TIME CONSTRAINTS EXTENSION ON FREQUENT SEQUENTIAL PATTERNS

A. Ben Zakour, M. Sistiaga
2MoRO, Bidart, France

S. Maabout, M. Mosbah
Université de Bordeaux, Bordeaux, France

Keywords: Frequent pattern, Time constraint, Relax constraint, Sequential pattern mining.

Abstract: Unlike frequent sets extraction for which only minimum support condition must be met, sequential patterns satisfy time constraints. Commonly, to consider two events as successive, these constraints are either to respect minimum and maximum time gap or to be included into a window size. In this paper, we introduce a new definition of "interesting sequences". This property suggests that temporal patterns, introducing concepts of sliding window, can be customized by the user so that the events chronology in the extracted sequences has not to strictly obey to the original event sequence.This definition is incorporated in the process of a conventional algorithm (Fournier-Viger et al., 2008). The extracted patterns have an interval time stamp form and represent an interesting palette of the original data.

1 INTRODUCTION

For Sequential Patterns Mining (SPM), the usage of the temporal factor depends on the needs of this component in the results. Existing extraction techniques consider only the space component, the "succession" of events in a sequence. In some cases, results can be "exuberant" e.g., during the analysis of the shopping basket, suppose that one customer buys product A and product B a day later, and another one buys product A and product B one month later. Based on these two sequences, the pattern extracted is "a customer who buys product A, buys the product B". This conclusion is not necessarily representative for the second customer, and the extracted information is not representative of the baseline data. Taking into account the temporal aspect in the SPM was introduced in (Srikant and Agrawal, 1996). Their constraints aim to (1) bringing together "close" events into an individual transaction and consider them as simultaneous and (2) regulating the minimal and maximal time gaps between two successive transactions. Considering the importance of the temporal component in the interpretation of patterns, (Hirate and Yamana, 2006) introduce temporal constraints to extract temporal patterns. They apply some improvements to Prefixspan (Pei et al., 2004): (1) an interval function to upgrade the timestamps of the bearing, (2) set the maximal and minimal time gaps between two successive transactions (3) min_whole_interval and max_whole_interval to regulate the minimum and maximum temporal spread of a sequence. Both of these works introduce time constraints to be satisfied by the extracted sequences. One should notice that in both cases, the extracted sequences respect the chronological order that appears in the underlying data. However, in some applications, close chronological ordering is not necessary important for events. For example the two temporal sequences $\langle(0,A)(1,B)(2,C)\rangle$ and $\langle(0,A)(1,C)(2,B)\rangle$ may represent the same information, that is A, B and C occur close to each other in the interval [0,2]. The two algorithms presented previously do not consider this kind of information.

In this paper, we present a new definition of interesting patterns. The idea is that we apply temporal relaxation on itemsets by using a backward sliding window size constraint. Such a constraint can take into account all *neighbor* events in a time window as simultaneous.

The paper is organized as follows. First, we present a brief state of the art. Then, we give a new definition of interesting sequences. The third section

Ben Zakour A., Sistiaga M., Maabout S. and Mosbah M..
TIME CONSTRAINTS EXTENSION ON FREQUENT SEQUENTIAL PATTERNS.
DOI: 10.5220/0003098702810287
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2010), pages 281-287
ISBN: 978-989-8425-28-7

describes a modification of the algorithm presented in (Fournier-Viger et al., 2008) to extract these sequences. A short evaluation of our approach is illustrated,a conclusion and perspectives in the last section.

2 RELATED WORK

In this section, we present some general definitions and two SPM works are detailed.

2.1 Terminology

A **transaction** is a timestamped itemset (or event set). It is denoted by $I_i = \{i_1, i_2, \ldots, i_p\}$. $time(I_i)$ denotes the timestamp of I_i. A **temporal sequence** is a timestamp ordered sequence of itemsets. It is denoted by $S = \langle (t_1, I_1), (t_2, I_2), \ldots, (t_n, I_n) \rangle$ where I_i is a transaction and $t_i = time(I_i)$. Hereafter, the timestamp t_i is related to the occurrence time of the first transaction of the sequence. Thus, it represents the time lag between the transaction I_i and I_1. A **sequence database** is a collection of sequences where each sequence has a unique identifier *id_sequence*. The **support** of a sequence s in a sequence database *SDB*, denoted by $support_{SDB}(s)$ is the percentage of sequences that contain s in SDB. Given a minimal support *minsupp*, s is **frequent** in *SDB* iff $support_{SDB}(s) \geq minsup$. Besides *minsupp* and depending on the business needs, the user may set **time constraints** that should be satisfied by the extracted patterns. The time constraints we consider are: mingap and maxgap: represent respectively the minimal and maximal time gap between two successive transactions: $time(I_{i+1}) - time(I_i) \geq mingap$, $time(I_{i+1}) - time(I_i) \leq maxgap$

