Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 05-043

Similarity-based Time-Profiled Association Mining: A Summary of Results

Jin Soung Yoo and Shashi Shekhar

December 30, 2005

Similarity-based Time-Profiled Association Mining: A Summary of Results

Jin Soung Yoo and Shashi Shekhar
Department of Computer Science and Engineering
University of Minnesota
200 Union ST SE 4-192
Minneapolis, MN 55414
jyoo,shekhar@cs.umn.edu

ABSTRACT

Given a time-stamped transaction database, time-profiled associations represent those subsets of items whose support time sequence satisfies a given specification. Considering the specification to be the similarity with the El Niño index sequence, the decreased fish-catch in Peru is an example of time-profiled associations. Classical association mining uses simple numeric interest measures (e.g., support) and a simple subset specification such that support is greater than a certain threshold. In contrast, time-profiled association mining uses a composite interest measure, i.e., a sequence of supports over a sequence of time slots and can use a richer subset specification, such as time series correlation and Euclidean distance. Mining time-profiled associations is computationally challenging due to the large size of datasets, composite interest measures, and richer subset specification. In this paper, we propose a monotonic lower bounding of $\mathcal{L}_p$ norm based similarity measure, and propose two novel algorithms to mine time-profiled associations. We show that the proposed algorithms are correct and complete in finding time-profiled associations. Experimental results show that the proposed algorithms outperform the naive approach.

1. INTRODUCTION

A *time-profiled association*[12] is an association pattern whose prevalence time sequence satisfies given a specification. Association patterns discovered by the traditional association rule mining[1] reveal the generic dependency among items of transactions, e.g., the sales of diapers and beer. However, transaction data are implicitly associated with time and the association patterns found might have different popularity levels over time. The time-profiled association problem is different from the traditional association mining in the following ways. First, traditional association mining uses simple numeric interest measures such as support and confidence but time-profiled association mining uses a composite interest measure such as a sequence of prevalence measures (e.g., support) at all time slots from a time-stamped transaction database. The composite interest measure can be used to capture the temporal variation of associations. Second, traditional association analysis uses a simple subset specification such that support is greater than a given threshold. In contrast, the time-profiled association mining can use a richer subset specification. For example, a subset specification may be defined using a query time sequence, a time sequence similarity function and a similarity threshold for global matching over times not each time slot.

Applications: Some applications for time-profiled associations include the following:

- Climate data analysis : Considering the similarity with El Niño index sequence as the specification, one example is the co-occurrences of climate events with the El Niño phenomenon, an abnormal warming in the eastern tropical Pacific Ocean[8] over the last 50 years [13]. Earth scientists are interested in the behavior of climates in a well-defined region of the land which shows a strong connection with the El Niño index. The behavior can be described by a time series which captures the time dependent assocation of some variables, e.g., temperature and precipitation. El Niño also has been linked to climate phenomena such as droughts and wild fires in Australia, decreasing fish-catch in the coastal regions of Peru, and heavy rainfall along the eastern coast of South America in the past 50 years [10].
- Sales data analysis : In market-basket dataset, El Niño might bring up a time-profiled association representing reduced sales of snow-board or other winter recreation items in the upper Midwest of the United States, where the El Niño may lead to milder winters with less snow. For other examples, given the stock prices of a company over times, there may be products whose sale behaviors are similar to the change of the stock. It may also be of interest to find products showing similar selling patterns with a product last year or in a different geographic region (e.g., WAL Mart at different area)
- Ecology data analysis : Consider the time sequence of fruit amount frequency per month. Ecologists are of interest to find animal behaviors showing similar frequent patterns with the food variation.

These applications require to find itemsets showing approximate rather than exact matching to a query sequence.

This work was partially supported by NSF grant 0431141 and DOE, Oak Ridge National Laboratory(ORNL). The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. We thank a previous spatial database group member, Pusheng Zhang, for his initial work and valuable feedback.
.

Related Work: To our knowledge, there is no prior work directly tackling the problem of mining time-profiled associations. Closest efforts have attempted to capture special temporal profiles of association patterns. Özden et al.[9] identified cyclic association rules, which discover frequent periodically repetitive patterns. Li et al.[6] discovered frequent calendar schemas, e.g., year, month, and day, based on repeated patterns satisfying given support and confidence thresholds. These methods might not be directly applicable to time-profiled association mining using a composite interest measure and similarity based subset specification which is different with minimum threshold based specification. Otherwise, Agrawal et al.[3] addresses the problem of monitoring the support and confidence of association rules. First all rules satisfying a minium threshold from different time periods are mined and collected into a rule base. Then some rules can be queried by specifying shape operators(e.g., ups and downs) in support or confidence over time. This method might applicable to time-profile association mining. However we use similarity based richer subset specification and generate much smaller subsets since our method uses one phase approach to combine the generation of candidate itemsets and similarity search. Other works [7, 5] present mining methods in change. Liu et al.[7] studies changes of each association rule from one time period to another using support and confidence. Ganti et al.[5] presents a framework for measuring difference in two sets of association rules from two datasets. Theses are different with our work which uses a specific query sequence and global approximate matching.

A naive approach to mining time-profiled associations can be characterized using a two-phase paradigm. The first phase generates the history of support for all possible itemsets at different time points. The second phase matches the sequences of supports with a query sequence to find time-profiled associations . However, exponentially increasing computational costs of generating the prevalence time sequences of all combinatorial candidate itemsets become prohibitively expensive.

Contributions: In our previous work [12], we took the first steps to introduce the problem of mining time-profiled associations and ideas towards designing an algorithm to identify time-profiled association patterns. In this paper, we emphasize the following contributions.

