Mining Multi-level Time-interval Sequential Patterns in Sequence Databases

Ya-Han Hu
Department of Information Management
National Chung Cheng University
Chia-Yi, Taiwan, R.O.C.
yahan.hu@mis.ccu.edu.tw

Fan Wu
Department of Information Management
National Chung Cheng University
Chia-Yi, Taiwan, R.O.C.
kfwu@mis.ccu.edu.tw

Chieh-I Yang
Department of Information Management
National Chung Cheng University
Chia-Yi, Taiwan, R.O.C.
g97530003@ccu.edu.tw

Abstract—**Mining sequential patterns is an important issue in data mining and has many applications. An extended work of sequential pattern mining, called time-interval sequential pattern mining, is proposed to retrieve time-interval information between successive items. However, previous work only considers single-level time-interval in pattern extraction, which means sequential patterns with cross-level time-intervals are completely ignored. Therefore, this study first defines multi-level time-interval sequential patterns and then presents a novel algorithm, named MLTI-PrefixSpan, for discovering the complete set of multi-level time-interval sequential patterns. Experimental results show that the proposed algorithm is effective on the test dataset.**

Keywords-component: data mining, sequential patterns, time-interval

I. INTRODUCTION

Recently, the rapid advancement of information technology has accumulated explosive data of enterprises. Many data mining techniques have been developed to extract potentially useful information from databases [1][2]. Among these techniques, sequential pattern mining is one of the most important data mining techniques [3][4], which discovers patterns occurred frequently in a sequence database. Since sequential patterns only unfold the time order among *events* (or itemsets), they do not provide the time span between any two successive events. To solve this problem, Chen et al. [5] extended traditional sequential pattern mining method that can find a time-interval sequential pattern. These time-interval sequential patterns reveal the information of not only the time order of the itemsets but also the time-interval between two successive itemsets. A typical example of time-interval sequential pattern is that after buying a laptop, a customer often returns to upgrade memory module within three months and then hard disk within a half year.

Although a time-interval sequential pattern carry more information than the plaint sequential patterns, the time-intervals information attached in sequential patterns may still be insufficient for some decision support problems. For example, assume there are two time-interval sequential pattern, < *a*, 1 week, *b* > and < *b*, 5 months, *c* >, which consider the time unit as week and day, respectively. In business, we may need to combine these two patterns as < *a*, 1 week, *b*, 5 months, *c* > to discover the long-term behaviors of customers. In the previous work [5], they only consider time-intervals of single unit in mining the patterns. In other words, they did not consider the different time units and their hierarchical relationship in their algorithm; it is of no way to get this useful pattern, like < *a*, 1 week, *b*, 5 months, *c* >, through the approach. This paper first defined multi-level time-interval patterns (or MLTI-pattern in short) and then proposed the associate algorithm (called MLTI-PrefixSpan algorithm) to overcome the problem addressed in the above. Users can define various granularities of time-intervals and find sequential patterns with arbitrary lengths of time-intervals according to their needs. Consider Fig. 1, which shows a hierarchical structure depicting the relationship of time interval between two events in the sequential patterns. In this example, the root node (quarter node) in level 1 is divided into three month nodes in level 2, and the month node in turn is divided into four week nodes in level 3, and the week node is divided into seven day nodes in last level. What kind of time-interval should we set is depended on the requirement of user. As a point view of a supermarket manager, he want to know how long did the customers buy milk by weeks, months, quarters.

This paper is organized as follows. In Section 2, the relevant literature of sequential pattern mining is discussed. The problem of mining multi-level time-interval sequential is formally defined in Section 3 and the MLTI-algorithm is introduced in Section 4. To show the performance of the proposed algorithm, a variety of experiments are performed. Section 5 provides the results of experimental evaluations, compared with ours and others for single-level time-interval. Section 6 gives the conclusion and future work.

