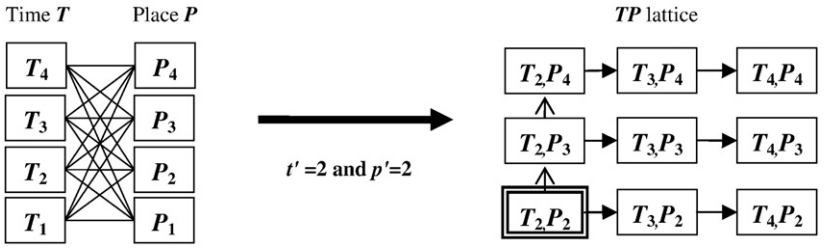


Fig. 2. An example of the *TP* lattice.

Based on the definitions of itemset and section, we define three types of context:

(1) The context of itemset X, denoted by $cx_1(X)$, is the set of all time-store pairs (combinations) where all items in X are on shelf. Note that X may consist of only one item.
(2) The context of section $S^k_{i,j}$, denoted by $cx_2(S^k_{i,j})$, is the set of all time-store pairs in the section.
(3) The context of the itemset X in section $S^k_{i,j}$, denoted by $cx_3(X, S^k_{i,j})$, is the set of all time-store pairs in $cx_2(S^k_{i,j})$, where all items of X are on shelf. In other words, $cx_3(X, S^k_{i,j}) = cx_1(X) \cap cx_2(S^k_{i,j})$.

2.2. Rule selection criteria

Similar to the traditional association rule selection, we use support and confidence as the selection criteria. However, these two criteria have to be time- and place-specific and, very importantly, to reflect whether the items are on shelf.

Let itemset $X = A \cup B$ and $A \cap B = \phi$. We define two supports for rule $A \Rightarrow B$ in section $S^k_{i,j}$. The first is *context support*, which is the relative frequency of occurrences of itemset X in given context $cx_2(S^k_{i,j})$, defined as

$$\xi_C\left(X, cx_2\left(S^k_{i,j}\right)\right) = \text{count}\left(X, cx_2\left(S^k_{i,j}\right)\right) \Big/ \left|cx_2\left(S^k_{i,j}\right)\right|,$$

where $|cx_2(S^k_{i,j})|$ is the number of transactions occurring in context $cx_2(S^k_{i,j})$ and count $(X, cx_2(S^k_{i,j}))$ is the number of transactions containing X in $cx_2(S^k_{i,j})$. If $\xi_C(X, cx_2(S^k_{i,j}))$ exceeds a given context support threshold δ_S, then we call X a *context-large itemset* in *section* $S^k_{i,j}$. We use $CL^k_{i,j}$ to denote the set of all context-large itemsets in $S^k_{i,j}$.

The second is *actual support*, which is the relative frequency of occurrences of itemset X in given context $cx_3(X, (S^k_{i,j}))$, defined by

$$\xi_A\left(X, cx_3\left(X, S^k_{i,j}\right)\right) = \text{count}\left(X, cx_3\left(X, S^k_{i,j}\right)\right) \Big/ \left|cx_3\left(X, S^k_{i,j}\right)\right|,$$

where $|cx_3(X, S^k_{i,j})|$ is the number of transactions occurring in context $cx_3(X, S^k_{i,j})$ and count$(X, cx_3(X, S^k_{i,j}))$ is the number of transactions containing X in $cx_3(X, S^k_{i,j})$. Note that the context support is the same as that defined for the traditional association rules, and the actual support is based on only the transactions in which the items in X are on shelf.

We further define the confidence of rule $A \Rightarrow B$ in *section* $S^k_{i,j}$, denoted by $\chi(A \Rightarrow B)$, as:

$$\chi(A \Rightarrow B) = \frac{\xi_A(A \cup B,\ cx_3(A \cup B,\ S^k_{i,j}))}{\xi_A(A,\ cx_3(A \cup B,\ S^k_{i,j}))} = \frac{count(A \cup B,\ cx_3(A \cup B,\ S^k_{i,j}))}{count(A,\ cx_3(A \cup B,\ S^k_{i,j}))}.$$

Accordingly, a rule is called a *section rule* in $S^k_{i,j}$ if its context support and confidence are, respectively, larger than pre-determined threshold δ_S and δ_C. For convenience, we use $SR^k_{i,j}$ to represent the set of all section rules in section $S^k_{i,j}$. Our goal is to extract all section rules from D for all combinations of time and place granularity levels defined in the *TP* lattice.

3. Algorithm

We present an efficient algorithm for extracting all section rules from database D. The algorithm is outlined in Fig. 3. We first discuss the main concepts of the algorithm and then use an example to illustrate the algorithm. A more detailed, technical description of the algorithm, including specific structures and functions, is given in the Appendix.

We first define several symbols used in describing the algorithm. Let PL represent the set of all context-large itemsets at time starting level t' and place starting level p'; i.e., PL combines all context-large itemsets in all sections associated with lattice node $(T_{t'}, P_{p'})$. It can be verified that if an itemset is not included in PL, it would not be context-large in any larger sections constructed afterwards by combining the sections in

1. **Input:** Temporal spatial database D and ***TP*** lattice, *SI*_table.
2. **Procedure:**
3. **Phase 0:**
4. Find ***PL*** ;
5. ***HT*** = creatTree(***PL***) ;
6. **Phase 1:**
7. For every section $S^k_{t',p'}$ {
8. $HT^k_{t',p'}$ = **HTree** (HT, $S^k_{t',p'}$) ;
9. $CL^k_{t',p'}$ = All itemsets which are context large in $HT^k_{t',p'}$;
10. $SR^k_{t',p'}$ = Generate the rules from $CL^k_{t',p'}$ by the information stored in $HT^k_{t',p'}$; }
11. **Phase 2:**
12. Following the broadcasting path in ***TP*** lattice to do the combination {
13. { $HT^k_{i,j}$ = **Combine**(i, j, k) ;
14. $CL^k_{i,j}$ = All the itemsets which are context large in $HT^k_{i,j}$;
15. $SR^k_{i,j}$ =Generate the rules from $CL^k_{i,j}$ by the information stored in $HT^k_{i,j}$; } }
16. **Output:**
17. All the rules stored in $SR^k_{i,j}$;

Fig. 3. The proposed algorithm.

