Knowledge-Based Systems 24 (2011) 1–9

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# Mining weighted sequential patterns in a sequence database with a time-interval weight

Joong Hyuk Chang *

*Dept. of Computer and Information Technology, Daegu University, Naeri Jillyang Gyeongsan, Gyeongbuk 712-714, Republic of Korea*

ARTICLE INFO

*Article history:*
Received 19 November 2009
Received in revised form 10 January 2010
Accepted 10 March 2010
Available online 18 March 2010

*Keywords:*
Weighted sequential pattern
Time-interval weight
TiWS support
TiWS pattern
Time-interval sequence database
Sequential pattern mining

ABSTRACT

Sequential pattern mining, including weighted sequential pattern mining, has been attracting much attention since it is one of the essential data mining tasks with broad applications. The weighted sequential pattern mining aims to find more interesting sequential patterns, considering the different significance of each data element in a sequence database. In the conventional weighted sequential pattern mining, usually pre-assigned weights of data elements are used to get the importance, which are derived from their quantitative information and their importance in real world application domains. In general sequential pattern mining, the generation order of data elements is considered to find sequential patterns. However, their generation times and time-intervals are also important in real world application domains. Therefore, time-interval information of data elements can be helpful in finding more interesting sequential patterns. This paper presents a new framework for finding time-interval weighted sequential (TiWS) patterns in a sequence database and time-interval weighted support (TiW-support) to find the TiWS patterns. In addition, a new method of mining TiWS patterns in a sequence database is also presented. In the proposed framework of TiWS pattern mining, the weight of each sequence in a sequence database is first obtained from the time-intervals of elements in the sequence, and subsequently TiWS patterns are found considering the weight. A series of evaluation results shows that TIWS pattern mining is efficient and helpful in finding more interesting sequential patterns.



## 1. Introduction

Sequential pattern mining aims to discover interesting sequential patterns in a sequence database, and it is one of the essential data mining tasks widely used in various application fields such as Web access pattern analysis, customer purchase pattern analysis, DNA sequence analysis, etc. In general, sequential pattern mining finds all frequent sequential patterns for a given sequence database and a frequency threshold whose occurrence frequency is no less than the threshold. In many of the previous researches on sequential pattern mining problems, sequential patterns and items in a sequential pattern have been considered uniformly. However, they have a different importance in real world applications, and thus more interesting sequential patterns can be found when the different importance is considered in sequential pattern mining. Based on this observation, weighted sequential pattern mining [13,25] has recently been proposed and actively studied. In contrast to general sequential pattern mining based on simple support counting, the weight of information is used in finding interesting sequential patterns in weighted sequential pattern mining.

For a sequence or a sequential pattern, not only the generation order of data elements but also their generation times and time-intervals are important. Therefore, for sequential pattern mining, the time-interval information of data elements can help to get more valuable sequential patterns. In [4] and [5], several sequential pattern mining algorithms have been presented which consider a time-interval between two successive items in a sequential pattern. However, they simply consider a time-interval between two successive data elements as an item, and thus they are unable to get weighed sequential patterns considering different weights of sequences in a sequence database. If the importance of sequences in a sequence database is differentiated based on the time-intervals in the sequences, more interesting sequential patterns can be found.

For example, in a sequence database gathered from a computer store, let us say the following two sequences were found:

[*Customer_A*] Having bought a laser printer, a customer returns to buy a scanner after 1 month and then a CD burner after 1 month.
[*Customer_B*] Having bought a laser printer, a customer returns to buy a scanner after 6 months and then a CD burner after 3 months.

* Tel.: +82 53 850 6588; fax: +82 53 850 6589.
*E-mail address:* jhchang@daegu.ac.kr

0950-7051/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2010.03.003

The sequences consist of the same items and the orders of items are the same in both customers, but the time-intervals between the items are different. Therefore, they may appear to be the same if only the order of items is considered, but to be totally different if the time-intervals are itemized. However, it is better advised and more close to a real world situation to consider them as having the same sequence with a different importance, since they bought the same items but the time-intervals between successive purchases are different for each other. In the above example, the sequence by a *Customer_A* can be considered more important than the sequence by a *Customer_B*, since the former has relatively smaller time-intervals than the latter. Accordingly, for a sequence in a sequence database, its importance, i.e. its weight, can be computed from the time-intervals in the sequence.

This paper proposes a new framework for finding weighted sequential patterns with a time-interval weight, i.e., time-interval weighted sequential (TiWS) patterns. In addition, based on the proposed framework, a TiWS pattern mining method is developed, and its efficiency and usefulness are verified through a series of experiments. The contributions of this paper are summarized as follows: First this paper presents a technique to find the weight of a sequence, which indicates its importance in a sequence database. The weight of a sequence is derived from the time-intervals of data elements in the sequence based on the assumption that a sequence with small time-intervals is more valuable. A technique to get the weight of a time-interval between two data elements in a sequence is also presented, which is used to get the weight of a sequence. Subsequently, a new framework for finding TiWS patterns in a sequence database is presented, and a new measure used to find TiWS patterns is also introduced. Finally, a TiWS pattern mining method is proposed.

The rest of this paper is organized as follows: Section 2 gives a brief summary of related work, and a problem definition is given in Section 3. In Section 4, a new framework for finding TiWS patterns is described. In addition, a TiWS pattern mining method, which is based on the new framework and the conventional sequential pattern mining algorithm, is presented in Section 5. Section 6 summarizes a series of experimental results, and finally conclusions and future researches are described in Section 7.

## 2. Related work

Many studies have contributed to efficient mining of sequential patterns, such as general sequential pattern mining [1,8,12,16,27], closed and maximal sequential pattern mining [14,19,20,23], constraint-based sequential pattern mining [3,9,15], approximate sequential pattern mining [10,11], sequential pattern mining in multiple data sources [11], sequence mining in biological data [7,21], incremental mining of sequential patterns [6], and sequential pattern mining in noisy data [24].

Among the algorithms for general sequential pattern mining, SPADE [27] and PrefixSpan [16] are more efficient than others in terms of processing time. SPADE is one of the vertical-format-based algorithms and uses equivalence classes in the mining process. PrefixSpan is one of the pattern-growth approaches. It recursively projects a sequence database into a set of smaller projected sequence databases and grows sequential patterns in each projected database by exploring only the locally frequent fragments. To improve the usefulness of mining results in real world applications, weighted pattern mining has been studied in association rule mining [2,17,18,22,26] and sequential pattern mining [13,25]. Most of the weighted pattern mining algorithms usually require pre-assigned weights, and the weights are generally derived from the quantitative information and the importance of items in a real world application. For example, in a retail database, the quantity and price of an item being sold is considered as its weight.