Example 1. *Let $s = \langle (0, I_1)(1, I_2)(10, I_3) \rangle$. If mingap=2 then transactions I_1 and I_2 are not considered as successive because they are "too close". If maxgap is set to 5 then I_2 and I_3 are not considered as successive because they are too distant.*

min_whole_interval and max_whole_interval: represent respectively minimal and maximal whole interval constraints. Let n be the number of itemsets in a sequence. Then: $time(I_n) - time(I_1) \geq min\_whole\_interval$, $time(I_n) - time(I_1) \leq max\_whole\_interval$

Example 2. *Let min_whole_interval=1, max_whole_interval=4 and $s = \langle (0, I_1)(1, I_2) \rangle$. s satisfies the min_whole_interval constraint and the max_whole_interval constraint.*

Window size: it allows to consider events (items) in different transactions, such as simultaneous (within a single transaction). These transactions should be relatively close to each other regarding to the size of the window.

Example 3. *Let $T = (0, AB)$ and $s = \langle (0, A)(2, B)(3, C) \rangle$. If ws=2 then s does contain T while if window = 1, T is not contained in s.*

2.2 GSP

In (Srikant and Agrawal, 1996), the authors improved their A Priori algorithm (Agrawal and Srikant, 1994) to bearing the absence of time constraint on timeless patterns extracting. This algorithm relaxes transaction definition by using a window notion and by integrating *mingap* and *maxgap* constraints.

Definitions. Let $d = \langle d_1 d_2 ... d_m \rangle$ and $s = \langle s_1 s_2 ... s_n \rangle$ be two sequences. d contains the sequence s with constraint if and only if: **Window size constraint** : There exist integers $l_1 \lesneq u_1 \lesneq l_2 \lesneq u_2 \ldots l_n \lesneq u_n$ such that: $s_i \subseteq \cup_{k=l_i}^{u_i} d_k, 1 \leqslant i \leqslant n$ and $time(d_{u_i}) - time(d_{l_i}) \leq$ window size, $1 \leqslant i \leqslant n$ **mingap and maxgap constraints** : There exist $l_1 \lesneq u_1 \lesneq l_2 \lesneq u_2 \ldots l_n \lesneq u_n$ such that: $s_i \subseteq \cup_{k=l_i}^{u_i} d_k, 1 \leq i \leq n$, $time(d_{l_i}) - time(d_{u_{i-1}}) \geq mingap$, for $2 \leq i \leq n$ and $time(d_{u_i}) - time(d_{l_{i-1}}) \leq maxgap$, for $2 \leqslant i \leqslant n$. $time(d_{l_i})$ is s_i start time, $time(d_{u_i})$ refer to s_i end time.

Algorithm Description. The main goal is to find all frequent sequences satisfying all user constraints. GSP is a level wise algorithm: first, it recovers L_1 the set of frequent 1-sequences. It generates the candidates sequences of size $k+1$ by self-joining L_{k-1}.

2.3 GSPM

(Hirate and Yamana, 2006) presents an approach for extracting frequent temporal sequences from a temporal sequences database. The algorithm applies time constraints different from those applied in GSP. The main goal is to extract temporal frequent patterns from sequence database by integrating a new time constraint that align interval timestamps into a same value. The algorithm is an improvement of PrefixSpan (Pei et al., 2004) where data-sequences may be either timestamped or just sorted.