$(T_{t'}, P_{p'})$. We also define HT as a hash tree with a specific node structure, and $HT^k_{i,j}$ the hash tree obtained by adding the count information in $S^k_{i,j}$ *to HT.*

We use Fig. 4 to explain the proposed algorithm. For convenience in describing the algorithm, we use $t'=2$ and $p'=2$. In phase 0 of the algorithm, we find the context-large itemsets in each section of (T_2, P_2) by the Apriori algorithm or the FP-tree algorithm and combine them to obtain PL. We then construct a hash tree HT from PL. At this point, HT does not contain any count information. This process is described by lines 4 and 5 in Fig. 3.

In phase 1, since items may have different on-shelf times in different stores, we scan the database again to compute the counts of the itemsets in PL in all contexts of lattice node (T_2, P_2). In producing the counting information, we use a Table, namely the *SI* table, which stores the item on-shelf information for each section in the lattice note. Fig. 5-a shows a simple example of the *SI* table, where $i_1, i_2, \ldots, i_n$ are items and $S_1, S_2, \ldots, S_m$ are unit sections in (T_1, P_1). In the table, "1" and "0" indicate, respectively, that the item is or is not for sale in the corresponding store and time. We store these count information of the itemsets into $HT^k_{2,2}$ for finding the rules under different granularities of time and place.

In phase 2, instead of scanning the database again, we use the hash tree with the count information in a higher granularity level to derive the hash tree with the count information in a lower granularity level by following the broadcasting path in *TP* shown in Fig. 4. Basically, an arc in the figure represents the process that a hash tree is derived from another hash tree, and the entire sequence

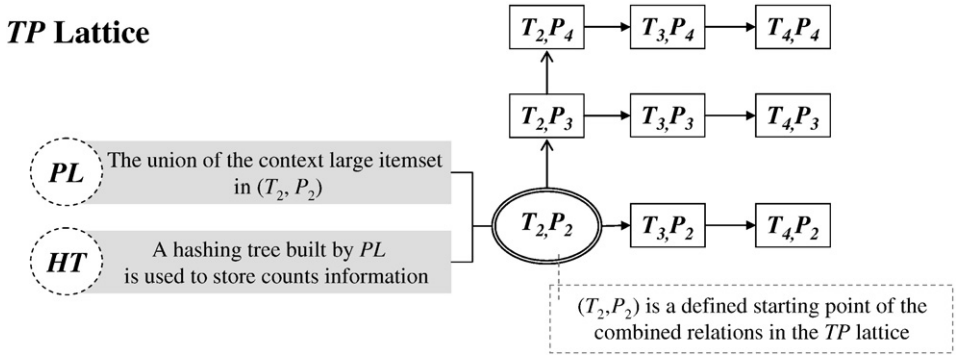


Fig. 4. The broadcasting path in the lattice.

a

	i_1	i_2	i_3	i_4	...	i_n
S_1	1	1	0	1	...	1
S_2	1	0	0	1	...	1
S_3	0	1	1	1	...	1
...	...	...	...	...	...	...
S_m	1	0	1	1	...	1

b

	a	*b*	*c*	*d*	...	*z*
May 1, *P*1	0	1	1	0	...	0
May 2, *P*1	1	0	0	1	...	1
May 3, *P*1	0	1	1	1	...	1
...	...	...	...	...	...	...
Feb. 28, *P*765	1	0	1	1	...	1

Fig. 5. a. An *SI* table. b. An example of the *SI* table.

of operations is performed by moving to the right first and upwards. A very important reason that these trees can be combined is because all the hash trees associated with each combination of time and place have the identical structure, i.e., the same nodes and arcs. Therefore, several trees can be combined into one by adding their counts in the corresponding nodes.

The proposed algorithm can be finished by scanning the database three times. If the FP-tree algorithm is used in phase 0 to find the context-large itemsets in each section, two scans of the database are sufficient, because the sections are non-overlapped with each other. In phase 1, we need to scan the database to add the count information to all the hash trees. As before, since these sections are non-overlapped, one scan of database is sufficient. Finally, no database access is required in Phase 2, because the count information is available in the hash trees constructed previously.

Having constructed $HT^k_{t',p'}$ *or* $HT^k_{i,j}$ in general, we use the count information in the tree to evaluate the support and confidence levels for all possible rules in the corresponding section or context in selecting the section rules.

Example. We use the example in Fig. 4 to illustrate the algorithm. Suppose we have found the context-large itemsets in each section of (T_2, P_2) and combined them into PL = {*a, ab, ac, ad, abc, abd, acd, abcd, b, bc, bd, bcd, c, cd, d*}. Then, we build a hash tree HT as shown in Fig. 6.

In order to compute the counts of the itemsets in PL in all contexts of lattice node (T_2, P_2) under different on-shelf times in different stores, we first check the *SI* table to find out which items are on-shelf within a certain context. Through the *SI* table, we can compute the count of the itemsets {*b, c, bc*} in $S^1_{2,2}$. Further assume that we find the counts of *b*, *c*, and {*b*, *c*} in $S^1_{2,2}$ are 2, 3, and 1, respectively, and the total number of transactions is 4. That is, count(b, $cx_2(S^1_{2,2})$)=2, count(c, $cx_2(S^1_{2,2})$)=3, count(bc, $cx_2(S^1_{2,2})$)=1, $|cx_3(bc, S^1_{2,2})|$=4. We store the above count information in $HT^1_{2,2}$. Fig. 7-a shows how to add the count information to the nodes corresponding to the itemsets. Meanwhile, we assume that $HT^1_{2,2} \subset HT^1_{3,2}$ and $HT^2_{2,2} \subset HT^1_{3,2}$. By combining $HT^2_{2,2}$ and $HT^1_{2,2}$, shown, respectively, in Fig. 7-b and a, we obtain a new hash tree $HT^1_{3,2}$ shown in Fig. 7-c. This illustrates that the count information in a lower granularity level of the lattice can be obtained from those in higher granularity levels.