Time-interval and gap information between items in a sequence have been used to find interesting sequential patterns, and there have been several studies on mining sequential patterns considering them. Pei et al. [15] and Ji et al. [9] have proposed constrained sequential pattern mining methods, which use time-interval and gap information as a constraint. In these methods, they are used only to confine the mining result of sequential patterns, but they do not give a mining result of weighted sequential patterns. Chen et al. have proposed sequential pattern mining algorithms, considering time-interval information between two successive items in a sequence [4,5], whose target database is a sequence database with single items and their corresponding time-intervals. However, the algorithms just consider time-interval information between two successive items as an item, so that they cannot support to get weighted sequential patterns based on different weights of sequences.

## 3. Problem definition

In general sequential pattern mining, only the order of data elements is considered, and so a sequence is represented as an ordered list of data elements without time stamp. However, for mining sequential patterns with time-interval, generally, the sequence of a sequence database has its corresponding time stamp [4,5].

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of all *items*. A *sequence* $S = \langle s_1, s_2, \ldots, s_l \rangle$ is an ordered list of itemsets, where $s_j$ $(1 \leqslant j \leqslant l)$ is an itemset, and its *time stamp list* $TS(S) = \langle t_1, t_2, \ldots, t_l \rangle$ is an ordered list of corresponding time stamps of the itemsets, which stand for the time when they occur, where $t_j$ $(1 \leqslant j \leqslant l)$ is the time stamp of $s_j$ and $t_{j-1} \leqslant t_j$ $(2 \leqslant j \leqslant l)$. An *itemset*, which is also called an *element* of a sequence, $s_j$ is a subset of $I$ and denoted as $(x_1, x_2, \ldots, x_m)$, where $x_k$ $(1 \leqslant k \leqslant m)$ is an item. For brevity, brackets of an itemset are omitted if an itemset has only one item. An item can occur at most once in an itemset of a sequence, but it can occur multiple times in different itemsets of a sequence. The number of items in a sequence is called the length of the sequence, and a sequence with $l$ items is called an $l$-sequence. A sequence database is a set of tuples $\langle sid, S \rangle$ where *sid* is a sequence identifier and $S$ is a sequence. For a sequence database *SDB*, the total number of tuples in *SDB* is called its size and denoted by $|SDB|$. Table 1 shows an example sequence database. It consists of four sequences, and their *sid*s are 10, 20, 30, and 40, respectively.

In practice, a sequence $A = \langle a_1, a_2, \ldots, a_n \rangle$ is called a subsequence of another sequence $B = \langle b_1, b_2, \ldots, b_m \rangle$ and $B$ is a super-sequence of $A$, if there exist integers $1 \leqslant j_1 < j_2 < \cdots < j_n \leqslant m$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \ldots$, and $a_n \subseteq b_{j_n}$. For a sequence database, a tuple $\langle sid, S \rangle$ is said to contain a sequence $A$ if $A$ is a subsequence of $S$. The count of a sequence $A$ in a sequence database is the number of tuples in the database containing $A$, and the support of $A$ is the ratio of the count over the size of the sequence database. The time-interval weighted sequential pattern mining proposed in this paper considers the weight of a sequence, and uses it to find the count of a sequential pattern and the size of a sequence database. In time-

**Table 1**
A sequence database with a time stamp list.

| *sid* | Sequence | Time stamp list |
|---|---|---|
| 10 | $\langle a, (abc), (ac), d \rangle$ | $\langle 0, 1, 2, 3 \rangle$ |
| 20 | $\langle (ad), c, (bc), (ae) \rangle$ | $\langle 1, 2, 3, 4 \rangle$ |
| 30 | $\langle (ad), (bc), (df) \rangle$ | $\langle 1, 3, 5 \rangle$ |
| 40 | $\langle a, (abc), d \rangle$ | $\langle 2, 3, 4 \rangle$ |

interval weighted sequential pattern mining, the weighted count of a sequence $A$ in a sequence database $SDB$ is the sum of weights of sequences in $SDB$ containing $A$. Likewise, its weighted support is the ratio of its weighted count over the sum of the weights of all sequences in $SDB$.

Given a support threshold $minSupport$ $(0 < minSupport \leqslant 1)$, a sequence $A$ is called a time-interval weighted sequential pattern in a sequence database $SDB$ if the weighted support of $A$ is no less than the support threshold. Accordingly, for a given sequence database and a support threshold, the problem of time-interval weighted sequential pattern mining is to find the complete set of all time-interval weighted sequential patterns whose weighted supports are no less than the threshold.

## 4. Time-interval weighted sequential (TiWS) patterns

In sequential pattern mining, not only the generation order of elements but also their generation times and time-intervals among them are important in real world application domains. Therefore, for a sequence or a sequential pattern, the time-interval information in the sequence can be a useful measure to decide its significance. That is, in mining sequential patterns, a sequence with small time-intervals among its elements can be considered more important than that with large time-intervals. Based on this observation, an interesting mining result of time-interval weighted sequential patterns is found for a sequence database by considering the weight of each sequence obtained from the time-intervals in the sequence. This section presents a new technique to get the time-interval weight of a sequence for a sequence database as well as a new framework for finding time-interval weighted sequential patterns.

### 4.1. A time-interval between a pair of itemsets

A sequence in the sequence database consists of itemsets and their corresponding time stamps. This section describes how the time-intervals of a sequence are found from the time stamps of itemsets in the sequence, which are used to get the time-interval weight of the sequence. For a sequence in a sequence database where the sequence consists of $n$ itemsets, there exist $\frac{n\times(n-1)}{2}$ pairs of itemsets in the sequence, and the time-interval between the itemsets of each pair is defined as in Definition 1. For a sequence whose $sid$ is 10 as in Table 1, there can be $\frac{4\times(4-1)}{2}$ pairs of itemsets, and the time-interval of each pair is found as shown in Table 2.

**Definition 1** (*Time-interval*). For a sequence $S = \langle s_1, s_2, \ldots, s_l\rangle$ and its time stamp list $TS(S) = \langle t_1, t_2, \ldots, t_l\rangle$, the time-interval between two itemsets $s_i$ and $s_j$ $(1 \leqslant i < j \leqslant l)$ in the sequence, i.e., $TI_{ij}$, is defined as follows:

$$TI_{ij} = t_j - t_i.$$

### 4.2. Time-interval weight of a pair of itemsets

The time-interval between a pair of itemsets is a positive value with no limitation. Therefore, to fairly enumerate the time-intervals of different pairs of itemsets in a sequence database, they need to be normalized. For this purpose, the time-interval weight of the pair is found for each pair of itemsets in a sequence based on its time-interval, and defined as in Definition 2.

