Available online at www.sciencedirect.com

SCIENCE DIRECT®

Decision Support Systems 40 (2005) 339–354

Decision Support Systems

www.elsevier.com/locate/dsw

Market basket analysis in a multiple store environment

Yen-Liang Chen[a], Kwei Tang[b,*], Ren-Jie Shen[a], Ya-Han Hu[a]

[a] Department of Information Management, National Central University, Chung-Li, 320 Taiwan, ROC
[b] Krannert Graduate School of Management, Purdue University, West Lafayette, IN 47907, USA

Received 1 December 2003; received in revised form 5 April 2004; accepted 5 April 2004
Available online 2 June 2004

Abstract

Market basket analysis (also known as association-rule mining) is a useful method of discovering customer purchasing patterns by extracting associations or co-occurrences from stores' transactional databases. Because the information obtained from the analysis can be used in forming marketing, sales, service, and operation strategies, it has drawn increased research interest. The existing methods, however, may fail to discover important purchasing patterns in a multi-store environment, because of an implicit assumption that products under consideration are on shelf all the time across all stores. In this paper, we propose a new method to overcome this weakness. Our empirical evaluation shows that the proposed method is computationally efficient, and that it has advantage over the traditional method when stores are diverse in size, product mix changes rapidly over time, and larger numbers of stores and periods are considered.
© 2004 Elsevier B.V. All rights reserved.

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

1. Introduction

Because of advances in information and communication technologies, corporations can effectively obtain and store transactional and demographic data on individual customers at reasonable costs. One of the challenges for corporations that have invested heavily in customer data collection is how to extract important information from their vast customer databases in order to gain competitive advantage. Market basket analysis (also known as association rule mining) is a method of discovering customer purchasing patterns by extracting associations or co-occurrences from stores' transactional databases. Discovering, for example, that supermarket customers are likely to purchase milk, bread, and cheese together, or that bank customers are likely to use a set of services jointly, can help managers in designing store layout, web sites, product mix and bundling, and other marketing strategies.

The methodology was introduced by Agrawal et al. [2] and can be stated as follows. Given 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 to select the association rules. Support is a measure of how often the transactional records in the database contain both X and Y,

* Corresponding author. Tel.: +1-765-494-4464; fax: +1-765-494-9658.
E-mail address: ktang@mgmt.purdue.edu (K. Tang).

0167-9236/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2004.04.009

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. By far, the Apriori algorithm [1] is the most known algorithm for mining the association rules from a transactional database, which satisfy the minimum support and confidence levels specified by users.

Since association rules are useful and easy to understand, there have been many successful business applications, including, for example, finance, telecommunication, marketing, retailing, and web analysis [5]. The method has also attracted increased research interest, and many extensions have been proposed in recent years, including (1) algorithm improvements [6,12,18,21]; (2) fuzzy rules [13,14]; (3) multi-level and generalized rules [7,10]; (4) quantitative rules [20,24,25]; (5) spatial rules [7,15]; (6) inter-transaction rules [19]; (7) interesting rules [4,9]; and (8) temporal association rules [3,16,17]. Brief literature reviews of association rules are given by Chen et al. [8] and Han and Kamber [11].

In today's business world, it is common for a company to have subsidiaries, branches, dealers, or franchises in different geographical locations. For example, Wal-Mart, the largest supermarket chain in the world, has more than 4400 stores worldwide. For a company with multiple stores, discovery of purchasing patterns that may vary over time and exist in all, or in subsets of, stores can be useful in forming marketing, sales, service, and operation strategies at the company, local, and store levels.

There are two main problems in using the existing methods in a multi-store environment. The first is caused by the temporal nature of purchasing patterns. An apparent example is seasonal products. Temporal rules [3,16,17] are developed to overcome the weakness of the static association rules that either find patterns at a point of time or implicitly assume the patterns stay the same over time and across stores. A literature review on temporal rules is given by Roddick and Spiliopoulou [22]. In temporal rules, selling periods are considered in computing the support value, where the selling period of a product is defined as the time between its first and last appearances in the transaction records. Furthermore, the common selling period of the products in a product set is used as the base in computing the "temporal support" of the product set. The results of the method may be biased, however, because a product may be on shelf before its first transaction and/or after the last transaction occurs, and a product may also be put on-shelf and taken off-shelf multiple times during the data collection period.

The second problem is associated with finding common association patterns in subsets of stores. Similar to the problem in using existing temporal rules in a multi-store environment, we have to consider the possibility that some products may not be sold in some stores, for example, because of geographical, environmental, or political reasons. This is seemingly related to spatial association rules. However, the focus of spatial rules is on finding the association patterns that are related to topological or distance information in, for example, maps, remote sensing or medical imaging data and VLSI chip layout data [23].

To overcome the problems, we develop an Apriori-like algorithm for automatically extracting association rules in a multi-store environment. The format of the rules is similar to that of the traditional rules. However, the rules also contain information on store (location) and time where the rules hold. The results of the proposed method may contain rules that are applicable to the entire chain without time restriction or to a subset of stores in specific time intervals. For example, a rule may state: "In the second week of August, customers purchase computers, printers, Internet and wireless phone services jointly in electronics stores near campus." Another example is: "In January, customers purchase cold medicine, humidifiers, coffee, and sunglasses together in supermarkets near skiing resorts." These rules can be used not only for general or localized marketing strategies, but also for product procurement, inventory, and distribution strategies for the entire store chain. Furthermore, we allow an item to have multiple selling time periods; i.e., an item may be put on-shelf and taken off-shelf multiple times. We further assume that different stores can have different product-mixes in different time periods. That is, each store can have its own product-mix, and the product-mix in a store can be dynamically changed over time.

Because the time and store (location) factors are considered, the rule generation procedure is more complicated than the Apriori algorithm. The simulation results presented in the paper show that the proposed method is computationally efficient and has significant advantage over the traditional association method when the stores under consideration are diverse in size and have product mixes that change rapidly over time.

The paper is organized as follows. We formally define the problem in Section 2 and in Section 3 propose an algorithm. In Section 4, we compare the results generated from the proposed algorithm and the traditional Apriori algorithm in a simulated multi-store environment. The conclusion is given in Section 5.

2. Problem definition

We consider a market basket database D that contains transactional records from multiple stores over time period T. Our objective is to extract the association rules from the database. For convenience in presentation, the cardinal of a set, say Σ, is denoted by $|\Sigma|$. Let $I=\{I_1, I_2,\ldots, I_r\}$ be the set of product items included in D, where I_k $(1\leq k\leq r)$ is the identifier for the kth item. Let X be a set of items in I. We refer X as a k-itemset if $|X|=k$. Furthermore, a transaction, denoted by s, is a subset of I. We use $W(X, D)=\{s\,|\,s\in D\wedge X\subseteq s\}$ to denote the set of transactions in D, which contain itemset X.

Definition 1. The *support* of X, denoted by *sup*(X, D), is the fraction of transactions containing X in database D; i.e., $sup(X, D)=|W(X, D)|/|D|$. For a specified support threshold σ_s, X is a *frequent itemset* if $sup(X, D)\geq\sigma_s$.

Note that the definitions of the support and the frequent itemset are those used in the traditional association rules, and, therefore, the store and time information is not considered in determining the support of an itemset.