Finally, we use the information stored in $HT^k_{i,j}$ to compute the actual support and the confidence of the rules in higher granularity levels to identify section rules in the corresponding section. Let us consider the node shown in Fig. 8 with label *c*, which immediately follows the node with label *b*. After scanning all the unit sections in $S^k_{2,2}$, the node contains all count information in section $S^k_{3,2}$. Therefore, we can obtain the confidence of the rules $b \Rightarrow c$ and $c \Rightarrow b$ as $\chi(b \Rightarrow c)=3/5=0.6$ and $\chi(c \Rightarrow b)=3/4=0.75$.

4. Performance evaluation

We perform numerical experiments to (1) evaluate the running time and the number of generated rules of the proposed algorithm, and (2) compare the rules generated from the proposed algorithm, the Apriori algorithm

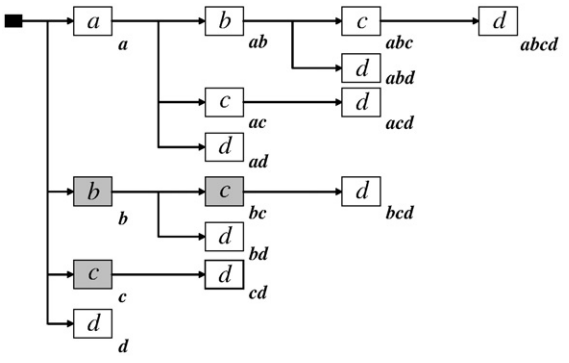


Fig. 6. An example of the hash tree.

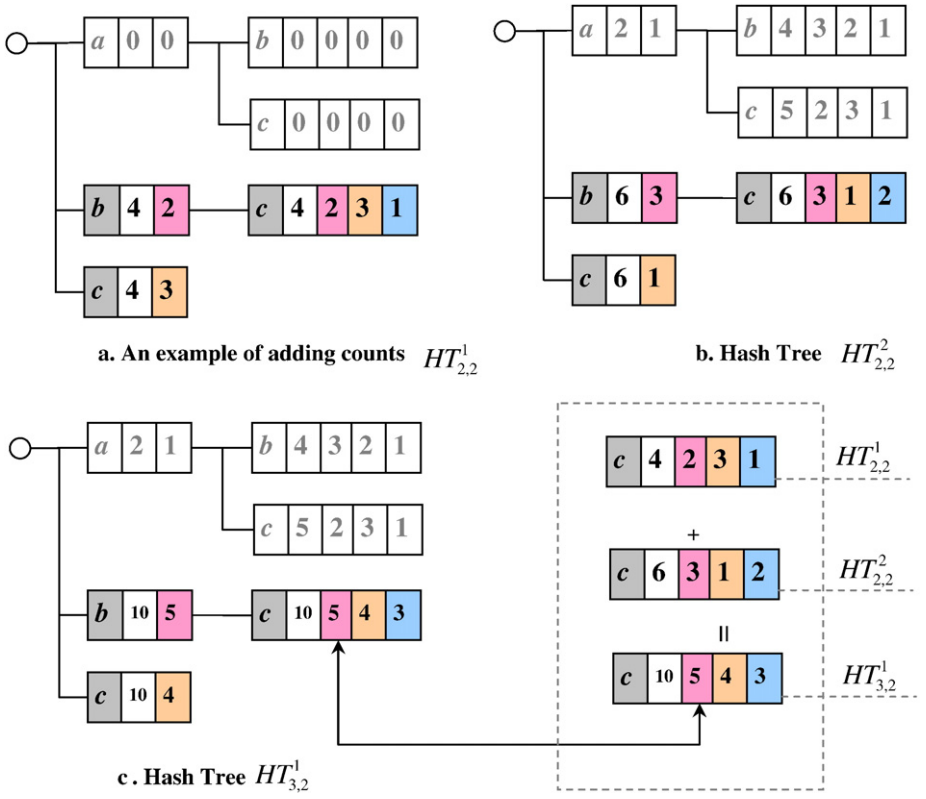


Fig. 7. An example of adding counts and combining tree.

[2] for traditional rules generation, and the store-chain association rules proposed by Chen, et al. [6]. For ease of reference, we use AR, SCAR, and CAR to represent the Apriori algorithm, the store-chain association rules algorithm, and the proposed context-based association rules algorithm, respectively, in presenting the results.

The first experiment is based on synthetic data sets, and, in the second experiment, we use a real sales data set of a supermarket chain in Taiwan. The experiments were carried out in the environment below:

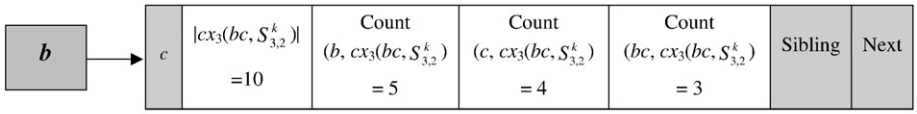
(1) Operation system: Windows XP.
(2) Hardware: Pentium 3.0G processor, 1024MB main memory.
(3) Tool: Borland C+ language and JBuilder language.

4.1. The first experiment

4.1.1. Data generation

In the first experiment, we randomly generated several synthetic transactional data sets, using the data generation algorithm proposed in [6] with some minor modifications. The parameters used in data generation are listed in Table 1, and the data generation process is described as follows.

First, we select several time and place concept hierarchies. For simplicity, we use the same number of granularity levels, H, and branching degrees, B, for the time and place hierarchies. Totally, we generate 8 data sets listed in Table 2. Consider the third data set (labeled as B3H4), for example, it has 4 higher



$$\chi(b \Rightarrow c) = count(bc, cx_3(bc, S^k_{3,2}))/\, count(b, cx_3(bc, S^k_{3,2})) = 3/5 = 0.6$$

$$\chi(c \Rightarrow b) = count(bc, cx_3(bc, S^k_{3,2}))/\, count(c, cx_3(bc, S^k_{3,2})) = 3/4 = 0.75$$

Fig. 8. Rule generation in a node.

