Beyond Intratransaction Association Analysis: Mining Multidimensional Intertransaction Association Rules

HONGJUN LU
Hong Kong University of Science and Technology
LING FENG
Tilburg University
and
JIAWEI HAN
Simon Fraser University

In this paper, we extend the scope of mining association rules from traditional single-dimensional intratransaction associations, to multidimensional intertransaction associations. Intratransaction associations are the associations among items within the *same transaction*, where the notion of the transaction could be the items bought by the *same customer*, the events happened on the *same day*, and so on. However, an intertransaction association describes the association relationships among *different transactions*, such as "*if (company) A's stock goes up on day 1, B's stock will go down on day 2, but go up on day 4.*" In this case, whether we treat company or day as the unit of transaction, the associated items belong to different transactions. Moreover, such an intertransaction association can be extended to associate multiple contextual properties in the same rule, so that *multidimensional* intertransaction associations can be defined and discovered. A two-dimensional intertransaction association rule example is "*After McDonald and Burger King open branches, KFC will open a branch two months later and one mile away,*" which involves two dimensions: *time* and *space*. Mining intertransaction associations poses more challenges on efficient processing than mining intratransaction associations. Interestingly, intratransaction association can be treated as a special case of intertransaction association from both a conceptual and algorithmic point of view. In this study, we introduce the notion of multidimensional intertransaction association rules, study their measurements—*support* and *confidence*—and develop algorithms for mining intertransaction associations by extension of Apriori. We overview our experience using the algorithms on both real-life and synthetic data sets. Further extensions of multidimensional intertransaction association rules and potential applications are also discussed.

Authors' addresses: H. Lu, Dept. of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China; email: luhj@cs.ust.hk (On leave from National University of Singapore); L. Feng, InfoLab, Dept. of Information Management, Center research institute, Tilburg University, B 302, PO Box 90153, 5000 LE Tilburg, Netherlands; email: ling@kub.nl; J. Han, School of Computing Science, Simon Fraser University, Canada; email: han@cs.sfu.ca.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Data mining, intra/intertransaction association rules, multidimensional context

---

1. INTRODUCTION

The discovery of association rules is an important data mining problem. The most often cited application of association rules is market basket analysis using transaction databases from supermarkets. These databases contain sales transaction records, each of which details items bought by a customer in the transaction. Mining association rules is the process of discovering knowledge such as

R_1: *80% of customers who bought diapers also bought beer.*

which can be expressed as *diapers* $\Rightarrow$ *beer* (20%, 80%), where 80% is the *confidence level* of the rule, and 20% is the *support level* of the rule indicating how frequently the customers bought both diapers and beer. In general, an association rule takes the form $X \Rightarrow Y\ (s, c)$, where X and Y are sets of items, and s and c are *support* and *confidence* respectively. The discovered knowledge can then be used in store floor planning, sales promotions, etc.

Since the problem of mining association rules was introduced in Agrawal et al. [1993], a large amount of work has been done in various directions, including efficient, Apriori-like mining methods [Agrawal and Srikant 1994; Klemettinen et al. 1994; Savasere et al. 1995; Park et al. 1995; Park et al. 1996; Toivonen 1996; Zaki et al. 1997; Fang et al. 1998], mining generalized, multilevel, or quantitative association rules [Srikant and Agrawal 1995; Srikant and Agrawal 1996; Han and Fu 1995a; Fukuda et al. 1996a; Fukuda et al. 1996b; Miller and Yang 1997; Lent et al. 1997; Kamber et al. 1997], association rule mining query languages [Meo et al. 1996; Tsur et al. 1998], constraint-based rule mining [Ng et al. 1998; Srikant et al. 1997; Tsur et al. 1998; Baralis and Psaila 1997; Han and Fu 1995b], incremental maintenance of discovered association rules [Cheung et al. 1996a], parallel and distributed mining [Agrawal and Shafer 1996; Han et al. 1997; Cheung et al. 1996b], mining correlations and casual structures [Brin et al. 1997; Silverstein et al. 1998], cyclic and interesting association rule mining [Özden et al. 1998; Ramaswamy et al. 1998], etc. Despite these efforts, there is an important form of association rule which is useful, but could not be discovered with the existing association rule mining framework.

Association among items from different transactions. Taking stock markets as an example, we can construct a share price movement database as follows: each trading day has one record in the database which contains the counters whose prices are up for that day. Applying the association rules as mentioned above, we can discover rules like

R'_1: *When the prices of IBM and SUN go up, 80% of the time the price of Microsoft increases on the same day.*

While the association rule such as R'_1 reflects some relationship among the prices, its role in price prediction is limited; and traders may be more interested in the following type of rules:

R_2: *If the prices of IBM and SUN go up, Microsoft's will most likely (80% of the time) increases the next day.*

Unfortunately, current association rule miners cannot discover this type of rule because of the fundamental difference between rules like R_2 and those such as R_1 and R'_1, which we will refer to as classical association rules. The classical association rules express the associations among items within the *same transaction*, such as items purchased by a customer or share price movement within a day. On the other hand, rule R_2 represents some association relationship *among the field values from different transaction records*. To distinguish these two types of associations, we name the classical associations as **intratransaction associations** and the latter as **intertransaction associations**.

It is worth pointing out that intertransaction associations cannot be converted to intratransaction associations by simply transforming the data. That is, no matter how we redefine transactions, intertransaction associations will remain associations across the boundaries of transactions. For example, even if we transform the stock data so that transactions correspond to the daily fluctuations of individual stocks, what Rule R_2 expresses will still be associations among transactions, since the ups and downs of different stocks are in different transaction records.

Multidimensional context. In classical association rule mining, records in a transaction database contain only items and are identified by their TIDs. Although transactions occurred under certain *contexts*, such as time, place, customers, etc., such contexts have been ignored in classical association rule mining. When we talk about intertransaction association, we are exploring the association of items along a certain dimension. If the records in the transaction database are organized by *transaction time*, the intertransaction association rules represent associations along the dimension of time. From the example of stock price movement mentioned earlier, we can see that in addition to items, the time of transactions also contributes to the validation of associations, and such time intervals actually convey a kind of contextual information.

Besides transaction time, other properties like spatial location also form an interesting context for the existence of associations. Therefore, we can enhance the traditional transaction model by associating records with a number of attributes that describe the context where the transaction happens. We call them *dimensional attributes*, because these attributes in fact form a multidimensional space, and transactions can be viewed as points in such a space. These dimensional attributes can be of any kind as long as they are meaningful to the applications. Time, distance, temperature, latitude, etc., are typical dimensional attributes. For a stock movement database, the dimensional attributes could be trading date and region. For a geographical database, we can have two numerical dimensional attributes, x- and y-coordinates, with respect to a certain reference point, and the items could be those objects that are of interest to a particular application, such as shops, gas stations, fast food restaurants, etc. In the current study, we assume that the domain of each dimensional attribute is ordinal and can be divided into equal-sized intervals. For instance, time can be divided into day, week, month, . . . and distance into yard, mile, etc. For dimensional attributes with continuous values, we can discretize them first in order to define such intervals to represent their relative relationship. For example, we can order different colors as red, green, blue, yellow, . . . in such a way that green is one unit away from red, and blue is one unit away from green but two units away from red, and so on.

With the definition of dimensional attributes, we can further extend intertransaction association to **multidimensional intertransaction association**. For example, if a database contains records about the timing and location of buildings and facilities of a new city under development, we may be able to find two-dimensional intertransaction association rules like

R_4:*After McDonald and Burger King open branches, KFC will open a branch two months later and one mile away*.

Since many real-world associations do happen in a certain context (in the above example, time and spatial intervals convey a kind of contextual information), integrating such *multidimensional contextual information* into association rule mining is important. We believe that mining multidimensional intertransaction association rules is a major data mining problem with broad applications. As discussed in Section 6, classical intratransaction association rule mining can be viewed as a special case of intertransaction association rule mining from both a conceptual and algorithmic point of view.

In this paper, we will give a formal definition of multidimensional intertransaction association rules. As a first phase of study, we will also compare two algorithms for mining one-dimensional intertransaction association rules from large databases.

The remainder of the paper is organized as follows. Section 2 defines multidimensional intertransaction association rules and related measurements. Section 3 describes the three phases in mining intertransaction association rules: (i) data preparation, (ii) large-itemset mining, and (iii) rule generation. Section 4 examines the large-itemset mining in detail and presents two algorithms for this phase. Our experiments that evaluate the performance of these algorithms on both synthetic and real-life data sets are presented in Section 5. Section 6 discusses some closely related work

Table I. A Transformed Single-Dimensional Extended Transaction Database

Trans.	Date	Items	Extended Trans.	Extended Items
t_1	day_1	a, b, c	$\Delta_0(t_1)$	$\Delta_0(a), \Delta_0(b), \Delta_0(c)$
t_2	day_2	c, d, e	$\Delta_1(t_2)$	$\Delta_1(c), \Delta_1(d), \Delta_1(e)$
t_3	day_3	a, b	$\Delta_2(t_3)$	$\Delta_2(a), \Delta_2(b)$
t_4	day_4	a, b, c, e	$\Delta_3(t_4)$	$\Delta_3(a), \Delta_3(b), \Delta_3(c), \Delta_3(e)$
t_5	day_5	b, c, d, e	$\Delta_4(t_5)$	$\Delta_4(b), \Delta_4(c), \Delta_4(d), \Delta_4(e)$

and addresses some issues related to mining multidimensional intertransaction association rules. Section 7 concludes the paper with a brief discussion of future work.

2. MULTIDIMENSIONAL INTERTRANSACTION ASSOCIATION RULES

In this section, we define single- and multidimensional intertransaction association rules and related *support* and *confidence* measurements.

2.1 Single-Dimensional Intertransaction Association Rules

We start with single-dimensional intertransaction association rules for easy explanation, and then extend them to the multidimensional context in the next subsection.

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_s\}$ denote a set of literals, called items. A transaction database $\mathcal{T}$ is a set of transactions $\{t_1, t_2, \ldots, t_n\}$, where t_i $(i = 1, 2, \ldots, n)$ is a subset of $\mathcal{I}$. A single-dimensional mining space can be represented by one dimensional attribute, whose domain is a finite subset of nonnegative integers. Let $n_i = \langle v \rangle$ and $n_j = \langle u \rangle$ be two points in the one-dimensional space, then a relative distance between n_i and n_j is defined as $\Delta(n_i, n_j) = \langle u - v \rangle$. In this paper, we also use $\Delta(n_i)$ or simply Δ_i for $\Delta(n_0, n_i)$, where n_0 is a reference point in the one-dimensional space. Note that n_i, $\Delta(n_i)$, and Δ_i can be used interchangeably with respect to a reference point. We call an item i_k, at the point Δ_j in the one-dimensional space, an **extended item** and denote it as $\Delta_j(i_k)$. In general, the two extended items $\Delta_i(i_k)$ and $\Delta_j(i_k)$ are not equal if $\Delta_i \neq \Delta_j$. In a similar fashion, we call a transaction t_k, at the point Δ_j in the one-dimensional space, an **extended transaction** and denote it as $\Delta_j(t_k)$. The set of all possible extended items, $\mathcal{I}_e$, is defined as a set of $\Delta_j(i_k)$ for any $i_k \in \mathcal{I}$, at all points Δ_j in the one-dimensional space. $\mathcal{T}_e$ is the set of all extended transactions in the one-dimensional space. The reference point of an extended transaction subset is defined to be the smallest Δ_j among all $\Delta_j(t_k)$ in this subset.