Let $\{T_1, T_2,\ldots, T_m\}$ be the set of mutually disjoint time intervals (periods) and form a complete partition of T. Furthermore, they are ordered, such that T_{i+1} immediately follows T_i for $i\geq 1$. Note that the time periods are defined according to specific needs of the problem, such as 1 h, 6 h, 1 day, 1 week, and so on. Let $P=\{P_1, P_2,\ldots, P_q\}$ be the set of stores, where P_j $(1\leq j\leq q)$ denotes the jth store in the store chain. We assume that each transaction s in D is attached with a timestamp, t, and store identifier, p, to indicate the store and time that the transaction occurs.

Let $S_k\subseteq P$ and $R_k\subseteq T$ be the sets of the stores and times that item I_k is sold, respectively. We define $V_{I_k}=S_k\times R_k$ as the *context* of item I_k; i.e., the set of the combinations of stores and times where item I_k is sold. Furthermore, the context of itemset X, denoted by V_X, is the set of the combinations of stores and times that all items in X are sold concurrently. For example, if itemset X consists of two items I_k and $I_{k'}$, the context of X is given by $V_X=V_{Ik}\cap V_{ik'}$.

Definition 2. Let X be an itemset in I with context V_X, and D_{V_X} the subset of transactions in D whose timestamps t and store identifiers p satisfy V_X. We define the *relative support* of X with respect to the context V_X, denoted by *rel_sup*(X, D_{V_X}), as $|W(X, D_{V_X})|/|D_{V_X}|$. For a given threshold for relative support σ_r, if a frequent itemset X satisfies $rel\_sup(X, D_{V_X})\geq\sigma_r$, we call X a *relative-frequent* (*RF*) *itemset*.

In the last definition, we require that a relative-frequent itemset X be frequent. We add this restriction for two reasons. First, it enables us to preserve the well-known downward-closure property, by which the candidate set of the next phase can be obtained by joining the frequent sets of the preceding phase; this will greatly improve the performance of the algorithm. Second, this restriction does not present any real problem to the mining algorithm, because none of the important patterns would be missing because of using a low σ_s value. Therefore, we prefer using a low σ_s value. However, it should not be too low because an itemset that occurs only in few transactions has no practical significance.

Furthermore, the minimum threshold for the relative support of an itemset is used to determine whether a sufficient percentage of transactions exists in its context to warrant the inclusion of the itemset as a relative-frequent (RF) itemset. Its use and purpose are similar to those of the traditional minimum support threshold. Consequently, we can set its value the same way as we set the traditional minimum support threshold.

Definition 3. Consider two itemsets X and Y. The relative support of X with respect to the context $V_{x\cup y}$, denoted by *rel_sup*(X, $D_{V_{X\cup Y}}$), is defined as $|W(X, D_{V_{X\cup Y}})|/|D_{V_{X\cup Y}}|$. The *confidence* of rule $X\Rightarrow Y$, denoted by $conf(X \Rightarrow Y)$, is defined as $rel\_sup(X \cup Y, D_{V_{X \cup Y}}) / rel\_sup(X, D_{V_{X \cup Y}})$.

The above definition implies that the context of rule $X \Rightarrow Y$ is $V_{x \cup y}$; i.e., the base used to compute the confidence of rule $X \Rightarrow Y$ is the common stores and time periods shared by all the items in $X \cup Y$.

Definition 4. Let Z be an RF itemset, where $Z = X \cup Y$, $X \subseteq I$, and $Y \subseteq I \backslash X$. Given a confidence threshold σ_c, if $conf(X \Rightarrow Y) \geq \sigma_c$, we call $X \Rightarrow Y$ a *store-chain* (*SC*) *association rule*, and $V_{x \cup y}$ as the *context* of the rule.

Based on Definitions 1 and 4, it is clear that the selection criteria and outputs for the store-chain association rules are different from those of the traditional association rules. For the store chain rules, the output includes the confidence, support, and a context indicating the stores and times the rules hold.

It can be shown that the traditional method underestimates the support and the confidence values (a proof is given in Appendix A). Consequently, important purchasing patterns that satisfy the criteria of the SC association rules may not be identified by the traditional association-rule methods.

3. Algorithm

We propose an Apriori-like algorithm for mining the store-chain association rules. The algorithm is outlined in Fig. 1. We first explain the general concept for developing the algorithm and then use five subsections to give detailed information on several key steps of the algorithm.

In describing the algorithm, we use RF_k to denote the set of all relative-frequent k-itemsets; F_k, the set of all frequent k-itemsets; and C_k, the set of candidate k-itemsets. Note that, in the traditional Apriori algorithm, a k-item candidate itemset must be a combination of $k-1$ frequent itemsets because of the anti-monotone property [1]. Therefore, the Apriori algorithm can generate the candidate itemsets in the kth phase by joining the frequent itemsets in the $(k-1)$th phase. However, for the SC association rule, a subset of an RF itemset may not be an RF itemset because the base for calculating the relative support value varies in different phases. Consequently, in the proposed algorithm, we generate candidate itemsets from the frequent itemsets, instead of the RF itemsets. Furthermore, when we use the frequent itemsets to generate the candidate set in the next phase, it still satisfies the anti-monotone property, because we use the same base to compute the supports for all itemsets.

As the first step of the algorithm, we build a table, called the *PT table*, for each item in I to associate the item with its context (i.e., the stores and times it is sold) and use the table to determine the context of an itemset. The algorithm proceeds in phases, where in the kth phase we generate F_k from C_k and RF_k from F_k. In the first phase, we scan the database for the first time and build a two-dimensional table, called the *TS table*. In this table, the entry at the position corresponding to T_i and P_j, denoted by $TS(T_i, P_j)$, records the number of transactions that occur at store P_j in period T_i. Using this table and the PT table for a given itemset X, we can determine the number of transactions associated with the context of X, i.e., $|D_{V_X}|$. In the kth phase of the algorithm, we first derive C_k, and, then, generate F_k by evaluating their supports, which can be done by scanning the database and removing all infrequent itemsets. Since an RF itemset must be a frequent itemset, we generate RF_k from F_k by evaluating the relative supports of the itemsets X in F_k.

In the following subsections, we give detailed descriptions for the key elements of the algorithm, including methods of (1) building the PT table, (2) building the TS table in the first phase, (3) finding RF_k, (4) generating candidate itemsets, and (5) generating the store-chain association rules.

3.1. The PT table

The purpose of the PT table is to efficiently store the time and store information for each product item in the database. We use a simple example to illustrate the procedure for constructing the table. Consider the bit matrices in Fig. 2 for items I_1, I_2, and I_3, in which there are six stores and six selling periods, and "1" and "0" indicate, respectively, that the item is or is not for sale in the corresponding store and time.

Because an item normally does not switch between on- and off-shelf very frequently in a typical application, we store an item's context information in the PT table instead of the bit matrix in order to conserve data storage space. In the PT table, we need only to record

```
1.  For each $I_m$, construct the PT table;
2.  For all transactions $s$ in database $D$
3.      { add 1 to the support count of 1-itemset $c$ contained in $s$;
4.        increase the value at $TS(T_i,P_j)$ by 1
          where the time of $s$ is in $T_i$ and the store of $s$ is at $P_j$; }
5.  $F_1$={$c$| $c$.count / |$D$| $\geq \sigma_s$};
6.  For ($k$=2; $F_{k-1} \neq \varnothing$ ; $k$++)
7.      {$C_k$=$CandGen(F_{k-1})$;
8.      If $k > 2$ then for each $z$ in $RF_{k-1}$ built its PT table;
9.      For all transactions $s$ in $D$
10.         {add 1 to the support counts of candidates $c$ in $C_k$ contained in $s$;
11.         add 1 to the support counts of all subsets $x$ of each $z$ in $RF_{k-1}$ w.r.t. $V_Z$ ;}
12.     $F_k$={$c$ in $C_k$ and ($c$.count / |$D$| ) $\geq \sigma_s$};
13.     $RF_k$={$g$ in $F_k$ and ($g$.count / |$D_{V_g}$|) $\geq \sigma_r$};
14.     compute the confidence of $x \Rightarrow y$ where $x \cup y$ in $RF_{k-1}$;
15.     For each ($x \cup y$) in $RF_{k-1}$
16.         {if $conf(x \Rightarrow y) \geq \sigma_c$
17.             output $x \Rightarrow y$ with context $V_{X \cup Y}$ ;}
18.     }
```

Fig. 1. Algorithm Apriori_TP.