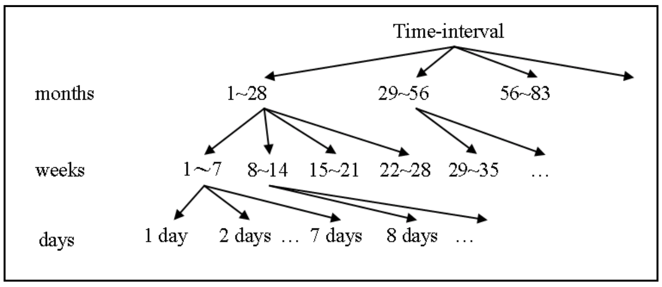

Fig. 1. A example of time-interval hierarchy

II. Related work

According to the literature, the algorithms for mining time-interval sequential patterns can be divided into three classes, namely, ignoring the time interval, time-window approach , and considering the intervals but without hierarchical concepts between two successive events.

First of all, methods by ignoring the time interval are introduced by [3][6][7].Those uncovered patterns are addressed on the temporal order of items instead of any time interval. For instance, there are four items in a sequence pattern (item1, item 2, item3, item4), however, we only know the item1 occurs at first, and then item 2 occurs after item 1, and so on.

Secondly, many researches provide ideas of specifying the length of time window. For example, [8] let user to define the width of the time window to find frequent episodes in sequences of event. By this way, we can find two types of episode: serial episodes and parallel episodes. For example, Assume episode α is a serial episodes; therefore, four items *a*, *b*, *c* and *d* happens by this order. But there is no evidence that these four items are separated by certain time intervals. On the contrary, assume episode β is a parallel episode, which has *a*, *b*, *c* and *d* occurs in a time range but in unknown order and separated by unknown interval. Wu et al. [9] devised a process for mining alarm sequential patterns from the alarm data of a mobile system. Particularly, an urgent window δ is defined as the user-specified time interval, which is set to determine the maximal time difference is allowed if two events are to be viewed relevant to each other. For instance, assume a sequential alarm pattern (*e*, *f*, *g*, *h*) with δ set to five seconds. Event *e* happens at first, then *f*, occurs after *e* and so on. And the time interval between *e* and *f*, are within five seconds, *f*, and *g* as well. However, this work didn't focus other potential time-intervals between events.

At last, [5] devised a method that concisely finds the time-intervals between successive items. It can obtain the time-interval sequential patterns like "(*a*, (1~7), *b* (15~21), *c*)". It means that a customer who has bought *a* will buy *b* within September 1th to September 7th. Similarly, after buying *b*, a customer will buy *c* with high probability within September 15th to September 21th. By this way, we can set different time-intervals according to our interests. This work developed two algorithms which are modified from Apriori [10] and PrefixSpan [7]. The former algorithm employs level-wise concept and incorporate support threshold to reduce unwanted patterns. And the latter adopts the tree-projection concept to divide the database into several sections. Nevertheless, we only can divide the time lines by single granularity of time-level. For example, time-level is determined as 7 days, a time-interval sequential pattern can be found as"(*a*, (1~7), *b*, (15~21), *c*)".Consequently, A pattern"(*a*, (1~7), *b*, (15~28), *c*)" can never be discovered by this approach.

Since [5] cannot find the different granularities of time-intervals between two successive items, this research proposes an efficient algorithm to discover multi-level time-interval sequential patterns, called MLTI-PrefixSpan which is modified from I-PrefixSpan [5].

III. Promlem definition

Let I denote the set of items in the database and $|I|$ denote the number of items in set I. The subset of I is called an *itemset*. A customer's data-sequence is an ordered list of itemsets associated with time stamps. For example, a data-sequence, α, can be represented as an ordered list, $<(a_1: t_1), (a_2: t_2), (a_3: t_3), \ldots, (a_m: t_m)>$, where a_i is an itemset (i.e., $a_i \subseteq I$), t_i $(1 \leq i \leq m)$ is the time stamp in which time a_i occurs, and $t_{i-1} \leq t_i$ for $2 \leq i \leq m$. Without loss of generality, the items in each itemset are sorted in alphabetical order. Thus, the expression of a sequence is unique in this paper.

