

COPYRIGHT NOTICE

© 2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

Similarity-Profiled Temporal Association Mining

Jin Soung Yoo, *Member, IEEE*, and Shashi Shekhar, *Fellow, IEEE*

Abstract—Given a time stamped transaction database and a user-defined reference sequence of interest over time, similarity-profiled temporal association mining discovers all associated item sets whose prevalence variations over time are similar to the reference sequence. The similar temporal association patterns can reveal interesting relationships of data items which co-occur with a particular event over time. Most works in temporal association mining have focused on capturing special temporal regulation patterns such as cyclic patterns and calendar scheme-based patterns. However, our model is flexible in representing interesting temporal patterns using a user-defined reference sequence. The dissimilarity degree of the sequence of support values of an item set to the reference sequence is used to capture how well its temporal prevalence variation matches the reference pattern. By exploiting interesting properties such as an envelope of support time sequence and a lower bounding distance for early pruning candidate item sets, we develop an algorithm for effectively mining similarity-profiled temporal association patterns. We prove the algorithm is correct and complete in the mining results and provide the computational analysis. Experimental results on real data as well as synthetic data show that the proposed algorithm is more efficient than a sequential method using a traditional support-pruning scheme.

Index Terms—Temporal data mining, temporal association pattern, support time sequence, similarity.

1 INTRODUCTION

GIVEN a time stamped transaction database and a user-defined reference sequence of interest over time, the goal of *similarity-profiled temporal association mining* is to discover all associated item sets whose prevalence variations over time are similar to the reference sequence under a threshold. Similarity-profiled temporal association mining can reveal interesting relationships of data items that co-occur with a particular event over time. For example, the fluctuation of consumer retail sales is closely tied to changes in weather and climate [1]. While bottled water and generator sales might not show any correlation on normal days, a sales association between the two items may develop with the increasing strength of a hurricane in a particular region [2]. Retail decision makers may be interested in such an association pattern to improve their decisions regarding how changes in weather affect consumer needs.

Recent advances in data collection and storage technology have made it possible to collect vast amounts of data every day in many areas of business and science. Common examples are recordings of sales of products, stock exchanges, web logs, climate measures, and so on. One major area of data mining from these data is association pattern analysis. Association rules discover interrelationships among various data items in transactional data. Following the work of Agarwal and Srikan [3], the discovery of association rules has been extensively studied in [4], [5], [6], and [7]. In particular, in [8], [9], [10], they have

paid attention to temporal information, which is implicitly related to transaction data, e.g., the time that a transaction is executed, and discovered association patterns that vary over time. However, most works in temporal association mining have focused on special temporal regulation patterns of associated item sets such as cyclic patterns [8] and calendar-based patterns [9]. For example, it may be found that beer and chips are sold together primarily in evening time on week days. The temporal regulation patterns can be explained with binary sequences where the 1's correspond to the time units in which the pattern is prevalent (i.e., its support value is greater than a minimum prevalence threshold), and the 0's correspond to the time units in which the pattern is not prevalent. For instance, when the unit of time is day, a repetitive temporal pattern on Monday is defined as 10000001000000... Fig. 1a illustrates an example of temporal regulation patterns. It shows the prevalence values of three item sets I_1 , I_2 , and I_3 over time, and the binary temporal sequences of I_1 and I_2 under a fixed prevalence threshold (e.g., support threshold = 0.5). We can notice that item sets I_1 and I_2 show the same temporal regulation pattern, 111000111000..., even if their actual prevalence strengths are quite different. In contrast, Fig. 1b illustrates an example of temporal similarity patterns. The prevalence values of item sets I_2 and I_3 show a very similar variation with a user guided reference sequence R . The reference sequence can represent the change of prevalence of an item of interest (e.g., a product sale in market basket data, a stock exchange in the stock market, a climate measure such as temperature or precipitation, and a scientific phenomenon), or a user guided prevalence sequence pattern showing special shapes (e.g., a seasonal, emerging, or diminishing pattern over time). Current methods for temporal regulation pattern mining cannot reveal these types of temporal patterns that are based on actual prevalence similarity.

Application examples. Our similarity-profiled temporal association patterns can give important insight into many

• J.S. Yoo is with the Department of Computer Science, Indiana University-Purdue University, 2101 E. Coliseum Blvd., Fort Wayne, IN 46805-1499. E-mail: yooj@ipfw.edu.

• S. Shekhar is with the Department of Computer Science, University of Minnesota, 4-192, EE/CS Bldg., 200 Union St. SE, Minneapolis, MN 55455. E-mail: shekhar@cs.umn.edu.

Manuscript received 25 Nov. 2005; revised 25 Sept. 2006; accepted 28 July 2008; published online 2 Sept. 2008.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0523-1105. Digital Object Identifier no. 10.1109/TKDE.2008.185.

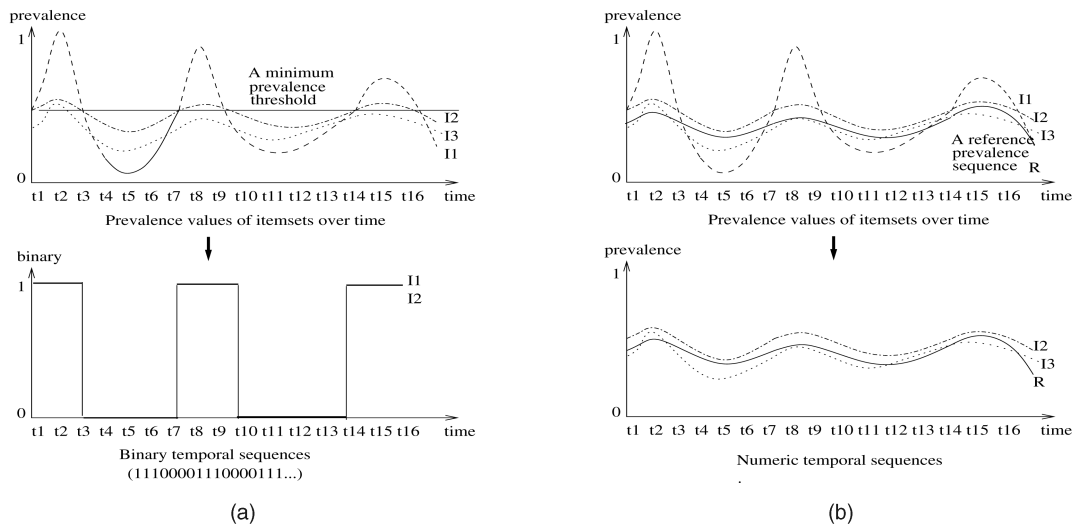


Fig. 1. A comparison of temporal association patterns. (a) Temporal regulation patterns. (b) Temporal similarity patterns.

application domains such as business, agriculture, earth science, ecology, and biology. They can also be used as filtered information for further analysis of market trends, prediction, and factors showing strong connections with a certain scientific event. For example, the weather-to-sales relationship is a very interesting problem in retail analysis. Managers in the merchandise and food sales division of Walt Disney World are hoping to find correlations between daily temperatures and sales [1]. Wal-Mart discovered a surprising customer buying pattern during hurricane season in one of its sales regions. Not only did survival kits (e.g., flashlights, generators, tarps) show similar selling patterns with bottled water, but also did the sales of Strawberry Pop-Tarts (a snack item) [2]. Our method can help in finding such item sets whose sales similarly change to that of a specific item (event) for a period of time. The mining results can improve supply chain planning and retail decision making for maximizing the visibility of items most likely to be in high demand during special time periods. We can find another example in the business domain from the online website Weather.com, which offers weather-related lifestyle information including travel, driving, home and garden, and sporting events, as well as weather information [11]. The website reports that almost 40 percent of weather.com visitors shop home improvement products when temperatures rise [11]. The website may attract more advertisers if it can analyze the relationships of visited websites through weather.com with changes of weather. Our temporal patterns may be used for finding such weather-related sponsor sites. In the scientific application domain, Earth scientists have been interested in the behavior of climates in a region that are often influenced by the El Niño phenomenon, an abnormal warming in the eastern tropical Pacific Ocean [12]. If we consider the El Niño related index values over the last 10 years, e.g., the Southern Oscillation Index (SOI) [12], as a reference sequence, one example of a similarity-profiled temporal association is a climate event pattern of low precipitation and low atmospheric carbon dioxide in Australia whose co-

occurrence over time is similar to the fluctuation of the El Niño index sequence.

In this paper, we formalize the similarity-profiled temporal association pattern. The problem of mining the temporal pattern is formulated with a *similarity-profiled subset specification*, which consists of a reference time sequence, a similarity function, and a dissimilarity threshold. The subset specification is used to define a user interest temporal pattern and guide the degree of approximate matching of prevalence values of associated item sets for it. We use a composite interest measure for the similar temporal patterns. The temporal prevalence variation of an associated item set is represented using support values of the item set over time. \mathcal{L}_p norm ($p = 1, 2, \dots, \infty$)-based similarity measures are used to capture how well the support time sequence of the item set matches the given reference sequence. Similarity-profiled temporal association mining presents challenges during computation. The straight-forward approach is to divide the mining process into two separate phrases. The first phrase computes the support values of all possible item sets at each time point and generates their support sequences. The second phrase compares the generated support time sequences with a given reference sequence and finds similar item sets. In this step, a multidimensional access method such as an R-tree family can be used for a fast sequence search [13]. However, the computational costs of first generating the support time sequences of all combinatorial candidate item sets and then doing the similarity search become prohibitively expensive with increase of items. Thus, it is crucial to devise schemes to reduce the item set search space effectively for efficient computation. We explore interesting properties for similarity-profiled association mining. First, to estimate support sequences without examining an input data set, we define tight *upper* and *lower bounds of true support sequences*. For early pruning of candidate item sets, we utilize the concept of a lower bounding distance, which is often used for indexing schemes in the time series literature and define the *lower bounding distance* with the upper and lower bounds of support sequences. Our *upper lower bounding distance* is

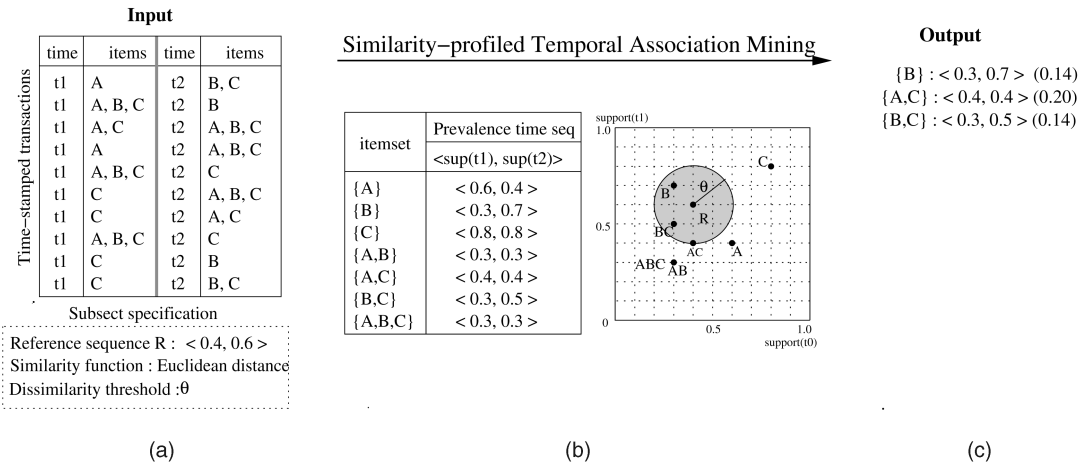


Fig. 2. An example of similarity-profiled temporal association mining: (a) input data, (b) generated prevalence (support) time sequences and sequence search, and (c) output item sets.

especially noteworthy because it is monotonically nondecreasing with the size of the item set. This property is used for further reducing the item set search space. A Similarity-Profiled temporal Association MINing mEthod (SPAMINE) algorithm is developed on our algorithmic design concept. For comparison, an alternative algorithm, a sequential method using a traditional support-pruning scheme, is also presented. We analytically show that the SPAMINE algorithm is *complete* and *correct*, i.e., there is no false dropping or false admission for similarity-profiled associated item sets. We also give a brief computational analysis. Finally, we experimentally evaluate the proposed algorithm using synthetic and real data. The experimental results show that the SPAMINE algorithm is more efficient and scalable than the sequential method.

