Relaxing Time Granularity for Mining Frequent Sequences

Asma Ben Zakour, Sofian Maabout, Mohamed Mosbah, and Marc Sistiaga

Abstract. In an industrial context application aiming at performing aeronautic maintenance tasks scheduling, we propose a frequent *Interval Time Sequences* (ITS) extraction technique from discrete temporal sequences using a sliding window approach to relax time constraints. The extracted sequences offer an interesting overview of the original data by allowing a temporal leeway on the extraction process. We formalize the ITS extraction under classical time and support constraints and conduct some experiments on synthetic data to validate our proposal.

1 Introduction

Sequential patterns mining is an important data mining task. It handles several kind of sequential information like network intrusion detection [Srinivasulu et al., 2010], identification of behavior trends [Rabatel et al., 2009] and tandem repeat DNA sequences [Ceci et al., 2011]. According to the target application, different forms of sequences may be extracted e.g., timestamped (see [Yi-Cheng et al., 2010] and [Fournier-Viger et al., 2008]), summarized [Pham et al., 2009], composite (see [Ceci et al., 2011]) and multidimensional sequences (see [Rabatel et al., 2009] and [Plantevit et al., 2007]). Considering timestamped patterns, time representation and time granularity are both more or less relevant regarding the application domain. We introduce through the work presented in this paper a new form of timestamped sequences and propose *ITS-PS*: an algorithm enabling their extraction. The sequences we want to extract aim to characterize aeronautic usage behaviors with respect to their impact on maintenance tasks application. They are intended to be used to predict maintenance application in order to perform its scheduling process. For example, for aircraft lives data, let V_i refer to the flight i,

Asma Ben Zakour · Sofian Maabout · Mohamed Mosbah · Marc Sistiaga
LaBRI, University of Bordeaux, CNRS UMR 5800, France and
2MoRO Solutions, Bidart, France

F. Guillet et al. (eds.), *Advances in Knowledge Discovery and Management,*
Studies in Computational Intelligence 527,
DOI: 10.1007/978-3-319-02999-3_4,

M_j refer to a maintenance task j and $\mathscr{S} = \{S_1, S_2\}$ be a set of historic sequences where $S_1 = \langle (0,V_1)(2,V_2)(3,V_3)(5,M_1)\rangle$ and $S_2 = \langle (0,V_1)(2,V_3)(3,V_2)(6,M_1)\rangle$. Let the minimal support constraint be equal to 2. Our method returns the sequence: $\langle ([0,0]V_1)([2,3]V_2\,V_3)([5,6]M_1)\rangle$. Its meaning is as follows: "If flight V_1 occurs, followed by both flights V_2 and V_3 *in any order* but in a time interval $[2,3]$ after V_1 then, maintenance task M_1 is performed in a time lying in the interval $[5,6]$ after V_1". Such a pattern allows to group V_2 and V_3 in the same relevant behavior.

For this propose, extracted patterns must convey three criteria: (1) the frequency of events chronology in order to describe frequent usage behaviors, (2) timestamped sequences to ensure relevance and precision of maintenance prediction and (3) relaxation of local order of events, i.e., if two events occur in a close time interval, then the chronology of their respective occurrences could be considered as irrelevant, then they may be considered as co-occurring.

To fulfill those three criteria we propose to *merge* temporally close and consecutive events (associated to a discrete timestamp) into an unique set of simultaneous events associated with an interval timestamp. This interval reflects an uncertainty on the occurrence time of events. The "closeness" of events is managed via a user defined maximal sliding window size. In the previous example, events V_2 and V_3 have been merged and timestamped with $[2,3]$ following the application of a sliding window size equal to 1. Note that V_1 cannot be merged with them because its two occurrences are "far" from those of V_2 and V_3. Related methods (some of them are presented in section 2) do not allow extracting such information. For instance, the *GSP* algorithm proposed in [Agrawal and Srikant, 1996] applied on the sequences of the previous example with the same minimal support constraint and window size, extracts the sequence: $\langle (V_1)(V_2\,V_3)(M_1)\rangle$. Even if it contains the same events chronology as ours, it does not provide any temporal information. Hence, the only information it carries is: "flight V_1 is followed by flights V_2 and V_3, in any order, and they are themselves followed by the application of the maintenance task M_1". This frequent sequence cannot be efficiently used by an aeronautic expert who needs to reduce maintenance costs and aircraft interruptions by forecasting the most precise maintenance task application moment since the sequence does not provide any temporal information.