Definition. Let f be a time function. It maps time intervals to integers. Let f be defined as follows:

$$f(x) = \begin{cases} f_0 \; if \; x \in [val_1, val_2[\\ f_1 \; is \; x \in [val_2, val_3[\\ \ldots \\ f_{n-1} \; if \; x \in [val_n, val_{n+1}[\end{cases}$$

Let $a = \langle (t_1, X_1), (t_2, X_2), \ldots, (t_m, X_m) \rangle$ and $b = \langle (t'_1, X'_1), (t'_2, X'_2), \ldots, (t'_n, X'_n) \rangle$ be tow time sequences.

b is a subsequence of a w.r.t f, if and only if there exist $1 \leq i_1 \lneq i_2 ... \lneq i_n \leq m$ such that

- $X_k \subseteq X'_{i_k}$ for all $1 \leq i \leq n$ and
- $f(t_k) = f(t'_{i_k})$

Let a sequence data base SDB, a sequence $\alpha = \langle (t_{1,1}, a_1), (t_{1,2}, a_2), \ldots, (t_{1,m}, a_m) \rangle$ and X_β an itemset. If exist an Integer j ($1 \leq j \leq m$) as $X_\beta \subset a_j$ and $I(t_{1,\beta}) = I(t_{1,j})$, than the of α prefix regards to $X_\beta, I(t_{1,\beta})$ is defined by: $prefix(\alpha, X_\beta, I(t_{1,\beta})) = \langle (t_{1,1}, a_1), (t_{1,2}, a_2) ..., (t_{1,j}, a_j) \rangle$
The α suffix regards to X_β, $I(t_{1,\beta})$ represents sequence events those occurs after X_β. It's defined by the: $suffixe(\alpha, X_\beta, I(t_{1,\beta})) = \langle (t_{j,j}, a'_j), (t_{j,j+1}, a_{j+1}), \ldots, (t_{j,m}, a_m) \rangle$.

Algorithm Description. The approach is a Prefixspan extension (Pei et al., 2004). The first step is to recover the frequent 1-sequences. Then, longer patterns are extracted through a patterns growth process. They are recovered by using a projection process to discover, for each build patterns, the possible continuations on the concerned set of the SDB. The SDB projection on a α pattern is denoted by $SDB|\alpha$ and defined by the equation: $SDB|\alpha = \{is \in BD | is = suffix(\gamma, 0, I)\}$ *avec* $\gamma \in SDB$ Iteration are stopped when there is no possible continuation or no more frequent items.
These two works extract frequent patterns by introducing time constraints. They relax the classical transaction definition to better represent baseline data. Relation is introduced through the application of the grouping window size (Srikant and Agrawal, 1996) and the temporal function in (Hirate and Yamana, 2006). Although those relaxation methods, it is impossible to associate unordered events.
In both methods, with a window size equals to 2 or with an equivalent time function, the sequence $\langle (0, ABC) \rangle$ does not contain the pattern $\langle (0, B)(1, AC) \rangle$. The algorithms look for item by item and apply the projection, they keep as a continuation only events that follow current item. In next section, we present our approach which allows to take into account this kind of data.

3 INTERESTING SEQUENCES: DEFINITIONS

In this paper, we propose to define a new type of interesting sequences extracted from a temporal sequences database. The difference with patterns extracted in works presented, is the relaxation of transaction definition without taking into account the order of items within itmsets.

An "interesting sequence" is denoted by $sp = \langle (\delta t_1, I_1), (\delta t_{1,2}, I_2) \ldots, (\delta t_{1,m}, I_m) \rangle$. where I_i is an itemset . $\delta t_{1,j}$ is a transaction time stamp, it is the temporal interval in witch I_j events occur. This interval is characterized by min_time (respectively max_temp), its lower (resp. upper) bound. $\delta t_{1,j}$ is a relative timestamps w.r.t to I_1 occurrence where: min_time and max_time are respectively the minimum and maximum time at which the events I_j may occur after those of I_1.

Example 4. *Let the sequence* $\langle (0, A)([2,3], B) \rangle$ *means that B occurs at earlier 2 times after A and at the latest 3 times after A.*

We consider the following time constraints:

- **mingap and maxgap**:

$$min(\delta t_{i+1}) - max(\delta t_i) \geq mingap$$
$$max(\delta t_{i+1}) - min(\delta t_i) \leq maxgap$$

- **min_whole_interval and max_whole_interval**:

$$min(\delta t_1) - max(\delta t_n) \geq min\_whole\_interval$$
$$min(\delta t_1) - max(\delta t_n) \leq max\_whole\_interval$$