The remainder of the paper is organized as follows: In Section 2, we formalize the problem of similarity-profiled temporal association mining. Our algorithmic design concepts are discussed in Section 3. Section 4 presents the proposed algorithm and also describes an alternative approach for comparison. Section 5 gives the proofs of correctness and completeness of the proposed algorithm and provides a computational analysis. The experimental results are presented in Section 6. Section 7 reviews the related work. In Section 8, we summarize our work and discuss future directions.

2 FRAMEWORK FOR SIMILARITY-PROFILED TEMPORAL ASSOCIATION MINING

In this section, we formulate the problem of mining similarity-profiled temporal association patterns. Fig. 2 shows a simple illustration of similarity-profiled temporal association mining.

2.1 Problem Statement

Given:

1. a finite set of items \mathcal{I} ;
2. an interest time period $\mathcal{T} = t_1 \cup \dots \cup t_n$, where t_i is a time slot by a time granularity, $t_i \cap t_j = \emptyset$, $i \neq j$;

3. a time-stamped transaction database $\mathcal{D} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_n$, $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$, $i \neq j$, wherein each transaction $d \in \mathcal{D}$ is a tuple $\langle \text{timestamp}, \text{itemset} \rangle$, where *timestamp* is a time $\in \mathcal{T}$ that the transaction is executed, and *itemset* $\subseteq \mathcal{I}$. \mathcal{D}_i is a set of transactions included in time slot t_i ; and
4. a subset specification:
 - 4a. a reference sequence $\vec{R} = \langle r_1, \dots, r_n \rangle$ over time slots t_1, \dots, t_n ,
 - 4b. a similarity function $f_{\text{Similarity}}(\vec{X}, \vec{Y}) \mapsto \mathbb{R}^n$, where \vec{X} and \vec{Y} are numeric sequences, and
 - 4c. a dissimilarity threshold θ .

Find. A set of item sets $I \subseteq \mathcal{I}$ that satisfy the given subset specification, i.e., $f_{\text{Similarity}}(\vec{S}_I, \vec{R}) \leq \theta$, where $\vec{S}_I = \langle s_1, \dots, s_n \rangle$ is the sequence of support values of an item set I over time slots t_1, \dots, t_n .

Objective. Find the complete and correct result set while reducing the computational cost.

2.2 Input Data

Items. We use the standard notion of *items* in traditional association rule mining [3]. Items can be supermarket items purchased by a customer during a shopping visit, product pages viewed in a Web session, climate events at a location, stocks bought and sold during an hour, etc. Items can be grouped to form an item set. An item set with k distinct items is referred to as a k *itemset*. The size of the item set space is $2^{|\mathcal{I}|} - 1$, where $|\mathcal{I}|$ is the number of distinct items.

Time period. We assume that we are interested in a fixed time period. A time period can be a particular year or any arbitrary period of time. We model time as discrete, and thus, a total time period can be viewed as a sequence of time slots by a certain time granularity [14]. For example, a one-year period can be divided into monthly unit time slots. We denote an i th time slot by t_i .

Transaction database. We assume that our database \mathcal{D} is a set of time-stamped transactions. Each transaction is a set of items over a finite item domain and has a time point when the transaction is executed. We call the time point associated with a transaction its *timestamp*. The transaction data set can be partitioned to disjoint groups of transactions by a time granularity. We denote a part of the transactions of \mathcal{D} executed in time slot t_i by \mathcal{D}_i . Fig. 2a

shows an example data set with three item types, A, B, and C. It is partitioned into two groups of transactions related to time slots t_1 and t_2 .

Subset specification. A subset specification is a set of conditions that item sets have to satisfy to become interesting patterns. Our subset specification consists of three components: a reference sequence, a similarity function, and a dissimilarity threshold. First, we assume that an arbitrary temporal pattern of interest can be defined as a reference sequence by a user. The reference sequence is a sequence of interesting values over time slots t_1, \dots, t_n . In the example in Fig. 2, $\langle 0.4, 0.6 \rangle$ is given as the reference sequence \vec{R} .

Second, we propose using a \mathcal{L}_p norm ($p = 1, 2, \dots, \infty$)-based similarity function. Many similarity measures have been discussed in the time series database literature [15]. A \mathcal{L}_p norm is the most popularly used distance measure in a similar time sequence search [16], [17], [18], [19] and can be used as basic building blocks for more complex similarity models as in [20].

Definition 1. For two time sequences $\vec{X} = \langle x_1, \dots, x_n \rangle$ and $\vec{Y} = \langle y_1, \dots, y_n \rangle$, the \mathcal{L}_p norm between \vec{X} and \vec{Y} is defined as $\mathcal{L}_p(\vec{X}, \vec{Y}) = (\sum_{t=1}^n |x_t - y_t|^p)^{\frac{1}{p}}$, where $p = 1, 2, \dots, \infty$.

When $p = 1$, the \mathcal{L}_1 norm is known as a *city block* or *Manhattan*. When $p = 2$, the \mathcal{L}_2 norm is called a *euclidean distance* and defined as $\mathcal{L}_2(\vec{X}, \vec{Y}) = (\sum_{t=1}^n |x_t - y_t|^2)^{\frac{1}{2}}$. It is known that the use of euclidean distance is optimal (in the Maximum Likelihood sense) when measurement differences are independent, identically distributed Gaussian [16]. In the extreme case when $p = \infty$, the \mathcal{L}_p norm is called the *maximum norm* and can be reformulated as $\mathcal{L}_\infty(\vec{X}, \vec{Y}) = \max_{t=1}^n |x_t - y_t|$. We use euclidean distance (\mathcal{L}_2 norm) for figure examples in this paper. The disadvantage of \mathcal{L}_p norm distances is that there is no obvious bound value of the maximum dissimilarity distance. It is hard to infer from their value whether the degree of dissimilarity is small or large. Thus, we also propose to use normalized \mathcal{L}_p norm distances that are divided by the number of time points. For example, the **normalized euclidean distance** between \vec{X} and \vec{Y} is defined as

$$\begin{aligned} \text{Normalized-}\mathcal{L}_2(\vec{X}, \vec{Y}) &= \left(\frac{1}{n}\right)^{\frac{1}{2}} * \mathcal{L}_2(\vec{X}, \vec{Y}) \\ &= \left(\frac{\sum_{t=1}^n |x_t - y_t|^2}{n}\right)^{\frac{1}{2}}, \end{aligned}$$

where n is the number of time slots.

The last component of the subset specification is a dissimilarity threshold. It indicates a maximum discrepancy allowable to become similarity-profiled temporal association patterns.

2.3 Interest Measure

Prevalence measures such as support have been successfully used in traditional association rule mining and temporal association rule mining [21]. Instead of a one-dimensional interest measure for measuring the prevalence of an item set in an entire data set, we use a composite interest measure that describes a discrepancy degree between its n -dimensional prevalence value sequence from a temporally divided data set and a given reference sequence. First, we define a support time sequence for the prevalence sequence of an item set.

Definition 2. Let $\mathcal{D} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_n$ be a disjoint time-stamped transaction data set. The **support time sequence** of an item set I is defined as

$$\vec{S}_I = \langle \text{support}(I, \mathcal{D}_1), \dots, \text{support}(I, \mathcal{D}_n) \rangle,$$

where $\text{support}(I, \mathcal{D}_t)$ is the support value of item set I in a transaction set \mathcal{D}_t , which is the fraction of transactions that contain the item set I in \mathcal{D}_t such that

$$\text{support}(I, \mathcal{D}_t) = |\{d \in \mathcal{D}_t | I \subseteq d\}| / |\mathcal{D}_t|, 1 \leq t \leq n.$$

For example, Fig. 2b shows the support time sequences of possible item sets from the data set in Fig. 2a.

Definition 3. Let I be an item set and $\vec{S}_I = \langle s_1, \dots, s_n \rangle$ be the support time sequence of I . Given a reference sequence $\vec{R} = \langle r_1, \dots, r_n \rangle$, an interest measure for the similarity-profiled temporal association pattern is defined as $D(\vec{R}, \vec{S}_I)$, which is a \mathcal{L}_p norm ($p = 1, 2, \dots, \infty$)-based dissimilarity distance between \vec{R} and \vec{S}_I .

Statistically, the distance can be thought of as the deviation of the support time sequence \vec{S}_I from the reference sequence \vec{R} . In the case of the normalized euclidean distance, *Normalized- $\mathcal{L}_2(\vec{R}, \vec{S}_I)$* is

$$\left(\frac{1}{n}\right)^{\frac{1}{2}} * \mathcal{L}_2(\vec{R}, \vec{S}_I) = \left(\frac{\sum_{t=1}^n |r_t - s_t|^2}{n}\right)^{\frac{1}{2}} = \sigma(\vec{S}_I).$$

The range of value is a set of real numbers between 0 to 1 inclusive since element values in the support sequence range from 0 to 1, and we assume the reference sequence values are in the same scale with the support measure, or can be transformed to the same scale. To allow a reference sequence in different baselines and scales, the reference sequence can be normalized to the range of support. Each value r_t in the reference sequence is transformed to $(r_t - r_{min}) / (r_{max} - r_{min}) * (L_{max} - L_{min}) + L_{min}$, where r_{min} is the minimum element value, and r_{max} is the maximum element value in the reference sequence, and L_{min} is the minimum value of the transformed range, i.e., 0, and L_{max} is the maximum value of the transformed range, i.e., 1. Several different similarity measures are proposed in the time series database literature [15]. For example, Dynamic Time Warping (DTW) allows phase shifts in time in order to detect similar shapes with different time phases [22], [23]. Similarity measures using Longest Common Subsequences (LCS) are also popular in subsequence matching [24]. More relaxed similarity models and the comparison of different similarity measures are beyond the scope of this paper. We plan to explore them in future work.

Definition 4. Let θ be a dissimilarity threshold. An item set I is called a similar item set if $D(\vec{R}, \vec{S}_I) \leq \theta$.

In Fig. 2, the output of similarity-profiled temporal association mining is $\{B\}$, $\{A, C\}$, and $\{B, C\}$ since their interest measure values do not exceed the dissimilarity threshold, 0.2.

3 ALGORITHMIC DESIGN CONCEPT

In this section, we discuss our algorithmic design concepts for mining similarity-profiled temporal association patterns.

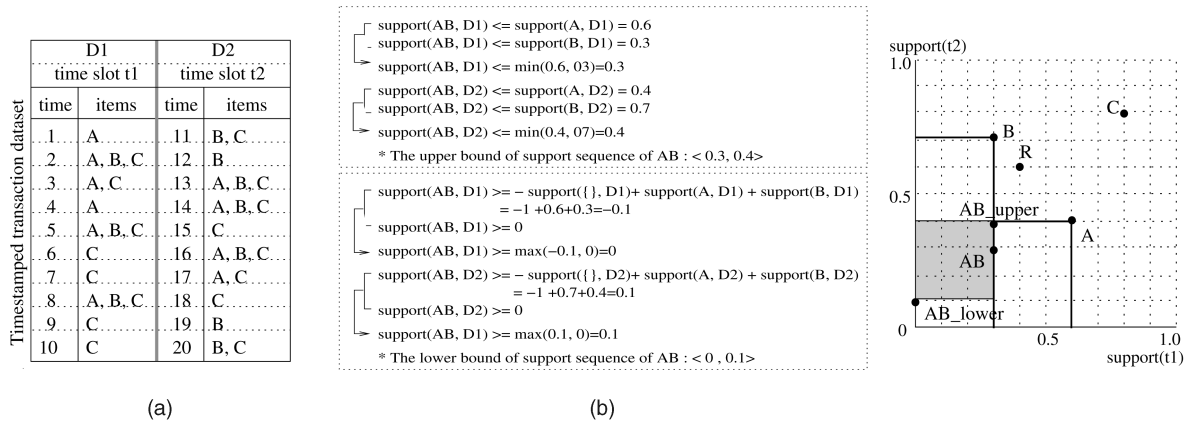


Fig. 3. An example of upper and lower bounds of support sequence of item set AB. (a) An example data set. (b) Bounds of supports and a support sequence.