In business, the concept of time interval can be seen as a hierarchy; for example, the business always divides the time interval for analyzing its sales in the in different granularities of time unit, such as a hierarchy of years, quarters, months and days. Let a time hierarchy denote as TI, TI^p denote the pth granularity level in TI, and $|TI^p|$ denotes the size of granularity of TI^p . For a time hierarchy TI, we assume that TI is composed by a set of different granularity of time unit, TI^1, TI^2 , …, TI^k, or denoted as TI = { TI^1, TI^2 , …, TI^k }, where TI^p $(1 \leqq p \leqq k)$ denotes the pth granularity level, and $|TI^p| < |TI^{p+1}|$. We let TI^k be the set of smallest time-interval units considered, and TI^{k-1} , TI^{k-2} , …, TI^1 be arranged in decreasing granularity. For example, a typical set of granularity level in transaction databases is TI = {years, quarters, months, days}. Let $TI^p = \{TI_1^p, TI_2^p, \ldots, TI_s^p\}$, where TI_q^p $(1 \leqq q \leqq s)$ stands for the q-th time-interval at level p and s is the number of time-intervals at level p. Each multi-level time-interval TI_q^p has only one *parent multi-level time-interval* at TI^{p-1} and the range of the time-interval TI_q^p must be smaller than that of its parent- multi-level -time-interval. Moreover, time lines at each level p are divided as follows.

- TI_1^p denotes time interval t satisfy $1 \leq t \leq TI_1^p$
- TI_2^p denotes time interval t satisfy $TI_1^p \leq t \leq TI_2^p$
- TI_1^p denotes time interval t satisfy $TI_2^p \leq t \leq TI_3^p$
- TI_1^p denotes time interval t satisfy $TI_{s-2}^p \leq t \leq TI_{s-1}^p$
- TI_1^p denotes time interval t satisfy $TI_{s-1}^p \leq t \leq TI_s^p$

TABLE I. TIME INTERVAL BETWEEN MULTI-LEVEL TIME-INTERVAL

level	Each interval at level p	Corresponded time
TI^1	TI_1^1	TI_1^1 : $1 \leq t \leq 27$
TI^2	TI_1^2, TI_2^2, TI_3^2	$TI_1^2 : 1 \leq t \leq 9 : TI_2^2 ; 9 < t \leq 18$; TI_3^2 : $18 < t \leq 27$
TI^3	TI_1^3, TI_2^3, TI_3^3, $\ldots TI_8^3, TI_9^3$	$TI_1^3 : 1 \leq t \leq 3 : TI_2^3 ; 3 < t \leq 6$;… TI_9^3 : $24 < t \leq 27$

Definition 1. Let $\beta = (b_1, \&_1, b_2, \&_2, b_3, \dots, b_{n-1}, \&_{n-1}, b_n)$ be a multi-level time-interval sequence (*MLTI-sequence*), where b_j is an itemset and $\&_j$ is a multi-level time-interval ($\&_j \in TI$, $1 \le j \le n$). Given a data-sequence α=((a_1: t_1), (a_2: t_2), (a_3: t_3), …, (a_m: t_m)), we say β occurs in α, or is a *subsequence* of α if integers $1 \le i_1 < i_2 < \dots < i_n \le m$ exist such that,

1. $b_1 \subseteq a_{i_1}$ a$_{i1}$, $b_2 \subseteq a_{i_2}$, …, $b_n \subseteq a_{i_n}$
2. t_{i_j} - $t_{i_{j-1}}$ satisfies the condition of interval $\&_{i-1}$ for $2 \le j \le n$.

Example 1. In Table. I ,Suppose TI={ TI^1 , TI^2 , TI^3 }, TI^1={ TI_1^1 }, TI^2={ TI_1^2 , TI_2^2 , TI_3^2 }, TI^3={ TI_1^3 , TI_2^3 , TI_3^3 , … TI_8^3 , TI_9^3 } , α =((a,1), (c,3) (ab,4), (ae,6), (c,10)).Then, β =(a, TI_1^2 ,e, TI_2^3 ,c) is a multi-level time-interval sequential subsequence of α .