- **Window size**:

$$max(\delta t_i) - min(\delta t_i) \leq ws$$

Let $s = \langle (t_1, I_1), \ldots, (t_n, I_n) \rangle$ and $s_1 = \langle (t'_1, s_1), \ldots, (t'_m, s_m) \rangle$. s contains s_1, denoted $s \models S_1$ if and only if: *For all* $1 \leq i \leq m$
$s_1 \subset (\bigsqcup_{u=l1_d}^{l1_f} I_u), s_2 \subset (\bigsqcup_{u=l2_d}^{l2_f} I_u), \ldots, s_m \subset (\bigsqcup_{u=lm_d}^{lm_f} I_u)$
$max(time(\bigsqcup_{u=li_d}^{li_f} I_u)) - min(time(\bigsqcup_{u=li_d}^{li_f} I_u) \leq ws$
$time(s_i) \in [min(time(\bigsqcup_{u=li_d}^{li_f} I_u)), max(time(\bigsqcup_{u=li_d}^{li_f} I_u)]$
$min(time(\bigsqcup_{u=li+1_d}^{li+1_f} I_u)) - max(time(\bigsqcup_{u=li_d}^{li_f} I_u)) \geq 0$

Example 5. *Let* $s_1 = \langle (0, A)(1, C)(2, B)(5, CDE)(6, F) \rangle$, $s_2 = \langle (0, AB)(5, FD) \rangle$ *and* $s_3 = \langle (0, A)(12, CD) \rangle$. *If window size ws is equal to 2, then* s_1 *contains* s_2. *If* $mingap = 4$ *then* s_1 *does not contain* s_2. *Finally if* $maxgap = 10$ *then* s_1 *does not contain* s_3*: the gap between the maximal timestamps of* (CD) *and the minimal time stamp of* (AB) *in* s_3 *is greater than maxgap (*$12 \geq 10$*).*

This definition allows to extract patterns that can't be considered frequent by classical constraint. They group, besides classical frequent patterns, those grouping frequent unordered events occurring beside a window interval.

4 COMPUTING INTERESTING SEQUENCES

Problem Definition. Given a sequences database, a support threshold *minsupp*, a window

size *ws* and time constraints: *mingap*, *maxgap*, *min_whole_interval* and *max_whole_interval*, find all interesting sequences.

Algorithm Description. The calculation of "interesting" sequences is implemented through a modification of the algorithm SPMF (Fournier-Viger et al., 2008). This is an improvement of the algorithm PrefixSpan (Pei et al., 2004). Initially, the algorithm extracts frequent items I from the sequences database. This provides the set $L_1 = \{s_i = (0,I) | support(I) \geq minsupp\}$. Then, for providing continuations of each 1-sequence, SDB is projected onto each 1-sequence $(0,I)$. This projection is intended to take into account the relaxation introduced by the window. The projection considers possible continuation of an event that occurred at a time t. It backwards to event occurring in the interval $[ws-t,t]$ concatenated to the "classical" continuation. This interval allows to consider as possible "simultaneous" continuation events appearing before the current event. Let $\alpha = \langle(t_{1,1},a_1),\ldots,(t_{1,m},a_m)\rangle$ and a pair $(\delta t, X_\beta)$. If there exists j such that $1 \leq j \leq m$ and $X_\beta \subset a_j$ and $t_{1,\beta} \in \delta t$ then the α prefix with regards to $X_\beta, t_{1,\beta}$ is defined as:

$wprefixe(\alpha, X_\beta, t_{1,\beta}) = \langle(t_{1,1},a_1),(t_{1,2},a_2),\ldots,(t_{1,j},a_j)\rangle$

The suffix of α wrt $X_\beta, I(t_{1,\beta})$ is defined as:

If $j = 1$ then $wsuffix(\alpha, X_\beta, t_{1,\beta}) = \langle(t_{j,j}, a_j \setminus X_\beta),(t_{j,j+1},a_{j+1}),\ldots,(t_{j,m},a_m)\rangle$, else it is equal to $\langle(t_{j,k},a'_k),\ldots,(t_{j,j},a_j \setminus X_\beta),(t_{j,j+1},a_{j+1})\ldots,(t_{j,m},a_m)\rangle 1 \leq k \leq m$ *as* $t_{jk} \leq ws$ *and* $t_{j(k-1)} \geq ws$.