Table 1
Parameters used in simulation

Symbol	Parameter
D	Number of transaction
Q	Number of stores
M	Number of time periods
R	Number of items
L	Average length of transactions
F_l	Average length of maximum potentially frequent itemsets
F_d	Maximum number of frequent itemsets
I_d	Replace rates of items
S_u, S_l	The maximum and minimum sizes of stores
H	The number of levels in the hierarchy
B	The branching degrees in the hierarchy

granularity levels and 3 branches at each granularity level. We further assume $t'=2$ and $p'=2$. As a result, the total number of contexts for the *TP* lattice for the data set is $(1+3+9)\times(1+3+9)=169$. Based on the hierarchies, the numbers of stores (q), and time units (m), are determined automatically as the number of leaf nodes. For data set B3H4, $q=m=27$.

Secondly, we determine the number of transactions in each context. The remaining part of data generation is the same as that used in [6]. We briefly describe it as follows. The store sizes are generated by a uniform distrubution between S_u and S_l. We assume that the total number of transactions and the number of products on shelf in a store are dependent on the store size. In addition, we allow the stores to have different product replacement ratios. In the data generation, these relationships are established by generating m random numbers for store i from a Poisson distribution with mean S_i, and we use the j-th number, denoted by W_{ij}, as the weight of store i in period j. Let D_{ij} denote the number of transactions in store i and period j. The total number of transactions, D, for the data sets are listed in the last column of Table 1 and distributed to the store i in period j, following this rule:

$$D_{ij} = \frac{D}{\sum_{m=0}^{P}\sum_{n=0}^{T} W_{mn}} \times W_{ij}.$$

Thirdly, we determine the number of products for each context. Assume that the number of product items for sale in a store is proportional to the square root of the store's size. Then, the number of products in store i, denoted by N_i, is determined by

$$N_i = \frac{r}{\mathrm{Max}\left(\sqrt{S_i}\right)} \times \sqrt{S_i}.$$

Note that the products sold in a store may change over time, although N_i is kept the same in all periods. Since I_d is the proportion of products that will be replaced in each period, store i replaces $N_i \times I_d$ products in each period. Furthermore, we follow the method used by Agrawal and Srikant [2] to generate the maximum number of frequent itemsets, F_d, with an average length of F_l.

Finally, for each context P_iT_j, we generate D_{ij} transactions. All the transactions in the data sets will be generated by a Poisson distribution with mean L and a series of maximum potentially frequent itemsets. If an itemset generated from the process has some items not sold at store i in period j, we remove these items, and then continue adding items to the transaction until we have reached the intended size. If the last itemset exceeds the limit of this transaction, we remove the part that exceeds the limit. When adding an itemset to a transaction, we use a "corruption level," $c=0.7$, to simulate the phenomenon that all the items in a frequent itemset do not always appear together. Information on how the corruption level affects the procedure of generating items for a transaction can be seen in the paper of Agrawal and Srikant [1]. The following parameter values are used in data generation: the average length of transactions=6, the average length of maximum potentially frequent itemsets=4, the maximum number of frequent itemsets=1,000, the number of items=1,000, the replacement rate=0.01, $S_u=100$, and $S_l=50$.

4.1.2. Results

We study the effects of the number of concept levels on the running time of and the number of rules generated by the three algorithms. In Fig. 9-a, the results of the running time are reported for three data sets, B2H3, B2H4, and B2H5, under several selected levels of support ranging from 0.1% to 1%. Note that the numbers of contexts associated with B2H3, B2H4 and

Table 2
Data sets

Data set	Branch degrees	Hierarchies	Number of Transaction	Number of Contexts
B2H3	2	3	100,000	9
B2H4	2	4	100,000	49
B2H5	2	5	100,000	225
B3H4	3	4	100,000	169
B4H3_D10	4	3	100,000	25
B4H3_D15	4	3	150,000	25
B4H3_D20	4	3	200,000	25
B4H3_D25	4	3	250,000	25