We consider three means to reduce the computation cost. The first is to estimate the support time sequences without examining the data set. The second is to reduce the item set search space. The last approach concerns about ways to scan the data set for computing the support values.

3.1 Envelope of Support Time Sequence: Upper Bound and Lower Bound

Generating the support time sequences of item sets is the core operation in similarity-profiled association mining algorithm. The operation, however, is very data intensive and sometimes can produce the sequences of all combinations of items. We explore a way for estimating support time sequences without examining an input data set. In [25], Calders proposes a set of rules for deducing best bounds on the support of an item set if the supports of all subsets of it are known.

Theorem 1. Let \mathcal{D} be a transaction data set and I be an item set, $support(I, \mathcal{D}) \in [L(I, \mathcal{D}), U(I, \mathcal{D})]$ with

$$L(I, \mathcal{D}) = \max\{\sigma_I(J, \mathcal{D}), 0 \mid J \subset I \text{ and } |J| \text{ is even}\},$$

$$U(I, \mathcal{D}) = \min\{\sigma_I(J, \mathcal{D}) \mid J \subset I \text{ and } |J| \text{ is odd}\},$$

$$\text{where } \sigma_I(J, \mathcal{D}) = \sum_{J' \subseteq J \subset I} (-1)^{|I-J'|+1} \cdot support(J', \mathcal{D}).$$

$L(I, \mathcal{D})$ means a lower bound of $support(I, \mathcal{D})$, and $U(I, \mathcal{D})$ means an upper bound of $support(I, \mathcal{D})$. For example, consider the bounds of support of an item set $I = \{A, B\}$. Suppose the supports of all subsets of I are $support(\{\}) = 1$, $support(\{A\}) = 0.6$ and $support(\{B\}) = 0.3$. By Theorem 1,

$$\sigma_I(\{\}) = (-1)^3 \cdot support(\{\}) + (-1)^2 \cdot support(\{A\}) + (-1)^2 \cdot support(\{B\}) = -1 \cdot 1 + 1 \cdot 0.6 + 1 \cdot 0.3 = -0.1$$

$\sigma_I(\{A\}) = (-1)^2 \cdot support(\{A\}) = (-1)^2 \cdot 0.6 = 0.6$, and $\sigma_I(\{B\}) = (-1)^2 \cdot support(\{B\}) = (-1)^2 \cdot 0.3 = 0.3$. Among all subsets of I , $\{A\}$ and $\{B\}$ are subsets whose size is odd, thus

$$U(I, \mathcal{D}) = \min\{\sigma_I(\{A\}), \sigma_I(\{B\})\} = \min\{0.6, 0.3\} = 0.3.$$

In the case of the lower bound, $L(I, \mathcal{D}) = \max\{\sigma_I(\{\}), 0\} = \max\{-0.1, 0\} = 0$ since $\{\}$ is a subset whose size is even. Thus, $0 \leq support(\{A, B\}) \leq 0.3$. The proof of the tight

bounds is described in [25]. We adopt this set of rules to derive the tight upper bound and lower bound of the support time sequence of an item set.

Definition 5. Let $\mathcal{D} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_n$ be a time-stamped transaction data set. The lower bound support time sequence of an item set I , \vec{L}_I , and the upper bound support time sequence of I , \vec{U}_I are defined as follows:

$$\vec{L}_I = \langle l_1, \dots, l_n \rangle = \langle L(I, \mathcal{D}_1), \dots, L(I, \mathcal{D}_n) \rangle,$$

$$\vec{U}_I = \langle u_1, \dots, u_n \rangle = \langle U(I, \mathcal{D}_1), \dots, U(I, \mathcal{D}_n) \rangle.$$

From the example data set in Fig. 3a, let us derive the lower and upper bound support sequences of an item set $I = \{A, B\}$. As shown in Fig. 3b, the upper bound support sequence of $\{A, B\}$, $\vec{U}_{AB} = \langle u_1, u_2 \rangle$, is $\langle 0.3, 0.4 \rangle$ since $u_1 = U(\{A, B\}, \mathcal{D}_1) = 0.3$ and $u_2 = U(\{A, B\}, \mathcal{D}_2) = 0.4$. The lower bound support sequence of $\{A, B\}$, $\vec{L}_{AB} = \langle l_1, l_2 \rangle$, is $\langle 0, 0.1 \rangle$ since $l_1 = L(\{A, B\}, \mathcal{D}_1) = 0$ and $l_2 = L(\{A, B\}, \mathcal{D}_2) = 0.1$. Fig. 3b also illustrates the bound sequences in a two-dimensional space. The dark area indicates the range in which the true support time sequence of $\{A, B\}$ can be located.

3.2 Lower Bounding Distance

We explore lower bounding distances to find item sets whose support sequences could not possibly be a best match with a reference sequence under a dissimilarity threshold. In our concept of lower bounding distance, if the lower bounding distance of an item set does not satisfy the dissimilarity threshold, its true distance also does not satisfy the threshold. Thus, the lower bounding distance can be used to prune item sets early without the computation of the true distances. We first define our lower bounding distance with upper and lower bound support time sequences. This lower bounding distance consists of two parts, an upper lower-bounding distance and a lower lower-bounding distance.

Definition 6. For a reference sequence \vec{R} and the upper bound support sequence \vec{U} of an item set, let $\vec{R}^U = \langle r_1, \dots, r_k \rangle$ be a subsequence of \vec{R} and $\vec{U}^L = \langle u_1, \dots, u_k \rangle$ be a subsequence of \vec{U} , where $r_t > u_t$, $1 \leq t \leq k$. The upper lower-bounding distance between \vec{R} and \vec{U} , $D_{Ulb}(\vec{R}, \vec{U})$, is defined as $D(\vec{R}^U, \vec{U}^L)$.

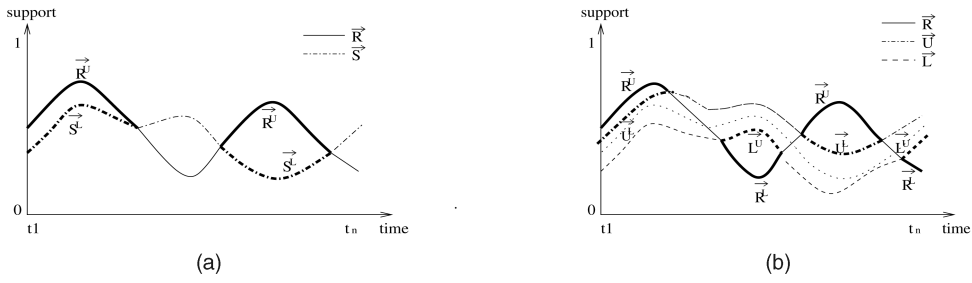


Fig. 4. Subsequences for a lower bounding distance (a) with true support sequence ($D_{Ulb}(\vec{R}, \vec{S}) = D(\vec{R}^U, \vec{S}^L)$) and (b) with upper and lower bound support sequences ($D_{Ulb}(\vec{R}, \vec{U}) = D(\vec{R}^U, \vec{U}^L)$, $D_{Llb}(\vec{R}, \vec{L}) = D(\vec{R}^L, \vec{L}^U)$).

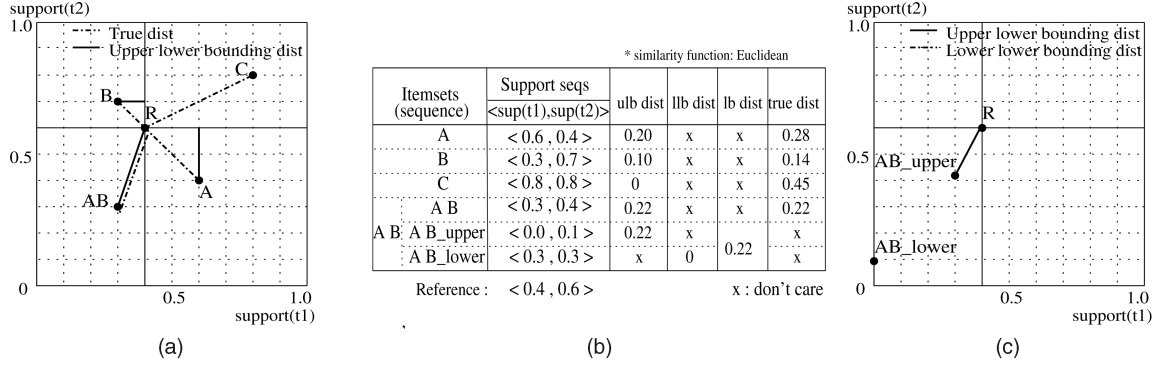


Fig. 5. An example of lower bounding distances. (a) Upper lower-bounding distances of true support sequences. (b) Distance table. (c) Upper lower-bounding distance and lower lower-bounding distance of support bound sequences.

The upper lower-bounding distance between \vec{R} and \vec{U} is a dissimilarity distance between a subsequence of \vec{R} , \vec{R}^U , and a subsequence of \vec{U} , \vec{U}^L , in which each element value r_t in \vec{R}^U is greater than the corresponding element value u_t of \vec{U}^L . For example, when euclidean distance is the similarity function, $D_{Ulb}(\vec{R}, \vec{U}) = D(\vec{R}^U, \vec{U}^L) = (\sum_{t=1}^n f(r_t, u_t))^{\frac{1}{2}}$, where if $r_t > u_t$, $f(r_t, u_t) = |r_t - u_t|^2$; otherwise, $f(r_t, u_t) = 0$. In the same way, the upper lower-bounding distance between a reference sequence \vec{R} and the true support time sequence \vec{S} of an item set, $D_{Ulb}(\vec{R}, \vec{S})$, is $D(\vec{R}^U, \vec{S}^L)$, where $\vec{S}^L = \langle s_1, \dots, s_k \rangle$ is a subsequence of \vec{S} , and $r_t > s_t$, $1 \leq t \leq k$. As shown in Fig. 4, the subsequence does not need to be a continuous sequence.

Definition 7. For a reference sequence \vec{R} and the lower bound support time sequence \vec{L} of an item set, let $\vec{R}^L = \langle r_1, \dots, r_k \rangle$ be a subsequence of \vec{R} and $\vec{L}^U = \langle l_1, \dots, l_k \rangle$ be a subsequence of \vec{L} , where $r_t < l_t$, $1 \leq t \leq k$. The **lower lower-bounding distance** between \vec{R} and \vec{L} , $D_{Llb}(\vec{R}, \vec{L})$, is defined as $D(\vec{R}^L, \vec{L}^U)$.

The lower lower-bounding distance between a reference sequence \vec{R} , and a lower bound support sequence \vec{L} is a dissimilarity distance between a subsequence of \vec{R} , \vec{R}^L , and a subsequence of \vec{L} , \vec{L}^U , in which each element value r_t in \vec{R}^L is less than the corresponding element value l_t of \vec{L}^U .

Definition 8. For a reference sequence \vec{R} , the upper bound support time sequence \vec{U} , and lower bound support time sequence \vec{L} of an item set, the **lower bounding distance** $D_{lb}(\vec{R}, \vec{U}, \vec{L})$ is defined as $D_{Ulb}(\vec{R}, \vec{U}) + D_{Llb}(\vec{R}, \vec{L})$.