	Item I_1						**Item I_2**						**Item I_3**					
Time / **STORE**	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**
P_1	1	1	1	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1
P_2	0	1	1	1	0	0	1	1	0	0	0	0	1	1	1	0	0	1
P_3	1	1	1	1	1	0	0	0	0	1	1	1	0	0	1	1	1	1
P_4	0	0	1	1	1	0	1	1	0	0	0	1	0	0	0	1	1	1
P_5	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
P_6	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1	1	1	0

Fig. 2. Bit matrices for I_1, I_2, and I_3.

	I_1	I_2	I_3
P_1	4	0	1 3
P_2	1 2 5	3	4 6
P_3	6	1 4	1 3
P_4	1 3 6	3 6	1 4
P_5	1 2 5	1 2	0
P_6	5	1 5	1 2 6

Fig. 3. The PT tables for I_1, I_2, and I_3.

the time that the item changes its status between on-shelf and off-shelf. (Initially, we assume that the item is on-shelf in all the stores.) For example, in Store P_1, item I_1 changes its status only in time T_4. Therefore, we need only to record "4" in the PT table to reflect the time and store information for item I_1 in Store P_1. Following this procedure, the information of the original bit matrices for items I_1, I_2, and I_3 is converted into the PT tables shown in Fig. 3.

As mentioned previously, the PT tables for individual items can be used to determine the PT table for a given itemset. The procedure given in Fig. 4 shows how to generate the jth row (store) of the PT table for an itemset X by combining the jth rows of the PT tables for all items in the itemset. We start with an m-dimension bit array, denoted by PT_j, for the itemset with initial values of "1" for each of the m time periods. These initial values are replaced by "0" found at the corresponding position in the PT tables for all the items in the itemset. Finally, we transform PT_j into PT(X, j), the jth row of the PT table for itemset X.

Let us use an itemset consisting of I_1 and I_2 as an example. In order to generate the second row (store) of the PT table, we start with an initial bit array: [1 1 1 1 1 1]. Since the corresponding row of the PT table for I_2 is [1 2 5], the first, fifth, and sixth elements of the bit array are replaced by "0", resulting a new bit array: [0 1 1 1 0 0]. Following the same method, the third through the sixth elements of the new bit array are replaced by "0" when I_2 is considered. As a result, the final bit array is [0 1 0 0 0 0], and the corresponding (second) row of the PT table for the itemset is [1 2 3].

Using the concept described above, we develop the procedure in Fig. 4. In the procedure, PT(k, j) denotes the jth row of the PT table for item k, and its ℓth element is PT(k, j, ℓ), where ℓ is an odd number. The elements of PT_j are replaced by 0's according to the rule stated in lines 4 through 6 in the algorithm: a segment of PT$_j$ is replaced by "0", starting from position PT(k, j, ℓ) and ending at position PT(k, j, $\ell+1$) − 1. PT(k, j) is inserted in sequence into PT_j for every item j in itemset X. The process of developing the PT table is included as line 8 in the algorithm given in Fig. 1.

3.2. The TS table

After building the PT table, the first phase of the algorithm is to build the TS table, where each entry at the position corresponding to T_i and P_j is the number of transactions that occur at store P_j in period T_i. This can be done by a scan through the database. An

```
1)  Fill 1's into the array of $PT_j$;
2)  For every item $k$ in $\boldsymbol{X}$
3)      $n1$=the number of elements in $PT(k, j)$
4)      for ( $\ell = 1$ ; $\ell \le n1$ ; $\ell + 2$ )
5)          for ( $i = PT(k, j, \ell)$; $i < PT(k, j, \ell+1)$ and $i \le m$ ; $i$++ )
6)              $PT_j\,[i]$=0;        /* $PT_j\,[i]$ is the $i$-th bit of array $PT_j$ */
7)  Transform $PT_j$ into $PT(X, j)$.
```

Fig. 4. The method to compute the jth row of the PT table for itemset X.

example of the table is given in Fig. 5. Using the TS and PT tables for itemset X, we can determine the value $|D_{V_X}|$ by summing all the values in the entries of the TS table according to the store and time information of the items in X. The process of constructing the table is described in lines 2 through 4 in Fig. 1.

3.3. Relative-frequent itemset

Because an RF itemset must be a frequent itemset, we can generate RF_k from F_k by computing the relative supports of those itemsets X in F_k. It is evident that $|W(X, D_{V_X})|$ equals $|W(X, D)|$ because it is not possible for X to appear in a transaction not in D_{V_X}. Further, $|D_{V_X}|$ can be obtained from the TS and PT tables of X. As a result, we can find the RF itemsets by first computing the relative supports of all X in F_k and then pruning those itemsets whose relative supports are less than σ_r.

3.4. Candidate itemsets

As discussed, we generate the candidate itemsets from the frequent itemsets, instead of the RF itemsets, from the last phase. Furthermore, when we use the frequent itemsets to generate the candidate set in the next phase, it still satisfies the anti-monotone property, because we use the same base to compute the supports for all itemsets. We illustrate the computation process by the following example.

Example 1. Suppose there are 15 periods, from T_1 to T_{15}, and the numbers of transactions occurring in these 15 periods are 19, 17, 14, 25, 20, 17, 15, 27, 21, 20, 22, 18, 25, 21, and 19, respectively. Assume that the selling periods of product A are from T_1 to T_{10}, and that there are 60 transactions containing product A. Furthermore, assume that the selling periods for product B are from T_6 to T_{15}, and that 80 transactions of them include product B. Finally, there are 50 transactions containing both products A and B, and they are sold in periods from T_6 to T_{10}.

In order to compute the supports and the relative supports for itemsets {A}, {B}, and {A, B}, we identify the following values: $|W(\{A\}, Dv_{\{A\}})| = |W(\{A\}, D)| = 60$, $|W(\{B\}, Dv_{\{B\}})| = |W(\{B\}, D)| = 80$, and $|W(\{A, B\}, Dv_{\{A,B\}})| = |W(\{A, B\}, D)| = 50$. Since the base for computing the support is $|D| = 300$, the supports for the three itemsets are given by $sup(\{A\}, D) = 60/300 = 0.2$, $sup(\{B\}, D) = 80/300 = 0.267$, and $sup(\{A, B\}, D) = 50/300 = 0.167$, respectively. On the other hand, the bases for computing the relative support are $|Dv_{\{A\}}| = 195$, $|Dv_{\{B\}}| = 205$, and $|Dv_{\{A\cup B\}}| = 100$, respectively, for the three itemsets. As a result, the relative supports are $rel\_sup(\{A\}, Dv_{\{A\}}) = 60/195 = 0.308$, $rel\_sup(\{B\}, Dv_{\{B\}}) = 80/205 = 0.39$, $rel\_sup(\{A, B\}, Dv_{\{A,B\}}) = 50/100 = 0.5$ for the itemsets.