Example 2.1. A simple traditional database is transformed into a single-dimensional extended transaction database as shown in Table I. Let $T'_e = \{\Delta_1(t_2), \Delta_2(t_3), \Delta_3(t_4)\}$. Following the definition, the reference point of T'_e is Δ_1, with which we say T'_e contains a set of extended items $\{\Delta_0(c), \Delta_0(d), \Delta_0(e), \Delta_1(a), \Delta_1(b), \Delta_2(a), \Delta_2(b), \Delta_2(c), \Delta_2(e)\}$. In other words, the above set of extended items has Δ_1 as its reference point.

With the above notations, we are now in a position to define single-dimensional intertransaction association rules and related measurements.

Definition 2.1. A single-dimensional intertransaction association rule is an implication of the form $X \Rightarrow Y$, where $X \subset \mathcal{I}_e$, $Y \subset \mathcal{I}_e$, and $X \cap Y = \emptyset$.

Based on Definition 2.1, a rule that predicts stock price movement, such as *"if the price of stock 'a' increases one day, and the price of stock 'c' increases the next day, then most probably the price of stock 'e' will increase on the fourth day,"* can be expressed by the single-dimensional intertransaction association rule "$\Delta_0(a)$, $\Delta_1(c) \Rightarrow \Delta_3(e)$."

Similar to intratransaction association rules, we use *support* and *confidence* as two major rule measurements. Traditionally, the support of a rule $X \Rightarrow Y$ is the fraction of transactions that contain $X \cup Y$ over the whole transactions, and the confidence of the rule is the fraction of transactions containing X that also contain Y. That is, we can have the following definition:

Definition 2.2. Given two subsets of $\mathcal{I}_e$, X, and Y, referencing to Δ_0. Let T_{xy} be a set of extended transactions that contain $X \cup Y$, and let T_x be a set of extended transactions that contain X. The *support* and *confidence* of a single-dimensional intertransaction association rule $X \Rightarrow Y$ are support$(X \Rightarrow Y) = |T_{xy}|/|\mathcal{T}_e|$ and confidence$(X \Rightarrow Y) = |T_{xy}|/|T_x|$.

Note that, for intertransaction association rules, the above definition is an approximation. To illustrate, let us look at the rule "$\Delta_0(a)$, $\Delta_1(c) \Rightarrow \Delta_3(e)$." If we apply the traditional method to the database in Table I, we get the support and confidence, 1/5 and 1/3 respectively, where 5 is the total number of transactions, and 3 is the number of transaction sets that contain $\{\Delta_0(a), \Delta_1(c)\}$. But in fact, there are only 2 sets of transactions, i.e., $\{\Delta_0(t_1), \Delta_1(t_2), \Delta_2(t_3), \Delta_3(t_4)\}$ and $\{\Delta_1(t_2), \Delta_2(t_3), \Delta_3(t_4), \Delta_4(t_5)\}$, that *possibly contain* $\{\Delta_0(a), \Delta_1(c), \Delta_3(e)\}$. It seems unfair to take total number of transactions "5" as the denominator to compute the support. Similarly, for the confidence computation, although "3" sets of transactions contain $\{\Delta_0(a), \Delta_1(c)\}$, only one of them is meaningful for $\{\Delta_0(a), \Delta_1(c), \Delta_3(e)\}$. So the reasonable support and confidence of the rule shall be 1/(5-3) and 1/(3-2) respectively. That is, the denominator used to compute the support and confidence level should be smaller. The difference is determined by the span of intervals involved in the itemset. However, observing that in most real databases, the total number of transactions is far larger than the maximum transaction interval in a rule, the difference between the above two calculations is actually very minor. Thus, we can still adopt the traditional method to compute support and confidence for intertransaction association rules.

Using the above approximation not only simplifies the computation of the support level of itemsets, but also maintains the important monotonic property of the support of itemsets. In other words, the support of an itemset will not be larger than the support of any of its subsets. We like to have the property, since it is the base of a large set of efficient association rule mining algorithms. If the denominator in the support computation formula depends on the size of the itemset, this monotonic property will not hold. It is worth pointing out that in order to maintain this monotonic property, we will assume the database in consideration contains no "holes" along the dimension. That is, there is a record in the database for every dimensional value. For example, in the database in Table I, there is a record for every date. In this study, we assume that there are no holes in the database, since they can be either filled by transactions with null item lists, or discarded by changing the value of the dimensional attribute. Consider a stock price movement database: each record has the form (*date, stock-list*), where *stock-list* contains those stocks whose prices are up compared to the previous trading day. Although we may find that there are certain holes in the database, i.e., no records for certain dates, based on the domain knowledge, we know that there are two possible reasons that cause such holes: either it is a holiday (no trading), or prices of all stocks dropped that day.[1] For the former case, the date should not be counted, since only the trading day is meaningful for association mining. For the latter, we need to insert a record for that trading day with a null item list, and, as a result, we will have a stock price movement database with no holes. Each record consists of an ID of the trading day and the stocks whose prices rise.

2.2 Multidimensional Intertransaction Association Rules

When increasing the number of dimensional attributes from 1 to m, we come to a multidimensional mining space. In practice, there are a wide range of application databases that can be viewed as multidimensional transaction databases. For example, an urban development project can use a two-dimensional transaction database, where the two dimensional attributes are *month* and *block number*, and the item lists include the buildings or facilities completed during the month in a particular block.

In general, a multidimensional space is a finite subset of N^m, where N is the set of nonnegative integers. Each dimension is represented by a dimensional attribute. A mapping function can be introduced which maps a database transaction onto a point in the m-dimensional space. Let $n_i = \langle v_1, v_2, \ldots, v_m \rangle$ and $n_j = \langle u_1, u_2, \ldots, u_m \rangle$ be two points in the m-dimensional space. The *relative distance* between n_i and n_j in the m-dimensional space can be defined as $\Delta(n_i, n_j) = \langle u_1 - v_1, u_2 - v_2, \ldots, u_m - v_m \rangle$, and the reference point of these two points is defined as $\langle \min(u_1, v_1), \min(u_2, v_2), \ldots, \min(u_m, v_m) \rangle$.

Figure 1 depicts a two-dimensional extended transaction database with two dimensional attributes x and y, whose domains have been discretized into 5 and 4 equal-lengthed intervals respectively. There are totally four items, a, b, c, and d, in the database. For simplicity, we also denote a point in the two-dimensional space using $\Delta_{x,y}$ with respect to a certain reference

[1]In fact, there is another possible reason: missing data. That is, we do not have a trading record for that day. We will leave the issue of handling missing data to further study for the moment.

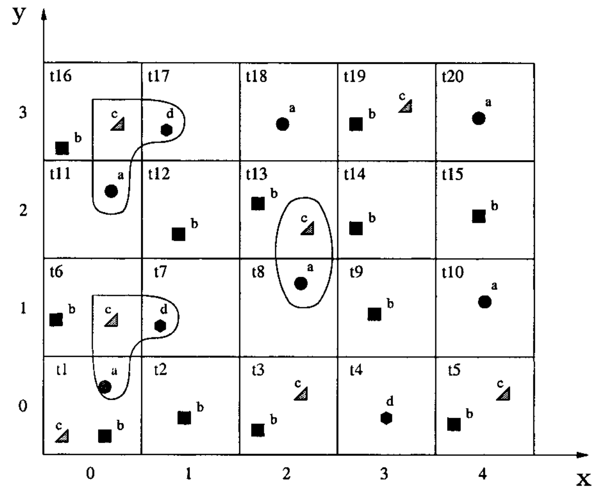


Fig. 1. Graphical representation of a two-dimensional transaction database.

point.[2] The database contains extended transactions such as $\Delta_{0,0}(t_1) = \{\Delta_{0,0}(a), \Delta_{0,0}(b), \Delta_{0,0}(c)\}$, $\Delta_{1,0}(t_2) = \{\Delta_{1,0}(b)\}$, ..., $\Delta_{4,3}(t_{20}) = \{\Delta_{4,3}(a)\}$. Upon such an extended transaction database, the support and confidence of the rule "$\Delta_{0,0}(a), \Delta_{0,1}(c) \Rightarrow \Delta_{1,1}(d)$" are 2/20 and 2/3 respectively, since there are totally 3 extended transaction sets: $\{\Delta_{0,0}(t_1), \Delta_{0,1}(t_6), \Delta_{1,1}(t_7)\}$, $\{\Delta_{0,2}(t_{11}), \Delta_{0,3}(t_{16}), \Delta_{1,3}(t_{17})\}$, and $\{\Delta_{2,1}(t_8), \Delta_{2,2}(t_{13}), \Delta_{3,2}(t_{14})\}$, containing $\{\Delta_{0,0}(a), \Delta_{0,1}(c)\}$, and 2 containing $\{\Delta_{0,0}(a), \Delta_{0,1}(c), \Delta_{1,1}(d)\}$.

Multidimensional intertransaction association rules and related measurements can be defined in a similar way as single-dimensional intertransaction association rules and related measurements, except that more dimensional attributes get involved in describing the mining context.

3. MINING ONE-DIMENSIONAL INTERTRANSACTION ASSOCIATION RULES

In this section, we give an overview of mining one-dimensional intertransaction association rules which can be divided into three phases: *data preparation, large extended itemset discovery*, and *association rule generation*.

—Phase-1 (Data preparation) The transaction database is prepared for mining from operational databases, and the major task is to organize the transactions based on intervals of the dimensional attribute(s). For example, to find the long-term movement regularities of stock prices across different weeks (months), we need to transform daily price movement into weekly (monthly) groups. After such transformation, each

[2]Recall that Δ_i is used for a point n_i in an m-dimensional space regarding a reference point. The (x, y) in $\Delta_{x,y}$ is used to index a point in a two-dimensional space.

record in the database will contain an interval value representing the relative address from a reference point in the dimensional space, and a list of items.