**Definition 2** (*Time-interval weight*). Let $u(u > 0)$ be the size of unit time and $\delta(0 < \delta < 1)$ be a base number to determine the amount of weight reduction per unit time $u$, for a sequence $S = \langle s_1, s_2, \ldots, s_l\rangle$ and its time stamp list $TS(S) = \langle t_1, t_2, \ldots, t_l\rangle$, the time-interval weight of a time-interval $TI_{ij}$ between two itemsets $s_i$ and $s_j$ $(1 \leqslant i < j \leqslant l)$, i.e., $w_g(TI_{ij})$, $w_l(TI_{ij})$, and $w_c(TI_{ij})$ are defined, respectively, as follows:

(i) *General scale weighting:* $w_g(TI_{ij}) = \delta^{\frac{TI_{ij}}{u}} = \delta^{\frac{t_j - t_i}{u}}$ *[WF_1].*

(ii) *Log scale weighting:* $w_l(TI_{ij}) = \delta^{\log_2\left(1+\frac{TI_{ij}}{u}\right)} = \delta^{\log_2\left(1+\frac{t_j-t_i}{u}\right)}$ *[WF_2].*

(iii) *General scale weighting with a ceiling:* $w_c(TI_{ij}) = \delta^{\left\lceil\frac{TI_{ij}}{u}\right\rceil} = \delta^{\left\lceil\frac{t_j-t_i}{u}\right\rceil}$ *[WF_3].*

**Table 2**
Possible pairs of itemsets.

| 1st Itemset | 2nd Itemset | Time-interval |
|---|---|---|
| $a$ | $(abc)$ | 1 |
| $a$ | $(ac)$ | 2 |
| $a$ | $d$ | 3 |
| $(abc)$ | $(ac)$ | 1 |
| $(abc)$ | $d$ | 2 |
| $(ac)$ | $d$ | 1 |

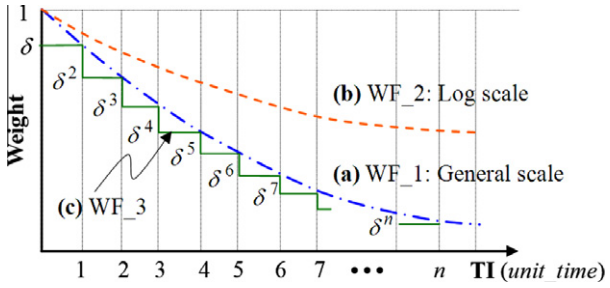


**Fig. 1.** Time-interval weighting functions.

The smaller the values of $\delta$ and $u$ are, the more sensitive a time-interval weight is to the increase of a time-interval. Among the three weighting functions shown in Definition 2, when the first weighting function (i.e., WF_1) is applied, the time-interval weight of a pair of itemsets is affected by its time-interval in general scale. That is, the weight decreases in general scale as the time-interval increases as in the line (a) in Fig. 1. In the case of the second weighting function (i.e., WF_2), the weight decreases in log scale. The third weighting function (i.e., WF_3) is similar to WF_1, but all pairs of itemsets whose time-intervals are in the same time unit have the same time-interval weight in WF_3. Lines (b) and (c) show the time-interval weights of WF_2 and WF_3, respectively.

### 4.3. Time-interval weight of a sequence

The time-interval weight of a sequence is computed from the time-intervals of pairs of itemsets in the sequence. In this process, for two different pairs of itemsets in the same sequence, a contribution of each pair to the sequence may be different even though they have the same time-interval weight. This is because the diversity of the itemsets in terms of the length of an itemset. That is, among the itemsets in a sequence, a large-sized itemset may contribute more to the sequence than a small-sized one. Based on this observation, for a pair of itemsets in a sequence, its strength of contribution to the sequence is defined as in Definition 3.

**Definition 3** (*Strength of a pair of itemsets*). For a sequence $S = \langle s_1, s_2, \ldots, s_l\rangle$ and its time stamp list $TS(S) = \langle t_1, t_2, \ldots, t_l\rangle$, the strength of a pair of two itemsets $s_i$ and $s_j$ $(1 \leqslant i < j \leqslant l)$ in the sequence, i.e., $ST_{ij}$, is defined as follows:

$ST_{ij} = length(s_i) \times length(s_j)$, where $length(s_i)$ denotes the number of items in $s_i$

Subsequently, for a sequence $S = \langle s_1, s_2, \ldots, s_l \rangle$ and its time stamp list $TS(S) = \langle t_1, t_2, \ldots, t_l \rangle$, the time-interval weight of the sequence is found as in Definition 4 considering the time-intervals in the sequence and their strengths to the sequence.

**Definition 4** (*Time-interval weight of a sequence*). For a sequence $S = \langle s_1, s_2, \ldots, s_l \rangle$ and its time stamp list $TS(S) = \langle t_1, t_2, \ldots, t_l \rangle$, the time-interval weight of the sequence, i.e., $W(S)$, is defined as follows:

$$W(S) = \begin{cases} \frac{1}{N}\sum_{i=1}^{l-1}\sum_{j=i+1}^{l}\{w(TI_{ij}) \times ST_{ij}\}, \text{ where } N = \sum_{i=1}^{l-1}\sum_{j=i+1}^{l} ST_{ij} \\ \quad \text{and } w(TI_{ij}) \text{ denotes a weighting function} & (l \geqslant 2) \\ 1 & (l = 1) \end{cases}$$

For a sequence whose *sid* is 10 in Table 1, the strength of each pair is found as follows: $ST_{12} = 1 \times 3$, $ST_{13} = 1 \times 2$, $ST_{14} = 1 \times 1$, $ST_{23} = 3 \times 2$, $ST_{24} = 3 \times 1$, and $ST_{34} = 2 \times 1$. Subsequently, the total sum of the strengths, $N$, and the time-interval weight of the sequence, $W(S_{sid=10})$, is found as follows:

$$\begin{aligned} N &= ST_{12} + ST_{13} + ST_{14} + ST_{23} + ST_{24} + ST_{34} \\ &= 3 + 2 + 1 + 6 + 3 + 2 = 17 \end{aligned}$$

$$\begin{aligned} W(S_{sid=10}) &= \frac{1}{N} \times \{w(TI_{12}) \times ST_{12} + w(TI_{13}) \times ST_{13} + w(TI_{14}) \times ST_{14} \\ &\quad + w(TI_{23}) \times ST_{23} + w(TI_{24}) \times ST_{24} + w(TI_{34}) \times ST_{34}\} \\ &= \frac{1}{17} \times \{w(1) \times 3 + w(2) \times 2 + w(3) \times 1 + w(1) \times 6 \\ &\quad + w(2) \times 3 + w(1) \times 2\} \\ &= \frac{\{w(1) \times 11 + w(2) \times 5 + w(3) \times 1\}}{17} \end{aligned}$$

Therefore, when the general scale weighting function $w_g(TI) = \delta^{TI/u}$ with $\delta = 0.9$ and $u = 1$ is applied, the value of $W(S_{sid=10})$ is found as 0.863. Under the same condition, the weights of the other sequences in Table 1 are found as shown in Table 3. In addition, the appearance of a sequence $(ad)(bc)$ in $S_{sid=20}$ is considered more important than that in $S_{sid=30}$ because the time-interval weight of $S_{sid=20}$ is greater than that of $S_{sid=30}$.