Suppose we set σ_s at 0.1 and σ_r at 0.35. Then, we find that {A}, {B}, and {A, B} are frequent. Furthermore, {A} is not relative-frequent, but {B} and {A, B} are relative-frequent.

3.5. The store-chain association rules

Having found the RF itemsets, we proceed to calculate the confidence values and to find all the SC association rules. As defined in Definition 3, the confidence value is given by $conf(X \Rightarrow Y) = rel\_sup(X \cup Y, D_{V_{X\cup Y}})/rel\_sup(X, D_{V_{X\cup Y}})$. If the confidence value exceeds σ_c, the SC association rule holds.

There is an issue that must be dealt with in computing the confidence value. In the calculation of $rel\_sup(X \cup Y, D_{V_{X\cup Y}})/rel\_sup(X, D_{V_{X\cup Y}})$, we obtain the numerator after the phase of processing $X \cup Y$. But the denominator is still undetermined after the phase of processing $X \cup Y$, because the length of X is smaller than that of $X \cup Y$, and we process the itemsets of the same length in a single phase. One possible solution to this problem is to add one step after the phase of processing $X \cup Y$. In this new step, we compute support levels of all subsets of $X \cup Y$ under the context of $V_{X\cup Y}$; i.e., the support levels of X in database $D_{V_{X\cup Y}}$, where X is a subset of $X \cup Y$.

	T_1	T_2	T_3	T_4	T_5	T_6
P_1	23	23	5	42	23	56
P_2	93	42	12	39	47	23
P_3	43	41	8	70	43	59
P_4	21	32	43	34	10	21
P_5	32	42	30	64	34	32
P_6	45	12	16	90	12	65

Fig. 5. An example of the TS table.

If the RF itemset produced in each phase needs another scan to produce the confidence value, then the number of scans of the database in this algorithm is twice that required by the Apriori algorithm. In order to reduce this requirement, we use another method: if Z is an RF itemset found in the kth phase, we compute $rel\_sup(X, D_{V_Z})$ in the $(k+1)$ phase by "hitchhiking," where X is a subset of Z. In other words, in phase $k+1$, we perform two operations: the first is to find the RF itemset of length $k+1$, and the second is to compute the relative supports, such as $rel\_sup(X, D_{V_Z})$. All these values are calculated during the same scan of the database. Consequently, the proposed method requires only one more scan than the Apriori algorithm to obtain the confidence value when the RF itemset of the last phase is produced. This process is included as line 11 in the algorithm, and in Fig. 6, we give the process of computing $rel\_sup(X, D_{V_Z})$ for all subsets X of Z.

In order to compute $rel\_sup(X, D_{V_Z})$ for all subsets X of Z, we must enumerate all the subsets, X, of each RF itemset, Z, in the previous phase—if the length of the RF itemset is k, the number of subsets is 2^k-2. Because each Z has its own PT table (built in line 8 of Fig. 1), we need to check whether a transaction happens in the PT tables of all RF itemsets Z every time a transaction is read in, after computing the supports of the candidates in C_k. If not, it indicates that this transaction does not happen under the context of V_z, and it can be ignored. On the other hand, if the answer is positive, it indicates that this transaction happens under the context of V_z, and, as a result, we need to check if the transaction includes any subset X of Z. This enables us to determine the support levels of all the subsets X of Z under the context of V_z.

For example, suppose that two RF itemsets are generated in the third phase: {A, B, C} and {C, D, E}. In the fourth phase, we build the PT tables for all the RF itemsets in RF_3. And when a transaction is read, we need to check whether it includes any candidates in C_4, as well as whether its time and store combination is in the contexts of {A, B, C} or {C, D, E}. If the time and store combination of the transaction does not conform to the context of {A, B, C}, we need to check whether it does to that of {C, D, E}. If it does, we proceed to check whether it includes any subsets like {C, D, E}: {C}, {D}, {E}, {C, D}, {C, E}, and {D, E}. If it does, the counters of all the matching subsets are increased by one.

Finally, line 14 in Fig. 1 shows the step for generating the store chain association rule $X \Rightarrow Y$, where $X \cup Y$ is in RF_{k-1}. It is not difficult to compute the confidence of the rule—i.e., $rel\_sup(X\cup Y, D_{V_{X\cup Y}})/rel\_sup(X, D_{V_{X\cup Y}})$—because $rel\_sup(X\cup Y, D_{V_{X\cup Y}})$ has already been found in the previous phase and $rel\_sup(X, D_{V_{X\cup Y}})$ found in the current phase.

3.6. Complexity analysis

In this section, we analyze the time complexity and memory space complexity of the algorithm. Let m be the number of items, n the number of transactions in the database, l the number of items in a transaction. Further, let Υ denote the largest value of $|C_k|$. Note that, although $|C_k|$ can theoretically be as large as $O(m^k)$, $|C_k|$ is very unlikely to be larger than $O(m^2)$ in practice. This is because, in an Apriori-like algo-

```
1)  for each $z$ in $RF_{k-1}$
2)      if Time and Store of a transaction $s$ is in the context of $z$
3)          for all subsets $x$ of $z$
4)              if $x \subseteq s$
5)                  increase $count(x, V_z)$ by 1
```

Fig. 6. Compute the support counts of all the subsets X of Z.

rithm [1,2,6,20], C_2 usually has the largest size among all candidate sets. We discuss the time complexities of the steps of Apriori_TP algorithm separately, as well as the total time complexity of the algorithm, as follows.

1. In step 1, we construct the PT table for each item. To produce the table for an item, its Bit Matrix table with $|P|$ rows and $|T|$ columns needs to be linearly scanned and processed. Thus, the time for step 1 is $O(m \times |P| \times |T|)$.
2. In steps 2 to 4, two operations are performed: (1) compute the supports of all itemsets in C_1, and (2) construct the TS table. Since the first operation requires a linear scan of all the items in every transaction, its time is $O(n \times l)$. The time needed for the second operation is $O(n)$ because we examine the attached time and store identifier of each transaction, it requires time $O(n)$. As a result, the total time for the three steps is $O(n \times l)$.
3. Step 5 is for determining F_1 by examining the support of every itemset in C_1. Since C_1 has n itemsets, the time needed for the step is O(n).
4. There is a loop from steps 6 to 18. The time complexities of the steps in iteration k of the loop are discussed as follows.
   4.1. In step 7, we generate C_k. Consequently, the required time is $O(|C_k|)$. Because we assume $O(|C_k|) \leq O(\Upsilon)$, the time is $O(\Upsilon)$.
   4.2. In step 8, we build a PT table for each itemset z in RF_{k-1}. We need $k-2$ merging operations for this step because the $k-1$ PT tables of every individual item in z need to be merged. Because each merging operation can be done in time $O(|P| \times |T|)$, the total time for step 8 is $O(|RF_{k-1}| \times k \times |P| \times |T|)$. Since $RF_{k-1} \subseteq C_{k-1}$, we have $O(|RF_{k-1}|) \leq O(\Upsilon)$ and the total time becomes $O(\Upsilon \times k \times |P| \times |T|)$.
   4.3. In steps 9 to 11, there are two tasks: (1) compute the supports of all itemsets in C_k, and (2) compute the supports of all subsets of itemsets in RF_{k-1}. The time required for the first task is $O(n \times l \times |C_k|)$, because it can be done by first reading every transaction and then adding the counts to the corresponding itemsets. In the second task, we add the counts to all subsets of itemsets in RF_{k-1} rather than all itemsets in RF_{k-1}. Therefore, it can be done by first reading every transaction and every itemset in RF_{k-1}, generating all subsets, and finally adding the counts. Performing these operations requires time $O(n \times l \times |RF_{k-1}| \times 2^{k-1})$. Since $O(|RF_{k-1}|) \leq O(|C_{k-1}|)$, the time required for this part is $O(n \times l \times \Upsilon \times 2^{k-1})$.
   4.4. Step 12 is used to generate F_k from C_k. Since the support of each itemset in C_k must be checked if it is no less than σ_s, the time is $O(|C_k|) = O(\Upsilon)$.
   4.5. Step 13 is for generating RF_k from F_k. Because the support of each itemset in F_k must be checked if it is no less than σ_r, the time is $O(|F_k|)$. Since $O(|F_k|) \leq O(|C_k|)$, we have the total time $O(\Upsilon)$.
   4.6. In steps 14 through 17, we compute the confidence of $x \Rightarrow y$ where $x \cup y$ in RF_{k-1}. That means, for each $z = x \cup y$ in RF_{k-1}, we need to check all of its subsets. Therefore, there are totally $|RF_{k-1}| \times 2^{k-1}$ possible combinations. Since each combination needs a simple division, the total time for this part is $O(|RF_{k-1}| \times 2^{k-1})$. Furthermore, because $O(|RF_{k-1}|) \leq O(|C_{k-1}|)$, the total time required is $O(\Upsilon \times 2^{k-1})$.