—Phase-2 (Large extended itemset discovery) In this phase, we find the set of all extended itemsets whose supports are above a user-specified *support* threshold. A k-extended itemset is of the form $\{\Delta_{x_1}(i_1), \Delta_{x_2}(i_2), \ldots, \Delta_{x_k}(i_k)\}$, where item i_t $(1 \leq t \leq k)$ appears at the point Δ_{x_t} in the contextual space. For example, a three-extended itemset $\{\Delta_0(a), \Delta_1(b), \Delta_3(c)\}$ describes three items and their relative addresses along the dimension. That is, taking a transaction containing item a as the reference transaction, $\Delta_1(b)$ represents an item b to be contained in a transaction 1 unit distance away from the reference transaction, and $\Delta_3(c)$ represents an item c in a transaction 3 unit distances away from the reference transaction.

To limit the search space, we set a maximum span threshold, *maxspan*, along each dimension for a particular application. In a single-dimensional case, it works like a sliding window; only those associations covered by the window are considered. There are two reasons for this. First, from the application viewpoint, we are often interested in the relationships within a certain range, such as share prices going up within a week (for investors seeking short-term profits), gas stations and fast-food outlets within 50 miles, etc. A rule like "*if IBM stock goes down, SUN stock will go down 243 days later*" is unlikely to inspire the confidence of stock traders. Second, from the view point of computational complexity, a threshold set reasonably will generate interesting results within a reasonable time. As we see later, mining intertransaction association rules is computationally much more complex than mining intratransaction association rules, which is already computationally intensive. Note that, with *maxspan* $= 0$, the intertransaction association rules degrade to the classical intratransaction association rules. Detailed algorithms for generating large extended itemsets are described in the following section.

—Phase-3 (Association rule generation) Using sets of large extended itemsets, we can find the desired intertransaction association rules, the generation of which is similar to that of the classical association rules [Agrawal and Srikant 1994].

4. LARGE EXTENDED ITEMSET DISCOVERY PHASE

In this section, we describe two algorithms, *E-Apriori* and *EH-Apriori*, for finding large extended itemsets by extension of Apriori [Agrawal and Srikant 1994].

Let L_k represent the set of large k-extended itemsets, and C_k the set of candidate k-extended itemsets. Both algorithms make multiple passes over the database, and each pass consists of two phases. First, the set of all large $(k - 1)$-extended itemsets L_{k-1}, found in the $(k - 1)$th pass, is used to generate the candidate extended itemset C_k. The candidate generation procedure ensures that C_k is a superset of the set of all large k-extended itemsets. The algorithms now scan the database. For each list of consecutive transactions, they determine which candidates in C_k are contained and increment their counts. At the end of the pass, C_k is examined to check which of the candidates are actually large, yielding L_k. The algorithms terminate when L_k becomes empty.

As previously reported in Park et al. [1995], the processing cost of the first two iterations (i.e., obtaining L_1 and L_2) dominates the total mining cost. The reason is that, for a given minimum support, we usually have a very large L_1, which in turn results in a huge number of itemsets in C_2 to process. Mining intertransaction association rules results in a much larger C_2, as many additional two-extended itemsets such as $\{\Delta_0(a), \Delta_1(a)\}$ will be added.

In order to construct a significantly smaller C_2, *EH-Apriori* adopts a similar technique of hashing as Park et al. [1995] to filter out unnecessary candidate two-extended itemsets. When the support of candidate C_1 is counted by scanning the database, *EH-Apriori* accumulates information about candidate two-extended itemsets in advance in such a way that all possible two-extended itemsets are hashed to a hash table. Each bucket in the hash table consists of a number to represent how many extended itemsets have been hashed to it thus far. Such a resulting hash table can be used to greatly reduce the number of two-extended itemsets in C_2.

In the following, we describe how *E-Apriori* and *EH-Apriori* generate candidates and count their supports for one-dimensional intertransaction association rules.

4.1 Candidate Generation

Pass 1. To generate candidate set C_1 of 1-extended itemsets, for every item we attach all possible relative addresses within the range of [0, *maxspan*] in the one-dimensional space. That is:

$$C_1 = \{\{\Delta_x(i_n)\} | (i_n \in \mathcal{I}) \wedge (0 \leq x \leq maxspan)\}$$

Starting from the transaction t at the reference point Δ_s, i.e., extended transaction $\Delta_s(t)$, the transaction t' at the point Δ_{s+x} in the dimensional space, i.e., extended transaction $\Delta_{s+x}(t')$, is scanned to determine whether item i_n exists. If so, the count of $\Delta_x(i_n)$ increases by one. One scan of the database will deliver the large set L_1.

Pass 2. We generate a candidate two-extended itemset $\{\Delta_0(i_m), \Delta_x(i_n)\}$ from any two large one-extended itemsets in L_1, $\Delta_0(i_m)$ and $\Delta_x(i_n)$, in this way:

$$C_2 = \{\{\Delta_0(i_m), \Delta_x(i_n)\} | (x = 0 \wedge i_m < i_n) \vee (x \neq 0)\}$$

Note that, of any two-extended itemset in C_2, the minimal relative address is always 0.

Pass k > 2. Given L_{k-1}, the set of all large $(k - 1)$-extended itemsets, the candidate generation procedure returns a superset of large k-extended itemsets. This procedure has two parts. In the *join* phase, we join L_{k-1} with L_{k-1}. Let $p = \{\Delta_{u_1}(i_1), \Delta_{u_2}(i_2), \ldots, \Delta_{u_{k-1}}(i_{k-1})\}$ and $q = \{\Delta_{v_1}(j_1), \Delta_{v_2}(j_2), \ldots, \Delta_{v_{k-1}}(j_{k-1})\}$, and we have

insert into	C_k
select	$p.\Delta_{u_1}(i_1), \ldots, p.\Delta_{u_{k-1}}(i_{k-1}), q.\Delta_{v_{k-1}}(j_{k-1})$
from	$L_{k-1}\, p, L_{k-1}\, q$
where	$(p.\Delta_{u_1}(i_1) = q.\Delta_{v_1}(j_1)) \wedge \ldots \wedge (p.\Delta_{u_{k-1}}(i_{k-1}) < q.\Delta_{v_{k-1}}(j_{k-1}))$

We define the comparison operators = and < between two extended itemsets as follows:

Definition 4.1. $\Delta_{x_m}(i_m) = \Delta_{x_n}(i_n)$, if and only if (1) $i_m = i_n$ and (2) $x_m = x_n$.

Definition 4.2. $\Delta_{x_m}(i_m) < \Delta_{x_n}(i_n)$, if and only if (1) $x_m < x_n$ or (2) $(x_m = x_n) \wedge (i_m < i_n)$.

Next, in the prune phase, we delete all those extended itemsets in C_k which have some $(k - 1)$-subsets with the supports less than *support*. If a subset's minimum interval is not 0, it can be subtracted from all the items' intervals, in order that they always start from 0.

4.2 Counting Support of Candidates

To facilitate the efficient support counting process, a candidate C_k of k-extended itemsets is divided into k groups, with each group G_o containing o number of items whose relative addresses (intervals) are 0 $(1 \leq o \leq k)$. For example, a candidate set of 3-extended itemsets

$$\begin{aligned} C_3 = \{ \; & \{\Delta_0(a), \Delta_1(a), \Delta_2(b)\}, \{\Delta_0(c), \Delta_0(d), \Delta_2(d)\}, \\ & \{\Delta_0(a), \Delta_0(b), \Delta_3(h)\}, \{\Delta_0(l), \Delta_0(m), \Delta_0(n)\}, \\ & \{\Delta_0(p), \Delta_0(q), \Delta_0(r)\} \end{aligned}$$

is divided into three groups:

$$G_1 = \{ \; \{\Delta_0(a), \Delta_1(a), \Delta_2(b)\} \; \}$$

$$G_2 = \{ \; \{\Delta_0(c), \Delta_0(d), \Delta_2(d)\}, \{\Delta_0(a), \Delta_0(b), \Delta_3(h)\} \; \}$$

$$G_3 = \{ \; \{\Delta_0(l), \Delta_0(m), \Delta_0(n)\}, \{\Delta_0(p), \Delta_0(q), \Delta_0(r)\} \; \}.$$

Each group is stored in a modified *hash-tree*. Only those items with interval 0 participate in the construction of this hash-tree. For instance, in G_2, only $\{\Delta_0(c), \Delta_0(d)\}$ and $\{\Delta_0(a), \Delta_0(b)\}$ enter the hash-tree. The construction process is similar to that of Apriori [Agrawal and Srikant 1994]. The rest of the items, $\Delta_2(d)$ and $\Delta_3(h)$, are simply attached to the

k=1

$C_1 = \{\{\Delta_x(i_n)\} \mid (i_n \in \mathcal{I}) \ \wedge \ (0 \le x \le maxspan)\}$
EH-Apriori: set all the buckets of $HashTable$ to 0;
foreach extended transaction $\Delta_s(t)$ **do**
    foreach candidate $c : \Delta_x(i_n) \in C_1 \ \ (\Delta_x(i_n) \in \Delta_{s+x}(t'))$ **do**
        c.count++;
        EH-Apriori: **foreach** 2-extended itemset $p : \{\Delta_0(i_m), \Delta_x(i_n)\}$
            $(\Delta_0(i_m) \in \Delta_s(t), \ \Delta_x(i_n) \in \Delta_{s+x}(t'))$ **do**
            $HashTable$[func(p)]++;
    endfor
$L_1 = \{c : \{\Delta_x(i_n)\} \mid (c \in C_1) \ \wedge \ (c.\text{count} \ge support)\}$

k=2

$C_2' = \{\{\Delta_0(i_m), \Delta_x(i_n)\} \mid (\Delta_0(i_m) \in L_1) \ \wedge \ (\Delta_x(i_n) \in L_1) \ \wedge \ ((x \neq 0) \vee (x = 0 \ \wedge i_m < i_n))\}$
EH-Apriori: $C_2 = \{c : \{\Delta_0(i_m), \Delta_x(i_n)\} \mid (c \in C_2') \ \ \wedge \ (HashTable[\text{func}(c)] \ge support)\}$
E-Apriori: $C_2 = C_2'$
foreach extended transaction $\Delta_s(t)$ **do**
    foreach candidate $c : \{\Delta_0(i_m), \Delta_x(i_n)\} \in C_2 \ \ \ (\Delta_0(i_m) \in \Delta_s(t), \ \Delta_x(i_n) \in \Delta_{s+x}(t'))$ **do**
        c.count++;
$L_2 = \{c : \{\Delta_0(i_m), \Delta_x(i_n)\} \mid (c \in C_2) \ \wedge \ (c.\text{count} \ge support)\}$

k>2

for $(k = 3; L_{k-1} \neq \phi; k++)$ **do**
    C_k=E-Apriori-Gen(L_{k-1});
    for $(o = 1; o \le k; o++)$ **do**
        G_o=E-Group(C_k, o);
        foreach extended transaction $\Delta_s(t)$ **do**
            $G_{\Delta_s(t)}$=E-Subset($G_o, \Delta_s(t)$); // candidates in group G_o contained in consecutive transactions starting from $\Delta_s(t)$
            foreach candidate $c : \{\Delta_{x_1}(i_1), \ldots, \Delta_{x_k}(i_k)\} \in G_{\Delta_s(t)}$ **do**
                c.count++;
        endfor
    endfor
    $L_k = \{c : \{\Delta_{x_1}(i_1), ..., \Delta_{x_k}(i_k)\} \mid (c \in C_k) \ \wedge \ (c.\text{count} \ge support)\}$
endfor
$L = \bigcup_k L_k$;

Fig. 2. E-Apriori and EH-Apriori algorithms.

corresponding itemsets, $\{\Delta_0(c), \Delta_0(d)\}$ and $\{\Delta_0(a), \Delta_0(b)\}$ respectively, in the leaves of the tree.