Definition 2. A sequence database S is formed by a set of records < sid, s >, where s is a data-sequence and sid is the identifier of this data-sequence. For a given sequence β, its support count in S can be defined as follows:

$support_S(\beta) = |\{(sid, s)| (sid, s) \in S \wedge \beta$ is a subsequence in $s\}$

Example 2. Consider the sequence database shown in Table. I and Table. II with TI={ TI^1 , TI^2 , TI^3 } The *MLTI-sequence* (a, TI_1^2 ,e, TI_2^3 ,c) includes three items, and therefore has a length of 3. It is called a 3-time-interval sequence. The multi-level time-interval sequence (a, TI_1^2 ,e, TI_2^3 ,c) is a multi-level time-interval subsequence of transaction 10. Additionally, (a, TI_1^2 ,e, TI_2^3 ,c) is also contained in transaction 20. As a result, its support is 50%. If *min_sup*=30% is set , then (a, TI_1^2 ,e, TI_2^3 ,c) is a multi-level time-interval sequential pattern in the database.

Definition 3. Each granularity level p in TI has its minimum support threshold, denote as *minsup*(TI^p). Given a time-interval sequence β = (b_1, $\&_1$, b_2, $\&_2$, b_3, …, b_{n-1}, $\&_{n-1}$, b_n), the *minsup* of time interval $\&_j$, denoted as *minsup*($\&_j$), is equal to *minsup*(TI^p) if $\&_j \in TI^p$. Moreover, the *minsup* of β, denoted as *minsup*(β), is equal to: min(*minsup*($\&_1$),*minsup*($\&_2$),…,*minsup*($\&_{n-1}$)).

Definition 4. follow Definition 3, a time-interval sequence β is called a multi-level time-interval sequential pattern (*MLTI*-pattern) iff $support_S(\beta) \geqq minsup(\beta)$.

TABLE II. A SEQUENCE DATABASE

Sid	Sequence
10	((a, 1), (c, 3), (ab, 4),(ae ,6), (c, 10))
20	((d, 4), (abe, 7), (d, 9), (e, 9), (cd, 14))
30	((ab, 8), (e, 11), (d, 13), (bc, 16), (c,20))
40	((b, 15) ,(f, 17), (e ,18), (bc, 22))

IV. THE PROPOSED ALGORITHM

This section presents our efficient algorithm called multi-level time-interval PrefixSpan(MLTI-PrefixSpan) for mining multi-level time-interval sequential patters from databases. The I-PrefixSpan algorithm [5] extends the classic PrefixSpan [7] so that it can tackle the time-intervals between items in sequence data. However, we extend the I-PrefixSpan algorithm to find multi-level time-interval sequential patterns , which can be determined from sequence data with time intervals across different levels. First of all, the definition prefix, projection, posfix and projected database are introduced .Then our algorithm can be determined on these definitions. Last, there is an example to illustrate our concept.

Definition 5. Given a data-sequence α =((a_1: t_1), (a_2: t_2), (a_3: t_3), …, (a_m: t_m)) and a *MLTI-sequence* β = (b_1, $\&_1$, b_2, $\&_2$, b_3, …, b_{n-1}, $\&_{n-1}$, b_n) ($n \le m$), where b_j is an itemset and $\&_j$ is a multi-level time-interval ($\&_j \in TI$, $1 \le j \le n$). β is a multi-level time-interval prefix (*MLTI-prefix*) of if and only if (1) $b_j = a_j$ for $1 \le j \le (n-1)$; (2) $b_n \subseteq a_n$; (3)all the items in (a_n - b_n) are alphabetically after those in b_n ; and(4) t_j - t_{j-1} satisfies the condition of $\&_{j-1}$ for $1 < i \le (n-1)$.

Example 3. Suppose TI={ TI^1 , TI^2 , TI^3 },where TI^1={ TI_1^1 }, TI^2={ TI_1^2 , TI_2^2 , TI_3^2 }, TI^3={ TI_1^3 , TI_2^3 , TI_3^3 ,… TI_8^3 , TI_9^3 }.Then, β =(ab, TI_1^3 ,ce, TI_1^3 ,a) is a multi-level time-interval sequential subsequence of α =((ab,1), (ce,3) (ab,4), (ae,6), (c,10)).The first and second in both α and β are matched and third itemsets of β , (a) , which is contained in the third itemsets of α , (ae). Furthermore, the multi-level time-interval between (ab) and (ce) is TI_1^3 which means that (ce) happens after (ab) in first day to ninth day.