From the above analysis, we know that two parts of the algorithm are most time consuming. The first is step 8, and the second is steps 9 through 11, which require times $O(\Upsilon \times k \times |P| \times |T|)$ and $O(n \times l \times \Upsilon \times 2^{k-1})$, respectively. Let K denote the total number of the phases in the loop from step 6 to step 17. Then the total time is $O(\Upsilon \times K^2 \times |P| \times |T|) + O(n \times l \times \Upsilon \times K \times 2^K)$.

Next, we analyze the memory space required for the algorithm. We perform the analysis by examining the space needed to store the data structures used in the algorithm.

1. Because the space requirement for the PT-Interval table for each item is $O(|P| \times |T|)$, the total requirement for all individual items is $O(m \times |P| \times |T|)$.
2. The requirement for the PT-Interval table for each itemset in RF_{k-1} is $O(|RF_{k-1}| \times |P| \times |T|)$.

Since $O(|RF_{k-1}|) \leq O(|C_{k-1}|)$, the total requirement for all itemsets is $O(\Upsilon \times |P| \times |T|)$.

3. The requirement for the TS table is $O(|P| \times |T|)$, because it is a single table.
4. The space requirements for C_k, RF_k, and F_k are $O(\Upsilon)$ individually. Note that, because the same space can be shared by different iterations in the loop from steps 6 to 17, we only need one copy of them rather than multiple copies.
5. The space for storing the supports of all subsets of itemsets in RF_{k-1} is $O(|RF_{k-1}| \times 2^{k-1})$. Since $O(|RF_{k-1}|) \leq O(|C_{k-1}|)$, the required space is $O(\Upsilon \times 2^{k-1})$.
6. Combining all the space requirements, we find the total space is $O(\Upsilon \times 2^K) + O(\Upsilon \times |P| \times |T|)$.

4. Performance evaluation

In this section, we perform a simulation study to empirically compare the proposed and traditional association-rule mining methods. The main objective of the simulation study is to identify the conditions under which the proposed method significantly outperforms the traditional method in identifying important purchasing patterns in a multi-store environment. Three factors are considered in the study: (1) the numbers of stores and periods, (2) the store size, and (3) the product replacement ratio. In addition, we evaluate the computational efficiency of the proposed Apriori_TP algorithm using the Apriori algorithm [1] as the baseline for comparisons. The proposed algorithm is implemented by Borland C+ language and tested on a PC with a Celeron 1.8 G processor and 768 MB main memory under the Windows 2000 operating system.

4.1. Data generation

In the experiment, we randomly generate the synthetic transactional data sets by applying the data generation algorithm proposed by Agrawal and Srikant [1]. The factors considered in the simulation are listed in Table 1. In addition, we generate the time and store information for each transaction in the data sets.

To generate the store sizes, we use two parameters, S_u and S_l, to represent the largest and smallest store sizes, respectively, and the size of store i for $1 \leq i \leq q$, denoted by S_i, is generated by a uniform distribution between S_u and S_l. We assume that the total number of transactions and the number of products are dependent on a store's size. In addition, we also allow the stores to have different product replacement (turnover) ratios. In the simulation, these relationships are established by generating m random numbers for the store i from a Poisson distribution with mean S_i, and we use the jth number, denoted by W_{ij}, as the weight of store i in period j. Let D_{ij} denote the number of transactions of store i in period j. The total number of transactions, D, is distributed to the store i, and period j is determined by:

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

Furthermore, we assume that the number of products in a store is proportional to the square root of its size. Thus, let $IS_i = \sqrt{S_i}$ for $i = 1, 2, \ldots, q$. Then, the number of products in store i, denoted by N_i, is determined by the following formula:

$$N_i = \frac{r}{\mathrm{Max}(\mathrm{IS}_i)} \times \mathrm{IS}_i$$

Note that the products sold in a store may change over time, although N_i is kept the same in all periods. Since the parameter I_d is the proportion of products that will be replaced in every period, store i replaces $N_i \times I_d$ products in each period. Furthermore, we follow the method used by Agrawal and Srikant [1] to generate F_d maximum potentially frequent itemsets with an average length of F_l.

Table 1
Parameters used in simulation

D	Number of transactions
q	Number of stores
m	Number of periods
r	Number of items
L	Average length of transactions
F_l	Average length of maximum potentially frequent itemsets
F_d	Number of maximum potentially frequent itemsets
S_u,S_l	The maximum and minimum sizes of stores
I_d	Replacement rates of items

Finally, we generate all the transactions in the data sets. To generate the transactions for store i in period j, we generate D_{ij} from 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 repetitively add the items into the transaction until we have reached the intended size. If the last itemset exceeds the boundary of this transaction, we remove the part that exceeds the boundary. 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 is included in the paper by Agrawal and Srikant [1]. To generate the nine types of data sets shown in Table 2, we use the following parameter values: $r=1000$, $D=100$ K, $L=6$, $F_l=4$, and $F_d=1000$. For each type of the data sets, 10 replications are generated for statistical analysis of the results.

4.2. Performance measures

As discussed in Section 2, the traditional method underestimates the support and the confidence values and, as a result, may fail to identify important purchasing patterns in a multiple-store environment. We define three measures (errors) for empirically assessing the magnitudes of the deviations in support, confidence, and the number of association rules when we use the traditional association rules for the store-chain data.