Upon reading one transaction of the database, every hash-tree is tested. If one itemset is contained, its attached itemsets whose intervals are larger than 0 will be checked against the successive transactions. In the above example, if $\{\Delta_0(a), \Delta_0(b)\}$ exists in the current extended transaction at the point Δ_s, then the extended transaction $\Delta_{s+3}(t')$ will be scanned to see whether it contains item h. If so, the support of 3-extended itemset $\{\Delta_0(a), \Delta_0(b), \Delta_3(h)\}$ will increase by 1. Figure 2 shows the algorithms *E-Apriori* and *EH-Apriori*.

E-Apriori and *EH-Apriori* share the same procedures, except that in Pass 1, *EH-Apriori* hashes all two-itemsets, such as $\{\Delta_0(i_m), \Delta_x(i_n)\}$, contained in the current series of extended transactions into the corresponding buckets of a *HashTable*, and prunes unnecessary two-extended itemsets from C_2 in pass 2, whose corresponding bucket values in the *HashTable* are less than *support*. The hash function that we used is as below.

FUNCTION 4.1. *Let "#buckets" denote the average number of hash buckets allocated to two-itemsets which have the same relative address combination (e.g.,* (Δ_0, Δ_x) *is the address combination of 2-itemset* $\{\Delta_0(i_m), \Delta_x(i_n)\}$*); we have*

$$\#buckets = \lfloor TotalHashEntry/(maxspan + 1) \rfloor.$$

For each bucket, we assign "#itemsets" number of two-itemsets, where

$$\#itemsets = \lceil (TotalItem \times TotalItem)/\#buckets. \rceil.$$

Given a two-itemset $p = \{\Delta_0(i_m), \Delta_x(i_n)\}$*, the hash function* $h(p)$ *is computed by the following formula:*

$$h(p) = \Delta_x \times \#buckets + (((i_m - 1)\ mod\ TotalItem) \times TotalItem + (i_n - 1))/\#itemsets$$

Functions E-Apriori-Gen( ), E-Group( ), and E-Subset( ) are used to generate candidates, group candidates, and count support of candidates when $k > 2$.

5. PERFORMANCE STUDY

To assess the performance of the proposed algorithms, we conducted a series of experiments on both synthetic and real-life data. The method used to generate synthetic data is described in Section 5.1, and Section 5.2 presents some experimental results from this. Results obtained from real data are described in Section 5.3.

5.1 Generation of Synthetic Data

The method used by this study to generate synthetic transactions is similar to the one used in Agrawal and Srikant [1994], with some modifications noted below. Table II summarizes the parameters used and their settings. For simplicity, we use *itemset* and *extended itemset* interchangeably in the following.

Transaction sizes are typically clustered around a mean, and a few transactions have many items. Typical sizes of large itemsets are also clustered around a mean, with a few large itemsets having a large number of items across different transactions.

We first generate a set L of the potentially large itemsets, which may span different transactions, e.g., $\{\Delta_0(a), \Delta_1(b), \Delta_2(c)\}$, and then assign a large itemset from L to corresponding transactions.

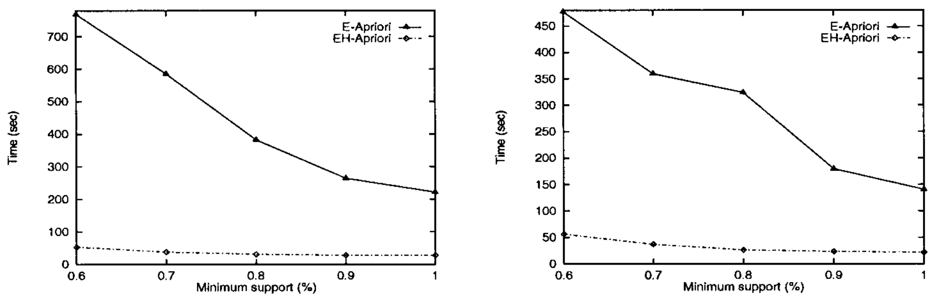
Table II. Parameters and Settings

Parameter	Meaning	Setting
$\|D\|$	number of transactions	60K - 100K
$\|T\|$	average size of the transactions	4 - 7
$\|MT\|$	maximum size of the transactions	7 - 10
$\|L\|$	number of potentially large extended itemsets	1000
$\|I\|$	average size of the potentially large extended itemsets	1.5, 3
$\|MI\|$	maximum size of the potentially large extended itemsets	4, 7
N	number of items	600 - 1000
W	maxspan (maximum interval)	0 - 4

The number of potentially large itemsets is set to $|L|$. A potentially large itemset is generated by first picking the size of the itemset from a Poisson distribution with mean equal to $|I|$. The maximum size of potentially large itemsets is $|MI|$. Items and their relative addresses (intervals) in the first large itemset are chosen randomly, where item is picked up from 1 to N, and its interval is picked up from 0 to W. To model the phenomenon that large itemsets often have common items and intervals, some fraction of items and their intervals in subsequent itemsets are chosen from the previous itemset generated. We use an exponentially distributed random variable with mean equal to the *correlation level* to decide this fraction for each itemset. The remaining items and their intervals are picked at random. In the datasets used in the experiments, the correlation level is set to 0.5. Having generated all the items and intervals for a large itemset, we revise each of its intervals by subtracting the minimum interval value of this large itemset. In this way, the minimum interval of each potentially large itemset is always 0.

After generating the set L of potentially large itemsets, we then generate transactions in the database. Each transaction is assigned a series of potentially large itemsets. However, upon the generation of one transaction, we must consider a list of consecutive ones starting from it, as items in a large itemset may span different transactions. For example, after selecting the large itemset $\{\Delta_0(a), \Delta_1(b), \Delta_2(c)\}$ for current transaction $\Delta_s(t)$, we should assign item a to t, item b to its next transaction t' which is one unit away, i.e., $\Delta_{s+1}(t')$, and item c to the transaction at the point Δ_{s+2}.

Before assigning items to a list of consecutive transactions, we should determine the sizes of those transactions. The size of each transaction is picked from a Poisson distribution with mean equal to $|T|$. The maximum size of transactions is $|MT|$. Each potentially large itemset has a weight associated with it, which corresponds to the probability that this itemset will be picked. The weight is picked from an exponential distribution with unit mean, and is then normalized so that the sum of the weights for all the itemsets in L is 1. The next itemset to be put in the transaction is chosen from L by tossing an $|L|$-sided weighted coin, where the weight for each side is the probability of picking the associated itemset.



(a) T5-I3-W3-N1000-D100K (b) T5-I1.5-W2-N1000-D100K

Fig. 3. Minimum support versus execution time.

If the large itemset picked on hand does not fit in the current, or any one of its successive transactions, it is put in these transactions anyway in half the cases, and enters an *unfit* queue for the next transaction in the rest of the cases. Each time, we pick itemsets from this queue first, according to the first-in-first-out principle. Only when the queue is empty, do we perform random selection from the set L. As in Agrawal and Srikant [1994], we use a corruption level during the transaction generation to model the phenomenon that all the items in a large itemset do not always occur together. This corruption level for an itemset is fixed and is obtained from a normal distribution, with mean 0.5 and variance 0.1.

We generated datasets by setting $N = 1000$ and $|L| = 1000$. We chose four values for $|T|$: 4, 5, 6, and 7. The number of transactions was set from 60K to 100K. We also chose two values for $|I|$: 3 and 1.5. The corresponding $|MI|$ was set as 7 and 4, respectively. The maximum interval W was varied from 0 to 4.

5.2 Experiments on Synthetic Data

We study the scalability of *E-Apriori* and *EH-Apriori* algorithms using synthetic generated data. Four sets of experiments were conducted on a Sun Ultra Sparc Workstation with a CPU clock rate of 143MHz and 64MB main memory. We report the running performance of both algorithms in terms of wall-clock time.

5.2.1 *Basic Experiment.* The first set of experiments studies the basic behavior of the algorithms when the minimum support changes. The results are shown in Figure 3.

As the minimum support increases, the execution times of both *E-Apriori* and *EH-Apriori* decrease because of reduction in the total number of candidate and large itemsets. Throughout the experiments, *EH-Apriori* is always far superior than *E-Apriori*. For example, in Figure 3(a), when minimum support is 0.7%, the mining time of *EH-Apriori* is about 38 seconds while that of *E-Apriori* is about 585 seconds. The former only needs to spend 6.5% of the time of the latter to get the same mining results. This is not surprising if we look at their running times at Pass 1 and 2 in Table

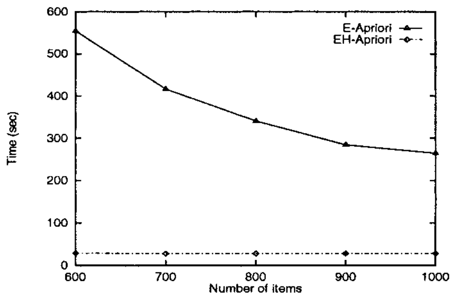
Table III. Comparison of E-Apriori and EH-Apriori

T5-I3-W3-N1000-D100K				
	support=0.7%		support=0.9%	
	E-Apriori	EH-Apriori	E-Apriori	EH-Apriori
t_1	2.5 s	21.1 s	2.5 s	21.1 s
$\|C_2\|$	27035	115	102258	14
t_2	576.4 s	10.8 s	257.6 s	3.0 s
$t_1 + t_2$	578.9 s	31.9 s	260.1 s	24.1 s
Total Mining Time	585.2 s	38.3 s	264.0 s	27.9 s
T5-I1.5-W2-N1000-D100K				
t_1	2.0 s	13.7 s	2.0 s	13.8 s
$\|C_2\|$	141491	182	59213	58
t_2	351.4 s	17.4 s	173.9 s	6.0 s
$t_1 + t_2$	353.4 s	31.1 s	175.9 s	19.8 s
Total Mining Time	359.6 s	37.2 s	179.4 s	23.3 s

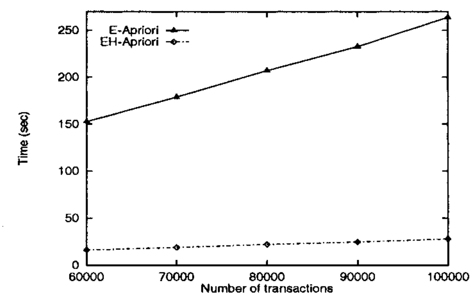
III. After Pass 2, both algorithms behave exactly the same. As shown, although the execution time of the first pass of *EH-Apriori* is slightly larger than that of *E-Apriori*, due to the extra overhead required for building *HashTable*, *EH-Apriori* incurs significantly smaller execution time than *E-Apriori* in the latter Pass 2, as $|C_2|$ is decreased greatly from 270355 to 115 by means of hash filtering (at T5-I3-W3-N1000-D100K, minimum support = 0.7%). Less $|C_2|$ results in much less time to test against each transaction of the database. Table III also shows the dominance of the first two passes over the overall mining performance. From this preliminary experiment, we can see that strategies aimed at pruning unnecessary candidates for intertransaction association rule mining can offer greater benefits than for the classical association rule mining [Park et al. 1995].