- We provide a formal problem definition of time-profiled association mining.
- We propose a *lower bounding distance* of $\mathcal{L}_p$ norm ($p = 1, 2, \ldots, \infty$) based similarity function and formally prove the monotonically non-decreasing property of the lower bounding distance. The monotonicity property makes it possible to effectively prune the itemset search space and efficiently find similar associate itemsets.
- We propose two time-profiled association mining algorithms with differing database scan methods: a lattice-dominant time-profiled association mining method(L-TPMINE) and a snapshot-dominant time-profiled association mining method(S-TPMINE).
- We analytically show that the proposed algorithms are complete and correct, i.e., there are no false droppings or false admissions for time-profiled association mining algorithms. We experimentally evaluate the proposed algorithms. The experimental results show that the proposed algorithms outperform the naive approach and scale with the length of time snapshots and number of items.

Scope and Outline: The remainder of the paper is organized as follows. Section 2 formally defines the problem of mining time-profiled association patterns and discusses modeling time-profiled associations. Algorithmic design concepts and efficient one-phase algorithms for mining time-profile associations are proposed in Section 3. The proofs of correctness and completeness of the algorithms are given in Section 4. Section 5 presents the experimental evaluation. We summarize our work and discuss future directions in Section 6.

2. MODELING OF TIME-PROFILED ASSOCIATIONS

In this section, we describe the problem definition of time-profiled association mining and discuss modeling time-profiled associations.

2.1 Problem Statement

The problem of mining time-profiled association patterns is to find all subsets of items which satisfy a given subset specification. The detailed definition is described as follows.

Given:
1) A time framework TF which is divided into a set of disjoint time slots, $TF = t_0 \cup \ldots \cup t_{n-1}$.
2) A time-stamped transaction database TD. Each transaction $d \in TD$ is a tuple $<$ *time-stamp*, *itemset* $>$, where *time-stamp* is a time $\in TF$ that transaction d is executed and *itemset* is a set of items $\subseteq E = \{e_1, \ldots, e_m\}$, and $TD = TD_0 \cup \ldots \cup TD_{n-1}$, where TD_i is a set of transactions belong to a time slot t_i.
3) A subset specification
  3a) A query sequence $\vec{Q} =< q_0, \ldots, q_{n-1} >$
  3b) A time-series similarity function:
    $f_{similarity} : (\vec{S}_1, \vec{S}_2) \rightarrow R$, where $\vec{S}_1$ and $\vec{S}_2$ are time sequences and R is a numeric value.
  3c) A similarity threshold θ.
Find:
A complete and correct set of itemsets $I \subseteq E$ which satisfy a given subset specification, i.e., $f_{similarity}(\vec{S_I}, \vec{Q}) \leq \theta$, where $\vec{S_I} =< s_0, \ldots, s_{n-1} >$ is the time sequence of prevalence measures of an itemset I over time slots $t_0, \ldots, t_{n-1}$.
Objective:
Minimize computational cost.

Figure 1 shows an example of time-profiled association mining with a simple time-stamped transaction database having two time slots t_0 and t_1. As in Figure 1 (a), the input are 1) a time frame $TF = \{t_0, t_1\}$, 2) a time-stamped transaction database and 3) a subset specification in which the query sequence is <0.4, 0.6> , the similarity function is the Euclidean distance and the similarity threshold is 0.2. When the prevalence time sequences of itemsets are generated using their supports as in Figure 1 (b), the output of time-profiled association mining is B, AC and BC since the Euclidean distances between their prevalence time sequences and the query time sequence are less than or equal to 0.2.

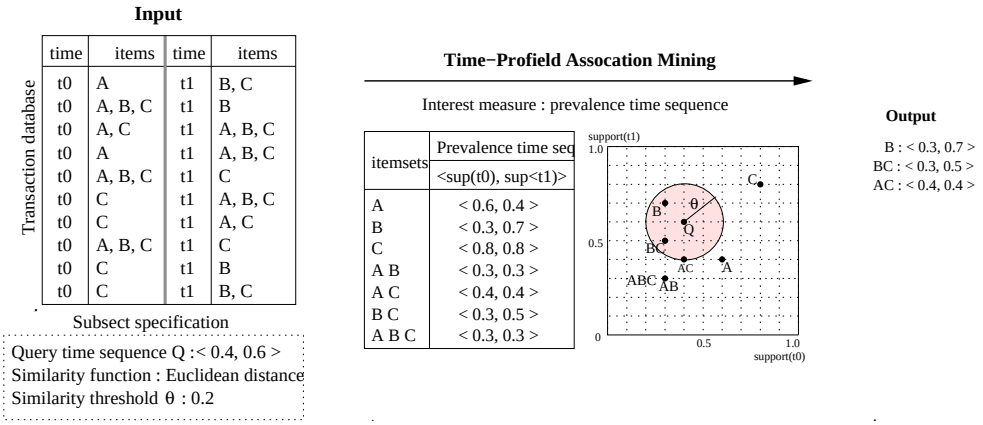


Figure 1: An example of time-profiled association mining (a) Input dataset (b) Generate prevalence time sequences and search similar itemsets (c) Output itemsets

2.2 Interest Measure

Prevalence measures such as support have been successfully used in traditional association rule mining since they represent the statistical significance of a pattern. However, rather than applying a mining algorithm to the whole data, the data is first partitioned according to time periods. The granularity of the time period is application dependent. We propose to adopt a time sequence of prevalence measures as an interest measure for time-profiled association mining.

DEFINITION 1. *Given a transaction database* $TD = TD_0 \cup \ldots \cup TD_{n-1}$ *where* TD_i *is a set of transactions executed in time slot* t_i*, the prevalence time sequence of an itemset* I*,* $\vec{S}_I =< s_0^I, \ldots, s_{n-1}^I >$ *is the sequence of prevalence values of the itemset* I *over time slots* $t_0, \ldots, t_{n-1}$*. If the prevalence measure is support, we call it the* **support time sequence** *of* I *such that*

$$\vec{S}_I =< support_{DT_0}(I), \ldots, support_{DT_{n-1}}(I) >,$$

where $support_{TD_i}(I)$*,* $0 \leq i < n-1$*, is the* **support** *of an itemset* I *at time slot* t_i *which is the fraction of transactions* d *that contain the itemset* I *in* TD_i *such that* $support_{TD_i}(I) = \frac{|\{d \in TD_i | I \subset d\}|}{|TD_i|}$.