Table 2
Data sets

Data set	Number of stores	Number of periods	Range of store sizes	Product replacement rate
1	5	5	50–100	0.001
2	10	10	50–100	0.001
3	50	50	50–100	0.001
4	50	50	10–100	0.001
5	50	50	50–100	0.001
6	50	50	90–100	0.001
7	50	50	50–100	0.001
8	50	50	50–100	0.005
9	50	50	50–100	0.010

The type A error measures the relative difference in the support levels of all frequent itemsets generated by the traditional and proposed methods. It is determined by $rel\_sup(X, D_{V_X}) - sup(X, D)/rel\_sup(X, D_{V_X})$. For example, if the support and relative support for an itemset X are $sup(X, D)=0.02$ and $rel\_sup(X, D_{V_X})=0.03$, respectively, then the type A error rate is $rel\_sup(X, D_{V_X}) - sup(X, D))/rel\_sup(X, D_{V_X})=33.33\%$. By averaging the error rates of all frequent itemsets, we obtain the overall type A error rate. Similarly, the type B error is used to compare the difference in confidence levels of all rules generated by the traditional and proposed methods. It is defined as $conf(X \Rightarrow Y) - conf'(X \Rightarrow Y))/conf(X \Rightarrow Y)$, where $conf'(X \Rightarrow Y)$ is the rule confidence computed by the traditional methods. By averaging the type B error rates of all common rules in the two methods, we obtain the overall type B error rate. Finally, the type C error is used to compare the relative difference in the numbers of rules generated by the two methods. Note that we set σ_s and σ_r at the same level when evaluating the types A and B error rates. It is because the frequent itemsets found by the two algorithms have to be the same in order to have a common base to compare the results produced by the two algorithms. Furthermore, we set σ_c at 1% in the comparison based on the type B error. Using this low value, we can include almost all possible rules in the comparison. However, because in a practical situation the minimum confidence threshold could be higher than this value, we also obtain the results for selected minimum confidence values ranging from 40% to 60%. Finally, we set σ_s at 0.5% in the comparison based on the type C error.

4.3. Simulation results

The first comparison is carried out based on the first three types of data sets in Table 2. Because these three types of data sets have different numbers of stores and periods, the results show the effects of the size of store chain and the length of time on the errors associated with using the traditional method. In order to study the effect of σ_s, we also obtain the results for selected minimum support thresholds ranging from 0.3% to 0.6%. The averages of the types A, B, and C errors are shown in parts (a), (b), and (c), respectively, of Fig. 7. The two-factor ANOVA model is used to

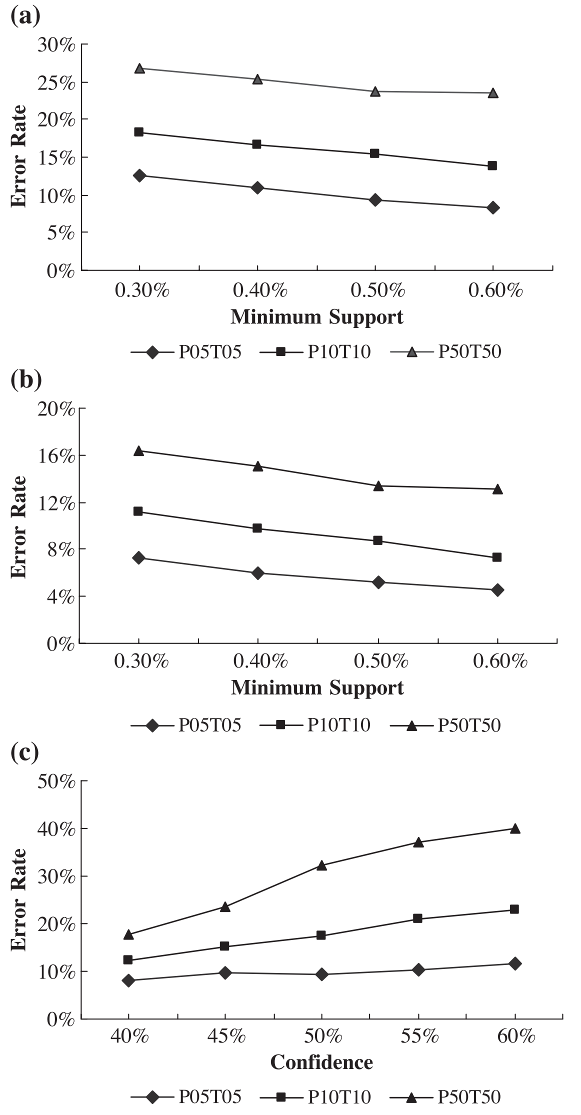

Fig. 7. (a) Effects of the numbers of stores and periods on the type A error rate. (b) Effects of the numbers of stores and periods on the type B error rate. (c) Effects of the numbers of stores and periods on the type C error rate.

analyze the results. We find that all the three error rates are significantly larger in the cases involving larger numbers of stores and periods. We also notice that the error rates generally increase as the minimum support decreases. All these results suggest that the traditional method is not suitable for the store-chain data. The result in part (c) of the figure further supports this conclusion, where, in the worst case reported, 40% of the SC rules are not successfully discovered when σ_c is 60%.

The second comparison is used to study the effects of the store size on the error rates. The data set types 4, 5, and 6 are used, and the average error rates are shown in Fig. 8. The results of statistical analysis based on the two-factor ANOVA model indicate that the error rates are significantly larger when the store size has a larger variation. As shown in parts (a) and

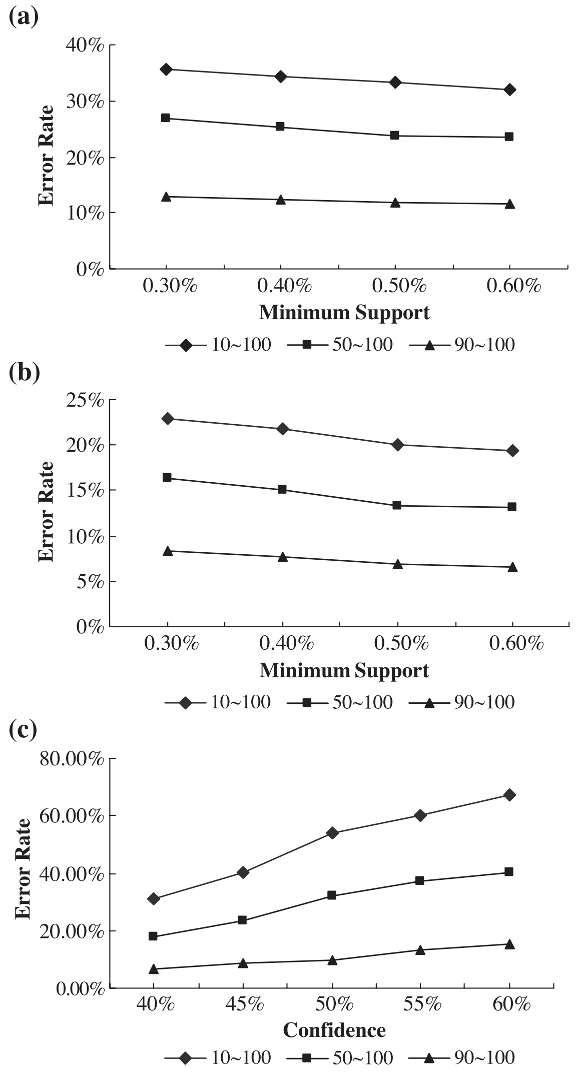

Fig. 8. (a) Effects of store size on the type A error rate. (b) Effects of store size on the type B error rate. (c) Effects of store size on the type C error rate.