5.2.2 *Scale-Up Experiment.* The second set of experiments is designed to study scale-up properties of *E-Apriori* and *EH-Apriori*. Figure 4(a) shows how *E-Apriori* and *EH-Apriori* behave as the number of items in a database increases from 600 to 1000.

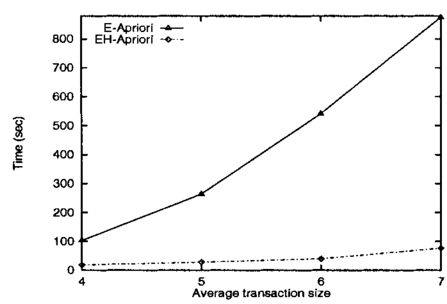
It is interesting to note that, when the number of items increases and keeping the minimum support unchanged, the execution time of *E-Apriori* decreases while that of *EH-Apriori* remains nearly unchanged. This is due to the fact that, with more items, each transaction under a certain average and maximum transaction size condition is more likely to have different items assigned, resulting in less number of large one-itemsets that meet the *support* threshold. The smaller $|L_1|$ leads directly to less candidates C_2 being generated, and therefore less time for *E-Apriori* to count them. However, for *EH-Apriori*, as the hash function being utilized can prune a large number of candidate two-itemsets, the real $|C_2|$ left is nearly the same, although the total number of items is different. From the experiment records, we find that $|C_2| = 15$ when there are 600 items in the database ($N = 600$), and $|C_2| = 14$ when $N = 1000$. Hence, the time spent on pass 2 changes little under *EH-Apriori*. This test shows us that setting an



T5-I3-W3-D100K, support=0.9%
(a) Number of items scale-up



T5-I3-W3-N1000, support=0.9%
(b) Number of transactions scale-up



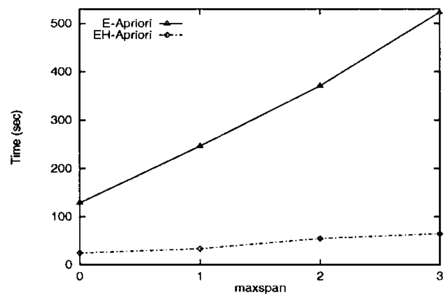
I3-W3-N1000-D100K, support=0.9%
(c) Average transaction size scale-up

Fig. 4. Results of scale up experiments.

appropriate hash function can make *EH-Apriori* robust to different item numbers.

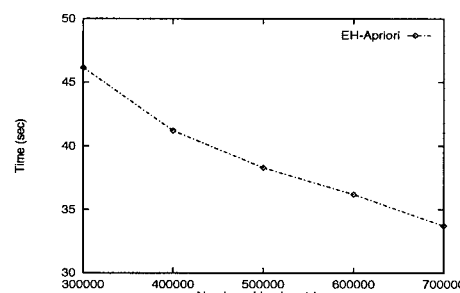
Next, we examine how both algorithms scale up with the number of transactions in the database. We increase the number of transactions from 60K to 100K. The results shown in Figure 4(b) coincide with our expectation: the execution times of both algorithms increase when more transactions in a database need to be scanned and checked. As shown, their mining times scale quite linearly.

Finally, we investigate the scale-up as we increase the average transaction size from 4 to 7. From the results presented in Figure 4(c), we may observe that for both algorithms, the more items per transaction, the more time needed to process. The reason is obvious: given a minimum support and a set of items, when the average transaction size is large, there is more $|L_1|$ generated; hence more $|C_2|$ needs to be counted. Also, the time needed to scan every transaction of the database becomes longer, resulting in higher processing costs. For example, at average transaction size 5, the execution times of *E-Apriori* and *EH-Apriori* are about 264 seconds and 28 seconds respectively, but at average transaction size 7, they increase dramatically to 875 seconds and 77 seconds.



T5-I3-N1000-D100K, support=0.8%

Fig. 5. Maximum interval versus time.



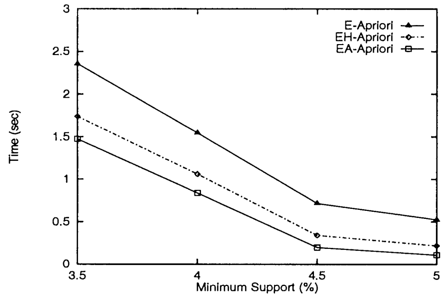
T5-I3-W3-N1000-D100K, support=0.7%

Fig. 6. Hash entries versus time.

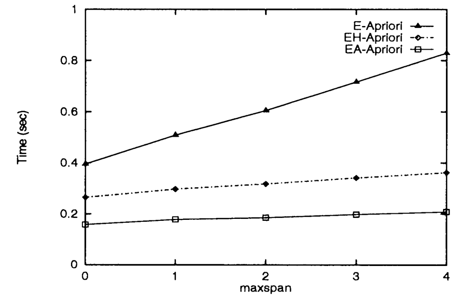
5.2.3 *Effect of the Maximum Interval.* In our third experiment, we study the impact of *maximum interval* (*maxspan*) on the performance of mining algorithms.

In Figure 5, when the maximum interval is 0, mining intertransaction association rules degrades to mining classical association rules. When the maximum interval increases, much more candidates, such as $\{\Delta_0(a), \Delta_x(a)\}(x \neq 0)$, are added into C_2. Thus, both *E-Apriori* and *EH-Apriori* have to spend more time to count candidate C_k. For instance, the execution time of *E-Apriori* increases from 129 seconds at maximum interval 0 to 523 seconds at maximum interval 3, and the execution time of *EH-Apriori* increases from 25 seconds to 64 seconds, about 75% and 60% more respectively. From this experiment, we may find that in the situation of more candidates, *EH-Apriori* can outperform *E-Apriori*, exhibiting more advantages of hash filtering techniques.

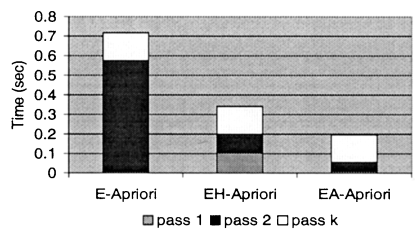
5.2.4 *Effect of the Size of a Hash Table.* Observing that the hash table size used by *EH-Apriori* affects the cardinality of C_2 generated, in the last experiment on synthetic data, we examine the performance of *EH-Apriori* under five different *HashTable* sizes, varying from 300K to 700K. As illustrated in Figure 6, with more hash entries, the problem of hash



maxspan=3
(a) Minimum support versus time



support=4.5%
(b) Maxspan versus time



maxspan=3, support=4.5%
(c) Time spent on different passes

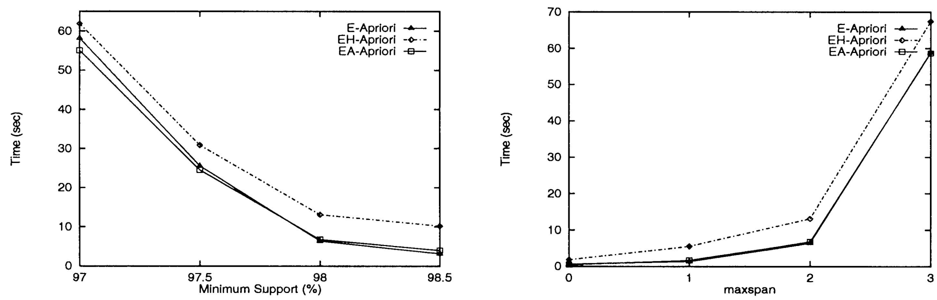
Fig. 7. Execution time on WINNER stock data (T10-N84-D250).

collision can be alleviated a little more. Therefore, more useless two-itemsets can be pruned away effectively before the database is scanned. *EH-Apriori* can thus achieve more improvement than *E-Apriori*.

5.3 Experiments on Real Data

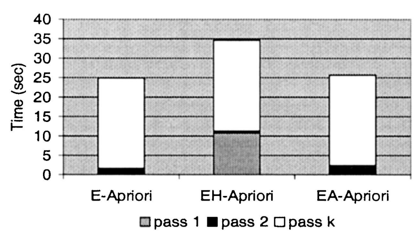
To study the applicability of intertransaction association rules, we conduct experiments on two different real datasets, obtained from Singapore Stock Exchange and Hong Kong Observatory respectively. In this subsection, we summarize the results of the experiments.

5.3.1 *Tests with 1996 Singapore Stock Data.* From the price data of 1996 Singapore Stock Exchange (SSE), we created two datasets: a WINNER set and a LOSER set. Records in both sets have the same format (*date*, *counter-list*). The counter list in the WINNER set contains those counters whose closing prices are 3% more than the previous closing prices; and the counter list in the LOSER set the other counters. Since there were 250 trading days in 1996, both datasets have 250 records. As there are some missing values in the original data we have, we only get 84 counters that have complete price information from every trading day. Furthermore, as the major trend for SSE in 1996 was down side, the average transaction



maxspan=2
(a) Minimum support versus time

support=98%
(b) Maxspan versus time



maxspan=3, support=98.5%
(c) Time spent on different passes

Fig. 8. Execution time on LOSER stock data (T70-N84-D250).

size of LOSER is pretty large (more than 70), while that of WINNER is relatively small (less than 10).

Since we do not have a large number of items (Total_Item = 84) in the datasets, the candidate C_2 fits in memory during the experiments. As pointed out by Ramaswamy et al. [1998], "*Using a* $L_1 * L_1$ *2-dimensional array instead of a hash-tree for discovering large 2-itemsets is much faster when memory permits.*" We implement such a technique in another algorithm called *EA-Apriori* as a comparison with *E-Apriori* and *EH-Apriori* algorithms.