### 4.4. Time-interval weighted support: TiW-support

Usually, sequential pattern evaluation by support is based on simple counting in the classical sequential pattern mining. In this section, a new term of *TiW-support* (*T*ime-interval *W*eighted *support*) of a sequence is introduced, and an evaluation process of time-interval weighted sequential patterns in a sequence database is also introduced, which uses TiW-support. Sections 4.1–4.3 have presented the way to get a time-interval weight of each sequence in a sequence database. Using the time-interval weight, the TiW-support of a sequence in a sequence database is defined as in Definition 5.

**Definition 5** (*TiW-support of a sequence*). For a sequence database *SDB*, the TiW-support of a sequence $X$ in the sequence database, i.e., *TiW-Supp*($X$), is defined as follows:

$$TiW\text{-}Supp(X) = \frac{\sum_{S:(X \subseteq S) \wedge (S \in SDB)} W(S)}{\sum_{S:S \in SDB} W(S)}.$$

Accordingly, the time-interval weighted sequential (TiWS) pattern is defined as in Definition 6.

**Definition 6** (*Time-interval weighted sequential pattern: TiWS pattern*). Given a support threshold *minSupport* ($0 < minSupport \leqslant 1$), a sequence $X$ is a time-interval weighted sequential pattern if $TiW\text{-}Supp(X)$ is no less than the threshold, i.e., $TiW - Supp(X) \geqslant minSupport$.

Table 4 shows the supports of several sequences derived from the sequence database in Table 1 when the general scale weighting function $w_g(TI) = \delta^{TI/u}$ with $\delta = 0.9$ and $u = 1$ is applied. Three sequences $a(abc)d$, $(ad)$, and $(ad)(bc)$ have the same support in simple support counting, but the TiW-support of $a(abc)d$ is greater than those of the others because the sequence $a(abc)d$ appears in the sequences whose time-interval weights are relatively large. In addition, for two sequences $a(abc)d$ and $(ad)(bc)$, if the support threshold is set to 0.5, all of them can be sequential patterns in the classical sequential pattern mining, but the sequence $(ad)(bc)$ cannot be a TiWS pattern since its TiW-support is less than the threshold.

Like a term of simple support to find general sequential patterns, the term of TiW-support to find TiWS patterns also has an anti-monotone property.

**Property 1** (*Anti-monotone property of TiW-support). Let A and B be sequences in a sequence database SDB and B is a super-sequence of A, i.e., $A \subseteq B$. The TiW-support of the sequence A is found by Definition 5 as follows:*

$$TiW\text{-}Supp(A) = \frac{\sum_{S:(A \subseteq S) \wedge (S \in SDB)} W(S)}{\sum_{S:S \in SDB} W(S)} \quad (1)$$

*In* Eq. (1), *since $A \subseteq B$ is satisfied, $\sum_{S:(A \subseteq S) \wedge (S \in SDB)} W(S)$ is always greater than or equal to $\sum_{S:(B \subseteq S) \wedge (S \in SDB)} W(S)$. Accordingly, the following inequality is satisfied.*

$$\begin{aligned} TiW\text{-}Supp(A) &= \frac{\sum_{S:(A \subseteq S) \wedge (S \in SDB)} W(S)}{\sum_{S:S \in SDB} W(S)} \geqslant \frac{\sum_{S:(B \subseteq S) \wedge (S \in SDB)} W(S)}{\sum_{S:S \in SDB} W(S)} \\ &= TiW\text{-}Supp(B) \end{aligned}$$

*Therefore, if TiW-Supp(A) is less than a predefined support threshold, TiW-Supp(B) is also less than the threshold. Consequently, TiW-support has the anti-monotone property.*

**Table 3**
A time-interval weight of a sequence.

| *sid* | Sequence weight |
|---|---|
| 10 | $\{w(1) \times 11 + w(2) \times 5 + w(3) \times 1\}/17 = 0.863$ |
| 20 | $\{w(1) \times 8 + w(2) \times 6 + w(3) \times 4\}/16 = 0.832$ |
| 30 | $\{w(2) \times 8 + w(4) \times 4\}/12 = 0.759$ |
| 40 | $\{w(1) \times 6 + w(2) \times 1\}/7 = 0.887$ |

($w_g(TI) = \delta^{TI/u}$, $\delta = 0.9$, and $u = 1$).

**Table 4**
Change of supports (Simple support vs. TiW-support).

| Sequences | Simple support | TiW-support |
|---|---|---|
| $a(bc)$ | 1.000 | 1.000 |
| $aa$ | 0.750 | 0.773 |
| $a(abc)d$ | 0.500 | 0.524 |
| $(ad)$ | 0.500 | 0.476 |
| $(ad)(bc)$ | 0.500 | 0.476 |

Similarly in the classical sequential pattern mining, an anti-monotone property of TiW-support can be used to prune the exponential search space to find TiWS patterns in a large sequence database.

## 5. Mining TiWS patterns in a large sequence database

In a framework for finding TiWS patterns presented in this paper, the time-interval weight of each sequence in a sequence database is first obtained from the time-intervals of elements in the sequence. Subsequently, the TiW-support of each sequential pattern is found based on the weight, and a set of TiWS patterns is found considering the TiW-support. The framework can be easily implemented using the conventional sequential pattern mining methods. This section presents a new *psTiWS* method, which is a mining method of TiWS patterns in a sequence database. It is based on the PrefixSpan [16], therefore its overall process is similar to that of the PrefixSpan. However, in the *psTiWS* method, the time-interval weight of each sequence in a sequence database is obtained in the first scan of the sequence database, and it is used to find TiWS patterns. Details are presented in Fig. 2.