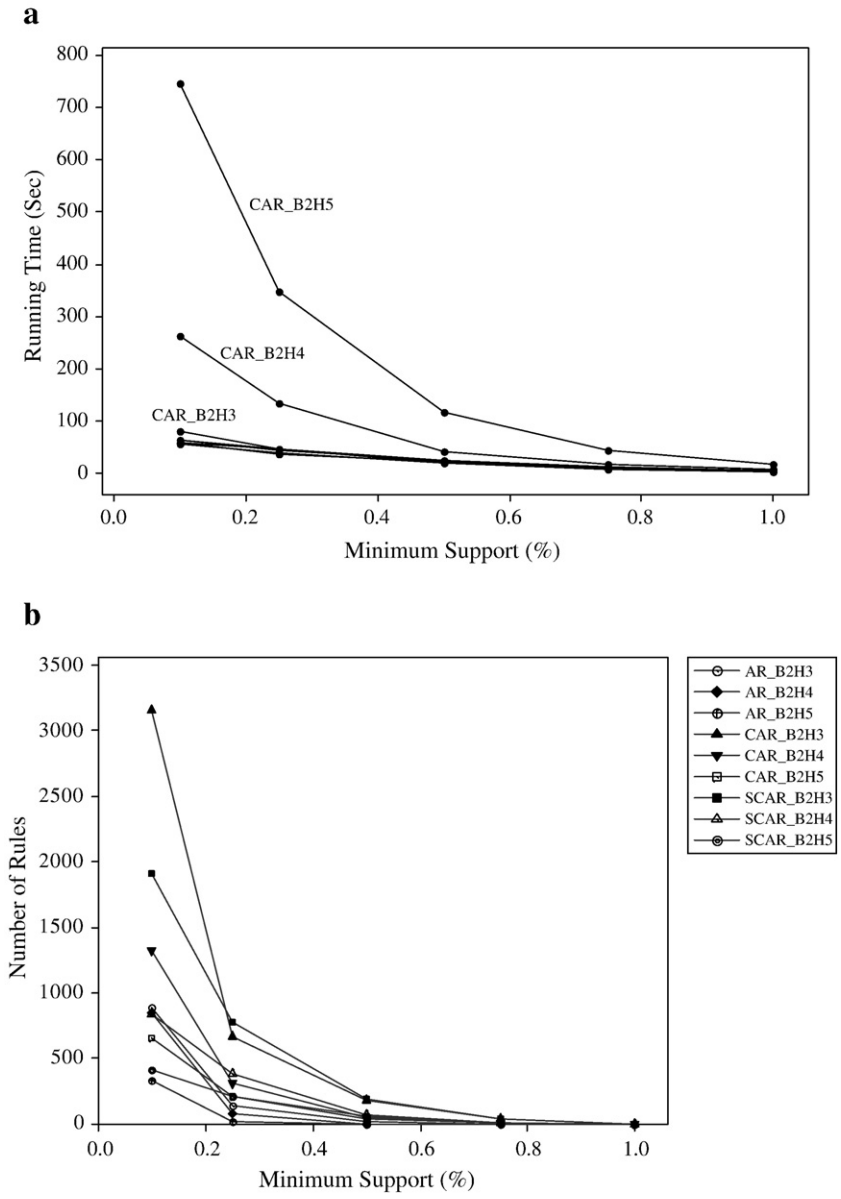


Fig. 9. a. Running Time of three algorithms. b. Number of rules generated by three algorithms.

B2H5 are 9, 49, and 225, respectively. Since the AR and SCAR algorithms do not generate rules based on contexts, their running times are not affected significantly by the number of concept levels. We found that the AR algorithm requires slightly less time than the SCAR algorithm. On the other hand, the running times of the CAR algorithm increase very rapidly as the number of concept levels increases from 3 to 5. This is expected because the CAR algorithm generates rules for each context. Furthermore, for data set B2H5, the CAR algorithm has to generate rules for 225 contexts. However, for the minimum support of 0.1%, its running time is only 12 times of that of the AR algorithm. These results further support that the CAR algorithm is efficient.

In Fig. 9-b, we report the numbers of rules generated by the three algorithms. Because the CAR algorithm generates rules for each context, the average number of rules per context is reported, while the total numbers of rules are reported for the other two algorithms. As expected, the results show that the numbers of rules increase as the minimum support decreases for all the

three algorithms. Furthermore, the numbers of rules generated by the three algorithms are very close when the minimum support is larger than or equal to 0.25%, but a large separation is observed when the minimum support is at 0.1%. This result suggests that many itemsets in the data sets have a low frequency. For example, the number of rules generated by the AR algorithm for B2H3 is 146 when the minimum support is .25%. Since the SCAR and CAR algorithms use smaller bases for calculating the support, we expect they produce larger numbers of rules. As the figure shows, when the minimum support is set at a low value, the numbers of rules generated by the three algorithms are very different.

The second part of the evaluation is on the effects of the complexity of the concept hierarchies on the performance of the CAR algorithm. We applied the CAR algorithm for data sets B2H3, B4H3, B2H4, B3H4, and B2H5. The results are reported in Fig. 10. Note that the numbers of the contexts associated with these data sets are, respectively, 9, 25, 49, 169, and 225. As expected, the running time increases as the number of contexts increases or the minimum support decreases. The results also indicate that the running time of the CAR algorithm does not increases as fast as the number of contexts does. For example, when the minimal support is 0.1%, the running time increases from 83 s to 740 s while the number of contexts increases from 9 to 225. The reason for this encouraging result is because in phase 2 of the CAR algorithm, the hash tree of a larger context is obtained by combining the hash trees of smaller contexts without additional database scans. This result justifies using hash trees of the same structure in the algorithm to store the count information for all contexts.

Finally, we study the effects of the data size on the performance of the CAR algorithm. We use the algorithm for three databases generated from B4H3, with the sizes ranging from 10 k to 1000 k. In Fig. 11, we show the running times of the algorithm for the database under minimum supports from 0.25% to 1.00%. As expected, the time increases as the database size increases. However, the results also suggest that the algorithm is efficient because the running time does not increase proportionally with respect to the size of the database.

4.2. Results with SC-POS data set

We evaluate the algorithm by using a real data set, called SC-POS, which is the sales data of a chain supermarket in Taipei, Taiwan. The transaction data of the SC-POS were obtained from twenty stores between July 1, 2002 and November 28, 2002. Each transaction in the SC-POS data set is a customer's shopping record, including the purchase date, time, and purchased items. A series of data pre-processing and cleaning tasks were performed, including generating the *SI* table. The final data set contains 123,323 transactions. Since items' on-shelf and off-shelf times are not available, we generate the *SI* table by assuming that an item is off-shelf if it is not sold in three days or longer. In Fig. 12, we show the running time of the CAR algorithm for the real data set under minimum support from 0.1% to 1.0%. The results indicate that the algorithm can have a

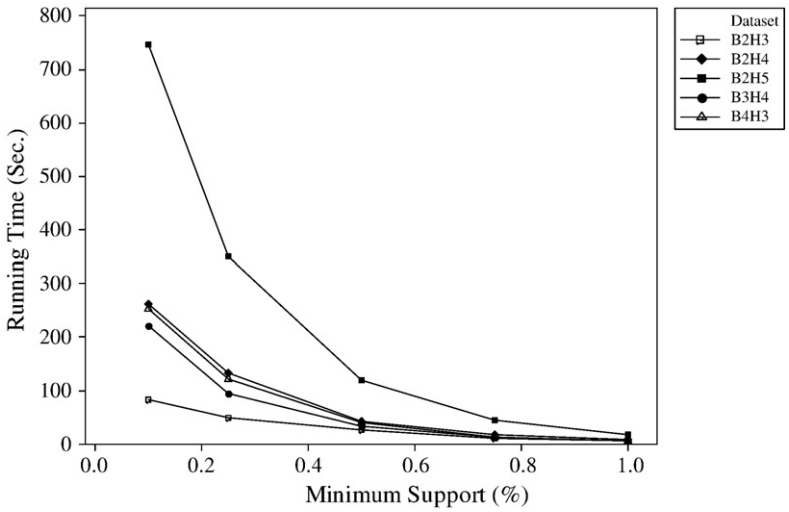


Fig. 10. The effects of the number of contexts on running time.