Fig. 4 gives an illustration of subsequences, \vec{S}^L , \vec{R}^U , \vec{R}^L , \vec{L}^U , and \vec{U}^L , for computing lower bounding distances. Fig. 5 shows lower bounding distances computed from that in Fig. 3a data set in a two-dimensional plot. Fig. 5a

shows the upper lower-bounding distances of several true support time sequences. Fig. 5c shows the upper lower-bounding distance of the upper bound sequence of {A, B}, \vec{U}_{AB} (AB_upper in Fig. 5), and the lower lower-bounding distance of the lower bound sequence of {A, B}, \vec{L}_{AB} (AB_lower in Fig. 5). The lower lower-bounding distance of \vec{L}_{AB} is $D_{Llb}(\vec{L}_{AB}, \vec{R}) = 0$ since no value in \vec{L}_{AB} is greater than the values of \vec{R} . The lower bound distance of the two bound sequences is $D_{Ulb}(\vec{U}_{AB}, \vec{R}) + D_{Llb}(\vec{L}_{AB}, \vec{R}) = 0.22 + 0 = 0.22$, as shown in Fig. 5b.

Lemma 1. For the upper bound support time sequence $\vec{U} = \langle u_1, \dots, u_n \rangle$, the lower bound support time sequence $\vec{L} = \langle l_1, \dots, l_n \rangle$, support time sequence $\vec{S} = \langle s_1, \dots, s_n \rangle$ of an item set I , and a reference sequence $\vec{R} = \langle r_1, \dots, r_n \rangle$, the lower bounding distance $D_{lb}(\vec{R}, \vec{U}, \vec{L})$ and the true distance $D(\vec{R}, \vec{S})$ hold the following inequality:

$$D_{lb}(\vec{R}, \vec{U}, \vec{L}) \leq D(\vec{R}, \vec{S}).$$

Proof. According to Definition 8, $D_{lb}(\vec{R}, \vec{U}, \vec{L}) = D_{Ulb}(\vec{R}, \vec{U}) + D_{Llb}(\vec{R}, \vec{L}) = (\sum_{t=1, r_t > u_t}^n (r_t - u_t)^2)^{\frac{1}{2}} + (\sum_{t=1, r_t < l_t}^n (r_t - l_t)^2)^{\frac{1}{2}}$, when euclidean distance is used as a similarity function. By Definition 5, $u_t = U(I, D_t)$, where $1 \leq t \leq n$. By Theorem 1, $s_t \leq U(I, D_t)$. Thus, $s_t \leq u_t$. Similarly, by Definition 5, $l_t = L(I, D_t)$, where $1 \leq t \leq n$. By Theorem 1, $s_t \geq L(I, D_t)$. Thus, $s_t \geq l_t$. Finally, $D_{lb}(\vec{U}, \vec{L}, \vec{R}) = D_{Ulb}(\vec{R}, \vec{U}) + D_{Llb}(\vec{R}, \vec{L}) = (\sum_{t=1, r_t > u_t}^n (r_t - u_t)^2)^{\frac{1}{2}} + (\sum_{t=1, r_t < l_t}^n (r_t - l_t)^2)^{\frac{1}{2}} \leq (\sum_{t=1, r_t > s_t}^n (r_t - s_t)^2)^{\frac{1}{2}} + (\sum_{t=1, r_t < s_t}^n (r_t - s_t)^2)^{\frac{1}{2}} = (\sum_{t=1}^n (r_t - s_t)^2)^{\frac{1}{2}} = D(\vec{R}, \vec{S})$. \square

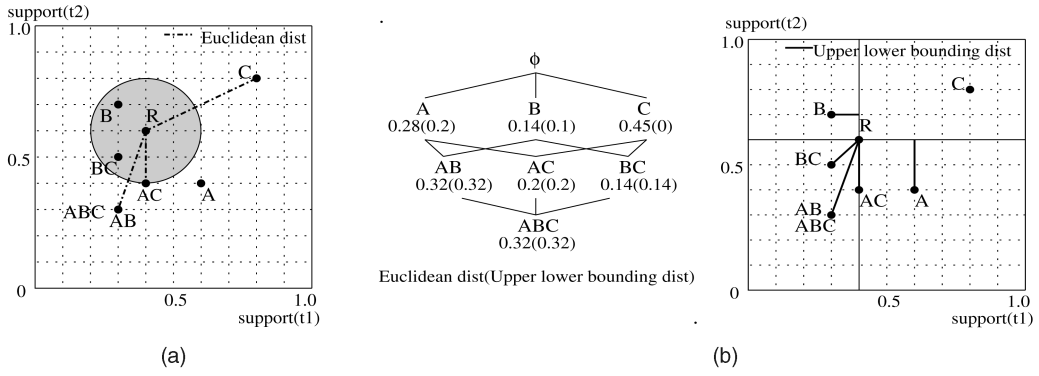


Fig. 6. (a) Nonmonotonicity of the euclidean distance. (b) Monotonically nondecreasing property of the upper lower-bounding distance.

3.3 Monotonicity Property of Upper Lower-Bounding Distance

Next, we explore a scheme to further reduce the item set search space for similarity-profiled temporal pattern mining. The most popular technique to reduce the item set search space in association pattern mining is to use the monotonicity property of the support measure [3]. The support values of all supersets of a given item set are not greater than the item set's support value. Thus, if an item set does not satisfy the support threshold, all supersets of the item set can be pruned. If our interest measure has a property similar with it, we can use it for reducing our item set search space. However, we observed that our interest measure does not show such a monotonicity. For example, Fig. 6a shows the euclidean distances between the support time sequences of $\{C\}$, $\{A, C\}$, and $\{A, B, C\}$; \vec{S}_C , \vec{S}_{AC} ; and \vec{S}_{ABC} , and a reference sequence \vec{R} . As can be seen, $D(\vec{S}_C, \vec{R}) = 0.45$, $D(\vec{S}_{AC}, \vec{R}) = 0.2$, and $D(\vec{S}_{ABC}, \vec{R}) = 0.32$. Thus, $D(\vec{S}_{ABC}, \vec{R}) > D(\vec{S}_{AC}, \vec{R})$, but $D(\vec{S}_{AC}, \vec{R}) < D(\vec{S}_C, \vec{R})$. However, we discovered an interesting property related to our upper lower-bounding distance. First, we present a related lemma.

Lemma 2. *The values of the support time sequence of an item set are monotonically nonincreasing with the size of the item set at each time slot.*

Proof. A support has a monotonically nonincreasing property with increasing size of an item set [3]. The support time sequence of an item set consists of supports computed from disjoint sets of transactions by Definition 2. Each element value of the sequence follows the same monotonicity property, i.e., if I and J are item sets in a time slot and $J \subseteq I$, then $support(J) \geq support(I)$. \square

Lemma 3. *The upper lower-bounding distance between the (upper bound) support time sequence of an item set and a reference time sequence is monotonically nondecreasing with the size of the item set.*

Proof. We prove the monotonicity of upper lower-bounding distances to true support time sequences using euclidean distance. According to Definition 6, the upper lower-bounding distance between $\vec{S}_I = \langle s_1, \dots, s_n \rangle$ for a size k item set I and $\vec{R} = \langle r_1, \dots, r_n \rangle$ is $D_{Ulb}(\vec{R}, \vec{S}_I) = (\sum_{t=1, r_t > s_t}^n (r_t - s_t)^2)^{\frac{1}{2}}$. For a size $k+1$ item set $I' = I \cup \{i'\}$, where $i' \notin I$, and its support time

sequence $\vec{S}_{I'} = \langle s'_1, \dots, s'_n \rangle$, we need to prove that $D_{Ulb}(\vec{S}_I, \vec{R}) \leq D_{Ulb}(\vec{S}_{I'}, \vec{R})$. According to Lemma 2, the support is nonincreasing with the size of item set at each time slot, i.e., the support of I' is equal to or less than the support of I at each time slot such that $s'_1 \leq s_1, \dots, s'_n \leq s_n$. If $s_t \geq s'_t$, $s_t < r_t$ and $s'_t < r_t$, then $r_t - s_t \leq r_t - s'_t$. Thus, we can get $(\sum_{t=1, r_t > s_t}^n (r_t - s_t)^2)^{\frac{1}{2}} \leq (\sum_{t=1, r_t > s'_t}^n (r_t - s'_t)^2)^{\frac{1}{2}}$, i.e., $D_{Ulb}(\vec{R}, \vec{S}_I) \leq D_{Ulb}(\vec{R}, \vec{S}_{I'})$. The monotonicity of upper lower-bounding distance to the upper bound support time sequence can be similarly proved. \square

For example, in Fig. 6b, $D_{Ulb}(\vec{S}_A, \vec{R}) = 0.2$, $D_{Ulb}(\vec{S}_B, \vec{R}) = 0.1$, and $D_{Ulb}(\vec{S}_{AB}, \vec{R}) = 0.32$. Thus, $D_{Ulb}(\vec{S}_A, \vec{R}) \leq D_{Ulb}(\vec{S}_{AB}, \vec{R})$ and $D_{Ulb}(\vec{S}_B, \vec{R}) \leq D_{Ulb}(\vec{S}_{AB}, \vec{R})$. We can also see that $D_{Ulb}(\vec{S}_{AB}, \vec{R}) \leq D_{Ulb}(\vec{S}_{ABC}, \vec{R})$, $D_{Ulb}(\vec{S}_{AC}, \vec{R}) \leq D_{Ulb}(\vec{S}_{ABC}, \vec{R})$, and $D_{Ulb}(\vec{S}_{BC}, \vec{R}) \leq D_{Ulb}(\vec{S}_{ABC}, \vec{R})$.

3.4 Item Set Filtering Strategy

We devised three pruning steps using the low bounding distance and the monotonicity property of upper lower-bounding distance to effectively reduce the item set search space.

Pruning by subset check. The first step is a pruning by the subset check of the candidate. If the upper lower bounding distance of any subset of a candidate item set does not satisfy the dissimilarity threshold, the candidate item set is pruned using the nondecreasing monotonicity property of upper lower bounding distance.

Pruning by lower bounding distance of bound support sequences. The second step is the pruning of a candidate item set using the lower bounding distance of its upper bound support sequence and lower bound support sequence. If the lower bounding distance does not satisfy the dissimilarity threshold, the candidate can be eliminated without examining the data set and computing its true distance value.

Pruning by upper lower bounding distance of true support sequence. The final step is the pruning of item sets for reducing the number of the next size candidate item sets generated. If the upper lower-bounding distance to the true support time sequence of an item set does not satisfy the threshold, the item set is not involved in generating the next size candidates.

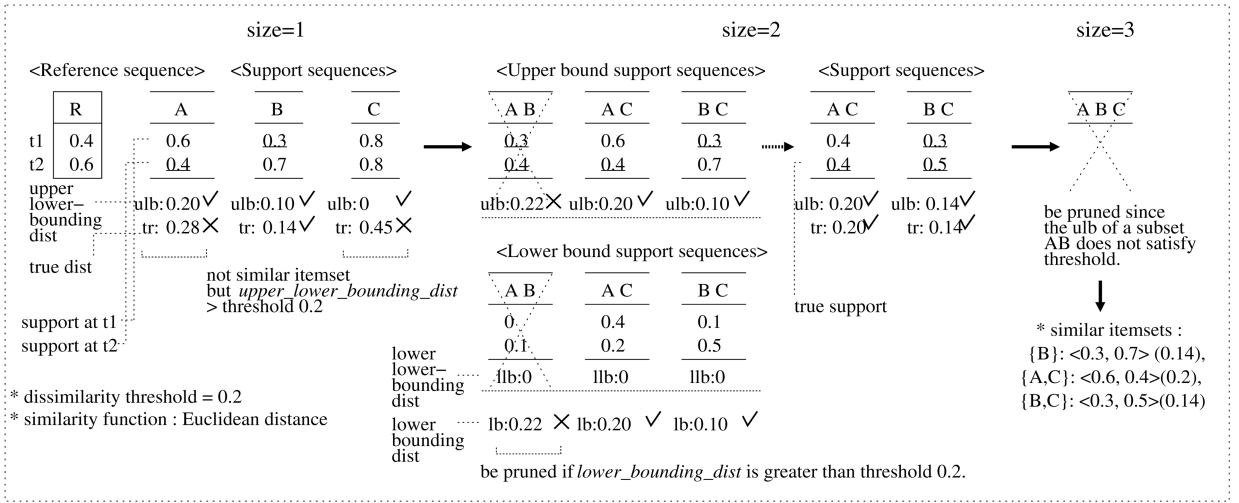


Fig. 7. A trace illustration of the SPAMINE algorithm.