Definition 6. Follow Definition 5, assume β is a *MLTI-subsequence* of α . Let $i_1 < i_2 < \dots < i_n$ be the indexes of the elements in α that match the element of . A subsequence α' =((a_1',t_1'), (a_2',t_2'), (a_3',t_3')..., (a_v',t_v')) of sequence α , where $v = (n+m-i_n)$, is called a projection of α with respect to β if and only if (1) β is a multi-level time-interval prefix of α' and (2) the last (m-i_n) elements of α' are the same as the last (m-i_n) elements of α' .

Example 4. If α = ((a,1), (c,3), (ab,4), (ae,6), (c,10)) is projected with respect to β =(a), consequently, we can find three difference projection α' . One is ((a,1), (c,3), (ab,4), (ae,6), (c,10)), another is ((ab,4), (ae,6), (c,10)),and the other is ((ae,6), (c,10)). The prefix β occurs in α at position 1 is α' = ((a,1), (c,3), (ab,4), (ae,6), (c,10)), where the last four elements are the same as those of α . Corresponding to position 3, α' = ((ab,4), (ae,6), (c,10)), where the last two elements are the same as those of α . And the last position is in the same way.

Definition 7. Let α' = ((a_1',t_1'), (a_2',t_2'), (a_3',t_3'),…(a_w',t_w')) be the projection of α with respect to a *MLTI-prefix* β = (b_1, $\&_1$, b_2, $\&_2$, b_3, …, b_{n-1}, $\&_{n-1}$, b_n'). Therefore, γ =

((a_n'' ,t_n'),(a_{n+1}',t_{n+1}'), (a_{n+2}',t_{n+2}'),…,(a_w',t_w')) is called the postfix of α with respect to prefix β , where <$a_n''= a_n' - b_n'$ >.

Example 5. We can extend the previous example. According to our definition of postfix , it can be found just by eliminating the prefix from the projection. The postfix γ of α'= ((*a*,1), (*c*,3), (*ab*,4), (*ae*,6), (*c*,10)) is (*c*,3), (*ab*,4), (*ae*,6), (*c*,10) by cutting the prefix. Eventually, the α -projected database is denoted by $S|_\alpha$, which is the collection of postfixes of sequence in database S with respect to α .

The classic PrefixSpan method use the strategy of divide and conquer. At the first step of algorithm, α is set as null and discover all of the frequent 1-patterns in α -projected database $S|_\alpha$ .Then , append each frequent 1-pattern called b to α in order to form a sequential pattern α' and construct the α' - projected database $S|_{\alpha'}$. By this way, we can find all the sequential patterns in S by recursively identifying the sequential patterns in $S|_{\alpha'}$. Moreover, I-PrefixSpan incorporate time-interval relationship between item *b* in $S|_\alpha$ and the last item in α . In order to deal with this problem, I-PrefixSpan create a *Table* which is used to store this type of time-interval relation such as "0, 1~3, 4~6, 7~∞". We use this *Table* in our lowest level of time granularity. For example in Fig.2, if we have three level of time-intervals, *TI*={ TI^1 , TI^2 , TI^3 }. A table *Table* is used to store the relationship between a frequent item b in $S|_\alpha$ and the last item in α , where each column corresponds to an item and each row correspond to a kind of multi-level time-interval relations in TI^3 ={ TI_1^3 , TI_2^3 , TI_3^3 ,… TI_8^3 , TI_9^3 }. Every cell *Table*(TI_q^3 ,b) in the table records the count of transactions in $S|_\alpha$ that contain item *b* and the time difference between this item and the last items of α lies with TI_q^3 . We can construct the table by operate teach each transaction in $S|_\alpha$ and identify the frequent cells. Cell *Table*(TI_q^3 ,*b*) can be appended to α only if the *Table*(TI_q^3 ,*b*) is a frequent one. After appending, we can form a multi-level time-interval sequential pattern α' and construct the α' - projected database $S|_{\alpha'}$. Additionally, we also can identify the frequent *Table*(TI_q^2 ,*b*) by accumulating the count of *Table*(TI_q^3 ,*b*) and construct α' -projected database as well. We can recursively find the multi-level time-interval sequential patterns in $S|_{\alpha'}$ and eventually produces all the multi-level time-interval sequential patterns in S. Fig. 3 presents our process of algorithm .