Figure 7 shows the experimental results of three algorithms on the WINNER dataset when support threshold and maxspan change. In accordance with results obtained from synthetic datasets, *EH-Apriori* always outperforms *E-Apriori*. Nevertheless, because the third *EA-Apriori* algorithm eliminates the effort of building the hash table and navigating the hash-tree to count candidate two-itemsets, it achieves the best performance of the three.

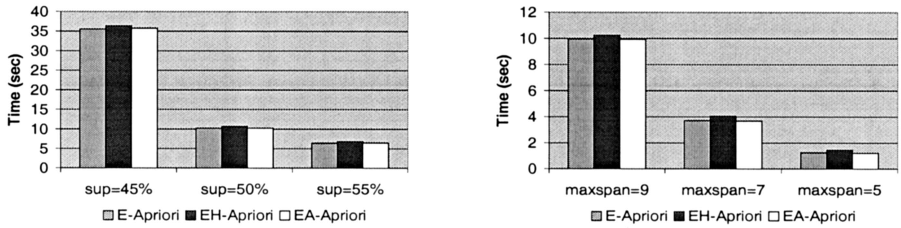
As the WINNER data set is small, with the average transaction size 10, we do not have rules with large support. However, we did find some interesting rules such as "Δ_0(*HAISUNWT*), Δ_0(*KIMENGWT*) $\Rightarrow$ Δ_1(*HAISUNWT*)" at support = 3.5% and confidence = 80%. That is, if HAISUN Warrant stock and KIMENG Warrant stock, belonging to the loans and

Table IV. Items Used in the Hong Kong Meteorological Datasets

Meteorological Element	ItemId	Value Range	Meaning
Wind Direction (^{o})	1	(0, 45]	North East
	2	(45, 90]	East
	3	(90, 135]	South East
	4	(135, 180]	South
	5	(180, 225]	South West
	6	(225, 270]	West
	7	(270, 315]	North West
	8	(315, 360]	North
Wind Speed (km/h)	9	(2, 12]	Light
	10	(12, 30]	Moderate
	11	(30, 40]	Fresh
	12	(40, 62]	Strong
Rainfall (cm)	13	[0, 0.05)	No Rain
	14	[0.05, 0.1)	Trace
	15	[0.1, 4.9)	Light
	16	[4.9, 25.0)	Moderate
	17	[25.0, 100.0)	Heavy
Relative Humidity (%)	18	[0, 50]	Very Dry
	19	(50, 70]	Dry
	20	(70, 90]	Medium Wet
	21	(90, 100]	Wet
Dry Bulb Temperature (^{o}C)	22	[0, 10]	Very Cold
	23	(10, 16]	Cold
	24	(16, 22]	Mild
	25	(22, 28]	Warm
	26	(28, 50]	Hot
Mean Sea Level Pressure (hPa)	27	(0, 9950]	Very Low
	28	(9950, 10000]	Low
	29	(10000, 10050]	Moderate
	30	(10050, 10100]	slightly High
	31	(10100, 10150]	High
	32	(10150, 20000]	Very High

bond sectors respectively, go up the same day, HAISUN Warrant will continue to increase the next day.

The test result of the LOSER data set, which has the average transaction size 70, is illustrated in Figure 8. Surprisingly, *EH-Apriori* becomes the worst algorithm. Looking at Figure 8(c), we find that *EH-Apriori* takes quite a lot of time at $k = 1$ to build up the hash table. Hence, the benefit of pruning out unnecessary candidates using the hash table disappears when transactions have more items on average. Another interesting observation made from this set of experiments is that the performance of both *EA-Apriori* and *E-Apriori* is very similar. By checking the experiment records



maxspan=11 support=45%

Fig. 9. Execution time on Hong Kong meteorological data (T6-T32-D736).

in detail, we note that $|C_2| = 305$ when support = 98%, and our *E-Apriori* parameter setting allows all the 305 candidate two-itemsets to be within one leaf. In this case, the hash tree contains only one leaf node. Thus, the efforts in navigating the hash tree to count candidates under *E-Apriori* decreases, making both *E-apriori* and *EA-Apriori* behave similarly. However, once one leaf cannot accommodate all the candidate itemsets, e.g., $|C_2| = 1090$ at support = 97%, *E-Apriori* needs to spend more time than *EA-Apriori*.

From the LOSER dataset, we found a total of 623 intertransaction association rules at support = 98.5% and confidence = 99%; one example rule is "$\Delta_0(UOL)$, $\Delta_1(SIA) \Rightarrow \Delta_2(DBS)$." It says that, if *UOL* goes down and *SIA* goes down the next day, *DBS* will go down on the second day. Here, *UOL* stock represents the land market, *SIA* stock represents loans and debentures, and *DBS* represents the banking market. This rule reveals the causal relationship among three major stocks in Singapore. As is known, land properties in Singapore play an important role in national economic development, and therefore their decay inevitably leads to a bad performance of loans and banking. Such a discovered rule reveals some of the characteristics of Singapore's economic structure.

5.3.2 *Tests with Hong Kong Meteorological Data.* To examine the potential applications of intertransaction association rules further, we run the algorithms against meteorological data obtained from the Hong Kong Observatory headquarters, which takes observations, including *wind direction, wind speed, dry bulb temperature, relative humidity, rainfall*, and *mean sea level pressure*, etc., every 6 hours each day. We try to find intertransaction association rules from the 1995 meteorological data, and examine their prediction accuracy using the 1996 meteorological data from the same area in Hong Kong. Considering seasonal changes of weather, we extract records from May to October for our experiments, and there are therefore totally 736 records (total_days * 4 = (31 + 30 + 31 + 31 + 30 + 31) * 4 = 736) for each year. These raw datasets, containing continuous data, are further converted into appropriate formats with which the algorithms can work, and Table IV lists 32 items after the data transformation. Each record has 6 meteorological elements (items). The interval of every

Table V. Some Significant Intertransaction Association Rules

Rule	Sup.	Conf.	Pred. Rate
$\Delta_0(2), \Delta_3(13) \Rightarrow \Delta_4(13)$ ("if there is an *east* wind direction and *no rain* in 18 hours, then there will also be a *no rain* in 24 hours.")	46%	92%	90.0%
$\Delta_0(2), \Delta_0(25), \Delta_2(25) \Rightarrow \Delta_3(25)$ ("if it is currently *warm* with an *east* wind direction, and still *warm* 12 hours later, then it will be continuously *warm* until 18 hours later.")	45%	95%	91.8%

two consecutive records is 6 hours. We set *maxspan* = *11* in order to detect the association rules happening within 3 days (i.e., (11 + 1)/4 = 3).

Figure 9 gives the mining times of *E-Apriori, EH-Apriori*, and *AH-Apriori* at different parameter settings. Comparatively, the difference between $|C_2|$ and $|L_2|$ is small in these experiments. Thus, the role of the hash table in *EH-Apriori* to eliminate unnecessary candidate two-itemsets is not obvious, although little effort is required to build the hash table at the average transaction size 6.

At support = 45% and confidence = 92%, from the 1995 Hong Kong meteorological data we found only 1 classical association rule—"*if the weather is medium wet, then there is no rain at the same time*" (which is quite obvious), but 580 intertransaction association rules (*maxspan* = *11*). Note that the number of intertransaction association rules depends on the *maxspan* setting. Table V lists some significant intertransaction association rules found. We measure their prediction capabilities using the 1996 meteorological data by *Prediction-Rate* $(X \Rightarrow Y) = sup(X \cup Y)/sup(X)$, which can achieve more than 90%.

Our study on real data is on-going. The results obtained so far indicate that, with intertransaction association rules, we can discover more comprehensive and interesting knowledge from the databases.

6. DISCUSSIONS

We have described the multidimensional intertransaction association rules, related mining algorithms, and some experimental results on synthetic and real data. For ease of understanding and simplicity, the discussion has been limited to one dimension, which happens to be time. In this section, we compare such association rule mining with several related data mining work, and then discuss how the technique addressed in this paper can scale beyond one dimension.

6.1 Related Work

The problem of multidimensional intertransaction association rule mining looks similar to the problems of sequential pattern and rule discovery when there is only one dimension involved. In this subsection, we will discuss the differences between the new problem and related work.

Traditional association rule mining. The original association rule mining proposed by Agrawal et al. [1993] is apparently a special case of the multidimensional intertransaction association rule mining: If we omit the dimensional attributes in the transactions, and set the window size to one, the multidimensional intertransaction association rule mining will degrade to intratransaction association rule mining.

Sequential pattern discovery. The problem of sequential pattern mining, as defined in Agrawal and Srikant [1995], is to find the maximal sequential patterns among all sequences that have a certain user-specified minimum support. Sequential pattern mining can be viewed as an extension of the classical association rule mining along two directions. First, the contents of the transaction database are different. In classical association rule mining, each record in the transaction database contains only the items bought by a single transaction of a particular customer. For sequential pattern mining, each record in the database contains items bought by a particular customer, in different transactions during a period of time. Second, while there is no ordering of items in the classical association rule mining, items are ordered according to the time when they were bought. One sequential pattern example is "*80% of customers bought shoes after they bought shirts.*"

Despite these amendments, the basic problem of sequential pattern mining has not been changed: the association to be discovered is still among the items within a record of the transaction database, with an extra requirement that items in itemsets should appear in the same order. On the other hand, intertransaction association mining is fundamentally different: mining associations among items from different transaction records (customers). Besides, sequential patterns only focus on *successive/precedent* relationships of items. No concrete intervals, such as 5 days later, 2 miles away, etc., between the occurrence of items are captured within sequential patterns.

Episodes and generalized episodes discovery. Mannila et al. introduced the problem of discovering frequent episodes from sequential data [Mannila and Toivonen 1996]. An episode is a collection of events that occur relatively close to each other in a certain (partial) order, whose total span of time is constrained by a window. Later in 1997, the problem was further generalized to allow events to have attributes [Mannila et al. 1997]. For example, a Web page access can be viewed as an event with two attributes, *page* and *host*, specifying the page accessed and the host that made the access respectively. With generalized events, frequent episode rules have the form of

$$P[V] \Rightarrow Q[W]$$

where P and Q are episodes; and V and W are real numbers representing time intervals, stating that if episode P has a minimal occurrence at interval $[t, t']$ with $t' - t \leq V$, then episode Q occurs at interval $[t, t'']$ for some t'' such that $t'' - t \leq W$. A rule discovered from the WWW log could be as follows:

$$\begin{aligned} & x.page = p_1 \wedge y.page = p_2 \wedge x.host = y.host\ [60] \\ \Rightarrow\ & x.page = p_1 \wedge y.page = p_2 \wedge z.page = p_3 \\ & \wedge x.host = y.host \wedge y.host = z.host\ [120] \end{aligned} \tag{1}$$

This rule expresses that if a host accesses page p_1 and p_2 within one minute, page p_3 is likely to be accessed by the same host within two minutes. Like multidimensional association rules proposed in this paper, generalized episode rules can associate multiple attributes to events, which is not possible for traditional association rules. Although multidimensional intertransaction association rules are quite similar to generalized episode rules, when one of the dimensions is time, the former is more concise and flexible than the latter in representing the quantitative temporal relationships of events or items. For example, with intertransaction association rules, we can easily specify

$$\Delta_{0,h}(p_1, p_2) \Rightarrow \Delta_{1,h}(p_3), \Delta_{3,h}(p_1). \tag{2}$$

That is, if page p_1 and p_2 are accessed by a host, the same host will likely access page p_3 one minute later, and access page p_1 again three minutes later. Such rules cannot be expressed by episode rules, since only time intervals like $V = [t, t']$, $W = [t, t'']$ are used to roughly specify the order of events in an episode, and these intervals are constrained to have the same starting time t.