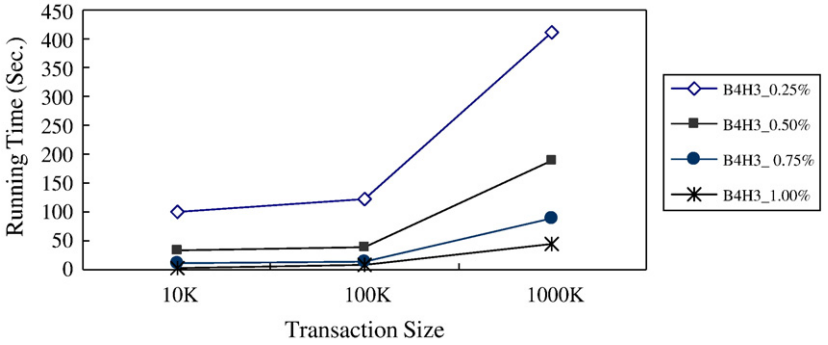


Fig. 11. The effects of the data size on running time.

reasonable running time even when the minimum support is low.

Because of the stores' homogeneity in location, we only use a time concept hierarchy in our analysis. The time concept hierarchy is defined as: year, quarter, and month. For comparison, we apply the SCAR and CAR algorithms to the data set and set the minimum support and confidence at 0.5% and 10%, respectively. Table 3 includes three cases where interesting differences were observed in the experiment.

The results in the first case suggest that although a rule may be extracted by both the SCAR and CAR algorithms, the CAR rules may carry more precise and richer information. For example, "Cakes⇒Milk" was obtained by both the algorithms. From the SCAR rule, we can only explain it in its overall context. In contrast, from the CAR rule, we know that the same rule is applicable in various different time-intervals, from monthly intervals to quarterly intervals. As shown in the table, the rule's support is stronger in October and November, and its confidence level is higher in July and August. In the second case, the rule is generated by SCAR, but it does not hold in CAR in several time periods. As shown in the table, the SCAR rule "ice cream⇒soft drink" held from July to November. However, the sales of these two items declined after October, which was the beginning of fall. The results of CAR show that the rule did not hold in October and November. In the third case, the SCAR rule "Health

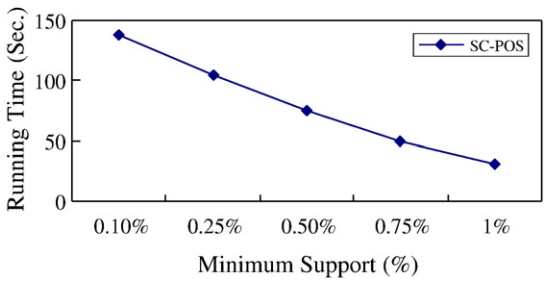


Fig. 12. Running Time of the real data set SC-POS.

Table 3
Results with SC-POS data set

Case	Rules	
1	SCAR	Cakes⇒Milk [(P1, 2002/7/1∼2002/11/28)], [support=0.56%, confidence=27.7%]
	CAR	P1, 2002/7: [Cakes⇒Milk (support=0.55%, confidence=32.57%)] P1, 2002/8: [Cakes⇒Milk (support=0.55%, confidence=32.93%)] P1, 2002/9: [Cakes⇒Milk (support=0.50%, confidence=25.54%)] P1, 2002/10: [Cakes⇒Milk (support=0.62%, confidence=26.07%)] P1, 2002/11: [Cakes⇒Milk (support=0.60%, confidence=25.36%)] P1, 2002/3rd quarter: [Cakes⇒Milk (support=0.53%, confidence=29.7%)]
2	SCAR	Ice-cream⇒soft drink [(P1, 2002/7/1∼2002/11/28)], [support=0.56%, confidence=10.1%]
	CAR	[a]P1, 2002/10: [Ice-cream⇒soft drink (support=0.27%, confidence=5.5%)] [a]P1, 2002/11: [Ice-cream⇒soft drink (support=0.14%, confidence=3.7%)]
3	SCAR	[a]Health Food⇒Vegetable [(P1, 2002/7/1∼2002/11/28)], [support=0.49%, confidence=26.2%]
	CAR	P1, 2002/10: [Health Food⇒Vegetable (support=0.55%, confidence=24.2%)] P1, 2002/11: [Health Food⇒Vegetable: support=0.78%, confidence=32.6%)]

[a] The rule does not meet the support requirement, the confidence requirement, or both.

Head	$\|cx_3(X, S^k_{i,j})\|$	Count $(y_1, cx_3(X, S^k_{i,j}))$	Count $(y_2, cx_3(X, S^k_{i,j}))$	.....	Count $(y_j, cx_3(X, S^k_{i,j}))$	Sibling	Next

Fig. A-1. An example of the node structure

1. HTree($HT, S^k_{t',p'}$) {
2. $HT^k_{t',p'} \leftarrow$ ***HT*** ;
3. For each $S^i_{1,1} \subseteq S^k_{t',p'}$ {
4. ***ForSale*** = check the *SI* table to get the items which are for sale in $cx_3(S^i_{1,1})$;
5. ***iForSale*** = ***PL*** ∩ Powerset(***ForSale***) ;
6. For each transaction in $S^i_{1,1}$
7. get the counts of the itemset in ***iForSale*** ;
8. add these counts to the corresponding node in $HT^k_{t',p'}$ }

Fig. A-2. Function HTree($HT, S^k_{t',p'}$).