3.5 Data Scan Strategy

Support time sequences can be constructed by different ways in scanning the time stamped transaction data set. We can consider two data scan methods.

Lattice-dominant scan. The lattice-dominant scan method reads a whole transaction data set from time slot t_1 to time slot t_n for candidate item sets of each size and generates their support time sequences.

Snapshot-dominant scan. The snapshot-dominant scan method repeats the scanning of transactions at each time slot. First, it counts the supports of all candidate item sets of different sizes in the first time slot, and then it moves to the next time slot and repeats the process. This method incrementally generates the support time sequences with the processed time slots.

4 ALGORITHM FOR SIMILARITY-PROFLED TEMPORAL ASSOCIATION MINING

We developed a Similarity-Profled temporal Association Mining mEthod (SPAMINE) based on our algorithm design concept discussed in Section 3. We present the SPAMINE algorithm first, and then, for comparison, we also present an alternative method using a support-pruning scheme.

4.1 SPAMINE Algorithm

Algorithm 1 shows the pseudocode of the SPAMINE. Fig. 7 provides a trace illustration of the SPAMINE execution using the example data set in Fig. 2a.

Algorithm 1. SPAMINE algorithm.

Inputs:

E : A set of single items.
 TD : A time-stamped transaction database
 \vec{R} : A reference sequence
 D : A similarity function
 θ : A dissimilarity threshold

Output: All item sets whose support sequences are similar to \vec{R} under D and θ

Variables:

k : item set size

C_k : A set of size k candidate item sets
 \vec{U}_k : A set of upper bound support sequences of size k item sets
 \vec{L}_k : A set of lower bound support sequences of size k item sets
 \vec{S}_k : A set of support sequences of size k item sets
 \vec{S} : A set of support sequences of all item sets
 B_k : A set of size k item sets whose upper lower-bounding distance $\leq \theta$
 A_k : A result set of size k item sets whose true distance $\leq \theta$

Main:

- 1) $C_1 = E$;
- 2) $\vec{S}_1 = \text{generate_support_sequences}(C_1, TD)$;
- 3) $(A_1, B_1) = \text{find_similar_itemsets}(C_1, \vec{S}_1, \vec{R}, D, \theta)$;
- 4) $k = 2$;
- 5) **while** (not empty B_{k-1}) **do**
- 6) $(C_k, \vec{U}_k, \vec{L}_k) = \text{generate_candidates_and_bound_sequences}(B_{k-1}, \vec{S})$;
- 7) $C_k = \text{prune_candidates_by_lbd}(C_k, \vec{U}_k, \vec{L}_k, \vec{R}, D, \theta)$
- 8) $\vec{S}_k = \text{generate_support_sequences}(C_k, TD)$;
- 9) $(A_k, B_k) = \text{find_similar_itemsets}(C_k, \vec{S}_k, \vec{R}, D, \theta)$;
- 10) $\vec{S} = \vec{S} \cup \vec{S}_k$; $k = k + 1$;
- 11) **end**
- 12) **return** $\bigcup(A_1, \dots, A_k)$;

Generate the support time sequences of single items and find similar items (Steps 1-3). All singletons ($k = 1$) become candidate items (C_1). The SPAMINE algorithm uses the lattice-dominant database scan method to generate the support time sequences. In the first scan of an entire input data set, the supports of singletons are computed per each time slot, and their support time sequences (\vec{S}_1) are generated. If distances between the support time sequences and a given reference sequence do not exceed a given dissimilarity threshold, the singletons are added to a result set (R_1). On the fly, if the upper lower-bounding distances of the support time sequences satisfy the threshold, the items are kept to B_1 for generating the next size candidates. In Fig. 7,

only item B is a similar item set, but items A and C are also kept for generating the next size candidate item sets.

Generate candidate item sets, and their upper and lower bound support sequences (Step 6). All size k ($k > 1$) candidate itemsets(C_k) are generated using size $k - 1$ itemsets(B_{k-1}) whose upper lower-bounding distances satisfy the dissimilarity threshold. If any subset of size $k - 1$ of a candidate item set is not in the B_{k-1} , the candidate is eliminated using the monotonically nondecreasing property of the upper lower-bounding distance. Otherwise, the upper and lower bound support sequences of candidates are generated. Fig. 7 shows the generated size 2 candidate item sets and their upper and lower bound support time sequences. Algorithm 2 shows the pseudocode of the *generate_candidates_and_bound_sequences* function.

Prune candidate item sets using their lower bounding distances (Step 7). If the lower bounding distance of the upper and lower bound support sequences of a candidate exceeds the dissimilarity threshold, the candidate is eliminated from the candidate set. For example, in Fig. 7, a candidate {A,B} is pruned since its lower bounding distance is greater than the threshold 0.2. Algorithm 3 shows the pseudocode of the *prune_candidates_by_lbd* function.

Scan the data set and generate the support time sequences (Step 8): The supports of candidate item sets are computed during the scan of the transaction data set from time slot t_1 to t_n , and their support time sequences(\vec{S}_k) are generated. Algorithm 4 shows the pseudocode of the *generate_support_sequences* function.

Find similar item sets (Step 9). The true distance between the support time sequence of an item set and the reference sequence is computed. If the value satisfies the threshold, the item set is included in the result set(R_k). In Fig. 7, {A,C} and {B,C} are similar item sets. On the fly, if the upper lower-bounding distances of candidate item sets satisfy the threshold, the item sets are added to B_k for generating the next size candidate item sets. Algorithm 5 shows the pseudocode of the *find_similar_item_sets* function. The size of examined item sets is increased to $k = k + 1$. The above procedures (steps 6-10) are repeated until no item set in B_k remains.

Algorithm 2. Generate_candidates_and_bound_sequences function.

```

generate_candidates_and_bound_sequences( $B_{k-1}, \vec{S}$ )
 $C_k$ : Size  $k$  candidate item sets
 $\vec{U}_k^c$ : Upper bound sequence of candidate item set  $c$ 
 $\vec{L}_k^c$ : Lower bound sequence of candidate item set  $c$ 
begin
    insert into  $C_k$ 
    select  $a.item_1, \dots, a.item_{k-1}, b.item_{k-1}$ 
    from  $B_{k-1}a, B_{k-1}b$ 
    where  $a.item_1 = b.item_1, \dots, a.item_{k-2} = b.item_{k-2}$  and
 $a.item_{k-1} < b.item_{k-1}$ ;

```

```

forall item set  $c \in C_k$  do
     $\vec{U}_k^c = 1$ ;
     $\vec{L}_k^c = 0$ ;
    forall size  $k - 1$  subset  $s$  of  $c$  do
        if  $s \notin B_{k-1}$  then delete  $c$  from  $C_k$ ;
        else  $\vec{U}_k^c = \text{make\_upper\_bound}(c, \vec{S})$ ;

```

```

         $\vec{L}_k^c = \text{make\_lower\_bound}(c, \vec{S})$ ;
    end
    return ( $C_k, \vec{U}_k, \vec{L}_k$ );
end

```

Algorithm 3. Prune_candidates_by_lbd function.

```

prune_candidates_by_lbd( $C_k, \vec{U}_k, \vec{L}_k, \vec{R}, D, \theta$ )
 $D_{Ulb}$  : upper lower-bounding distance function
 $D_{Llb}$  : lower lower-bounding distance function
begin
    for all candidates  $c \in C_k$  do
        if  $D_{Ulb}(\vec{U}_k^c, \vec{R}) + D_{Llb}(\vec{L}_k^c, \vec{R}) > \theta$  then  $C_k = C_k - c$ ; end
    end
    return  $C_k$ ;
end

```

Algorithm 4. Generate_support_sequences function.

```

generate_support_sequences( $C_k, TD$ )
 $TD_i$ : transactions in time slot  $i$ 
 $\vec{S}_k = |C_k| * < s_1, \dots, s_n >$ 
begin
    for ( $i = 1; i \leq n; i++$ ) do
        for all candidates  $c \in C_k$  do
             $\vec{s}_i^c = \text{calculate\_support}(c, TD_i)$ ;
        end
    end
    return  $\vec{S}_k$ ;
end

```

Algorithm 5. Find_similar_item_sets function.

```

find_similar_itemsets( $C_k, \vec{S}_k, \vec{R}, D, \theta$ )
begin
    for all candidates  $c \in C_k$  do
        if  $D_{Ulb}(\vec{S}_k^c, \vec{R}) \leq \theta$  then  $B_k = B_k \cup c$  end
        if  $D(\vec{S}_k^c, \vec{R}) \leq \theta$  then  $A_k = A_k \cup c$  end
    end
    return ( $A_k, B_k$ );
end

```

4.2 A Sequential Method

To our knowledge, there is no existing temporal association mining method in the literature that can be directly applied to our similarity-profiled association mining. To compare the effect of our design decision in developing the similarity-profiled association mining algorithm, we devised an alternative method that reads the input data set sequentially (i.e., snapshot-dominant scan) and filters candidate item sets using the support monotonicity property, which is popularly used in traditional association rule mining [3]. However, this method uses self-computed variable minimum support thresholds over time slots per item set. If the support of an item set at a time slot is less than the computed minimum support threshold, the item set can be pruned without examining its support values in other time slots and computing its similarity with a reference sequence. For example, in the case of normalized \mathcal{L}_1 norm distance, at the first time slot t_1 , the minimum support threshold for an item set is $\max\{r_1 - |T| \times \text{dissimi}, 0\}$, where r_1 is an element value at time t_1 of a reference sequence \vec{R} , $|T|$ is the total

number of time slots, and $dissimi$ is a dissimilarity threshold. $|T| \times dissimi$ denotes a maximum discrepant value allowed at time t_1 . If the support value of an item set at time slot t_1 is less than the computed minimum support threshold, we know the difference between the support sequence of the item set, and the reference sequence exceeds the dissimilarity threshold without examining other timeslots. The candidate item set would not be a similar item set. The supersets of the item set will also not satisfy the dissimilarity threshold since their support values are never greater than the item set's support by the monotonicity property of support, and their dissimilarity values increase. The minimum support threshold values at the other time slots are adjusted using the support values in the processed time slots. At time slot t_i , the minimum support threshold of an item set is $max\{r_i - (|T| \times dissimi - \sum_{j=1}^{i-1} g(r_j, s_j)), 0\}$, where s_j is a support value at time slot j , and $g(r_j, s_j) = r_j - s_j$ if $r_j > s_j$, otherwise 0, where $1 \leq j < i$. For example, the minimum support threshold of an item set I at time slot t_2 is adjusted with its support value s_1 in the time slot t_1 , that is, $max\{r_2 - (|T| \times dissimi - (r_1 - s_1)), 0\}$. In the same way, the minimum support threshold at time slot t_3 is $max\{r_3 - (|T| \times dissimi - ((r_1 - s_1) + (r_2 - s_2))), 0\}$ if $r_1 > s_1$ and $r_2 > s_2$.

5 ANALYTICAL ANALYSIS

5.1 Completeness and Correctness

We analyze the SPAMINE algorithm in terms of correctness and completeness. Correctness means that the distances between the support time sequences of all result item sets, and the reference sequence are not greater than a given dissimilarity threshold. Completeness means that the algorithm can find all item sets whose support sequences are similar to the reference sequence under the dissimilarity threshold. First, we introduce a related lemma.