Another difference to the discovery of frequent episodes is that our mining algorithms can find all association rules within the specified time span, regardless of the ordering of events, with reasonable amount of time. As mentioned by the authors, only certain types of episodes with predefined predicates (such as simple episodes where no binary predicates are included) are easily detected using their mining algorithms. The efficient mining of more general episode rules with arbitrary time bounds from a large sequence remains an open problem.

Temporal relationship mining in time sequence. Compared to episode sequences, Bettini et al. looked for more complex event sequences from time sequential data [Bettini et al. 1996; 1998]. Unlike episodes where only the order, but not the concise quantitative relationships among events, can be expressed, Bettini's model allows temporal relationship among events to be quantitatively defined, even using different granularities. Compared with multidimensional intertransaction association rules, their model has two limitations. First, they only considered the mining task where an event structure is given, and only some of its event variables, including the starting event variable, is instantiated. Therefore, the mining process only discovers possible event instances that match the given structure based on the frequence on which the corresponding events occur in the event sequence. No algorithms are given to discover all event structures with frequence that exceeds a threshold. Second, their work focused on event sequences. It is obvious that rules above a certain confidence threshold can show the connections between events more clearly than event sequences alone [Mannila et al. 1997]. However, neither definitions nor mining algorithms regarding the rules were discussed in their context. It is worth pointing out that, with proper data preprocessing and careful treatment, multiple time granularities can be realized in multidimensional intertransaction association model.

Rule discovery from time-series data. In a more recent work, Das et al. studied the problem of finding rules which relate patterns from a time series to other patterns in the same series, or patterns in one series to patterns in another series [Das et al. 1998]. Their example rule "*if the Microsoft stock price goes up and Intel stock price falls, then IBM stock price goes up the next day*" looks very similar to intertransaction association rules. However, compared to the intertransaction association rules, the rules they studied are rather simple and limited to the following form:

$$\textit{If A occurs, then B occurs within time T} \tag{3}$$

where A and B are basic temporal patterns detected from those sequences. Although the authors mentioned that the rules can be extended to a more complex form, such as

$$A_1 \wedge \ldots \wedge A_h[V] \Rightarrow B[T] \tag{4}$$

stating *if* A_1 *and ... and* A_h *occur within V units of time, then B occurs within time T*, the rules are still limited to two windows only.

As a summary, we believe that multidimensional intertransaction association rule mining proposed in this paper not only describes a new problem that has not been addressed before, but also gives a uniform treatment to a number of association- and pattern-related mining problems.

6.2 Mining Intertransaction Association Rules in Multidimensional Space

Although the performance study in this paper focused on one-dimensional intertransaction association rule mining, the methods can be generalized to mine multidimensional intertransaction association rules, which involve multiple dimensional attributes and associated intervals. We start our discussion from two dimensions for ease of explanation, and then extend it to m-dimension.

Figure 10 illustrates a two-dimensional space. To avoid dealing with a large number of dimensional values, we discretize the domains of its dimensional attributes x and y into $|D_1|$ and $|D_2|$ equal-sized intervals, and use $\Delta_{i,j}$ to denote a point in the space. Each transaction in the database can be mapped through a function *dim* to a certain point. Transactions converg-

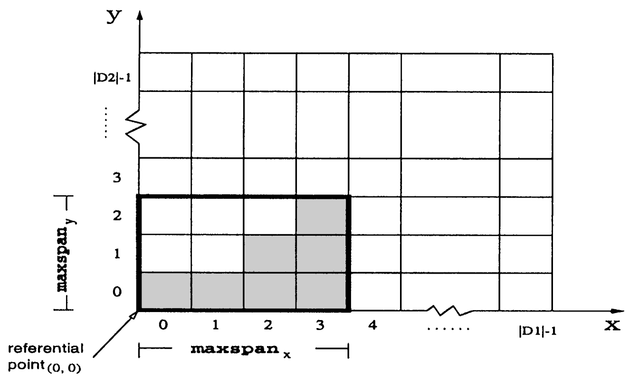


Fig. 10. Sliding region for a two-dimensional transaction database.

ing on the same point are combined into one, leading to $|D_1| * |D_2|$ total number of transactions in the extended database.

Similarly, we find two-dimensional intertransactions association rules based on large itemsets. The candidate itemsets under two dimensions can be generated in the same way as those under one dimension. However, the candidate counting effort increases dramatically with the number of dimensions. For example, to get the support of a three-itemset $\{\Delta_{0,0}(a), \Delta_{0,1}(a), \Delta_{1,2}(b)\}$, taking $\Delta_{0,0}$ as the reference point, we need scan transactions at $\Delta_{0,0}$, $\Delta_{0,1}$, and $\Delta_{1,2}$. Moving one step *up* (or *right*) and taking $\Delta_{0,1}$ (or $\Delta_{1,0}$) as the reference point, we need check another three transactions at $\Delta_{0,1}$, $\Delta_{0,2}$, and $\Delta_{1,3}$ (or $\Delta_{1,0}$, $\Delta_{1,1}$, and $\Delta_{2,2}$), and so on. Approximately, $|D_1| * |D_2|$ number of points can be taken as the reference point inside a two-dimensional context, and for each such point, the scanning scope covers several transactions, leading to a huge search space as compared to the case of one dimension.

To reduce the mining space, we can adopt a *sliding region*, which is similar to the sliding window in the one-dimensional context, to indicate the interesting spans (e.g., $maxspan_x$, $maxspan_y$) of associations along different dimensions. Figure 10 illustrates a 4×3 square-shaped sliding region. Besides the length limitation, more flexible dimension constraints can also be posed through sliding regions. For example, the shaded sliding region in Figure 10, excluding the reference point, tells that only associations whose x-axis dimensional value changes more quickly than their y-axis dimensional value ($x > y$) are of interest.

Generally, we can use a boolean expression $\mathcal{B}$ to represent a sliding region r, such that any point $\Delta_{x,y}$ in r except the reference point $\Delta_{0,0}$ must satisfy $\mathcal{B}$. Without loss of generality, we can assume that $\mathcal{B}$ is in disjunctive normal form $C_1 \vee C_2 \vee \ldots \vee C_s$, where each C_i is of the form $d_{i_1} \wedge d_{i_2} \wedge \ldots \wedge d_{i_t}$. Each element d_{i_j} can be either a condition on individual dimensional value (e.g., $x < 3$, $y < 2$), or a condition involving several dimensional values (e.g., $x > y$, $y = x + 2$). For instance, the shaded sliding region in Figure 10 can be described using the boolean expression $\mathcal{B}$: $(x < 5) \wedge (y < 4) \wedge (x > y)$.

Under the m-dimensional space where $m > 2$, totally $|D_1| * |D_2| * \ldots * |D_m|$ number of points could be taken as the reference point, based on which a series of transactions may be checked in order to find large itemsets. We can also use an m-dimensional *sliding cube* to reduce the search space. However, the mining cost is still significantly higher, sometimes even intractable, than for those under one or two dimensions, due to the huge space.

Inspired by the interesting work of Fukuda et al. on mining optimized association rules, which involve both numeric and boolean attributes [Fukuda et al. 1996a; 1996b], another possible method to tackle the efficiency problem of multidimensional intertransaction association mining is to explore the use of techniques developed in the computational geometry area, in order to perform fast context-based search and matching [Aggarwal et al. 1987].

6.3 Reducing Search Space Using Templates and Concept Hierarchies

Mining multidimensional intertransaction association rules is computationally more complex than mining traditional association rules. One of the major difficulties is the huge search space, and the overwhelming number of rules to be generated, than that in the traditional association rule mining situation. In order to make such rule mining truly practical and computationally tractable, besides the sliding-window approach adopted in this paper, we can explore the use of template models and conduct mining at multiple levels of abstraction.

—**Template-Guided Intertransaction Association Mining.** As seen in our discussion, sometimes it is still costly to mine intertransaction associations, and mining may also generate a large number of association rules because of a too large size of transaction database, too many itemsets contained in many transactions, or too large sliding windows. However, a user is often interested in only a small number of *particular* intertransaction associations. As an alternative to mining many rules and then selecting interesting ones from them, a query-directed, constraint-based approach can be adopted.

One way is to provide users with a template model to restrict the discovered rules to those which they are interested in. Essentially, a template is an expression used for specifying constraints on the intertransaction association rules. Similar to the studies of constraint-based mining of intratransaction associations [Ng et al. 1998], one can push constraints deeply into the intertransaction association mining process. For example, in a weather database, a user may only be interested in how *Central American* hurricanes influence the nearby weather, both in time (hours) and region (miles). In this case, only the region "*Central America*" and the weather pattern "hurricane" will be selected, and the study will be focused only to a limited-size window related to these hurricanes. A similar method can be applied to finding intertransaction associations related to highway traffic jam patterns. A user may only be interested in the situations of bad weather conditions and major accidents, instead of searching for all the cases from a large traffic database. Also, in stock movement prediction, a rule like "*When stock 'a' rises. and within the next two days another different stock also rises, then which stock will most likely rise one week later, following the rise of the second stock?*" might receive more attentions from investors.

Apparently, with the guidance of templates, mining can be more focused, and the number of rules generated can thus be reduced substantially. More detailed discussion can be found in our other paper [Feng et al. 1999].

—**Mining Intertransaction Associations at Multiple Levels.** Besides the sliding window and template-guided mining methods, we can also consider mining intertransaction association relationships at multiple levels of abstraction. Similar to mining multilevel intratransaction associations [Agrawal and Srikant 1995; Han and Fu 1995a], data involved in mining can be organized into concept hierarchies, and mining can be performed at multiple levels by drilling up or down along them to find more general or special rules. This is especially important in mining intertransaction associations, since sometimes there exist very few intertransaction associations with high support at high-level concepts. It is helpful to first mine at a high level and progressively drill down to catch interesting patterns. For example, for a stock exchange database, one may organize the concepts of stock price fluctuation like {*slightly-up, moderately-up, sharply-up* $\subset$ *UP*}, based on which one may first find the intertransaction associations at the rough granularity {*UP*, *DOWN*}, and then drill-down to see more detailed changes, such as *sharply-up* versus *moderately-down*. For the same reason, one may partition geographic regions into different hierarchies (such as Asia, North America, Europe, etc.), or partition stock categories into refined subcategories (such as oil, computer, automobile, etc.) so as to facilitate progressive deepening in the data mining process.