If $j = 1$ the projection of SDB on $\alpha = \langle(0,I)\rangle$ is defined by:

$$SDB|\alpha = \{s / s = suffix(\gamma, I, ws)\}, \gamma \in SDB$$

otherwise

$$SDB|\alpha = \{s / s = wsuffix(\gamma, I, ws)\}, \gamma \in SDB$$

The database projection on an item is described in Algorithm 1. On each projection, frequent pairs are calculated by using **Find_Frequent_Pairs** function. A pair $(\delta t, I)$ is a combination of a temporal interval and an item. The interval is the period in which the item occurs in the projection. The depth of this interval is at most equals to the window size. Once a frequent pair $(\delta t, i)$ is identified, it is concatenated to the last generated pattern. If the resulting pattern satisfies the time constraints, then it is a new frequent sequence. This new pattern generates a new iteration: the projection $(SDB|\alpha)|(\delta t, i)$ is computed and becomes the new research space of frequent pairs.

Example 6. *Let us consider example 5. The projections of* s_1 *and* s_2 *w.r.t.* $\langle(0,A)\rangle$ *provide resp.* $(SDB|_{(0,A)})$*:* $s_1 : \langle(1,C)(2,B)(5,CDE)(6,F)\rangle$, $s_2 : \langle(0,B)(5,DF)\rangle$ *where the frequent pairs are:* $([0,2],B),([5,5],D)$, *and* $([5,6],F)$.

Algorithm 1: Projection.

Input: SDB, $(\delta t, I)$
foreach *sequence s of SDB* **do**
  foreach *itemset IS of s* **do**
    if $I \in I_k$ *and* $t_k \in \delta t$ **then**
      if $IS \setminus \{I\} = \emptyset$ **then**
        add to projection $S = \langle((t_k - ws), I_i)...((t_{k+1} - t_k), I_{k+1}),...((t_n - t_k), I_n)\rangle$;
      else
        add to projection $S = \langle(t_k - ws), I_i)...(0, I_k \setminus \{i\}),((t_{k+1} - tk), I_{k+1}),...((t_n - t_k), I_n)\rangle$;

For $([0,2],B)$ *iteration, the projection* $SDB|_{([0,2],AB)}$ *is:* $s_1 : \langle(-1,C)(3,CDE)(4,F)\rangle$, $s_2 : \langle(5,DF)\rangle$ *where frequent pairs are:* $([3,5],D),([4,5],F)$.

- *For* $([3,5],D)$ *iteration, the concatenation of* $([0,2],AB)$ *and* $([3,5],D)$ *is defined by:*
  - *Items A and B are considered simultaneously and take place in* $[0,2]$.
  - *Item D is directly successive to AB and takes place earlier than three time units after AB, so D holds earlier than the time 5 = (2+3). Moreover, D occurs within 5 time units after AB, so D will be held no later than the time 5 = (0+5).*

Thus, the projection on $([3,5],D)$ *provides* $SDB|_{([0,2],AB)([5,5],D)} = s_1 : \langle(0,CE)(1,F)\rangle, s_2 : \langle(0,F)\rangle$.

The frequent pair $([0,1],F)$. *It provides the frequent sequence:* $\langle([0,2],AB)([5,5],D)([5,6],F)\rangle$ *and the following projection:* $S_1 : \langle(-1,CE)\rangle$. *There is no frequent pair, so no new iteration is executed. The same process is executed for the pairs* $([4,5],F))$, $([5,5],D)$ *and* $([5,6],F)$.

Conclusion. The approach we have presented provides frequent temporal patterns. Their timestamps are in the form of intervals whose widths are adjustable by the user. These intervals allow a time occurrence approximation of events. As GSP (Srikant and Agrawal, 1996), our approach uses also as input: a set of sequences, support threshold and time constraints. The two main differences between both approaches are: (1) the extraction process used. The effectiveness of PrefixSpan over GSP was demonstrated in various works (Fournier-Viger et al., 2008) (Hirate and Yamana, 2006) (Pei et al., 2004). (2) the quality of data. GSP patterns are timeless. In some areas, lack of timestamps represents a major handicap to data understanding and interpretation. In addition, the number of patterns returned by our approach

is more important. Indeed, the application of backward window allows to expand continuation patterns with those containing unordered events on a window size interval. Then, where GSP considers no frequent patterns, our approach searches through backward window redundant information and extracts a frequent pattern. The approach presented in (Hirate and Yamana, 2006) has as input a sequences database, a value of minsupp, time constraints and a time function to align timestamps. This approach and ours use the same process of extracting patterns based on PrefixSpan algorithm. The main difference concerns the amount of data. While the use of the sliding window can group events by degrees relative to the size of the window, the function level has only the events whose timestamps are in the same level. So, we end up with more frequent patterns due to the sliding form of the window, which groups gradually close events. Concrete results are presented in the next section.