For a sequence database, a memory space to store the time-interval weight of each sequence is required in the proposed method, so that the memory usage for performing the proposed method may increase a little compared with the PrefixSpan, but it is a negligible quantity. In terms of the processing time, the proposed method takes a time to get the time-interval of each sequence, but it is also a negligible quantity. In TiWS pattern mining for a sequence database, sequential patterns with large time-intervals

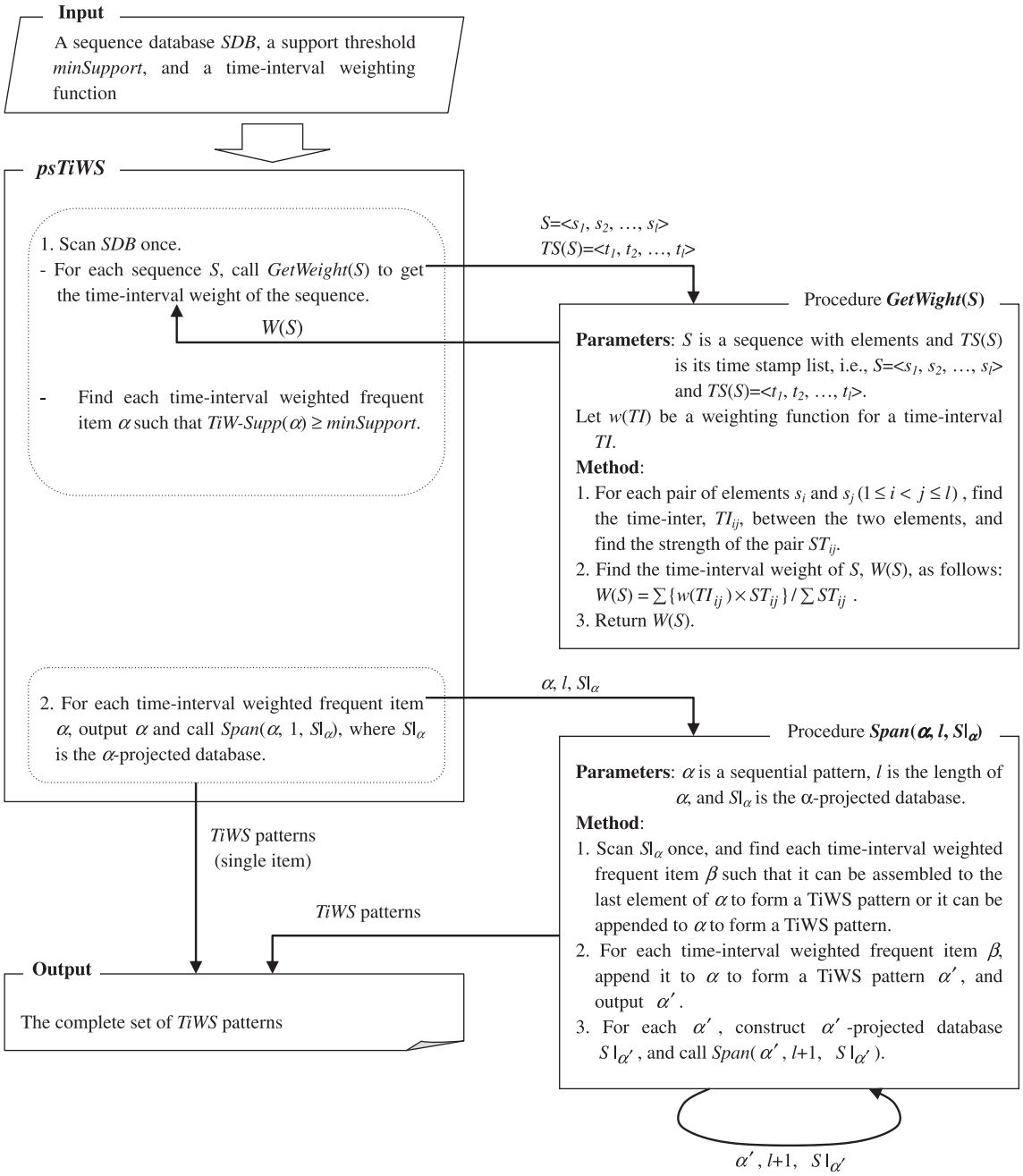


**Fig. 2.** psTiWS method: A method of mining TiWS patterns.

may be excluded from its mining result, so that the number of scans for the sequence database or its subset can be reduced. As a result, the processing time to get the mining result can also be reduced.

## 6. Experimental results

To verify the efficiency of a new framework for mining TiWS patterns and the *psTiWS* method, several experimental results on synthetic data sets and a real data set are presented in this section. To generate synthetic data sets, the IBM data set generator [1] was used, and their names were given following the conventions in [1] (see Table 5). The data sets generated by the IBM data generator do not have any time stamp. Therefore, for a synthetic data set to be used in mining TiWS patterns, a corresponding time stamp should be assigned to each data element in the data set. For this purpose, an approach using a probability distribution function or that using a randomization function can be considered, but these approaches almost never affect the performance of the new framework of TiWS pattern mining and the proposed *psTiWS* method. In addition, for a sequence in a sequence database, only its elements have their corresponding generation time since the items in the same data element are considered to be generated at the same time.

For synthetic data sets used in this paper, a randomization function is used to assign the corresponding generation times of elements in a sequence, and the difference of generation time between two successive elements in a sequence is in the range of 0–1000 ms for all synthetic data sets except the data set *D20C7T7S7I7-AB*. The data set *D20C7T7S7I7-AB* is composed of two consecutive subparts *part_A* and *part_B*. *Part_A* is a set of sequences generated by a set of items *set_A*, and *part_B* is a set of sequences generated by a set of items *set_B*. The two subparts are generated by the same method described in [1], but there is no common item between *set_A* and *set_B*. The time-interval between two successive itemsets in a sequence is in the range of 0–1000 ms in *part_A*, while it is in the range of 1000–2000 ms in *part_B*. That is, the sequences in *part_B* have relatively larger time-intervals than those in *part_A*. The data set *WebSEQ* is a real data set used to verify the efficiency of the proposed method in a real world application domain, and details of the data are described in the experimental result using the data set.

**Table 5**
Data sets.

| Data set | # of items |
|---|---|
| *D10C7T7S7I7* | 1000 |
| *D20C7T7S7I7-AB* | 1000 + 1000 |
| *D20C10T2.5S4I2.5, D40C10T2.5S4I2.5 D60C10T2.5S4I2.5, D80C10T2.5S4I2.5 D100C10T2.5S4I2.5* | 1000 |
| *WebSEQ* | 436 |

*D*, number of customers in the sequence data set (in K).
*C*, average number of transactions per customer.
*T*, average number of items per transaction.
*S*, average length of maximal sequences.
*I*, average length of transactions within the maximal sequences.
Details can be found in [1].