Lemma 4. For a reference sequence \vec{R} and an (upper bound) support time sequence \vec{S} of an item set, the true distance $D(\vec{R}, \vec{S})$ and the upper lower-bounding distance $D_{Ulb}(\vec{R}, \vec{S})$ hold the following inequality: $D_{Ulb}(\vec{R}, \vec{S}) \leq D(\vec{R}, \vec{S})$.

Proof. According to Definition 6,

$$D_{Ulb}(\vec{R}, \vec{S}) = \left(\sum_{t=1, r_t > s_t}^n (r_t - s_t)^2 \right)^{\frac{1}{2}} \leq \left(\sum_{t=1}^n (r_t - s_t)^2 \right)^{\frac{1}{2}} = D(\vec{R}, \vec{S}),$$

if euclidean distance is used as a similarity function. \square

Theorem 2. The SPAMINE algorithm is complete.

Proof. First, we will show that in Step 6 of Algorithm 1, the *generate_candidates_and_bound_sequences* function using the property of upper lower-bounding distance doesn't miss candidates, which can be true similar item sets. Let J be a subset of a size k candidate item set I . According to Lemma 3 and Lemma 4, $D_{Ulb}(\vec{R}, \vec{S}_J) \leq D_{Ulb}(\vec{R}, \vec{S}_I) \leq D(\vec{R}, \vec{S}_I)$. Thus, if $D_{Ulb}(\vec{R}, \vec{S}_J) > \theta$, then $D(\vec{R}, \vec{S}_I) > \theta$ and I cannot be a similar item set. Next, we will show that in Step 7 of Algorithm 1, the *prune_candidates_by_lbd* function using the property of the lower bounding distance does not remove a true item set. According to

Lemma 1, $D_{lb}(\vec{R}, \vec{U}_I, \vec{L}_I) \leq D(\vec{R}, \vec{S}_I)$. If $D_{lb}(\vec{R}, \vec{U}_I, \vec{L}_I) > \theta$, then $D(\vec{R}, \vec{S}_I) > \theta$. Thus, Step 7 does not prune a true item set. Steps 2 and 8 generate support time sequences from time stamped transactions correctly, and Steps 3 and 9 do find true similar item sets whose distances satisfy the dissimilarity threshold. \square

Theorem 3. The SPAMINE algorithm is correct.

Proof. The correctness can be guaranteed by Steps 3 and 9 in Algorithm 1. The *fine_similar_item_sets* function calculates true distances and includes only item sets whose distance is not greater than the dissimilarity threshold in the result set. \square

5.2 Computational Analysis

Next, we briefly compare the computation costs of SPAMINE and the alternative approach, the sequential method. First, we examine the computational complexities of the two methods. Let $T_{spamine}$ and $T_{sequential}$ be the total computation cost of the SPAMINE and of the sequential method, respectively:

$$\begin{aligned} T_{spamine} = & \sum_{k=1, m} \{ T_{gen_k_candi} + T_{gen_sup_bounds}(C_k) \\ & + T_{filter_coarse_itemsets}(C_k) \\ & + T_{gen_sup_seqs}(C'_k, D) \\ & + T_{find_true_itemsets}(C'_k) \}. \end{aligned}$$

$T_{gen_k_candi}$ represents the cost of generating size k candidate item sets. Let C_k be a set of the generated size k candidates. $T_{gen_sup_bounds}(C_k)$ is the cost of generating the support bound sequences of the candidate item sets. $T_{filter_coarse_itemsets}(C_k)$ is the cost of filtering the candidates using their lower bounding distances. Let C'_k be a set of the filtered size k candidate item sets. $T_{gen_sup_seqs}(C'_k, D)$ is the cost of scanning the whole data set D and generating the true support time sequences of the filtered candidates C'_k . $T_{find_true_itemsets}(C'_k)$ is the cost of finding a similar item set of size k that satisfies a given dissimilarity. Finally, m denotes the largest size of a candidate item set that can be generated in the SPAMINE algorithm:

$$\begin{aligned} T_{sequential} = & \sum_{i=1, t} \left\{ \sum_{k=1, n} \left\{ T_{gen_k_candi} + T_{filter_coarse_itemsets'}(C''_k) \right. \right. \\ & \left. \left. + T_{gen_partial_sup_seqs}(C''_k, D_i) \right\} \right\} \\ & + T_{find_true_itemsets}. \end{aligned}$$

First, t represents the total number of time slots. D_i is a set of transactions included in a time slot i . n denotes the largest size of candidate item sets generated at each time slot. Let C''_k be a set of size k candidates generated at each time slot. $T_{filter_coarse_itemsets'}(C''_k)$ is the cost of filtering the candidate item sets using variable minimum support thresholds. $T_{gen_partial_sup_seqs}(C''_k, D_i)$ is the cost of scanning the current data subset D_i and generating partial support sequences until the current time slot i of the filtered candidate set C''_k . The sequential method repeats the procedure to generate candidates, filter item sets coarsely, and generate partial support sequences with increasing candidate item set size(k) in each time slot. $T_{find_true_itemsets}$

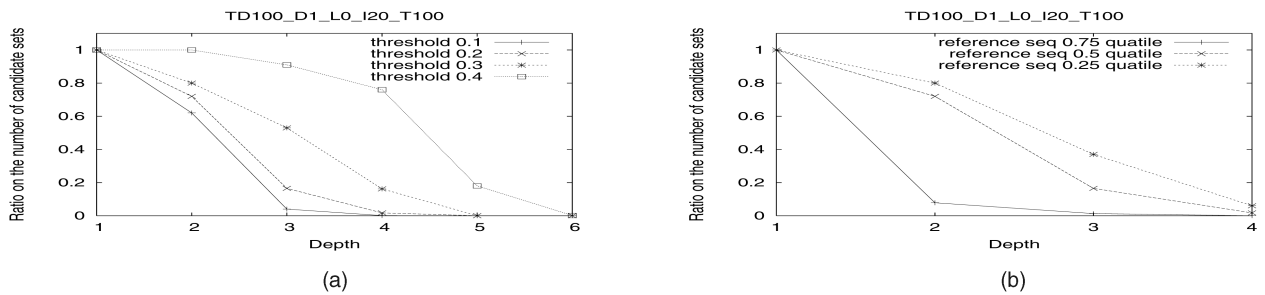


Fig. 8. Effect of overall pruning scheme in SPAMINE.

is the cost of finding all similar item sets from the computed true support sequences.

Next, we briefly compare the computation cost of the two different methods:

$$\begin{aligned}
 & T_{spamine}/T_{sequential} \\
 & \approx \left\{ \sum_{k=1,m} \left\{ T_{gen_k_candi} + T_{gen_sup_bounds}(C_k) \right. \right. \\
 & \quad \left. \left. + T_{filter_coarse_itemsets}(C_k) \right. \right. \\
 & \quad \left. \left. + T_{gen_sup_seqs}(C'_k, D) \right\} \right. \\
 & \quad \left. + T_{find_true_itemsets} \right\} \\
 & / \left\{ \sum_{k=1,n} \left\{ t \times (T_{gen_k_candi} + T_{filter_coarse_itemsets}'(C''_k) \right. \right. \\
 & \quad \left. \left. + T_{gen_partial_sup_seqs}(C''_k, D_i)) \right\} \right. \\
 & \quad \left. + T_{find_true_itemsets} \right\}.
 \end{aligned}$$

Recall that m denotes the largest size of candidate item sets that can be generated in SPAMINE, and n denotes the largest size of candidate item sets of each time slot that can be generated in the sequential method. We can expect $n > m$ since the pruning in time slots which are examined earlier in the sequential method is not effective due to the low support threshold values, and thus, in the worst case, item sets of the largest size possible (i.e., with the number of distinct items) could be generated. As a result, the total number of generated candidates in the sequential method, $\sum_{i=1,t} \sum_{k=1,n} C''_k$, would be much bigger than the total number of generated candidates in SPAMINE, $\sum_{k=1,m} C_k$. Thus, $t \times T_{gen_partial_sup_seqs} \gg T_{gen_sup_seqs}$ due to the number of candidate item sets examined in the sequential method. Overall, we can expect $T_{sequential} \gg T_{spamine}$ even if SPAMINE has $T_{gen_sup_bounds}$.

6 EXPERIMENTAL EVALUATION

We evaluate the SPAMINE algorithm to discover similarity-profiled temporal association patterns using synthetic and real data sets. All experiments were performed on a workstation with Intel Xeon 2.8 GHz with 2 Gbytes of memory running on Linux operating system.

6.1 Experiment Data Sets

Our experiments were performed on both synthetic and real data. Synthetic data sets were generated using a general transaction data generator used in [3]. We modified it for generating time stamped transactions, where each transaction has a time slot value. In the rest of paper, we use the following parameters to characterize the synthetic data sets we used. TD is the total number of transactions ($\times 1,000$), D is the number of transactions per time slot ($\times 1,000$), I is the number of distinct items, L is the average size of transactions, and T is the number of time slots. A reference time sequence was generated by choosing randomly a support sequence of an item set or by selecting a support value near a quantile, e.g., 0.25, 0.5, and 0.75, of the sorted supports of single items at each time slot. The default reference time sequence was chosen near the 0.5 quantile. For the experiment with real data, we used an Earth climate data set, a website click stream data set (BMS-Webview-1) and a land cover type data set (CoverType). The Earth climate data set includes monthly measurements of various climate variables (e.g., temperature and precipitation) and other related variables (e.g., Net Primary Production). This data set is nonpublic [26]. The BMS-Webview-1 data set contains click stream data from an e-commerce website [27]. The CoverType data set is a forest cover type data set, which is available in the UCI Repository [28]. Throughout the experiment, we used the normalized distances for similarity functions. Thus, the dissimilarity values ranged from 0 to 1.

6.2 Experiment Results

We present the results of the series of experiments with SPAMINE and the sequential method.

6.2.1 Evaluation with Synthetic Data

Effect of overall pruning scheme. First, we examined the overall effect of our pruning scheme on SPAMINE using the TD100-D1-L10-I20-T100 data set. Fig. 8a shows the ratios of number of candidate sets per depth (i.e., item set size) at different dissimilarity thresholds. The ratio value is the number of candidate item sets whose true supports are computed over the total number of possible item sets per depth. As can be seen, the SPAMINE substantially reduces the number of candidate sets. The effect is particularly significant with increase of depth. Fig. 8b shows the pruning effect with different reference sequence types. The dissimilarity threshold was set to 0.2. Our pruning scheme proves most effective when overall reference sequence values are higher than the support sequence values of item sets.

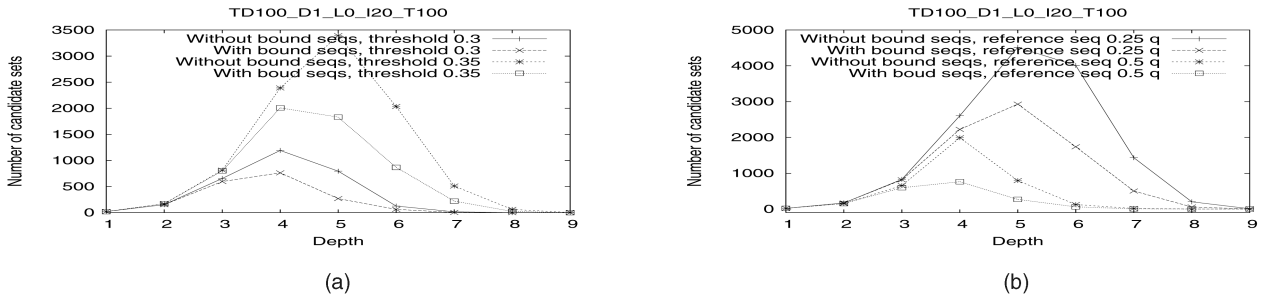


Fig. 9. Effect of lower bounding distance pruning of bounds of support sequence in SPAMINE.