Food"⇒Vegetable" does not have sufficient support from July to November. However, these two items are commonly consumed together during the winter in Taiwan. The rule is identified as a CAR rule in October and in November.

5. Conclusion

We propose a new approach for extracting association rules from transactional records in a multiple-store and multiple-period environment. For implementing the algorithm, concept hierarchies of time and place (location) are defined first for forming a systemic structure of contexts. The proposed algorithm can efficiently extract the rules which meet the support and confidence requirements for all the contexts derived from the time and place hierarchies.

The numerical evaluation of the proposed method shows that the algorithm is efficient in running time and may generate more specific and richer information than the store-chain rules and the traditional rules.

The paper has several possible extensions. The first is to consider using clustering and segmentation tools to building concept hierarchies or the *TP* lattice. The obvious benefit of the approach is to relieve the user from providing adequate time and place hierarchies. More importantly, the result could simplify the hierarchy structure and may significantly reduce the number of contexts. The next possible extension is to develop an item conceptual hierarchy or tree and base on which to enhance the expressions of the rules. Finally, we can apply the context hierarchy structure in extracting sequential rules in a multiple-store environment.

Appendix

We give the following detailed information on several key steps of the algorithm.

[1]. Hash tree

A specific structure, hash tree HT, is used to store the count information of itemsets in PL. In different sections, the hash tree will have different count information on

1. **Combine(*i,j,k*)** {
2. build a tree $HT^k_{i,j} \leftarrow HT$
3. if (***j***==*p'*) {
4. For all *k'* satisfying ($HT^{k'}_{i-1,j} \subset HT^k_{i,j}$) $HT^k_{i,j} \leftarrow HT^k_{i,j} + HT^{k'}_{i-1,j}$;
5. }
6. else {
7. For all *k'* satisfying ($HT^{k'}_{i,j-1} \subset HT^k_{i,j}$) $HT^k_{i,j} \leftarrow HT^k_{i,j} + HT^{k'}_{i,j-1}$;
8. }
9. }

Fig. A-3. Function *combine*(*i,j,k*).

1. For each itemset X in $S^k_{i,j}$ {
2. $\xi_C(X, cx_2(S^k_{i,j})) = \frac{count(X, cx_2(S^k_{i,j}))}{|cx_2(S^k_{i,j})|}$;
3. If $\xi_C(X, cx_2(S^k_{i,j})) > \delta_S$
4. then add X into $CL^k_{i,j}$; }

A-4. Finding *CL* itemsets.

For each itemset X in $CL^k_{i,j}$ {

$X = A \cup B$; $A \subseteq X$; $B \subseteq I/X$;

$\chi(A \Rightarrow B) = \frac{count(AB, cx_3(X, S^k_{i,j}))}{count(A, cx_3(X, S^k_{i,j}))}$;

if $\chi(A \Rightarrow B) > \delta_C$,

then add { $S^k_{i,j}$, $A \Rightarrow B$, $\chi(A \Rightarrow B)$} into $SR^k_{i,j}$; }

A-5. Finding section rules.

Fig. A-4 Finding *CL* itemsets. A-5 Finding section rules.

each node. However, all these hash trees have the same structure. This property facilitates us to combine the count information along the broadcasting path.

We build a hash tree HT according to all itemsets in PL, and each itemset in PL will have a counterpart node in HT. The labels of the path from the root to a node is denoted by $X = \{i_1, i_2, i_3 \ldots i_k\}$, where i_j is the label of the j-th node in the path and i_k is the label of this node.

The information maintained in each node is $|cx_3(X, S^k_{i,j})|$ and count($y_j, cx_3(X, S^k_{i,j})$) for $\forall y_j \subseteq X$; so, the size of a node depends on its depth in the tree. Fig. A-1 contains an example of the structure of a node in a hash tree. We use this tree structure to store the count information of each itemset in section $S^k_{i,j}$ into the corresponding node in $HT^k_{i,j}$.

[2]. The function HTree (HT,$S^k_{t',p'}$)

Considering the algorithm shown in Fig. 3, after phase 0 of the procedure, shown from line 3 to line 5, we have a hash tree HT constructed from PL.

Based on PL and the *SI* table, we can evaluate function HTree($HT, S^k_{t',p'}$) in phase 1. The detailed steps are shown in Fig. A-2.

The purpose of this procedure is to obtain the actual count of each itemset in PL in section $S^k_{t',p'}$ and store it in hash tree $HT^k_{t',p'}$, so that we can calculate the actual support and confidence more efficiently. To this end, the procedure will repeatedly add the counts of every unit section $S^2_{1,1} \subseteq S^k_{t',p'}$ into $HT^k_{t',p'}$. For each unit section $S^i_{1,1}$, it only adds the counts to the nodes where corresponding itemsets are on shelf.

[3]. The function combine (i,j,k)

To have a better performance, we have to discover the rules in a lower granularity without additional scans of the database. To this end, we can build a hash tree in a lower granularity level by combining hash trees associated with higher granularity levels. The procedure is shown in Fig. A-3.

Let us use $HT^{k'}_{i,j-1} \subset HT^k_{i,j}$ to indicate the contexts of $HT^{k'}_{i,j-1}$ are a subset of those of $HT^k_{i,j}$. In this case, we say $HT^{k'}_{i,j-1}$ is a child tree of $HT^k_{i,j}$. We can combine all child trees of $HT^k_{i,j}$ to form $HT^k_{i,j}$. In step 4 of the procedure, $HT^k_{i,j} \leftarrow HT^{k_1}_{i,j-1} + HT^{k_2}_{i,j-1}$ means $HT^{k_1}_{i,j-1}$ and $HT^{k_2}_{i,j-1}$ combine to obtain $HT^k_{i,j}$. And the *combine* operation refers to summing the support counts stored in corresponding nodes in all trees involved. Specifically, the operation is done by $|cx_3(X, S^k_{i,j})| = \sum_{k'} |cx_3(X, S^{k'}_{i,j-1})|$ and $count(y_i, cx_3(X, S^k_{i,j})) = \sum_{k'} count(y_i, cx_3(X, S^{k'}_{i,j-1}))$.