Paper Organization

The following section presents a concise overview of related work, especially the difference between our method and other approaches extracting interval timestamped sequences. Then, we formally define the semantics of sequences with uncertainty time intervals. Section 4 details the extraction process. We conclude our work by comparing our approach with an existing method, the *GSPM* algorithm proposed in [Hirate and Yamana, 2006]). Finally, we present some avenues for future work.

2 Related Work

Several works found in the literature deal with grouping events and frequent interval sequences extraction. The approach proposed in [Pham et al., 2009] merges some sequences events by using a sliding window. Grouping is performed during a pre-processing step, and then an extraction algorithm is applied. The resulting grouped events are however timestamped with a discrete time reference which is an arbitrary choice motivated by treatment simplicity. This represents an information loss w.r.t events occurring times. Moreover, applying the sliding window during a pre-processing phase increases the size of initial sequences since several grouping possibilities for the same sequence. Hence this introduces an ambiguity on the support counting.

Other approaches consider the initial data as timestamped with intervals. These intervals represent *duration* times not uncertainty about the exact discrete occurrence time. [Giannotti et al., 2006] extracts frequent sequences by using an *Apriori* like algorithm [Agrawal and Srikant, 1996]. It first identifies frequent patterns apart from timestamps. Then, for an extracted frequent pattern, it intersects intervals events occurrences in order to provide a succession of intervals associated with the frequent sequence.

In [Guyet and Quiniou, 2011], a timestamped sequence is represented by a hypercube whose axes are the sequence events. The similarity between sequences is expressed by hypercubes intersection volume. Sequences are grouped using this similarity. If there are enough grouped sequences, then a representative one is extracted and considered as a interesting pattern.

Extraction algorithms presented in [Wu and Chen, 2007], [Yi-Cheng et al., 2010] use Allen's interval relationships [Allen, 1983]. A *PrefixSpan*-like [Pei et al., 2001] algorithm is applied on interval timestamped sequences. Both algorithms results presented in [Wu and Chen, 2007] and in [Yi-Cheng et al., 2010] consist in relationships sequences between events and not on timestamped sequences. To the best of our knowledge, the closest work to ours is the one of [Hirate and Yamana, 2006]. Authors extract frequent sequences with interval timestamps from discrete timestamped sequences. They use a *level function* which is actually a non sliding window. Intuitively, events belonging to the same time interval are merged. But since close events may belong to consecutive but different intervals, they cannot be merged. This is due to the fact that the window is fixed.

Concerning the extraction technique itself, we find two main procedures in the literature: the first one is level wise, or breadth first, like *A priori* technique [Agrawal et al., 1994; Agrawal and Srikant, 1995]. This method has been used in many works, for instance, in [Rabatel et al., 2009] and [Agrawal and Srikant, 1996] it was applied to discrete temporal sequences and in [Giannotti et al., 2006] was applied to interval temporal sequences. The principal limitations of the Apriori extraction approach are (1) the number of generated candidates and (2) the number of the whole database scanning which is equal to the number of the levels used during the extraction process. Its principal advantage is its relatively low memory consumption since it maintains only one copy of the database in the main memory. The breadth

first method uses a *divide and conquer* strategy by progressively reducing the search space and selecting at each step 1-patterns (candidates of size 1). Each such selected 1-pattern P is actually a witness of an $i+1$-pattern $Q.P$ where Q is previously evaluated. Evaluating P, e.g its support, is equivalent to that of $Q.P$ because the former is performed in a *projected* data set, i.e., the part that we already know that it contains Q. Hence, intuitively, during the computation advancement, the underlying data is progressively reduced while the patterns keep their size equal to 1 making the evaluation as simple as possible. This second strategy has been used in e.g., [Hirate and Yamana, 2006; Pei et al., 2001; Yi-Cheng et al., 2010; Fournier-Viger et al., 2008; Wu and Chen, 2007; Guyet and Quiniou, 2008].