Algorithm 2: Principal.

Input: SDB, *minsupp*, mingap, max gap, min_whole_interval, max_whole_interval, ws, Patterns
Patterns = null;
Find frequent items in SDB ;
foreach *frequent item I* **do**
  prefix = (0, I);
  $SDB\,|_{(0,I)}$= Projection(SDB, (0,I), 0) ;
  foreach *pair* $(\delta t_f, I_f)$ *in Find_Frequent_Pairs* $(SDB|_{(0,I)}, C1, C2)$ **do**
    newprefix = concat(prefix, $(\delta t_f, I_f)$);
    if *newprefix satisfies min_whole_interval and max_whole_interval* **then**
      $SDB|_{(0,I)|(\delta t_f, I_f)}$ = Projection($SDB|_{(0,I)}$, (t_f, I_f), ws) ;
      Projection*($SDB|_{(0,I)|(\delta t_f, I_f)}$, *minsupp*,; *mingap*, *maxgap*, *min_whole_interval*,; *max_whole_interval*, *newFprefix*, *FSeq*) ;
      if *newprefix* $\notin$ *Patterns* **then**
        Add newprefix to Patterns ;

5 EXPERIMENTS

In this section, we present a qualitative experimentation of our approach. In a first part, the data used for our experimentation are described. Then, we detail a performance evaluation of the process used by the approaches, to motivate the method that we use to implement our work. In a third part, we compare our implementation to a GSPM implementation of patterns growth process.

Algorithm 3: Projection*

Input: *SDB*, *minsupp*, *mingap*, *maxgap*, *min_whole_interval*, *max_whole_interval*, *prefix*, *Patterns*)
foreach *Pair (f(t), t) in Find_Frequent_Pairs* $(SDB, mingap, maxgap)$ **do**
  newprefix = concat(prefix, (f(t),I));
  if *newprefix satisfies min_whole_interval and max_whole_interval* **then**
    if $support(f(t), I) \geq minsupp$ **then**
      Projection*($SDB|_{(f(t)_p, I_p)}$, mingap, maxgap,; min_whole_interval, max_whole_interval,; newprefix, patterns); Add newprefix to Patterns;

Data Description. We applied our algorithms to real aeronautical data related to a life history of six same aircraft. These data represent missions, reports carried out on different part of the vehicles and equipments maintenance tasks execution. It is organized on temporal sequences. A sequence is built by accumulating successive occurred events on an aircraft between occurrence of a specific maintenance task. Preprocessed sequences, from all vehicles and ended with the application of a same maintenance task, represent lists of temporal events preceding the execution of the task. Extracting patterns from this database consists in identifying commonly usages that lead to the application of this maintenance task. It allows to distinguish maintenance operations that use common root causes. Table 1 represents a sequences history sample for the task op_m1. We used a GSP implementation available in WEKA[1] without any time constraints implementation. We also modified an implementation of (Fournier-Viger et al., 2008) [2] to obtain the GSPM implementation. We modified the same

Table 1: Sample of preprocessed sequences.

ID	Sequences
S_1	⟨(t=0, taxi, sale),(t=223, PARAPUBLIC, sandy), (t=300, EMS, normal), (t=330, report_1),(t=490, PARAPUBLIC, normal),(t=520, **op_m1**)⟩
S_2:	⟨(t=0, PARAPUBLIC, sandy), (t=190, taxi,normal), (t=324, OEM, salt), (t=500, **op_m1**) ⟩
S_3:	⟨(t=0, EMS, normal), (t=190,taxi,salt), (t=340, PARAPUBLIC, normal)(t=390, report_1),(t=400 , **op_m1**)⟩

[1] http://www.cs.waikato.ac.nz/ ml/weka/
[2] http://www.philippe-fournier-viger.com/spmf

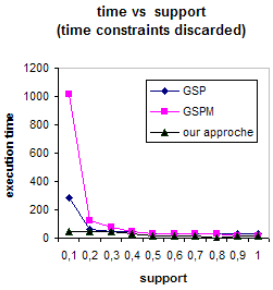

Figure 1: Process evaluation.