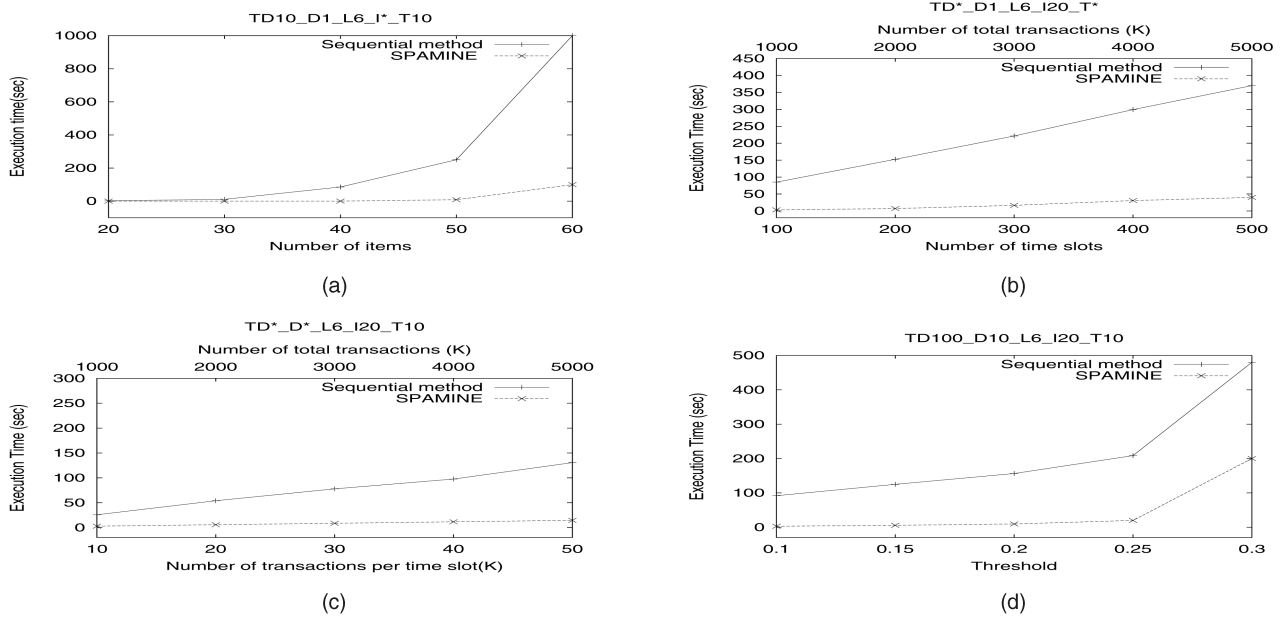


Fig. 10. Evaluation of scalability. (a) Effect of number of items. (b) Effect of number of time slots. (c) Effect of number of transactions per time slot. (d) Effect of threshold.

Effect of pruning by bounds of support sequences. In the next experiment, we examined the pruning effect by low bounding distance of the upper and lower bound support sequences. We compared two versions of the SPAMINE algorithm, one which uses the bounds of support sequences and one which does not. We used the same data set, TD100-D1-L10-I20-T100 and set the dissimilarity threshold at 0.2. Fig. 9a shows the number of candidates whose supports are needed to compute at different thresholds. As can be seen, the algorithm using the pruning scheme based on bounds of support sequences and the lower bound distance produces fewer candidate item sets. In Fig. 9b, we show the results with different reference sequence types.

Effect of number of items. We examined the effect of number of items with synthetic data sets of different number of items, TD10_D1_L6_I*_T10. Since the number of candidate item sets in the sequential method increases exponentially with the increase of number of items, we limited the execution to depth 5 and the threshold was fixed at 0.1. As seen in Fig. 10a, SPAMINE showed a similar execution time with less effect on the increase of small numbers of items in this experiment. In contrast, the execution time of the sequential method dramatically increased with increase of number of items. The reason is that the support pruning was not effective due to low

support threshold values, especially in earlier processed time slots.

Effect of number of time slots. In this experiment, we examined the effect of number of time slots using synthetic data sets, TD*_D1_L6_I20_T* having different numbers of time slots. The number of transactions per time slot was fixed, but the total data set size was increased with the number of time slots. Reference sequences were chosen near the 0.5 quantile in each data set, and the threshold was 0.1. As seen in Fig. 10b, SPAMINE receives much less effect from number of time slots. In contrast, the sequential method rapidly increases with the number of time slots.

Effect of number of transactions per time slot. In this experiment, we increased the number of transactions per time slot under a fixed number of time slots. The synthetic data sets were TD*_D*_L6_I20_T10. The total data set size increases with increase of number of transactions per time slot. In Fig. 10c, the overall execution time of the two algorithms increased with the increase of number of transactions. When compared with the results shown in Fig. 10b, in the same size data set, the execution times of the sequential method were more sensitive with regard to the number of time slots.

Effect of similarity threshold. We examined the performance of the two algorithms with different threshold values using the TD10_D1_L10_I20_S10 data set. As can be seen in

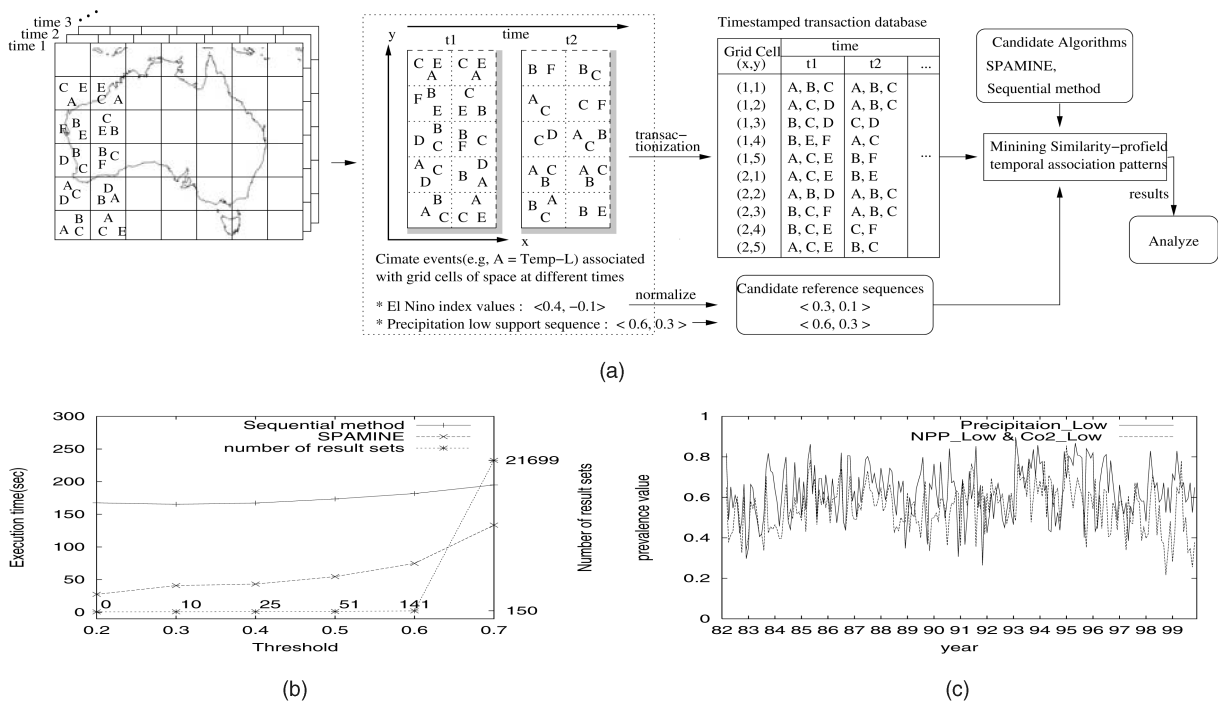


Fig. 11. Experimental result with an Earth climate data set. (a) Experimental setup. (b) Evaluation by thresholds (Reference : SOI index). (c) An illustration of result item set (Reference: precipitation low).

Fig. 10d, the execution time of the two algorithms increase with the increase of dissimilarity threshold values.

6.2.2 Evaluation with Real Data

An Earth climate data set. In the first experiment with real data, we evaluated the two algorithms with an Earth climate data set. The data set consists of global snapshots of measurement values for a number of variables (e.g., temperature, precipitation, NPP, CO₂, and Solar). The data was measured on latitude-longitude spherical grids of 0.5 degree \times 0.5 degree. For our analysis, we used measurement values in the Australia region since the climate phenomena in Australia is known to be linked to El Niño, the anomalous warming of the eastern tropical region of the Pacific [29]. The total number of grids in the Australia data, i.e., the number of transactions per time slot, was 2,827. First, we removed seasonal variation from the time series measurement data using a monthly Z score, i.e., by subtracting off the mean and dividing by the standard deviation. We defined event items based on four percentiles from the time series data of each variable (e.g., PRECI-LL, PRECI-L, PRECI-H, and PRECI-HH). The total number of items for our analysis was 50. The data set is available at monthly intervals from 1982 to 1999. We used 214 months of data, i.e., the number of time slots was 214. The total number of transactions was 604,978. For reference sequences, we used two sequences, the sequence of SOI, which is one of the indexes related to the El Niño phenomenon, and the prevalence sequence of low precipitation. Since the SOI index range is different from the support range, we first normalized the index values to the range of 0 to 1. The transformation of the raw value x was calculated as $(x - x_{min})(1 - 0)/(x_{max} - x_{min}) + 0$, where x_{min} is the minimum value of raw value x , and x_{max} is the maximum value of x . In the case of the SOI index, it was

$((x - (-7.6)) * (1 - 0))/(3.4 - (-7.6)) + 0$. Fig. 11a shows the experimental setup.

First, we used the El Niño index sequence (SOI index) for the reference sequence and discovered event item sets whose prevalence variations are similar to the El Niño index sequence. Fig. 11b shows the execution time and number of result sets by different thresholds. In addition, we found in our mining results that the prevalence variations of PRECI-L(0.20), CO₂-L(0.21), Solar-H(0.25), NPP-L(0.26), PRECI-L & CO₂-L(0.21), NPP-L & CO₂-L(0.23), PRECI-L & NPP-L(0.27), PRECI-L & NPP-L & CO₂-L(0.29), etc. were very related with the El Niño index sequence, with dissimilarity values of around 0.2. For example, it is known that Australia experiences low precipitation during El Niño, which leads eventually to abnormal climate phenomena such as droughts and wide fires in the region [26]. In the second experiment, we used the prevalence sequence of low precipitation as the reference sequence and discovered event item sets whose prevalence variations are related to the prevalence of low precipitation. Some of these results are in line with real phenomena; for example, a decrease of precipitation is known to lead to a decrease in NPP and atmospheric carbon dioxide (CO₂). Fig. 11c shows the reference sequence of PRECI-L and the support sequence of NPP-L and CO₂-L event set. We can notice that the two sequences change similarly, with a dissimilarity value of 0.14.