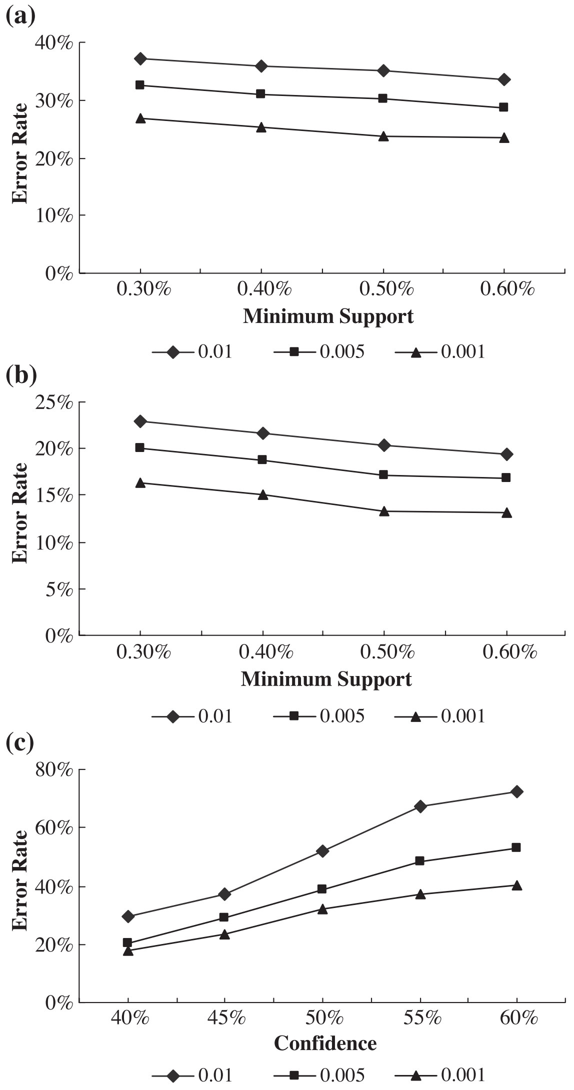

Fig. 9. (a) Effects of product replacement ratio on the type A error rate. (b) Effects of product replacement ratio on the type B error rate. (c) Effects of product replacement ratio on the type C error rate.

(b) of the figure, when the variation of the store size is the largest, the types A and B errors rates are close to 35% and 23%, respectively. In part (c), we find that more than 50% of the SC rules are not generated by the traditional method when σ_c is 50%, and the error rate reaches almost 70% when σ_c is 60%.

In the third part of the simulation study, we compare the error rates under different replacement rates. We use data set types 7, 8, and 9 for the comparison. The results, shown in Fig. 9, indicate that the error rates associated with larger replacement ratios are significantly higher than those associated with smaller replacement ratios. We also notice that the error rates increase as the minimum support decreases. These observations are supported by the results of our statistical analysis. Consequently, we conclude that the performance of the traditional method deteriorates as the product replacement ratio increases.

In the second part of the simulation study, we observe how the type B error rate changes when σ_c is varied from 40% to 60%. In this experiment, we set σ_s at 0.5% and use data set types 2, 4, 5, and 9 for comparison. We use data set type 5 as the baseline; data set type 2 to study the effect of smaller numbers of time periods and stores; and data set types 4 and 9, to study a larger variation in store size and a larger product replacement rate, respectively.

The simulation results are summarized in Fig. 10, where lines 1, 2, 3, and 4 correspond to the results of data sets 5, 2, 4 and 9, respectively. The result indicates that the error rate decreases significantly as we increase σ_c. This is because, when σ_c is higher, only those rules with higher confidence values are used in comparison, causing the type B error rate to decrease. Furthermore, we found that the effect of the product replacement rate is very similar to that of the numbers of periods and stores, and both factors are stronger than that of the variation in store size.

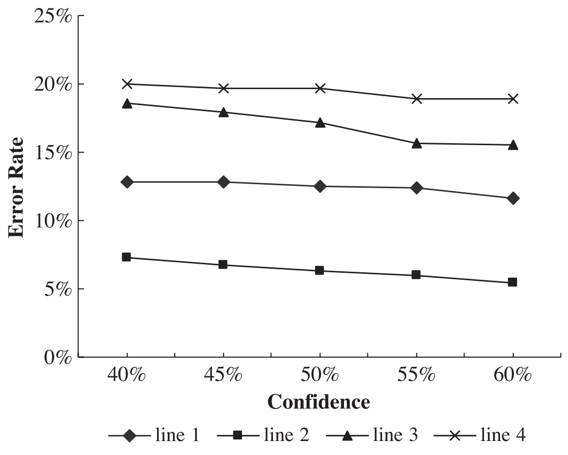

Fig. 10. The type B error rates vs. minimum confidence thresholds.

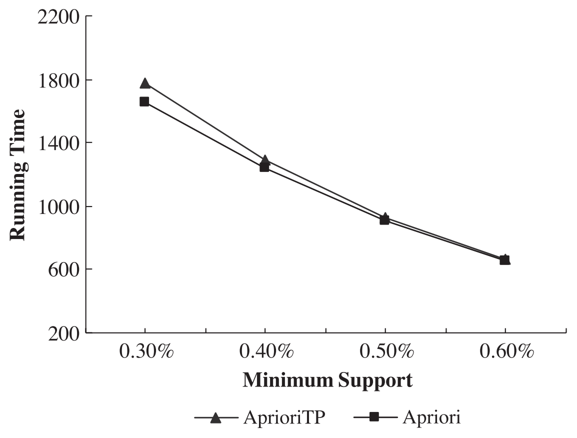

Fig. 11. Run times.

To summarize the simulation study, we conclude that the traditional association rules may not be able to extract all important purchasing patterns for a multi-store chain, especially when there are large numbers of stores and periods, a large variation in store sizes, and high product replacement ratios. This finding is significant because many store chains are growing in size to maintain the economy of scale and, at the same time, dynamically localize their product-mix strategies. All these trends support the need for the proposed method.

Finally, we evaluate the computational efficiency of the proposed algorithm by comparing it with the Apriori algorithm. We show the result in Fig. 11, where the running time is obtained by averaging the running times of all the data sets in Table 2. From the figure, we find that the proposed algorithm requires larger process times, but the differences are not substantial. This result is reasonable, because the proposed algorithm requires one more scan of the data than does the Apriori algorithm, and also requires additional basic operations in each phase of the algorithm.

5. Conclusion

Association-rule mining is a useful method of discovering customer purchasing patterns by extracting associations or co-occurrences from stores' transactional databases. Since the method was first proposed by Agrawal et al. [1] in 1993, it has become an established and active research area. The existing methods, however, may fail to discover important purchasing patterns in a multi-store environment, because of an implicit assumption that products under consideration are on-shelf all the time across all stores.

To overcome the problem, a new method, called store-chain association rules, is proposed specifically for a multi-store environment, where stores may have different product-mix strategies that can be adjusted over time. The format of the rules is similar to that of the traditional rules. However, the rules also contain information on store (location) and time where the rules hold. The rules extracted by the proposed method may be applicable to the entire chain without time restriction, but may also be store- and time-specific. These rules have a distinct advantage over the traditional ones because they contain store (location) and time information so that they can be used not only for general or local marketing strategies (depending on the results), but also for product procurement, inventory, and distribution strategies for the entire store chain.

An Apriori-like algorithm is developed for mining chain-store association rules. A simulation is used to empirically compare the proposed and traditional association-rule mining methods. Three factors are considered in generating stores' sales data: (1) the numbers of stores and periods, (2) the store size, and (3) the product replacement ratio. The analysis of the simulation result suggests that the proposed method has advantages over the traditional method especially when the numbers of stores and periods are large, stores are diverse in size, and product mix changes rapidly over time. Furthermore, the time complexity of the proposed algorithm is discussed, and the simulation results show that the algorithm is computationally efficient.

Store-chain association rules represent a promising research area in data mining. The results of this paper can be extended by considering time constraints, spatial constraints, quantitative attributes and/or taxonomy, and other kinds of time- or location-related knowledge. Furthermore, it is important to explore the strategies of generating the store-chain association rules incrementally, in an on-line model, in a distributed environment, or in parallel models.