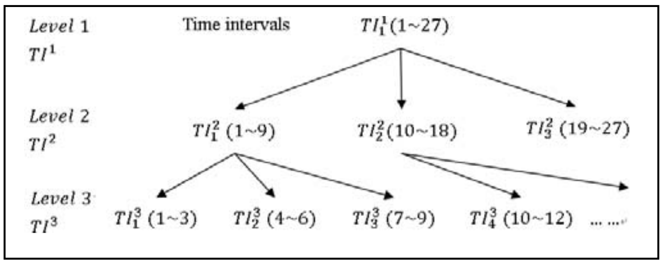

Fig. 2.An example of multi-level time-interval hierarchy

```
Subroutine: I-PrefixSpan (α, l, $S|_\alpha$ )
Parameter:
  $\alpha$        a multi-level time-interval sequential pattern
  $l$        The length of α. S|α:the α-projected database
  $S$        Sequence database
  $TI_q^p$     The interval at level $p(1 \le p \le k)$ and the q-th item
           $1 \le q \le s$ in this level
Methods:
  ➢ Scan $S|_\alpha$ one time
      — If l=0, then find all frequent items in S|α. For every
        frequent item b, append b to α as α′. Output all α′.
      — If l>0, then scan all the transactions in $S|_\alpha$ and
        accumulate the support of each cell Table( $TI_q^k$ , b).
        Compute the supports of cells Table( $TI_q^p$ , b),
        $1 \le p \le (k-1)$ according to Table( $TI_q^k$ , b).
        Identify the frequent cell Table( $TI_q^p$ , b) with
        different min_sup ( $TI^p$ ). Append them to $\alpha$ as $\alpha'$.
  ➢ For each $\alpha'$, construct $\alpha'$-projected database $S|_{\alpha'}$,and call I-
    PrefixSpan ($\alpha'$, l +1, $S|_{\alpha'}$).
```

Fig. 3.The MLTI-PrefixSpan algorithm

Example 6. Suppose *TI*={ TI^1 , TI^2 , TI^3 },where TI^1 ={ TI_1^1 }, TI^2 ={ TI_1^2 , TI_2^2 , TI_3^2 }, TI^3 ={ TI_1^3 , TI_2^3 , TI_3^3 ,… TI_8^3 , TI_9^3 } and each time-interval is in Table. I. Consider the sequence database shown in Fig 4.The minimum support at level 3 set to 2, and minimum support at level 2 set to 3. First of all, α = null yields five different α' , and the method have to be called again with different parameters for each different α' . Suppose the case with α' =(a). The process *MLTI*-PrefixSpan((*a*),1, $S|_{(a)}$) is used where the projected database $S|_{(a)}$ is demonstrated as follows.

[10:1] ((*c*, 3) (*a b*, 4), (*a*, 6), (*e*, 6), (*c*, 10))

[10:4] ((*b*, 4), (*a e*, 6), (*c*, 10))

[10:6] ((e, 6), (c, 10))

[20:7] ((b e, 7), (d e, 9), (c d, 14))

[30:8] ((b, 8) (e, 11) (d, 13), (b c, 16), (c, 20))

In the process, Table III has to be constructed first. Table III shows that the frequent cells are (TI^0 ,b),(TI^0 ,e) ,(TI_3^3 ,c) and (TI_1^3 ,e). Notably, we denote concurrence of two items as TI^0 for convenience. After appending frequent cells, TI^0 is not used because we denote concurrence of a and b as (ab). Table IV shows that the frequent cells at second level are (TI_1^2 ,c), (TI_1^2 ,d), (TI_1^2 ,e).

TABLE III. THE TABLE CONSTRUCTED IN *MLTI*-PREFIXSPAN ((A),1, $s\,|_{(A)}$)

a-projected	*a*	*b*	*c*	*d*	*e*
TI^0	0	3	0	0	2
TI_1^3	1	1	1	1	3
TI_2^3	1	0	1	1	1
TI_3^3	0	1	3	1	0
TI_4^3	0	0	0	0	0
TI_5^3	0	0	0	0	0
TI_6^3	0	0	0	0	0
TI_7^3	0	0	0	0	0
TI_8^3	0	0	0	0	0
TI_9^3	0	0	0	0	0