[Li et al., 2012] used it in order to approximate the set of close patterns extracted from a long sequence by introducing the gap constraint.The main limitation of this approach is its relative great memory consumption since at each pattern extension a physical projection of the database is created. Its advantages are (1) the fewer number of candidates generated at each step and (2) the increasingly reduced data to be scanned during the extraction process. We adopt this second method for applying our algorithm which is inspired by *PrefixSpan* proposed in [Pei et al., 2001].

3 Preliminaries

We first recall some of the standard definitions regarding simple temporal sequences as formulated in previous works, e.g., [Hirate and Yamana, 2006], [Pei et al., 2001] and [Fournier-Viger et al., 2008]. Let $\omega = \{e_1, e_2, \ldots, e_k\}$ be a set of events. A transaction is defined a set of events supposed as occurring simultaneously. A temporal sequence is a succession of chronologically ordered transactions. Each transaction in a temporal sequence is associated with a discrete timestamp, it is denoted by $S = \langle (t_1, I_1), (t_2, I_2) \ldots (t_n, I_n) \rangle$, $n \in \mathbb{N}$ where $\forall 1 \leq i \leq n$, I_i is a transaction and t_i its timestamp. A timed sequences database is a set of temporal sequences where each of them is identified by a unique identifier denoted by *id_sequence*.

Definition 1. Subsumption Let $S' = \langle (t'_1, I'_1), (t'_2, I'_2) \ldots (t'_m, I'_m) \rangle$ and $S = \langle (t_1, I_1), (t_2, I_2) \ldots (t_n, I_n) \rangle$ be two temporal sequences. S subsumes S' iff there exist $1 \leq i_1 \leq \cdots \leq i_m \leq n$ with $I'_1 \subset I_{i_1} \ldots I'_m \subset I_{i_m}$ and $t'_1 = t_{i_1}, \ldots, t'_m = t_{i_m}$.

If S subsumes S' we also say that S is a super-sequence of S'. The support of a sequence S in a database sequences D is the percentage of sequences from D which are super-sequences of S. It is denoted by $support_D(S)$. S is said frequent if its support is greater than a fixed minimum threshold *minsupp*.

Now, we define uncertainty interval temporal sequences. We recall that sequences with interval timestamps consider the transaction's interval as an uncertainty during which the transactions events do occur. If we consider for example the 1-sequence $S = \langle ([t_b, t_e], e) \rangle$, intuitively S means that: "the event e occurs punctually between times t_d and t_e".

Definition 2 (uncertainty Interval Temporal Sequences (ITS)). An n length interval temporal sequence S (ITS) is denoted by:

$$S = \langle([m_1, M_1], I_1), ([m_2, M_2], I_2) \ldots ([m_n, M_n], I_n)\rangle$$

where $([m_i, M_i], I_i)$ is a transaction with interval timestamp such that:

- $m_i \leq occurrence\_time(e_j) \leq M_i$. for all $e_j \in I_i$;
- An interval temporal sequence is consistent if, $m_i \leq m_{i+1}$ and $M_i \leq M_{i+1}$;
- $m_1 = 0$, i.e., timestamps in S are relative to m_1.

Example 1. Let $S1 = \langle([0,1], A)([2,2], BC)\rangle$. It means: "$A$ occurs between time points 0 and 1, B and C occur simultaneously between 1 and 2 temporal units after A". The lower bound of the interval associated to A is the time reference of $S1$. Each of B and C, occur *at most* $2\ (2-0)$ time units after A and *at least* $1\ (2-1)$ time unit after A.

$S2 = \langle([0,3], A)([1,2], B)([2,5], C)\rangle$ is not consistent since the upper bound of the second interval is lower than the upper bound of the first interval$(2 < 3)$.