7. CONCLUSION

While the classical association rules have demonstrated strong potential values, such as to improve market strategies for retail industry, they are limited to finding associations among items within a transaction. In this paper, we propose a more general form of association rules, named *multidimensional intertransaction association rules*. These rules can represent not only the associations of items *within* transactions, but also associations of items *among* different transactions. The classical association rule can be viewed as a special case of multidimensional intertransaction association rules. We implement several algorithms for finding such intertransaction association rules by extensions of Apriori, and present some performance results by applying the algorithms to both synthetic and real datasets.

One interesting observation made from the experiments is that, given a specific minimum support, the candidate set of two-itemsets in the intertransaction association mining is much larger than that in the classical intratransaction association mining. Therefore, strategies to prune unnecessary two-itemsets from the candidate set is more beneficial to the overall performance, than it is in the case of classical association mining. Our experimental results show the great advantages of hash and array techniques [Park et al. 1995; Ramaswamy et al. 1998], which are adopted in the previous studies, on this aspect. However, when datasets to be mined have a large average transaction size, say around 70, the benefits of using the hash technique to reduce the number of candidates disappear, since building a hash table itself takes quite a long time.

We view this work as a first step, with a number of interesting problems and opportunities remaining for future work. First, associations among transactions were studied at specific points in the m-dimensional space. We may consider to extend such point context to cover wider ranges and subspaces, so that more general rules like "*After McDonald and Burger King open branches, KFC will likely open a branch within two months and less than one mile away*" can be discovered. Such extension might also be helpful to handle those transactions with unequal intervals and holes in the m-dimensional space. Furthermore, to prevent mining uninteresting rules at high computational costs, we plan to provide a more general framework, to enable users to specify certain contexts under which the association rules are to be mined. Besides, development of efficient discovery algorithms for two- and m-dimensional intertransaction association rules is another issue worth exploration. It might also be interesting to study the intertransaction association rule mining in distributed and parallel environments.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for many valuable comments and suggestions that have improved the quality of this paper.

REFERENCES

Agrawal, R. and Shafer, J. 1996. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering 8*, 6, 962–969.

Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proc. of the 20th Intl. Conf. on Very Large Data Bases* (Santiago, Chile, September 1994), pp. 478–499.

Agrawal, R. and Srikant, R. 1995. Mining sequential patterns. In *Proc. of the International Conference on Data Engineering* (Taipei, Taiwan, March 1995), pp. 3–14.

Aggarwal, A., Klawe, M., Moran, S., Shor, P., and Wilbur, R. 1987. Geometrics applications of a matrix-searching algorithm. *Algorithmica 2*, 209–233.

Agrawal, R., Imielinski, T., and Swami, A. 1993. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (Washington, D.C., USA, May 1993), pp. 207–216.

Baralis, E. and Psaila, G. 1997. Designing templates for mining association rules. *Journal of Intelligent Information Systems 9*, 1, 7–32.

Bettini, C., Wang, X., and Jajodia, S. 1996. Testing complex temporal relationships involving multiple granularities and its applications to data mining. In *Proc. of the Fifteenth*

ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (Montreal, Canada, June 1996), pp. 68–78.

BETTINI, C., WANG, X., AND JAJODIA, S. 1998. Mining temporal relationships with multiple granularities in time sequences. *Data Engineering 21*, 1 (March), 32–38.

BRIN, S., MOTWANI, R., AND SILVERSTEIN, C. 1997. Beyond market basket: generalizing association rules to correlations. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (Tucson, Arizona, USA, June 1997), pp. 265–276.

CHEUNG, D., HAN, J., NG, V., AND WONG, C. 1996a. Maintenance of discovered association rules in large databases: an incremental updating technique. In *Proc. of the Intl. Conf. on Data Engineering* (New Orleans, Louisiana, USA, February 1996), pp. 106–114.

CHEUNG, D., NG, V., FU, A., AND FU, Y. 1996b. Efficient mining of association rules in distributed databases. *IEEE Transactions on Knowledge and Data Engineering 8*, 6, 911–922.

DAS, G., LIN, K.-I., MANNILA, H., RENGANATHAN, G., AND SMYTH, P. 1998. Rule discovery from time series. In *Proc. of the second International Conference on Knowledge Discovery and Data Mining* (New York, USA, August 1998), pp. 16–22.

FANG, M., SHIVAKUMAR, N., GARCIA-MOLINA, H., MOTWANI, R., AND ULLMAN, J. 1998. Computing iceberg queries efficiently. In *Proc. of the 24th Intl. Conf. on Very Large Data Bases* (New York, USA, August 1998), pp. 299–310.

FENG, L., LU, H., YU, J., AND HAN, J. 1999. Mining inter-transaction association rules with templates. In *Proc. ACM CIKM Intl. Conf. Information and Knowledge Management* (November 1999), pp. 225–233.

FUKUDA, T., MORIMOTO, Y., MORISHITA, S., AND TOKUYAMA, T. 1996a. Mining optimized association rules for numeric attributes. In *Proc. of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Montreal, Canada, June 1996), pp. 182–191.

FUKUDA, T., MORIMOTO, Y., MORISHITA, S., AND TOKUYAMA, T. 1996b. Data mining using two-dimensional optimized association rules: Schema, algorithms, and visualization. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (Montreal, Canada, June 1996), pp. 13–23.

HAN, E.-H., KARYPIS, G., AND KUMAR, V. 1997. Scalable parallel data mining for association rules. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (Tucson, Arizona, USA, June 1997), pp. 277–288.

HAN, J. AND FU, Y. 1995a. Discovery of multiple-level association rules from large databases. In *Proc. of the 21st Intl. Conf. on Very Large Data Bases* (Zurich, Switzerland, September 1995), pp. 420–431.

HAN, J. AND FU, Y. 1995b. Meta-rule-guided mining of association rules in relational databases. In *Proc. of the 1st Intl. Workshop on Integration of Knowledge Discovery with Deductive and Object-Oriented Databases* (Singapore, December 1995), pp. 39–46.

KAMBER, M., HAN, J., AND CHIANG, J. 1997. Metarule-guided mining of multi-dimensional association rules using data cubes. In *Proc. of the International Conference on Knowledge Discovery and Data Mining* (California, USA, August 1997), pp. 207–210.

KLEMETTINEN, M., MANNILA, H., RONKAINEN, P., TOIVONEN, H., AND VERKAMO, A. 1994. Finding interesting rules from large sets of discovered association rules. In *Proc. of the 3rd Intl. Conf. on Information and Knowledge Management* (Gaithersburg, Maryland, November 1994), pp. 401–408.

LENT, B., SWAMI, A., AND WIDOM, J. 1997. Clustering association rules. In *Proc. of the Intl. Conf. on Data Engineering* (Birmingham, England, April 1997), pp. 220–231.

MANNILA, H. AND TOIVONEN, H. 1996. Discovering generalized episodes using minimal occurrences. In *Proc. of the second International Conference on Knowledge Discovery and Data Mining* (Portland, Oregon, August 1996), pp. 146–151.

MANNILA, H., TOIVONEN, H., AND VERKAMO, A. 1997. Discovering frequent episodes in event sequences. Technical Report Series of Publications C, Report C-1997-15 (February), Department of Computer Science, University of Helsinki.

MEO, R., PSAILA, G., AND CERI, S. 1996. A new SQL-like operator for mining association rules. In *Proc. of the 22nd Intl. Conf. on Very Large Data Bases* (Mumbai, India, September 1996), pp. 122–133.

MILLER, R. AND YANG, Y. 1997. Association rules over interval data. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (Tucson, Arizona, USA, June 1997), pp. 452–461.

NG, R., LAKSHMANAN, L., HAN, J., AND PANG, A. 1998. Exploratory mining and pruning optimizations of constrained association rules. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (Seattle, Washington, June 1998), pp. 13–24.

ÖZDEN, B., RAMASWAMY, A., AND SILBERSCHATZ, A. 1998. Cyclic association rules. In *Proc. of the Intl. Conf. on Data Engineering* (1998), pp. 412–421.

PARK, J.-S., CHEN, M.-S., AND YU, P. 1995. An effective hash based algorithm for mining association rules. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (San Jose, CA, May 1995), pp. 175–186.

PARK, J.-S., CHEN, M.-S., AND YU, P. 1996. Data mining for path traversal patterns in a web environment. In *Proc. of the 16th Conference on Distributed Computing Systems* (Hong Kong, May 1996), pp. 385–392.

RAMASWAMY, S., MAHAJAN, S., AND SILBERSCHATZ, A. 1998. On the discovery of interesting patterns in association rules. In *Proc. of the 24th Intl. Conf. on Very Large Data Bases* (New York, USA, August 1998), pp. 368–379.

SAVASERE, A., OMIECINSKI, E., AND NAVATHE, S. 1995. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21st Intl. Conf. on Very Large Data Bases* (Zurich, Switzerland, September 1995), pp. 432–443.

SILVERSTEIN, C., BRIN, S., MOTWANI, R., AND ULLMAN, J. 1998. Scalable techniques for mining causal structures. In *Proc. of the 24th Intl. Conf. on Very Large Data Bases* (New York, USA, August 1998), pp. 594–605.

SRIKANT, R. AND AGRAWAL, R. 1995. Mining generalized association rules. In *Proc. of the 21st Intl. Conf. on Very Large Data Bases* (Zurich, Switzerland, September 1995), pp. 409–419.

SRIKANT, R. AND AGRAWAL, R. 1996. Mining quantitative association rules in large relational tables. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (Montreal, Canada, June 1996), pp. 1–12.

SRIKANT, R., VU, Q., AND AGRAWAL, R. 1997. Mining association rules with item constraints. In *Proc. of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining* (Newport Beach, California, August 1997), pp. 67–73.

TOIVONEN, H. 1996. Sampling large databases for association rules. In *Proc. of the 22nd Conference on Very Large Data Bases* (Mumbai, India, September 1996), pp. 134–145.

TSUR, D., ULLMAN, J., ABITBOUL, S., CLIFTON, C., MOTWANI, R., AND NESTOROV, S. 1998. Query flocks: a generalization of association-rule mining. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data* (Seattle, Washington, June 1998), pp. 1–12.

ZAKI, M., PARTHASARATHY, S., OGIHARA, M., AND LI, W. 1997. New algorithms for fast discovery of association rules. In *Proc. of the 3rd International Conference on Knowledge Discovery and Data Mining* (Newport Beach, CA, USA, August 1997), pp. 283–286.

Received June 1998; revised August 1998, March 1999, and July 1999; revised May 2000