[4]. Finding CL itemset and section rules

After the first two phases, we have the count information. Thus, we use the information stored in $HT^k_{i,j}$ to compute the context supports of the itemsets in $S^k_{i,j}$ and then compute the confidence of the rules. If the confidences of the rules exceed a given threshold δ_C, then we add the rules into $SR^k_{i,j}$. The main process is shown in Figs. A-4 and 5.

References

[1] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, Proceedings of the ACM SIGMOD International Conference on Management of Data, (SIGMOD '93), Washington DC, ACM New York, NY, USA, 1993, pp. 207–216.

[2] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, Proceedings of the 20th Very Large DataBases Conference (VLDB'94), Chili, Santiago, 1994, pp. 487–499.

[3] Terri C Albert, Paulo B Goes, Alok Gupta, GIST: a model for design and management of content and interactivity of customer-centric web sites, MIS Quarterly 28 (2) (2004) 161–183.

[4] Y.L. Chen, J.M. Chen, C.W. Tung, A data mining approach for retail knowledge discovery with consideration of the effect of shelf-space adjacency on sales, Decision Support Systems 42 (3) (2006) 1503–1520.
[5] M.-S. Chen, J. Han, P.S. Yu, Data mining: an overview from a database perspective, IEEE Transactions on Knowledge and Data Engineering 8 (6) (1996) 866–883.
[6] Y.L. Chen, K. Kang, R.J. Shen, Y.H. Hu, Market basket analysis in a multiple store environment, Decision Support Systems 40 (2) (2005) 339–354.
[7] L. Churilov, A. Bagirov, D. Schwartz, et al., Data mining with combined use of optimization techniques and self-organizing maps for improving risk grouping rules: application to prostate cancer patients, Journal of Management Information System 21 (4) (2005) 85–100.
[8] J. Han, M. Kamber, Data Mining, Morgan Kaufmann, San Francisco, 2001.
[9] E. Kim, W. Kim, Y. Lee, Combination of multiple classifiers for the customer's purchase behavior prediction, Decision Support Systems 34 (2) (2003) 167–175.
[10] Y.S. Kim, W.N. Street, An intelligent system for customer targeting: a data mining approach, Decision Support Systems 37 (2) (2004) 215–228.
[11] N. Kumar, A. Gangopadhyay, G. Karabatis, Supporting mobile decision making with association rules and multi-layered caching, Decision Support Systems 43 (1) (2007) 16–30.
[12] I.S.Y. Kwan, J. Fong, H.K. Wong, An e-customer behavior model with online analytical mining for internet marketing planning, Decision Support Systems 41 (1) (2005) 189–204.
[13] Q.Y. Lin, Y.L. Chen, J.S. Chen, Y.C. Chen, Mining inter-organizational retailing knowledge for an alliance formed by competitive firms, Information and Management 40 (5) (2003) 431–442.
[14] X.B. Li, A scalable decision tree system and its application in pattern recognition and intrusion detection, Decision Support Systems, 41 (1) (2005) 112–130.
[15] D.R. Liu, Y.Y. Shih, Integrating AHP and data mining for product recommendation based on customer lifetime value, Information & Management 42 (3) (2005) 387–400.
[16] S.S. Robbins, A.C. Stylianou, Global corporate web sites: an empirical investigation of content and design, Information & Management 40 (3) (2003) 205–212.
[17] R. Sarathy, K. Muralidhar, The security of confidential numerical data in databases, Information Systems Research, 13 (4) (2002) 389–403.
[18] M.J. Shaw, C. Subramaniam, G.W. Tan, M.E. Welge, Knowledge management and data mining for marketing, Decision Support Systems 31 (1) (2001) 127–137.
[19] S. Spangler, J.T. Kreulen, Lessler, J. Li, Xb, Author, Reprint Author Xiao-Bai Li, Xiao-Bai, S. Sarkar, Et Al, Privacy protection in data mining: A perturbation approach for categorical data, Information Systems Research 17 (3) (2006) 254–270.
[20] K.Y. Tam, S.Y. Ho, Web personalization as a persuasion strategy: an elaboration likelihood model perspective, Information Systems Research 16 (3) (2005) 271–291.
[21] S. Viaene, G. Dedene, R.A. Derrig, Auto claim fraud detection using Bayesian learning neural networks, Expert Systems with Applications 29 (3) (2005) 653–666.
[22] H. Wang, A.S. Weigend, Data mining for financial decision making, Decision Support Systems 37 (4) (2004) 457–460.
[23] Y. Xue, De. Brown, Spatial analysis with preference specification of latent decision makers for criminal event prediction, Decision Support Systems 41 (3) (2006) 560–573.

Kwei Tang is the Allison and Nancy Schleicher Chair of Management and Associate Dean for Programs and Student Services at the Krannert School of Management, Purdue University. He received a BS from National Chiao Tung University, Taiwan, an MS from Bowling Green State University, and a PhD in management science from Purdue University. He teaches data mining, quality management, applied statistics, and research methods. His current research interests are data mining, on-line process control, integration of quality functions in a manufacturing organization, e-commerce, and supply chain management.

Yen-Liang Chen is a Professor of Information Management at National Central University of Taiwan. He received his PhD degree in computer science from National Tsing Hua University, Hsinchu, Taiwan. His current research interests include data modeling, data mining, data warehousing and operations research. He has published papers in Decision Support Systems, Operations Research, IEEE Transactions on Software Engineering, Computers & OR, European Journal of Operational Research, Information and Management, Information Processing Letters, Information Systems, Journal of Operational Research Society, and Transportation Research.

Hsiao-Wei Hu is currently a PhD student in the Department of Information Management, National Central University, Taiwan. She received the MS degree in information management from National Central University of Taiwan. Her research interests include data mining, information retrieval and EC technologies. Currently she is focusing on classification techniques.