TABLE IV. TABLE CONSTRUCTED BY ACCUMULATING THE CELL IN TABLE. III

a-projected	*a*	*b*	*c*	*d*	*e*
TI_1^2	2	2	5	3	4
TI_2^2	0	0	0	0	0
TI_3^2	0	0	0	0	0

Then we append these seven cells to the end of (a) yields seven different α', which are (ab),(ae), (a , TI_3^3 , c) , (a , TI_1^2 c) , (a , TI_1^2 , d) , (a , TI_1^2 , e) , (a , TI_1^3 e). Suppose we choose (a ,TI_1^2 , e) as the next step of example, *MLTI*-PreFixSpan((a , TI_1^2 , e), 2 ,$S\,|_{(a,TI^2,e)}$) , where the projected database $S\,|_{(a,TI^2,e)}$ shows as follows.

[10:6] ((c, 10))

[20:9] ((c, 14), (d, 14))

[30:11] ((d, 13), (b c, 16), (c, 20))

The table *Table* must be constructed shown in Table IIV and Table VI as well, where the frequent cells are (TI_2^3 ,c) and (TI_1^2 ,c) . Then ,two 3-length multi-level time-interval sequential patterns (a, TI_1^2 , e, TI_2^3 , c) , (a, TI_1^2 , e, TI_1^2 , c) are generated. .We can do by this way and finally identifies all the sequential patterns in database. The pattern (a, TI_1^2 , e, TI_2^3 ,c) means that e happens after a in first day to ninth day and c happens after e in fourth day to ninth day. It can prove our approach can derive different time level in sequence data and has more specific time-intervals than another. Although (a TI_1^2 , e, TI_1^2 , c) didn't achieve our purpose, there will be more itemsets occurred after it with higher probability than (a TI_1^2 , e, TI_2^3 ,c). This type of patterns is still useful for mining.

TABLE V. THE TABLE CONSTRUCTED IN MLTI-PREFIXSPAN $(a,TI_1^2,e),2,S\,|_{(a.TI_1^2,e)})$

(a,TI_1^2,e) - projected	*a*	*b*	*c*	*d*	*e*
TI^0	0	0	0	0	0
TI_1^3	0	0	0	1	1
TI_2^3	0	1	3	1	0
TI_3^3	0	0	1	0	0
TI_4^3	0	0	0	0	0
TI_5^3	0	0	0	0	0
TI_6^3	0	0	0	0	0
TI_7^3	0	0	0	0	0
TI_8^3	0	0	0	0	0
TI_9^3	0	0	0	0	0

THE TABLE CONSTRUCTED BY ACCUMULATING THE CELL IN TABLE. IIV

(a,TI_1^2,e) - projected	*a*	*b*	*c*	*d*	*e*
TI_1^2	0	1	4	2	1
TI_2^2	0	0	0	0	0
TI_3^2	0	0	0	0	0

V. PERFORMANCE EVALUATION

This section compares the performance between the I-PrefixSpan[5] algorithm and the MLTI-PrefixSpan algorithms on a synthetic dataset. Both two algorithms are implemented in Java language and tested on a Pentium E2200 2.2G Windows XP system with 768MB of main memory and JVM(J2RE 1.40_01) as the Java execution environment. Table VII lists the parameters of our data generator. The parameter setting is as follows: $|D|$ =125,000, $|C|$ =10, $|T|$ =2.5, $|S|$ =4, $|I|$ =1.25, N_S =5000, N_I =25,000, N =10,000, TIL =3. The TIL is a new parameter designed for setting the number of time-interval level. Consequently , the multi-level time-interval in our experiments is set to TI={ TI^1 , TI^2 , TI^3 }, TI^1 ={ TI_1^1 , TI_2^1 , TI_3^1 }, TI^2 ={ TI_1^2 , TI_2^2 … TI_3^2 }, TI^3={ TI_1^3, TI_2^3, TI_3^3,… TI_{11}^3 , TI_{12}^3 }.

The runtime comparison is showed in Fig.4 and Fig.5, where the minimum support are 0.75%, 1% and 1.5%. As our expectation, the performance of MLTI-PrefixSpan is inferior to I-PrefixSpan. This is because the I-PrefixSpan only find single-level time-interval sequential patterns at a time while MLTI-PrefixSpan simultaneously finds three-level time-intervals as well as their correlations at the same time. The results also show that MLTI-PrefixSpan can discover more time-interval patterns, especially cross-level time-interval sequential patterns, and provide more information for users.