LEMMA 1. *The supports of the support time sequence of an itemset are monotonically non-increasing with the size of itemset at each time slot.*

2.3 Subset Specification

The subset specification is a set of conditions that itemsets have to satisfy. For time-profiled association pattern mining, the subset specification is composed of three components: a query time sequence, a time series similarity function and a similarity threshold. First, the query time sequence is a sequence of values over time slots $t_0, \ldots, t_{n-1}$. We assume that the query sequence value is in the same scale as the prevalence measure of itemsets or can be transformed to the same scale. Second, we propose using $\mathcal{L}_p$ norms ($p = 1, 2, \ldots, \infty$) between the query time sequence and our interest measure, i.e., support time sequence, for similarity function. Many measures of time-series similarity have been discussed in literature [4]. Among them, $\mathcal{L}_p$ norm is the most popular class of similarity measure.

DEFINITION 2. *For two time sequences* $\vec{S} =< s_0, \ldots, s_{n-1} >$ *and* $\vec{Q} =< q_0, \ldots, q_{n-1} >$*, the* $\mathcal{L}_p$ *norm between* $\vec{S}$ *and* $\vec{Q}$ *is defined as*

$$\mathcal{L}_p(\vec{S}, \vec{Q}) = (\sum_{i=0}^{n-1} |s_i - q_i|^p)^{\frac{1}{p}}, \text{ where } p = 1, 2, \ldots, \infty$$

It is called the *city-block* or the *Manhattan* norm when p=1, and the *Euclidean* norm(*Euclidean distance*) when p=2.

$$\mathcal{L}_2(\vec{S}, \vec{Q}) = (\sum_{i=0}^{n-1} |s_i - q_i|^2)^{\frac{1}{2}}$$

It is known that Euclidean distance is optimal(in the Maximum Likelihood sense) when measurement differences are additive, *i.i.d.* (independent, identically distributed) Gaussian [11]. It has been the most popular similarity measure in similar time sequence matching [11, 2, 4]. Linear correlation coefficient can be transformed to Euclidean distance [11]. We also propose *Average Euclidean distance* defined as

$$Avg\,\mathcal{L}_2(\vec{S}, \vec{Q}) = (1/n)^{\frac{1}{2}} * \mathcal{L}_2(\vec{S}, \vec{Q})$$

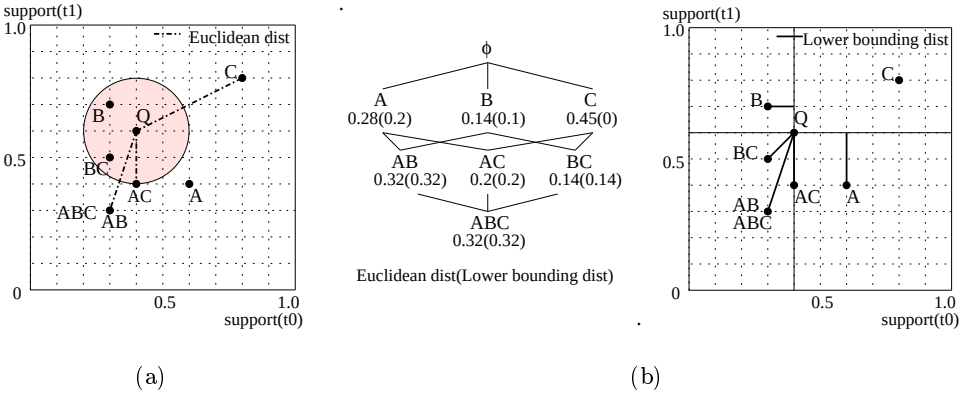
The intuition behind average Euclidean distance is that it represents an average distance between a support time sequence and a query sequence. The average Euclidean distance of a support time sequence $\vec{S}$ can be thought of as average support deviation to a query sequence $\vec{Q}$.

$$\sigma(\vec{S}) = (1/n)^{\frac{1}{2}} * \mathcal{L}_2(\vec{S}, \vec{Q}) = (\frac{\sum_{i=0}^{n-1} |s_i - q_i|^2}{n})^{\frac{1}{2}}$$

In the extreme case when $p = \infty$, it is called the *maximum* norm and can be reformulated as follows:

$$D_\infty(\vec{S}, \vec{Q}) = \max_{i=0}^{n-1} |s_i - q_i|$$

In the rest of paper, we denote $\mathcal{L}_p(\vec{S}, \vec{Q})$ norms or $Avg\,\mathcal{L}_2(\vec{S}, \vec{Q})$ to $D(\vec{S}, \vec{Q})$. We focus on Euclidean distance($\mathcal{L}_2$ norm) in



(a) (b)

Figure 2: (a) Non-monotonicity of the Euclidean distance with itemset size (b) Monotonically non-decreasing property of the lower bounding distance with itemset size

this paper. Other $\mathcal{L}_p$ and average Euclidean distance can be used without any loss.

The last component of the subset specification is a similarity threshold. If $D(\vec{S}, \vec{Q})$ is below a given similarity threshold, we say that the two sequences $\vec{S}$ and $\vec{Q}$ are similar.

3. ALGORITHM FOR TIME-PROFILED ASSOCIATION MINING

In this section, we discuss algorithmic design concepts for mining time-profiled association patterns and describe our time-profiled association mining algorithms.

3.1 Design Concepts

3.1.1 Upper Bound Support Time Sequence

The upper bound support time sequence can be used for approximately estimating the true support time sequence without scanning the time-stamped database. We define the upper bound of the support time sequence of an itemset using the support time sequences of its subsets.