A website click stream data set. In the second experiment with real data, we used a website click data set (BMS-Webview-1). Each transaction in this data set is a Web session consisting of all the product pages viewed in that session. Each product detail view is an item. The click data recorded in the data set covers nine weeks. When we used a week as a time interval, the number of time slots was nine. The total number of transactions was 59,602, the number of items was 497, the maximum transaction size was 267, and

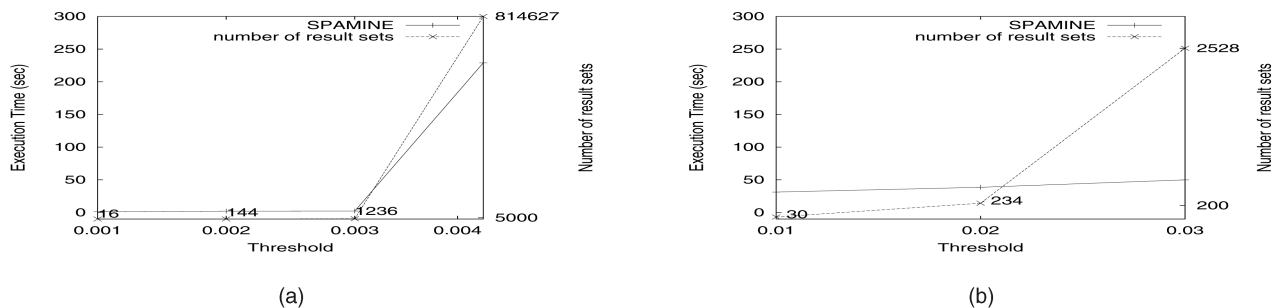


Fig. 12. Experimental results with other real data sets. (a) BMS-WebView-1. (b) CoverType.

the average transaction size was 2.5. The goal was to find web pages whose visits had similar frequency with a specific site over time. We excluded the sequential method in this experiment since it generated a huge number of candidate sets with this data set. Fig. 12a shows the execution time of SPAMINE and number of result sets by different thresholds. Even with the small increase of threshold, this data set generates many similar item sets with a randomly chosen reference, and the execution time is dramatically increased.

A *land cover type* data set. In the last experiment, we intended to show our SPAMINE algorithm can be applied to discover similar associated item sets from a data set that is distinctly divided by other attributes (e.g., land class type) instead of time. We used a land cover data set (CoverType), which contained seven class types, i.e., the data set was divided to seven distinct subsets. The number of original attributes was 54. Since some of the attributes were numeric variables, we converted them to nominal variables. The total number of items in this experiment was 64. The total number of transactions was 581,012, and the average transaction size was 12. As a reference sequence, we used the support sequence of a soil type item, i.e., “Leighcan.” We examined item sets whose support variations are similar with the reference sequence over different land class types. Fig. 12a shows the execution time and the number of result sets of SPAMINE with different thresholds. As an example from the output result, we can notice that the “Typic Cryaqualls complex” soil type in the “Comanche Peak Wilderness” area shows a similar prevalence population with the “Leighcan” soil type.

7 RELATED WORK

Although much work has been done on finding association patterns and similar time series, little attention has been paid to temporal association patterns that can discover similar variance groups over time. The closest related efforts have attempted to capture special temporal regulations of frequent association patterns such as cyclic association rule mining and calendar-based association rule mining [8], [10], [9], [30] in temporal association mining. Özden et al. [8] examined cyclic association rule mining, which detects periodically repetitive patterns of frequent item sets over time. Cyclic associations can be considered as item sets that occur in every cycle with no exception. The work of Özden et al. [8] was extended in Ramaswamy et al. [10] for relaxed matches. Li et al. [9] explored the problem of finding frequent item sets along with calendar-based patterns. The calendar-based patterns are defined with a calendar schema, e.g., (year, month, and

day). For example, (*, 10, 31) represents the set of time points each corresponding to the 31st day of October. However, real-life patterns are usually imperfect and may not demonstrate any regular periodicity. In the work of Li et al. [30], a temporal pattern is defined with the set of time points where the user expects discovered item sets to be frequent. In contrast, our temporal patterns are searched with a user defined numeric reference sequence and consider the prevalence similarities of all possible item sets, not only frequent item sets. Bettini et al. [31] finds frequent event patterns from time sequences using a user-specified skeleton. The user-specified skeleton is defined with a reference event and temporal constraints with time granularities. For example, the work can find events which frequently happen within two business days after a reference event, e.g., a rise in the stock price of IBM. The user-specified skeleton information may be used to generate a reference sequence of our concept. However, the sequence would be a type of binary sequence which has “1” values at time points within a time granularity from reference event instances. Hidber [32] introduced online association rule mining which gives the user the freedom to change the support threshold during the first scan of data set. Agrawal and Srikant [33] addressed the problem of monitoring the support and confidence of association rules. First, all association rules satisfying the minimum thresholds over time are mined and collected into a rule base. Then, interesting patterns are queried by specifying shape operators (e.g., ups and downs) in support or confidence over time. In contrast, we use numeric query sequences for our interesting patterns. There are many other studies in temporal data mining. Dong and Li [34] presented the problem of mining emerging patterns, which are item sets whose supports increase significantly from one data set to another. Liu et al. [35] studied the change of fundamental association rules between two time periods using support and confidence. Recent work has applied mining techniques in a data streaming context. The temporal frequency counting problem for a data stream environment was proposed by Teng et al. [36].

8 CONCLUSION AND FUTURE WORK

We formulated the problem of mining similarity-profiled temporal association patterns and proposed a novel algorithm to discover them. The proposed SPAMINE algorithm substantially reduced the search space by pruning candidate item sets using the lower bounding distance of the bounds of support sequences, and the monotonicity property of the upper lower bounding distance without compromising the

correctness and completeness of the mining results. Experimental results on synthetic and real data sets showed that the SPAMINE algorithm is computationally efficient and can produce meaningful results from real data. However, our pruning scheme effect depends on data distribution, dissimilarity threshold, and type of reference sequence.

In the future, we plan to explore different similarity models for our temporal patterns. The current similarity model using a \mathcal{L}_p norm-based similarity function is a little rigid in finding similar temporal patterns. It may be interesting to consider not only a relaxed similarity model to catch temporal patterns that show similar trends but also phase shifts in time. For example, the sale of items for clean-up such as chain saws and mops would increase after a storm rather than during the storm. We currently consider whole-sequence matching for the similar temporal patterns. Sub-sequence matching models may be more flexible for the patterns.

REFERENCES

[1] NOAA Economics, <http://www.ncdc.noaa.gov/oa/esb/?goal=climate&file=users/business/>, 2008.

[2] "After Katrina: Crisis Management, The Only Lifeline Was the Wal-Mart," *FORTUNE Magazine*, Oct. 2005.

[3] R. Agarwal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 1994.

[4] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," *Proc. ACM SIGMOD*, 2000.

[5] J. Han and Y. Fu, "Discovery of Multi-Level Association Rules from Large Databases," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 1995.

[6] J. Park, M. Chen, and P. Yu, "An Effective Hashing-Based Algorithm for Mining Association Rules," *Proc. ACM SIGMOD*, 1995.

[7] R. Srikant and R. Agrawal, "Mining Generalized Association Rules," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 1995.

[8] B. Ozden, S. Ramaswamy, and A. Silberschatz, "Cyclic Association Rules," *Proc. IEEE Int'l Conf. Data Eng. (ICDE)*, 1998.

[9] Y. Li, P. Ning, X.S. Wang, and S. Jajodia, "Discovering Calendar-Based Temporal Association Rules," *J. Data and Knowledge Eng.*, vol. 15, no. 2, 2003.

[10] S. Ramaswamy, S. Mahajan, and A. Silberschatz, "On the Discovery of Interesting Patterns in Association Rules," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 1998.

[11] Weather.com, <http://www.weather.com/aboutus/adsales/research.html>, 2008.

[12] NOAA, El Niño Page, <http://www.elnino.noaa.gov/>, 2008.

[13] D. Gunopulos and G. Das, "Time Series Similarity Measures and Time Series Indexing," *SIGMOD Record*, vol. 30, no. 2, 2001.

[14] C. Bettini, S. Jajodia, and X. Wang, *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Springer, 2000.

[15] D. Gunopulos and G. Das, "Time Series Similarity Measures," *Tutorial Notes of the ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, 2000.

[16] B. Yi and C. Faloutsos, "Fast Time Sequence Indexing for Arbitrary \mathcal{L}_p Norms," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 2000.

[17] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Database," *Proc. ACM SIGMOD*, 1993.

[18] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," *Proc. Int'l Conf. Foundations of Data Organization (FODO)*, 1993.

[19] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *J. Knowledge and Information Systems*, vol. 3, no. 3, 2001.

[20] R. Agrawal, K.I. Lin, H.S. Sawhney, and K. Shim, "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Database," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 1995.

[21] P. Tan, V. Kumar, and J. Srivastava, "Selecting the Right Interestingness Measure for Association Patterns," *Proc. ACM SIGKDD*, 2002.

[22] D. Berndt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," *Proc. AAAI Workshop on Knowledge Discovery in Databases*, 1994.

[23] E. Keogh and M. Pazzani, "Scaling Up Dynamic Time Warping for Data Mining Applications," *Proc. ACM SIGKDD*, 2000.

[24] N. Yazdani and Z. Ozsoyoglu, "Sequence Matching of Images," *Proc. Int'l Conf. Scientific and Statistical Database Management (SSDBM)*, 1996.

[25] T. Calders, "Deducing Bounds on the Frequency of Itemsets," *Proc. EDBT Workshop Database Techniques in Data Mining (DTDM)*, 2002.

[26] *Discovery of Changes from the Global Carbon Cycle and Climate System Using Data Mining*, <http://www.cs.umn.edu/old-ahpcrc/nasa-umn/>, 2008.

[27] Z. Zheng, R. Kohavi, and L. Mason, "Real World Performance of Association Rule Algorithms," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery in Databases*, 2001.

[28] UCI Repository of Machine Learning Databases, <http://www.ics.uci.edu/~mlearn/MLRepository.html>, 2008.

[29] G.H. Taylor, "Impacts of El Niño on Southern Oscillation on the Pacific Northwest," http://www.ocs.orst.edu/reports/enso_pnw.html, 2008.

[30] Y. Li, S. Zhu, X.S. Wang, and S. Jajodia, "Looking into the Seeds of Time: Discovering Temporal Patterns in Large Transaction Sets," *J. Information Sciences*, vol. 176, no. 8, 2006.

[31] C. Bettini, X. Wang, S. Jajodia, and J. Lin, "Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences," *IEEE Trans. Knowledge and Data Eng.*, vol. 10, no. 2, Mar.-Apr. 1998.

[32] C. Hidber, "Online Association Rule Mining," *Proc. ACM SIGMOD*, 1998.

[33] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. IEEE Int'l Conf. Data Eng. (ICDE)*, 1995.

[34] G. Dong and J. Li, "Efficient Mining of Emerging Patterns: Discovering Trends and Differences," *Proc. ACM SIGKDD*, 1999.

[35] B. Liu, W. Hsu, and Y. Ma, "Discovering the Set of Fundamental Rule Change," *Proc. ACM SIGKDD*, 2001.

[36] W. Teng, M. Chen, and P. Yu, "A Regression-Based Temporal Pattern Mining Scheme for Data Streams," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 2003.



Jin Soung Yoo received the BS degrees in statistics and computer science and engineering from Korea University, Korea, in 1990 and 1992 and the MS and PhD degrees in computer science from the University of Minnesota, Minneapolis, in 2006 and 2007. She worked as a software engineer for Samsung SDS, Inc. Seoul, Korea, from 1992 to 1999. She is an assistant professor in the Department of Computer Science, Indiana University-Purdue University, Fort Wayne. Her research interests include databases, data mining, spatial/temporal/spatio-temporal data mining/data management, network/graph mining, etc. She is a member of the IEEE and the IEEE Computer Society.



Shashi Shekhar received the BTech degree in computer science from the Indian Institute of Technology, Kanpur, India, in 1985, and the MS degree in business administration and the PhD degree in computer science from the University of California, Berkeley, California, in 1989. He is a McKnight Distinguished University professor at the University of Minnesota, Minneapolis, Minnesota. His research interests include spatial databases, spatial data mining, geographic information systems (GIS), and intelligent transportation systems. He has served on the editorial boards of the *IEEE Transactions on Knowledge and Data Engineering* and the *IEEE-CS Computer Science and Engineering Practice Board* and is serving as a coeditor-in-chief of the *Geoinformatica* journal. He is a fellow of the IEEE, the IEEE Computer Society and a member of the ACM.