Acknowledgements

The first author was supported in part by the Ministry of Education (MOE) Program for Promoting Academic Excellence of Universities under Grant No. 91-H-FA07-1-4 and National Science Council Grant No. 91-2416-H-008-003.

Appendix A

The support values and the confidence values obtained by the traditional association mining are underestimated, compared with the true value discussed in this paper. First, it is easy to see that the traditional support value is lower because its base is larger. As to the confidence value, say *conf*($X \Rightarrow Y$), the traditional approach defines it as follows.

$$\begin{aligned} conf(X \Rightarrow Y) &= sup(X \cup Y, D)/sup(X, D) \\ &= [|W(X \cup Y, D)|/|D|]/[|W(X, D)|/|D|] \\ &= |W(X \cup Y, D)|/|W(X, D)|. \end{aligned} \tag{A1}$$

But the correct one should be

$$\begin{aligned} conf(X \Rightarrow Y) &= rel\_sup(X \cup Y, D_{V_{X \cup Y}})/rel\_sup(X, D_{V_{X \cup Y}}) \\ &= [|W(X \cup Y, D_{V_{X \cup Y}})|/|D_{V_{X \cup Y}}|] \\ &\quad /[|W(X, D_{V_{X \cup Y}})|/|D_{V_{X \cup Y}}|] \\ &= |W(X \cup Y, D_{V_{X \cup Y}})|/|W(X, D_{V_{X \cup Y}})| \end{aligned} \tag{A2}$$

By comparing Eq. (A1) with Eq. (A2), we find that the numerators are the same, because it is not possible that $X \cup Y$ appears in a transaction not in $D_{V_{X \cup Y}}$, and that the denominator of Eq. (A1) is no less than that of Eq. (A2), because $|W(X, D)| \geq |W(X, D_{V_{X \cup Y}})|$. Thus, we conclude that the confidence value of Eq. (A1) is no larger than that of Eq. (A2).

References

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994, pp. 478–499.

[2] 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, Washington, D.C., 1993, pp. 207–216.

[3] J.M. Ale, G.H. Rossi, An approach to discovering temporal association rules, Proceedings of the 2000 ACM Symposium on Applied Computing (Vol. 1), Villa Olmo, Como, Italy, 2000, pp. 294–300.

[4] R.J. Bayardo Jr., R. Agrawal, Mining the most interesting rules, Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 1999, pp. 145–154, Aug.

[5] I. Bose, R.K. Mahapatra, Business data mining—a machine learning perspective, Information and Management 39 (2001) 211–225.

[6] S. Brin, R. Motwani, J.D. Ullman, S. Tsur, Dynamic itemset counting and implication rules for market basket data, Proceedings of the 1997 ACM-SIGMOD Conference on Management of Data, Tucson, Arizona, USA, May 1997, pp. 255–264.

[7] E. Clementini, P.D. Felice, K. Koperski, Mining multiple-level spatial association rules for objects with a broad boundary, Data and Knowledge Engineering 34 (3) (2000) 251–270.

[8] M.-S. Chen, J. Han, P.S. Yu, Data mining: an overview from a database perspective, IEEE Transactions on Knowledge and Data Engineering 8 (1996) 866–883.

[9] A. Freitas, On rule interestingness measures, Knowledge-Based Systems 12 (5) (1999) 309–315.

[10] J. Han, Y. Fu, Mining multiple-level association rules in large databases, IEEE Transactions on Knowledge and Data Engineering 11 (5) (1999) 798–805.

[11] J. Han, M. Kamber, Data Mining, Morgan Kaufmann, San Francisco, 2001.

[12] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, Proceedings of the 2000 ACM-SIGMOD Int. Conf. on Management of Data, Dallas, TX, 2000, May.

[13] H. Ishibuchi, T. Nakashima, T. Yamamoto, Fuzzy association rules for handling continuous attributes, Proceedings of the IEEE International Symposium on Industrial Electronics, Pusan, Korea, 2001, pp. 118–121.

[14] C.M. Kuok, A.W. Fu, M.H. Wong, Mining fuzzy association rules in databases, SIGMOD Record 27 (1) (1998) 41–46.

[15] K. Koperski, J. Han, Discovery of spatial association rules in geographic information databases, Proc. 4th International Symposium on Large Spatial Databases (SSD95), Portland, Maine, Aug. 1995, pp. 47–66.

[16] C.H. Lee, C.R. Lin, M.S. Chen, On mining general temporal association rules in a publication database, Proceedings of the 2001 IEEE International Conference on Data Mining, San Jose, California, 2001, pp. 337–344.

[17] Y. Li, P. Ning, X.S. Wang, S. Jajodia, Discovering calendar-based temporal association rules, Proceedings of the Eighth International Symposium on Temporal Representation and Reasoning, Cividale Del Friuli, Italy, 2001, pp. 111–118.

[18] J. Liu, Y. Pan, K. Wang, J. Han, Mining frequent item sets by opportunistic projection, Proceedings of the 2002 Int. Conf.

on Knowledge Discovery in Databases, Edmonton, Canada, 2003, July.

[19] H. Lu, L. Feng, J. Han, Beyond intra-transaction association analysis: mining multi-dimensional inter-transaction association rules, ACM Transactions on Information Systems 18 (4) (2000) 423–454.

[20] J.-S. Park, M.-S. Chen, P.S. Yu, Using a hash-based method with transaction trimming for mining association rules, IEEE Transactions on Knowledge and Data Engineering 9 (1997) 813–825.

[21] R. Rastogi, K. Shim, Mining optimized association rules with categorical and numeric attributes, IEEE Transactions on Knowledge and Data Engineering 14 (2002) 29–50.

[22] J.F. Roddick, M. Spiliopoilou, A survey of temporal knowledge discovery paradigms and methods, IEEE Transactions on Knowledge and Data Engineering 14 (2002) 750–767.

[23] S. Shekhar, S. Chawla, S. Ravadam, A. Fetterer, X. Liu, C. Lu, Spatial databases—accomplishments and needs, IEEE Transactions on Knowledge and Data Engineering 11 (1999) 45–55.

[24] R. Srikant, R. Agrawal, Mining quantitative association rules in large relational tables, Proceedings of the ACM-SIGMOD 1996 Conference on Management of Data, Montreal, Canada, 1996, pp. 1–12, June.

[25] J. Wijsen, R. Meersman, On the complexity of mining quantitative association rules, Data Mining and Knowledge Discovery 2 (1998) 263–281.

Yen-Liang Chen is Professor of Information Management at National Central University of Taiwan. He received his PhD degree in Computer Science from the 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 Operations Research, IEEE Transaction on Software Engineering, Computers and OR, European Journal of Operational Research, Information and Management, Information Processing Letters, Information Systems, Journal of Operational Research Society, and Transportation Research.

Kwei Tang is Professor of Management and the area coordinator of quantitative methods in the Krannert Graduate School of Management at 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. His current research interests include data mining, supply chain management, and quality management.

Ren-Jie Shen is a system analyst and designer in Data Systems Consulting, a leading commercial software company in Taiwan. He received his MS degree in Information Management from National Central University of Taiwan. His research interests include data mining, information systems and EC technologies.

Ya-Han Hu is currently a PhD student in the Department of Information Management, National Central University, Taiwan. He received the MS degree in Information Management from National Central University of Taiwan. His research interests include data mining, information systems and EC technologies.