DEFINITION 3. *Let I_{k+1} be a size $k+1$ itemset $\subseteq E$ and $B = \{I_k^1, \ldots, I_k^{k+1}\}$ be a set of all size k subsets of I_{k+1}, where $I_k^j \subset I_{k+1}$, $1 \leq j \leq k+1$. Let $\vec{S}_{I_k}^j = < s_0^{I_k^j}, \ldots, s_{n-1}^{I_k^j} >$ be the support time sequence of $I_{kj} \in B$. The* **upper bound support time sequence** *of I_{k+1}, $\vec{U}_{I_{k+1}}$ is $< min\{s_0^{I_k^1}, \ldots, s_0^{I_k^{k+1}}\}, \ldots, min\{s_{n-1}^{I_k^1}, \ldots, s_{n-1}^{I_k^{k+1}}\} >$.*

When $\vec{S}_A$ is <0.6, 0.4> and $\vec{S}_B$ is <0.3, 0.7>, the upper bound support time sequence $\vec{U}_{AB}$ of AB is <min(0.6, 0.3), min(0.4, 0.7)>=<0.3, 0.4>. In the rest of paper, we use the upper bound support (time) sequence and the upper bound of the support (time) sequence interchangeably.

3.1.2 Lower Bounding Distance

The lower bounding distance is defined as $\mathcal{L}_p$ norm between a support time sequence and a query time sequence at time slots where the element support is less than the element query value. It can be defined as follows:

DEFINITION 4. *For a (an upper bound) support time sequence $\vec{S} = < s_0, \ldots, s_{n-1} >$, a query time sequence $\vec{Q} = < q_0, \ldots, q_{n-1} >$ and $D(\vec{S}, \vec{Q})$, a $\mathcal{L}_p$ norm similarity function between $\vec{S}$ and $\vec{Q}$, the* **lower bounding distance** *between $\vec{S}$ and $\vec{Q}$ is defined as*

$$D_{lb}(\vec{S}, \vec{Q}) = D(\vec{S}, \vec{Q}) \quad where\ s < q$$

For instance, when Euclidean distance is the similarity function, the lower bounding distance is

$$D_{lb}(\vec{S}, \vec{Q}) = (\sum_{i=0}^{n-1} f(s_i, q_i))^{\frac{1}{2}},\ where\ f(s,q) = \begin{cases} 0, & if\ s \geq q \\ (q-s)^2, & if\ s < q \end{cases}$$

3.1.3 Monotonicity Property of the Lower Bounding Distance

OBSERVATION 1. *The $\mathcal{L}_p$ norms between the support time sequence of an itemset and a query sequence do not show any monotonicity with the size of the itemset.*

Figure 2 (a) shows the Euclidean distances between the support time sequences $\vec{S}_C$, $\vec{S}_{AC}$ and $\vec{S}_{ABC}$, and a query sequence $\vec{Q}$. $D(\vec{S}_C, \vec{Q})$=0.45, $D(\vec{S}_{AC}, \vec{Q})$=0.2 and $D(\vec{S}_{ABC}, \vec{Q})$ =0.32. We can notice that $D(\vec{S}_C, \vec{Q}) > D(\vec{S}_{AC}, \vec{Q})$ but $D(\vec{S}_{AC}, \vec{Q}) < D(\vec{S}_{ABC}, \vec{Q})$. In other $\mathcal{L}_p$ norms, it is difficult to find any monotonicity.

LEMMA 2. *The lower bounding distance between the (upper bound) support time sequence of an itemset and a query time sequence is* **monotonically non-decreasing** *with the size of the itemset.*

Proof. We prove using Euclidean distance. Other $\mathcal{L}_p$ norms can be proved similarly. First, we prove the monotonicity of lower bounding distance to the support time sequences. According to Definition 4, the lower bounding distance between $\vec{S}_I =< s_0^I, \ldots, s_{n-1}^I >$ for a size k itemset $I = \{i_1, \ldots, i_k\}$ and $\vec{Q} =< q_0, \ldots, q_{n-1} >$ is $D_{lb}(\vec{S}_I, \vec{Q}) = (\sum_{i=0, s_i^I < q_i}^{n-1} (s_i^I - q_i)^2)^{\frac{1}{2}}$. For a size $k+1$ itemset $I' = I \cup \{i'\}$, where $i' \notin I$ and its support time sequence $\vec{S}_{I'} =< s_0^{I'}, \ldots, s_{n-1}^{I'} >$, we need to prove that $D_{lb}(\vec{S}_I, \vec{Q}) \leq D_{lb}(\vec{S}_{I'}, \vec{Q})$. According to Lemma 1, the support is non-increasing with the size of itemset. Thus the support of I' is equal to or less than the

support of I at all time slots such that $s_0^I \geq s_0^{I'}, \ldots, s_{n-1}^I \geq s_{n-1}^{I'}$, and $q_i - s_i^I \leq q_i - s_i^{I'}$ where $s_i^I < q_i$ and $s_i^{I'} < q_i$, $0 \leq i < n$. We can get $(\sum_{i=0, s_i^I < q_i}^{n-1}(s_i^I - q_i)^2)^{\frac{1}{2}} \leq (\sum_{i=0, s_i^{I'} < q_i}^{n-1}(s_i^{I'} - q_i)^2)^{\frac{1}{2}}$, i.e., $D_{lb}(\vec{S}_I, \vec{Q}) \leq D_{lb}(\vec{S}_{I'}, \vec{Q})$. Second, the monotonicity of lower bounding distance to the upper bound support time sequnece can be proved similarly.

In Figure 2 (b), we can notice the monotonicity property of the lower bounding distances of the support time sequences. For example, $D_{lb}(\vec{S}_A, \vec{Q})$=0.2, $D_{lb}(\vec{S}_{AB}, \vec{Q})$=0.32 and $D_{lb}(\vec{S}_{ABC}, \vec{Q})$=0.32. Thus $D_{lb}(\vec{S}_A, \vec{Q}) \leq D_{lb}(\vec{S}_{AB}, \vec{Q}) \leq D_{lb}(\vec{S}_{ABC}, \vec{Q})$. Lemma 2 ensures that the lower bounding distance can be used to effectively prune the itemset search space and efficiently find similar itemsets.