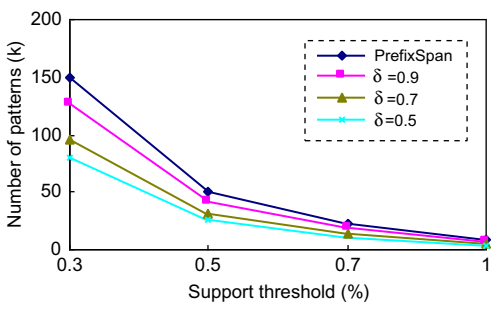


**Fig. 3.** Number of sequential patterns in function of $\delta$ ($u = 500$ ms).

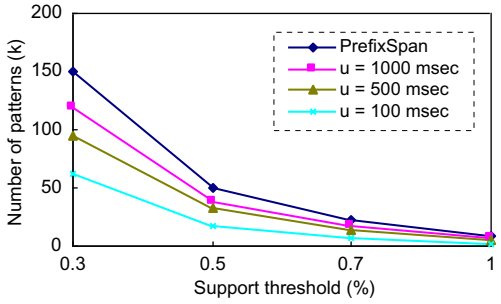


**Fig. 4.** Number of sequential patterns in function of $u$ ($\delta = 0.7$).

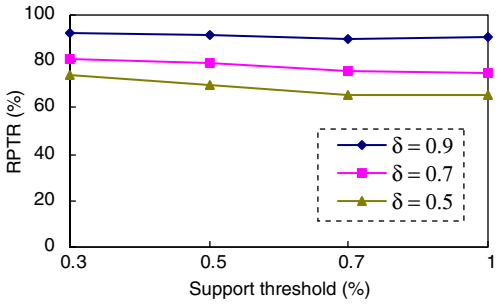


**Fig. 5.** Relative processing time ($u$ = 500 ms).

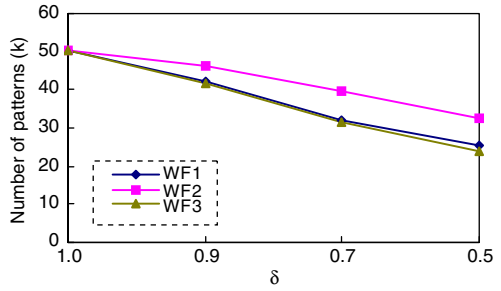


**Fig. 6.** Comparison of weighting functions ($u$ = 500 ms).

All experiments were performed on a 2.8 GHz Pentium machine with 1 GB of main memory running on Linux, and all programs were implemented in C.

Figs. 3–6 and Table 6 show the basic performance of the *psTiWS* method based on the data set *D10C7T7S7I7*. In this experiment, a support threshold is set to 0.005. This is because it is a proper value for getting sufficient number of sequential patterns to verify the effectiveness of the proposed framework clearly and fairly. Fig. 3 and Table 6 show the number of TiWS patterns in function of $\delta$. As shown in the results, the number of TiWS patterns decrease as the value of $\delta$ becomes smaller. That is, among the sequential patterns found in mining sequential patterns based on simple support counting, several sequential patterns whose time-intervals are relatively large are not found in a resulting set of TiWS patterns. Fig. 4 shows the number of TiWS patterns in function of $u$. Similarly in the experimental result w.r.t. $\delta$, the number of TiWS patterns decrease as the value of $u$ becomes smaller. These are

**Table 6**
Distribution of sequential patterns on data set *D10C7T7S7I7* ($u$ = 500 ms).

| $\delta$ | *L1* | *L2* | *L3* | *L4* | *L5* | *L6* | *L7* | *L8* | *L9* | *L10* | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PrefixSpan | 885 | 48,957 | 176 | 47 | 18 | 3 | 0 | 0 | 0 | 0 | 50,086 |
| 0.9 | 880 | 40,950 | 85 | 31 | 12 | 2 | 0 | 0 | 0 | 0 | 41,960 |
| 0.7 | 872 | 31,025 | 37 | 16 | 6 | 1 | 0 | 0 | 0 | 0 | 31,957 |
| 0.5 | 868 | 24,437 | 27 | 15 | 6 | 1 | 0 | 0 | 0 | 0 | 25,354 |

because a time-interval weight is more sensitive to the increase of a time-interval when the values of $\delta$ or $u$ are set to be small.

To evaluate the performance of the *psTiWS* method in terms of processing time, its relative processing time compared with the PrefixSpan is verified. For a sequence database, when $PT_{psTiWS}$ denotes the processing time to get a complete set of sequential patterns of the sequence database by the *psTiWS* method while $PT_{PrefixSpan}$ denotes the processing time by the *PrefixSpan* [16], the *relative processing time rate*(*RPTR*) of the *psTiWS* method for the sequence database is the ratio of $PT_{psTiWS}$ over $PT_{PrefixSpan}$. The detailed evaluation result on an actual processing time is not shown in this paper since it can be found in the evaluation results of the PrefixSpan. Fig. 5 shows the RPTR of the *psTiWS* method in function of $\delta$. Since the sequential patterns with large time-intervals are excluded from the mining result of the *psTiWS* method as described in Section 5, the number of scans for a target sequence database or its subset is reduced. As a result, for a sequence database, the processing time of the *psTiWS* method to get its mining result is less than that of the PrefixSpan, i.e., the RPTR of the psTiWS method is less than 100%. The smaller the value of $\delta$ is, the more the RPTR of the *psTiWS* decreases.

Fig. 6 shows the changes in the number of TiWS patterns for three weighting functions used in this paper. In general, the reduction rate of a time-interval weight according to the increase of a time-interval is relatively large in general scale weighting, i.e., WF_1, compared to log scale weighting, i.e., WF_2. This means that, when the general scale weighting function is applied, there may be a greater number of sequential patterns considered as less interesting because of larger time-intervals, compared to the case when the log scale weighting function is applied. Consequently, the number of TiWS patterns in the case of general scale weighting is less than that in the case of log scale weighting as shown in this figure. However, the result in the case of general scale weighting with a ceiling, i.e., WF_3, where all time-intervals in the same time unit have the same time-interval weight, is almost the same as the result in the case where the general scale weighting function is used.

To verify the efficiency of TiWS pattern mining on a data set with large time-intervals, the data set *D20C7T7S7I7-AB* is experimented. In this experiment, a support threshold and the value of $u$ were set to 0.005 and 1000 ms, respectively. Fig. 7 shows the changes in the number of sequential patterns on the data set *D20C7T7S7I7-AB*. As aforementioned, its subparts were generated by the same method in the same condition except the difference of a generation time between two successive elements in a sequence. Therefore, when the time-interval is not considered in mining sequential patterns, i.e., when the value of $\delta$ is set to 1.0, the number of sequential patterns derived from *part_A* and that derived from *part_B* are the same in the mining result on the data set *D20C7T7S7I7-AB* composed of the two subparts. However, when a framework of TiWS pattern mining is applied, the number of sequential patterns derived from the subpart *part_A* increases compared with the case without the framework, while the number of sequential patterns derived from the subpart *part_B* decreases. This difference is attributed to the fact that the sequence in *part_A* has a less time-interval than that in *part_B*, so that the time-interval weight of the sequence in *part_A* is greater than that in *part_B*. In other words, the sequence in *part_A* is considered as relatively more important. Consequently, more interesting sequential patterns that appear in the sequences with small time-intervals can be effectively found by using a new framework for finding TiWS patterns using TiW-support. Conversely, the possibility of finding less interesting sequential patterns in a mining result decreases. Fig. 8 shows the TiW-support and the classical simple support of several sequential patterns in the mining result on the data set *D20C7T7S7I7-AB* when the value of $\delta$ is set to be 0.9. Fig. 8 clearly shows that the TiW-support of the sequential pattern derived from *part_A* is greater than its classical simple support, while the TiW-support of the sequential pattern derived from *part_B* is less than its classical simple support for the same reason as illustrated in Fig. 7.