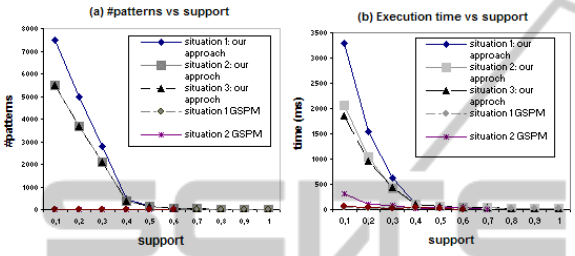

Figure 2: Performance evaluation.

code to implement our approach. First, we evaluate the quality and results provided by our approach compared to those provided by GSP. In a second step, we evaluate the of performances cost of interesting sequences approach compared to GSPM, since the two techniques are based on the same basic algorithm PrefixSpan. We will then assess the quality of the results of these two approaches.

Process Evaluation. We execute the three approaches by discarding time constraints to evaluate the performance of allApriori method compared to pattern-growth method. Figure 1 shows that execution time of pattern-growth (GSPM) is less than All Priori(GSPM). These results reinforce our choice to choose the Prefix Span approach.

Algorithm Evaluation. In this evaluation, we compare execution time and the number of extracted sequences with varying *minsupp*. We compare SPMF with backward window size, our proposed algorithm, and generalized Sequential Patterns Mining with item Interval. We have tested 3 situations:
situation 1 with f(t) = t/2, min gap= 3, max gap = 5 and ws=2, **situation 2** with f(t) = t, min_whole_interval = 3, max_whole_interval = 7 and ws = 0, **situation 3** with f(t)=E(t/2), min gap= 1, max gap = 5, min_whole_interval = 3, min_whole_interval = 7 and ws = 2.
As shown in Figure 2(a), using our backward sliding window allows to have a large number of patterns. The number of extracted sequences increases exponentially as *minsupport* decreases. Our approach is interesting in high values of *minsupport* because it provides patterns that are not extracted with GSPM. For the lowest support values our approach execution time is higher than that of GSPM (shown in Figure 2(b)). It is due to the greater number of possible continuations provided by the backward window size.

Table 2: Result patterns.

GSPM results	support
(fn(t)=0, taxi)(fn(t)=3, op_m3)	0,5
Interesting sequences results	
(t in [0.0, 0.0], taxi)(t in [2.0, 4.0], op_m3)	1
(t in [0.0, 0.0], taxi)(t in [5.0, 7.0], op_m3)	0,7
(t in [0.0, 0.0], taxi)(t in [4.0, 6.0], op_m3)	0,8

Patterns Quality Evaluation. Table 2 shows the resulting patterns provided by GSPM in the first column and by our algorithm in the second one. We can see that when GSPM provides a unique pattern our approach shows 3 because of the sliding windows. It allows the user to see all frequent possible combinations of patterns regarding to the user parameters (windows size). So, our interesting sequences approach has more exhaustive representation of the data.

6 CONCLUSIONS

In this paper, we presented a new definition of interesting sequences based on the principle of sliding windows which takes into account any order within transactions. This definition is important for sequence data that do not require high timing precision. It allows to gather as much information as possible to represent the actual data in a richer way without loss of information. The definition presented here is integrated into the process of the algorithm (Fournier-Viger et al., 2008) and provides satisfying results quality. Future work will focus on improving performance. Another issue is the huge number of extracted sequences. Extracting *maximal* interesting sequences may be a solution to reduce the result size without information loss. This approach is currently applied on aeronautic vehicles life history to identify common sequences preceding maintenance operations. These same behaviors will be used for better maintenance management and vehicle stops forecasting.

REFERENCES

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *VLDB*

Proceedings.

Fournier-Viger, P., Nkambou, R., and Nguifo, E. M. (2008). A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. In *Proceedings of the 7th MICAI Conf.)*.

Hirate, Y. and Yamana, H. (2006). Generalized sequential pattern mining with item intervals. *Journal of Computers*, 1(3).

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng.*

Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Proc. of EDBT*.