3.1.4 Database Scan Method

The method to generate support time sequences can be different in scanning the time-stamped transaction database. We propose two database scan methods: a lattice-dominant scan method and a snapshot-dominant scan method. The lattice-dominant scan method scans a whole transaction database from time t_0 to time t_{n-1} at each level of lattice itemsets and generate the support time sequences of itemsets over all time slots. Otherwise, the snapshot-dominant scan method repeats the scanning of transactions at a time slot, e.g, from the first time slot, with counting the supports of itemsets until it finds all candidate itemsets of different sizes. It then moves to the next time slot and repeats the process. This method incrementaly generates support time sequences with the processed time slots.

3.2 Time-Profiled Association Mining Algorithms

A naive method for finding time-profiled association patterns follows a two-step procedure. First, it counts the supports of all possible itemsets over all time slots and generates their support time sequences. Second, it searches support time sequences similar to a query sequence. In this step, advanced time series search methods using indexing can be used [4]. However, as the number of items and transactions increases and the time points increase, the computation cost of all combinations of itemsets becomes prohibitively expensive. We propose a one-step approach to combine the generation of support time sequences and the sequence search. We reduce the itemset search space using a three-step pruning. First is the pruning of the subset check of candidate itemsets and second is the pruning of candidate itemsets using the lower bounding distances to their upper bound support sequences. This allows eliminating a candidate itemset without scanning the database or even calculating the similarity measure. The final step is the pruning of candidate itemsets using the lower bounding distance to their support time sequences. This allows to reduce the number of the next size candidate itemsets generated. We propose two *Apriori*-like algorithms different in the database scan method for the time-profiled association mining method(TPMINE): a lattice-dominant method(L-TPMINE) and a snapshot-dominant method(S-TPMINE).

3.2.1 Lattice-dominant TPMINE Algorithm

The lattice-dominant TPMINE(L-TPMINE) algorithm uses the lattice-dominant database scan method to generate the support time sequences. Algorithm 1 shows the pseudocode.

Inputs:
E : Set of single items.
TD: A time-stamped transaction database
$\vec{Q}$: A query sequence
D : A similarity function
θ : A similarity threshold
Output: All itemsets whose support sequences are similar to $\vec{Q}$ under D and θ
Variables :
k : Itemset size
C_k : Set of size k candidate itemsets
$\vec{U}_k$: Set of upper bound sequences of size k itemsets
$\vec{S}_k$: Set of support sequences of size k itemsets
L_k : Set of size k itemsets whose lower bounding distance $\leq \theta$
R_k :Set of size k itemsets whose similarity measure $\leq \theta$
Main:
1) $k = 1$; $C_1 = E$;
2) $\vec{S}_1$= generate_support_sequences(C_1);
3) L_1= prune_candidate_itemsets(C_1, $\vec{S}_1$);
4) R_1= find_similar_itemsets(L_1, $\vec{S}_1$);
5) **while** (not empty L_k) **do**
6) $(C_{k+1}, \vec{U}_{k+1})$=generate_candidate_itemsets(L_k, $\vec{S}_k$);
7) C_{k+1}=prune_candidate_itemsets(C_{k+1}, $\vec{U}_{k+1}$);
8) $\vec{S}_{k+1}$=generate_support_sequences(C_{k+1});
9) L_{k+1}=prune_candidate_itemsets(C_{k+1},$\vec{S}_{k+1}$);
10) R_{k+1}=find_similar_itemsets(L_{k+1}, $\vec{S}_{k+1}$);
11) $k = k + 1$;
12) **end**
13) **return** $\bigcup(R_1, \ldots, R_{k+1})$;

Algorithm 1: The L-TPMINE algorithm

Generate the support time sequences of single items and find similar items (Steps 1 - 4): All singletons($k = 1$) become candidate items(C_1). In the first scan of a whole time-stamped database, the supports of singletons are counted per each time slot and their support time sequences($\vec{S}_1$) are generated. If the lower bounding distance between the support time sequence and a query sequence is greater than a given similarity threshold, the singleton is pruned from the candidate set. The others(L_1) are kept for generating the next size candidate itemsets. If true similarity distance between the support time sequence and the query sequence satisfies the threshold, the singleton is added to a result set(R_1). Figure 3 shows the L-TPMINE algorithm trace using the dataset in Figure 1 (a). Euclidean distance is used as similarity function. In this example, only item B is included in the result set since the Euclidean distance between $\vec{S}_B$ and $\vec{Q}$, 0.14 is less than 0.2. Notice that the Euclidean distances of $\vec{S}_A$ and $\vec{S}_C$ to $\vec{Q}$ do not satisfy the threshold value but since their lower bounding distances do satisfy it, they are kept for generating the next size candidate itemsets.

Generate candidate itemsets and the upper bound support sequences (Step 6) : All size $k+1$ ($k \geq 1$) candidate itemsets(C_{k+1}) are generated using size k itemsets(L_k) whose lower bounding distances satisfy the threshold. If a size k subset of a generated itemset is not in the L_k, the candidate itemset is eliminated. The upper bound support sequences of the candidate itemsets are generated using the support time sequences of their subsets on the fly.