To test the scalability of the proposed *psTiWS* method with the number of sequences in a sequence database, five data sets *D20C10T2.5S4I2.5*, *D40C10T2.5S4I2.5*, *D60C10T2.5S4I2.5*, *D80C10T2.5S4I2.5*, and *D100C10T2.5S4I2.5* were used. The experimental results on the data sets are shown in Fig. 9. A support threshold and the value of $u$ were set to 0.003 and 500 ms, respectively. As can be seen in the figure, the number of sequential

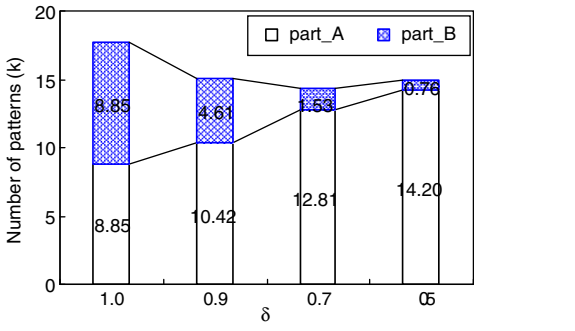


**Fig. 7.** Number of TiWS patterns on the data set *D20C7T7S7I7-AB*.

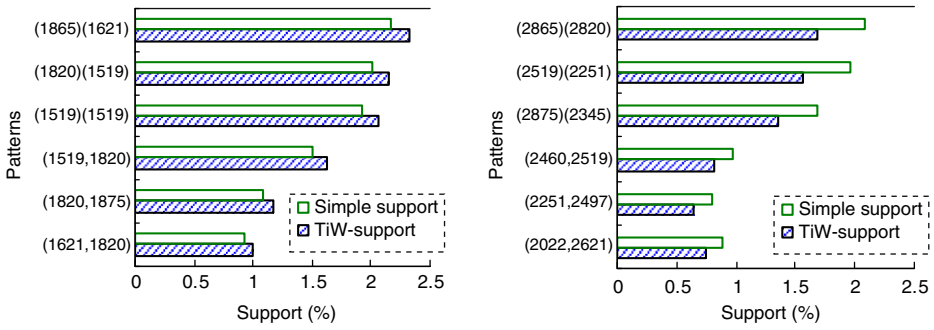


**Fig. 8.** Support comparison of TiW-support and classical simple support. (a) Patterns derived from *part_A*. (b) Patterns derived from *part_B*.

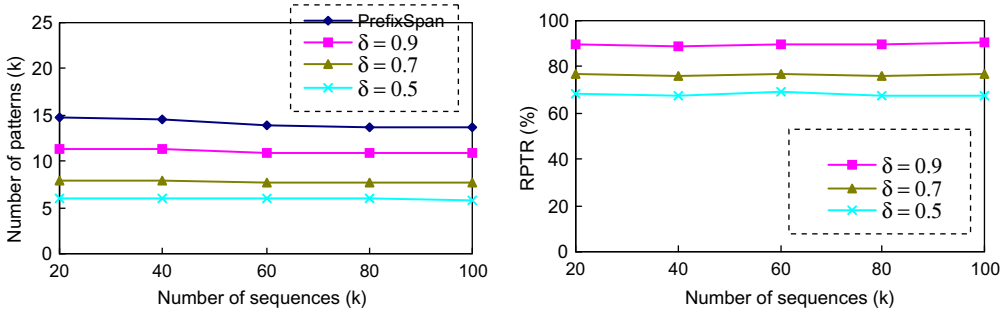


**Fig. 9.** Scalability test of the *psTiWS*. (a) Number of sequential patterns. (b) Relative processing time.

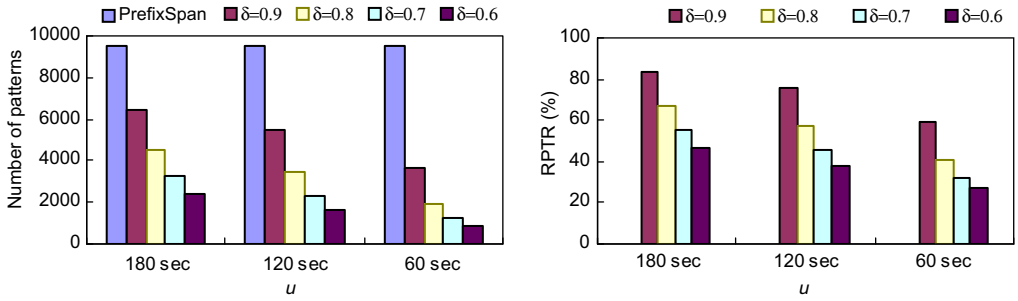


**Fig. 10.** Experimental results on the real data set *WebSEQ*. (a) Number of sequential patterns. (b) Relative processing time.

patterns in the mining result and the RPTR of the proposed method are scarcely affected by the size of a target sequence data set, i.e., the number of sequences in the data set. In short, the proposed method can efficiently be performed on a large-sized sequence data set.

The real data set *WebSEQ* was used to verify the efficiency of the *psTiWS* method in a real world application domain. It was generated from the web page access logs of a portal web site. In general, the consecutive web pages accessed by a user are considered as a semantically atomic unit of activities, i.e., a sequence. However, if a user does not access any web page for a certain period of time, the corresponding sequence is considered as being terminated and a new sequence is defined for any new web page accessed by the user. In the data set *WebSEQ*, the period was set to 60 s, i.e., a sequence was terminated if a user did not access any web page for 60 s. As a result, in the data set, the time-interval between two successive elements in a sequence is in a range of 0–60 s. The number of items is 436, and total number of sequences is 186042. Fig. 10 shows the experimental results on *WebSEQ*, and a support threshold is set to 0.005 in this experiment. As in the experimental results on a synthetic data set, the number of sequential patterns found in a framework for finding TiWS patterns, i.e., by the *psTiWS* method, is less than that found in general sequential pattern mining based on simple support counting, i.e., by the PrefixSpan. In addition, the processing time of the *psTiWS* method is less than that of the PrefixSpan. These findings lead us to conclude that the proposed method can be efficiently applied to the real world data set.

## 7. Concluding remarks

In this paper, a new framework for mining weighted sequential patterns with a time-interval weight has been developed. The time-interval weight of a pair of elements in a sequence based on its time-interval is proposed for the framework. Subsequently a process to get the weight of the sequence is also presented, and the strength of each pair of elements to the sequence is considered in this process. Based on the time-interval weight of a sequence, a novel measure of TiW-support and a term of a time-interval weighted sequential pattern are defined.