TABLE VI. PARAMETER

|D|	Number of customers
|C|	Average number of transactions per customer
|T|	Average number of items per transaction
|S|	Average length of maximal potentially large sequences
|I|	Average size of itemsets in maximal potentially large sequences
N_S	Number of maximal potentially large sequences
N_I	Number of maximal potentially large itemsets
N	Number of items
TI	Average length of time-intervals
TIL	Level of time-intervals

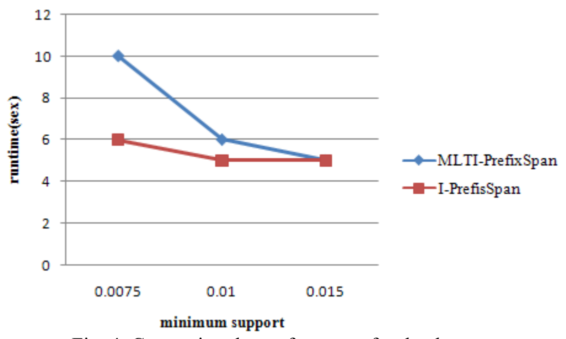

Fig. 4. Comparing the performance for the dataset

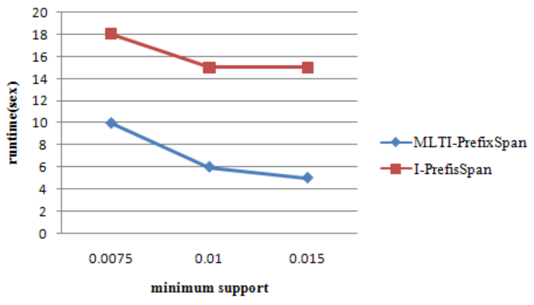

Fig. 5. Multiple three times for the runtime of I-PrefixSpan

VI. CONCLUSIION

This work proposes a novel multi-level time-interval sequential pattern mining algorithm. We incorporate the multi-level time-interval concept into sequential pattern mining. Specifically, our method can find cross-level patterns from sequence database instead of single-level time-interval. Traditional method can only set single-level time-interval at a time. It simply assumes the time-interval in patterns cannot involve various time-interval granularities, which is not often the case in real-life applications. In this study, we define MLTI-patterns and propose an efficient algorithm, called **MLTI-PrefixSpan**, to find all multi-level time-interval information in sequence databases. This method can be used in various applications, such as marketing strategy, analyzing web log and so on.

REFERENCES

[1] Chen, M.S., Han, J., & Yu, P.S. (1996). Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, *8* (6), (pp. 866-883).

[2] Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1991). *Knowledge discovery in databases: an overview,* AAAI/MIT press.

[3] Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. *Proceedings of 1995 Int. Conf. Data Engineering*, (pp. 3-14).

[4] Han, J., & Kamber, M. (2001). *Data mining: concepts and techniques,* Academic Press.

[5] Chen, Y.L., Chiang, M.C. and Ko, M.T. (2003). “Discovering time-interval sequential patterns in sequence databases,” *Expert Syst*. Appl., Vol. 25, No. 3, (pp. 343–354).

[6] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., & Hsu, M.-C. (2000). FreeSpan: frequent pattern-projected sequential pattern mining. *Proceedings of 2000 Int. Conf. on Knowledge Discovery and Data Mining*, (pp. 355-359).

[7] Pei, J., Han, J., Pinto, H., Chen, Q., Dayal, U., & Hsu, M.-C. (2001). PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. *Proceedings of 2001 Int. Conf. on Data Engineering* (pp. 215-224).

[8] Mannila, H., Toivonen, H., & Inkeri Verkamo, A. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery, 1*(3), (pp. 259 –289).

[9] Wu, P.-H., Peng,W.-C., & Chen, M.-S. (2001). Mining sequential alarm patterns in a telecommunication database. *Proceedings of Workshop on Databases in Telecommunications (VLDB 2001)* (pp. 37-51).

[10] Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of 1994 Int. Conf. Very Large Data Bases*, (pp. 487-499).