Prune candidate itemsets using the lower bounding distances to upper bound support sequences(Step 7) : If the lower bounding distance to the upper bound support sequence of a candidate itemset is greater than the

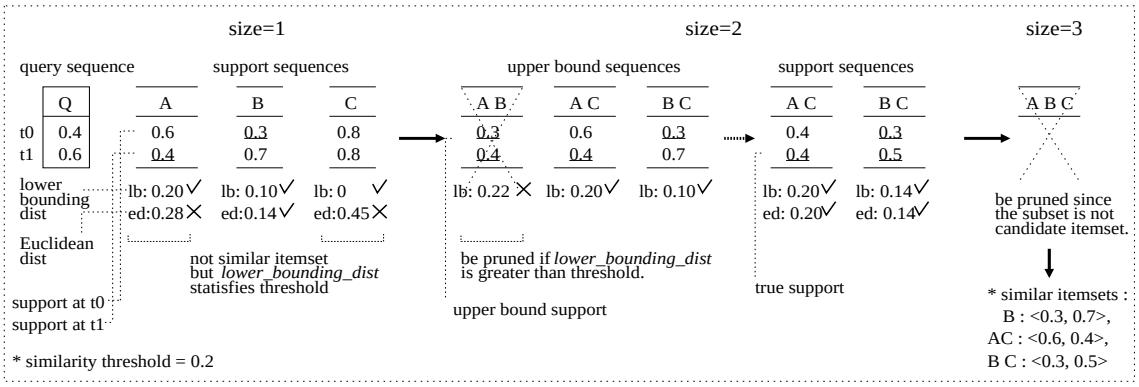


Figure 3: A example of a lattice-dominant TPMINE algorithm trace

threshold, the candidate itemset is eliminated from the set of candidate itemsets. For example, in Figure 3, the lower bounding distance of $\vec{U}_{AB}$ to $\vec{Q}$ is 0.22 which is greater than the threshold 0.2. Thus the itemset AB is removed from the set of candidate itemsets.

Scan database and generate the support time sequences(Step 8) : The supports of candidate itemsets are counted during the scan of database from time t_0 to time t_{n-1} and their support time sequences($\vec{S}_{k+1}$) are generated.

Prune candidate itemsets using the lower bounding distance to the support time sequences(Step 9) : Before finding the support time sequences similar to the query sequence, we do another pruning using the lower bounding distance of the support sequences for reducing the number of the next size candidate itemsets. This step uses the same *prune_candidate_itemsets* in step 7 except its parameter is the support sequences. For example, in Figure 3, the lower bounding distances of $\vec{S}_{AC}$ and $\vec{S}_{BC}$ satisfy the threshold 0.2, thus they are kept in L_2.

Find similar itemsets(Step 10) : The true similarity value between the support time sequence of an itemset and the query sequence are calculated. If the value satisfies the threshold, the itemset is included in the result set(R_{k+1}). Step 9 and 10 can be done on the fly. The size of the examined itemsets is increased to $k = k+1$. The above procedures are repeated until no itemset in L_{k+1} remains.

3.2.2 Snapshot-dominant TPMINE algorithm

The snapshot-dominant TPMINE(S-TPMINE) algorithm uses the snapshot-dominant scan method. It first repeats scanning transactions at time t_0 with increasing itemset size until all candidate itemsets at time t_0 are found. The pruning strategy is the same as those of L-TPMINE except S-TPMINE uses the accumulated distances up to the current time slot. If the accumulated lower bounding distance is greater than a given threshold, the candidate itemset is pruned at the current time slot. After finding all size candidate itemsets, we move to the next time slot and repeat the procedures until the last time slot. Finally itemsets whose accumulated true similarity value is less than the threshold are included in the result set. The detail of the S-TPMINE algorithm is omitted due to the length restriction of paper.

4. ANALYTICAL ANALYSIS

We analyze the two TPMINE algorithms in terms of correctness and completeness. Correctness means that the similarity distance between the support time sequences of all found itemsets and a query sequence is below a given similarity threshold. Completeness means that the algorithms find all itemsets which have similarity with the query sequence under the similarity threshold. We focus on the analysis of the L-TPMINE algorithm and then discuss the analysis of the S-TPMINE algorithm.

LEMMA 3. *For a $\mathcal{L}_p$ norm, $D(\vec{S},\vec{Q})$ and the lower bounding distance $D_{lb}(\vec{S},\vec{Q})$ between two sequences $\vec{S}$ and $\vec{Q}$, the following inequality holds:*

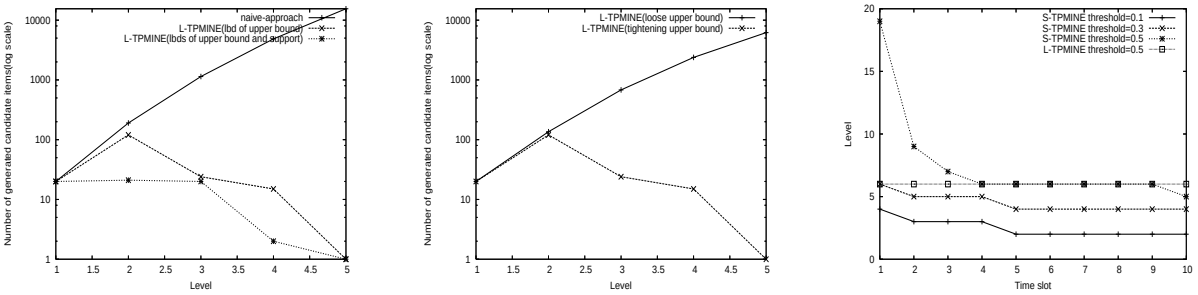
$$D_{lb}(\vec{S},\vec{Q}) \leq D(\vec{S},\vec{Q})$$

LEMMA 4. *Let $\vec{U}_{I_{k+1}} = < u_0^{I_{k+1}}, \ldots, u_{n-1}^{I_{k+1}} >$ be the upper bound support time sequences of I_{k+1} and $\vec{S}_{I_{k+1}} = < s_0^{I_{k+1}}, \ldots, s_{n-1}^{I_{k+1}} >$ be the support time sequence of an itemset I_{k+1}, $u_0^{I_{k+1}} \geq s_0^{I_{k+1}}, \ldots, u_{n-1}^{I_{k+1}} \geq s_{n-1}^{I_{k+1}}$.*