TiW-support differs from the conventional simple support in taking time-intervals of elements in a sequence as well as the order of elements in the sequence into account. In general, not only the generation order of data elements but also their generation times and time-intervals are important in real world application domains. Therefore, time-interval information of data elements can be helpful in finding more interesting sequential patterns. Extensive experimental results show that the new framework can give us more interesting mining results than the conventional framework based on simple support and it can be efficiently applied to find time-interval weighted sequential patterns.

Time-interval weighted sequential pattern mining can be efficiently used in application domains which generate time-related sequence data such as retail data analysis, web access log analysis, and financial data analysis, etc. When TiW-support is applied, the importance or interestingness of mined sequential patterns can be precisely verified by considering the time-interval in a sequence or sequential pattern.

A promising direction for future research to make the proposed term and framework more useful is the optimal selection of $\delta$ and $u$ in the time-interval weighting function. Currently, in the proposed framework, the values of $\delta$ and $u$ are set by a user. However, if their optimal value can be set considering the characteristics of target databases or application domains, it may be used more widely and effectively. Another interesting direction for future research could be in defining other approaches to get the time-interval of a sequence and other weighting functions. In addition, unlike the proposed framework defining the time-interval weight by a sequence in a sequence database and using the weight to get weighted sequential patterns, an approach defining the time-interval weight by each sequential pattern considering the time-intervals of itemsets in the sequential pattern could also be an interesting future research.

## Acknowledgements

I would like to thank the editor of the 'Knowledge-Based Systems' and anonymous reviewers for their constructive comments on an earlier version of this paper. This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2009-0064582).

## References

[1] R. Agrawal, R. Srikant, Mining sequential patterns, in: Proceedings of the 1995 International Conference on Data Engineering (ICDE '95), 1995, pp. 3–14.
[2] C.H. Cai, A.W.C. Fu, C.H. Cheng, W.W. Kwong, Mining association rules with weighted items, in: Proceedings of the IEEE International Database Engineering and Applications Symposium (IDEAS '98), 1998, pp. 68–77.
[3] E. Chen, H. Cao, Q. Li, T. Qian, Efficient strategies for tough aggregate constraint-based sequential pattern mining, Information Sciences 178 (6) (2008) 1498–1518.
[4] Y.-L. Chen, M.-C. Chiang, M.-T. Ko, Discovering fuzzy time-interval sequential patterns in sequence databases, IEEE Transactions on Systems Man and Cybernetics – Part B: Cybernetics 35 (5) (2005) 959–972.
[5] Y.-L. Chen, T.C.-H. Huang, Discovering time-interval sequential patterns in sequence databases, Expert Systems with Applications 25 (1) (2003) 343–354.
[6] H. Cheng, X. Yan, J. Han, IncSpan: incremental mining of sequential patterns in large databases, in: Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04), 2004, pp. 527–532.
[7] M. Ester, A top-down method for mining most specific frequent patterns in biological sequence data, in: Proceedings of the 2004 SIAM Internationlal Conference on Data Mining (SDM '04), 2004, pp. 90–101.
[8] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu, FreeSpan: frequent pattern-projected sequential pattern mining, in: Proceedings of the 2000 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00), 2000, pp. 355–359.
[9] X. Ji, J. Bailey, G. Dong, Mining minimal distinguishing subsequence patterns with gap constraints, Knowledge and Information Systems 11 (3) (2007) 259–296.
[10] H.-C. Kum, J. Pei, W. Wang, D. Duncan, ApproxMAP: approximate mining of consensus sequential patterns, in: Proceedings of the 2003 SIAM International Conference on Data Mining (SDM '03), 2003, pp. 311–315.
[11] H.-C. Kum, J.H. Chang, W. Wang, Sequential pattern mining in multi-databases via multiple alignment, Data Mining and Knowledge Discovery 12 (2+3) (2006) 151–180.
[12] M.-Y. Lin, S.-C. Hsueh, C.-W. Chang, Fast discovery of sequential patterns in large databases using effective time-indexing, Information Sciences 178 (22) (2008) 4228–4245.
[13] S. Lo, Binary prediction based on weighted sequential mining method, in: Proceedings of the 2005 International Conference on Web Intelligence, 2005, pp. 755–761.
[14] C. Luo, S.M. Chung, Efficient mining of maximal sequential patterns using multiple samples, in: Proceedings of the 2005 SIAM Internationall Conference on Data Mining (SDM '05), 2005, pp. 64–72.
[15] J. Pei, J. Han, W. Wang, Mining sequential patterns with constraints in large databases, in: Proceedings of the 2002 ACM International Conference on Information and Knowledge Management (CIKM '02), 2002, pp. 18–25.
[16] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu, Mining sequential patterns by pattern-growth: the PrefixSpan approach, IEEE Transactions on Knowledge and Data Engineering 16 (11) (2004) 1424–1440.
[17] K. Sun, F. Bai, Mining weighted association rules without preassigned weights, IEEE Transactions on Knowledge and Data Engineering 20 (4) (2008) 489–495.
[18] F. Tao, F. Murtagh, M. Farid, Weighted association rule mining using weighted support and significance framework, in: Proceedings of ACM SIGKDD '03, 2003, pp. 661–666.
[19] P. Tzvetkov, X. Yan, J. Han, TSP: mining Top-K closed sequential patterns, Knowledge and Information Systems 7 (4) (2005) 438–457.
[20] J. Wang, J. Han, C. Li, Frequent closed sequence mining without candidate maintenance, IEEE Transactions on Knowledge and Data Engineering 19 (8) (2007) 1042–1056.
[21] K. Wang, Y. Xu, J.X. Yu, Scalable sequential pattern mining for biological sequences, in: Proceedings of the 2004 ACM International Conference on Information and Knowledge Management (CIKM '04), 2004, pp. 178–187.
[22] W. Wang, J. Yang, P.S. Yu, WAR: weighted association rules for item intensities, Knowledge and Information Systems 6 (2) (2004) 203–229.
[23] X. Yan, J. Han, R. Afshar, CloSpan: mining closed sequential patterns in large datasets, in: Proceedings of the 2003 SIAM International Conference on Data Mining (SDM '03), 2003, pp. 166–177.
[24] J. Yang, P.S. Yu, W. Wang, J. Han, Mining long sequential patterns in a noisy environment, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD '02), 2002, pp. 406–417.
[25] U. Yun, A new framework for detecting weighted sequential patterns in large sequence databases, Knowledge-Based Systems 21 (2) (2008) 110–122.
[26] U. Yun, An efficient mining of weighted frequent patterns with length decreasing support constraints, Knowledge-Based Systems 21 (8) (2008) 741–752.
[27] M.J. Zaki, SPADE: an efficient algorithm for mining frequent sequences, Machine Learning 42 (1/2) (2001) 31–60.