THEOREM 1. *The L-TPMINE algorithm is complete.*

Proof. First, we will show that in step 6 of Algorithm 1, the *generate_candidate_itemsets* function using itemsets whose lower bounding distance satisfies the threshold doesn't miss a true itemset. Let I_k be a subset of a size $k+1$ candidate itemset I_{k+1} and its upper bounding distance between the support time sequence of I_k, P_k and $\vec{Q}$, $D_{lb}(\vec{S}_{I_k},\vec{Q}) > \theta$. According to Lemma 3 and Lemma 2, $D_{lb}(\vec{S}_{I_k},\vec{Q}) \leq D_{lb}(\vec{S}_{I_{k+1}},\vec{Q}) \leq D_(\vec{S}_{I_{k+1}},\vec{Q})$. Thus if $D_{lb}(\vec{S}_{I_k},\vec{Q}) > \theta$, $D(\vec{S}_{I_{k+1}},\vec{Q}) > \theta$ and I_{k+1} is not similar itemset. Next, we will show that in step 3, 7 and 9 of Algorithm 1, the *prune_candidate_itemsets* function using lower bounding distance of the (upper bound) support sequences does not remove a true itemset. First, according to Lemma 4 and Definition 4, $D_{lb}(\vec{U},\vec{Q}) \leq D_{lb}(\vec{S},\vec{Q})$. If $D_{lb}(\vec{U},\vec{Q}) > \theta$, $D_{lb}(\vec{S},\vec{Q}) > \theta$. Thus step 7 doesn't prune a true itemset. Second, according to Lemma 3, $D_{lb}(\vec{S},\vec{Q}) \leq D(\vec{S},\vec{Q})$. If $D_{lb}(\vec{S},\vec{Q}) > \theta$, $D(\vec{S},\vec{Q}) > \theta$. Thus steps 3 and 9 do not miss a true item.

THEOREM 2. *The L-TPMINE algorithm is correct.*



(a) (b) (c)

Figure 4: Algorithmic design decision (a) Lower bounding pruning (a) Upper bound support sequence (c) Database scan methods

Proof. The correctness can be guaranteed by steps 4 and 10 in Algorithm 1. The *fine_similar_itemsets* calculates exact similarity and includes itemsets whose similarity is not greater than the user-specified threshold into the result set.

THEOREM 3. *The S-TPMINE algorithm is complete and correct.*

Proof. First, S-TPMINE algorithm is correct since it includes itemsets whose accumulated similarity values are not greater than the threshold after processing the last time slot. Second, S-TPMINE is complete since it uses the same pruning strategies as those of L-TPMINE except it uses the accumulated lower bounding distance.

5. EXPERIMENTAL EVALUATION

We conducted a series of experiments to examine the following:

- The algorithmic design decision on upper bound support sequence, lower bounding distance and database scan method.
- The scalability of our algorithms in similarity threshold, number of time slots and number of items.

All experiments were performed on a workstation with 2 processors, each an Intel Xeon 2.8 GHz with 2 Gbytes of memory running the Linux operating system.

5.1 Experiment Datasets

Our experiments were performed on synthetic datasets. Synthetic datasets were generated using the transaction generator designed by the IBM Quest project used in [1]. We added a time slot parameter for generating time-stamped transactions. Table 1 shows the parameters used to generate the synthetic data set. In the table, D is the total number of transactions($\times$ 1,000) , N is the number of items, T is the average size of transactions and S is the number of time slots.

Query time sequences were randomly generated in the scale of the support value at each time slot or generated by choosing its median near several quantiles e.g., 0.1, 0.25, 0.5, 0.75, 0.9 of the sorted supports at each time slot after calculating the supports of single items. The Euclidean distance is used for similarity function in this experiment.

Data	D	N	T	S	Data	D	N	T	S
S1	100	20	10	10	S3-1	100	15	10	100
S2-1	500	20	10	50	S3-2	100	20	10	100
S2-2	1,000	20	10	100	S3-3	100	25	10	100
S2-3	2,000	20	10	200	S3-4	100	30	10	100
S2-4	3,000	20	10	300					

Table 1: Synthetic dataset characteristics

5.2 Experiment Results

We compared the performance of L-TPMINE, S-TPMINE and the naive two-phase approach. Since the naive approach generates huge candidate itemsets with increase of itemset size, we used small datasets, especially in the number of items, in comparisons with the naive approach.

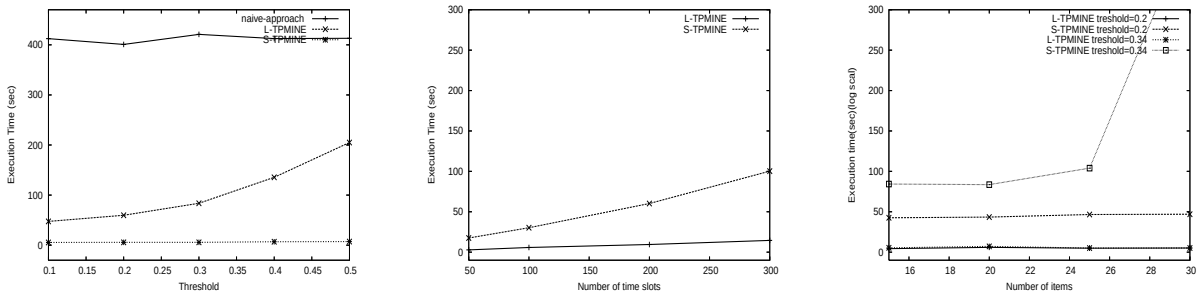
5.2.1 Evaluation of Algorithmic Design Decision

In a series of experiments for evaluation of algorithmic design decision, we used a synthetic dataset S1, a query sequence near the 0.5 quantile and 0.2 similarity threshold.

Effect of lower bounding pruning : Figure 4 (a) shows the number of generated candidate itemsets per each level. L-TPMINE generated dramatically fewer candidate itemsets compared with the naive approach with no pruning scheme. We can also notice the pruning by the lower bounding distances of both the upper bound sequence and the support time sequence is more effective in generating less candidate itemsets.

Effect of upper bound sequence design : In this experiment, we compared the algorithmic design of upper bound support time sequences. Our upper bound support of an itemset is generated using the minimum value of supports of its previous size subsets, e.g, $min(AB, AC, DB)$ for ABC. It showed more effective compared with another loose upper bound e.g., the minimum of supports of component singletons of itemsets, e.g., $min(A, B, C)$. Figure 4 (b) shows the L-TPMINE algorithm pruned many candidate itemsets when the upper bound was tightened.

Effect of database scanning method : We examined the effect of choice of database scan method. As shown in Figure 4 (c), S-TPMINE showed a similar lattice level



(a) (b) (c)

Figure 5: Scalability (a) Similarity thresholds (b) Number of time slots (c) Number of items

with L-TPMINE at thresholds 0.1 and 0.3. However, with the increase of the similarity threshold (threshold 0.5), S-TPMINE showed a dramatically large itemset search space in the beginning time slots since the accumulated lower bounding distance does not reach the threshold value. Otherwise, L-TPMINE showed a constant deployment of the itemset lattice independent with the time slot number.

5.2.2 Scalability Evaluation

Effect of similarity threshold : The effect of similarity threshold was examined using different threshold values with a synthetic dataset S1 and a query sequence near the 0.5 quantile. As can be seen in Figure 5 (a), the L-TPMINE and S-TPMINE algorithms showed much less execution time compared with the naive approach. L-TPMINE showed little effect with small increases in the similarity threshold in this dataset but the execution time of S-TPMINE increased with increase of threshold. This is the reason the pruning ability of S-TPMINE decreases in the beginning time slots and keeps many lattice subsets. Otherwise, the naive approach showed a stable execution time because it generates the support time sequences of all combination itemsets independent of the thresholds and then compare with a query sequence.

Effect of number of time slots : In this experiment, we examined the scalability of our algorithms with synthetic datasets, $S2-1$, $S2-2$, $S2-3$ and $S2-4$ having different numbers of time slots. Query sequences were chosen near the 0.5 quantile in each dataset and the threshold was the same value, 0.2 in each dataset. Figure 5 (b) shows that both algorithms increased with increases in the number of time slots. L-TPMINE was more scalable with number of time slots.

Effect of number of items : We examined the effect of number of items with synthetic datasets of different number of items, i.e., $S3-1$, $S3-2$, $S3-3$ and $S3-4$. In this experiment, we used two thresholds, 0.2 and 0.34 since the effect of number of items showed different results with different thresholds. Under the 0.2 threshold which prunes most itemsets before around level(pass) 4 in these datasets, L-TPMINE and S-TPMINE showed similar execution time and received little effect with the increase of itemset size. However, when the threshold value is increased to 0.34, S-TPMINE showed dramatically increased execution time. The reason is that S-TPMINE did not much prune in the beginning time slots under this threshold and generated dramatically increased number of itemsets at each level with increasing number of items. Figure 5 (c) shows the results.

6. CONCLUSIONS AND FUTURE WORK

We defined the problem of mining time-profiled association patterns and proposed novel algorithms to discover time-profiled associations. The proposed algorithms substantially reduced the search space by pruning candidate itemsets based on the monotonicity property of the lower bounding distances. We showed that the algorithms are correct and complete in finding time-profiled association patterns. Experimental results on synthetic datasets showed that our algorithms outperformed the naive approach. We plan experimental evaluation with real datasets.

In future work, we would like to generalize time-profiled association problem for *space-profiled associations*. The prevalence of associations over geospace can be also presented using a composite interest measure, e.g., a support sequence over spatial regions. A space-profiled association could characterize the variations of associations over geospace. An example of space-profiled associations is the interaction of sales of emergency preparedness products (e.g., canned food and survival kits) in eastern coastal regions of the United States and hurricane activities. We also plan to explore the extension of time-profiled associations to *space-time-profiled associations* to capture the spatio-temporal variations of associations in spatio-temporal databases.

7. REFERENCES

[1] R. Agarwal and R. Srikant. Fast algorithms for Mining association rules. In *Proc. of the 20th VLDB*, 1994.

[2] R. Agrawal, C. Faloutsos, and A.Swami. Efficient Similarity Search in Sequence Databases . In *Proc. Int. Conference on Foundations of Data Organization(FODO)*, 1993.

[3] R. Agrawal and G. Psaila. Active Data Mining. In *Proc. The First International Conference on Knowledge Discovery and Data Mining*, 1995.

[4] C. Faloutsos, M. Ranganathan, and Y.Manolopoulos. Fast subsequence matching in time-series database. In *Proc. ACM SIGMOD Conference*, 1993.

[5] V. Ganti, J. Gehrke, and R. Ramakrishnan. A framework for measuring changes in data characteristics. In *Proc. POPS*, 1999.

[6] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering Calendar-Based Temporal Assocation Rules. In *Proc. Int. Symposium Temporal Representation and Reasoning(TIME)*, 2001.

[7] B. Liu, W. Hsu, and Y. Ma. Discovering the set of fundamental rule change. In *Proc. KDD Conference*, 2001.

[8] NOAA. El Nino Page. http://www.elnino.noaa.gov/.

[9] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic Association Rules. In *Proc. of IEEE Int. Conference on Data Engineering*, 1998.

[10] G. H. Taylor. Impacts of el nino on southern oscillation on the pacific northwest. http://www.ocs.orst.edu/reports/enso_pnw.html.

[11] B. Yi and C. Faloutsos. Fast Time Sequence Indexing for Arbitrary $\mathcal{L}_p$ norms. In *Proc. VLDB Conference*, 2000.

[12] J. Yoo, P. Zhang, and S. Shekhar. Mining Time-Profiled Associations: An Extended Abstract. In *the Proc. of the Pacific-Asia Conference on Data Mining and Knowledge Discovery*, 2005.

[13] P. Zhang, M. Steinbach, V. Kumar, S. Shekhar, P. Tan, S. Klooster, and C. Potter. Discovery of Patterns of Earth Science Data Using Data Mining. In M. M. Kantardzic and J. Zurada, editors, *to appear in Next Generation of Data Mining Applications*. IEEE Press, 2004.