One may note that Interval Temporal Sequences are special cases of classical temporal sequences. Indeed, $S = \langle(t_1, I_1) \ldots (t_n, I_n)\rangle$ is equivalent to:

$$ITS(S) = \langle([t_1, t_1], I_1), \ldots, ([t_n, t_n], I_n)\rangle$$

In order to fit temporal parameters on extracted sequences and patterns formulation needs, we consider temporal constraints. They aim to: (i) set a maximum threshold of uncertainty, (ii) control minimum and maximum temporal delays between successive transactions and (iii) control the minimum and maximum temporal whole pattern length.

- *Gap* controls the earlier delay between two successive transactions and fix two thresholds: (1) the *mingap* represents the minimum delay below which the successive transactions are considered as too close to represent significant dissociated events and (2) the *maxgap* is the maximum delay between two transactions above which they are considered as too far to be directly correlated (strictly consecutive). Let *SI* be an n length ITS. *SI* satisfies the temporal constraints: *mingap*, *maxgap* if and only if $\forall\ 2 \leq i \leq n$:

$$mingap \leq (m_i - M_{i-1}) \leq maxgap$$

- *Whole_interval* controls the whole length time of a sequence by fixing two thresholds: (1) the *min_whole_interval* fixing the minimum temporal extent of the sequence below which the behavior conveyed by the sequence is not considered as complete and (2) the *max_whole_interval* fixing the maximal temporal extent of the sequence above which the behavior conveyed by the sequence is considered as more than a single one. The *Whole_interval* regulates the time length of a sequence such that for *SI* an n length ITS, *SI* satisfies the temporal constraints: *min_whole_interval*, *max_whole_interval* iff:

$$min\_whole\_interval \leq |m_1 - M_n| \leq max\_whole\_interval$$

- *Sliding Window* enables grouping successive transactions into a single one timestamped with an interval. The size of the sliding window fixes a maximum group spreading and so the maximum interval width. The window size regulates a maximum uncertainty threshold. Let SI be an n length ITS. Then SI satisfies the window size ws iff $\forall\ 1 \leq i \leq n$

$$|M_i - m_i| \leq ws$$

Example 2. Consider the ITS $S = \langle([0,1],A)([2,3],BC)([6,10],D)\rangle$ and the time constraints $mingap$ and $maxgap$ respectively equal to 2 and 3. SI does not satisfy $mingap$ because $m_2 - M_1 = 2-1 = 1 \leq 2$. On the other hand, S satisfies $maxgap$ since for all its successive transactions the maxgap constraint is satisfied ($m_2 - M_1 = 2-1 \leq 3$; $m_3 - M_2 = 6-3 \leq 3$). For a sliding window size equal to 3, S is not valid since $M_3 - m_3 = 10-6 > 3$. For a sliding window constraint fixed to 4, S satisfies it by all its timestamps.

The temporal constraints allow us to manage the temporal parameter into an ITS; they control minimum (respectively maximum) temporal leeway between two successive transactions since the correlation between both of them can be meaningful. Actually, minimum (respectively maximum) gap avoids considering too close (respectively too far) transactions as successive. These constraints are used in several algorithms, e.g., [Agrawal and Srikant, 1996; Fournier-Viger et al., 2008; Rabatel et al., 2009; Li et al., 2012]. In the same way, the *whole_interval* constraint fixes a minimum (respectively maximum) threshold for the whole sequence duration in order to guarantee a meaningful correlation between the transactions belonging to the same sequence[Hirate and Yamana, 2006; Fournier-Viger et al., 2008]. On the other hand, the sliding window manages a balance between the events grouping and uncertainty of their occurrences [Agrawal and Srikant, 1996; Rabatel et al., 2009].

In order to combine these temporal constraints in a consistent manner, we fix the following relationships between them:

$$ws < mingap;\ \ mingap \leq maxgap$$

Temporal constraints enumerated above are fixed by the user in order to extract relevant ITS. In the rest of this section we focus our work on the application of the *ws* constraint.

3.1 Merging Sequences

In this section, we define a $\diamondsuit$ operator that merges successive transactions belonging to the same sequence. For an ITS, $\diamondsuit$ starts from a position j and merges spreading transactions covered by a window size until the last transaction of the ITS. Hence, $\diamondsuit$ has three parameters: a sequence S, a position j in S and a window size ws. More formally:

Definition 3. Let $S = \langle([m_1,M_1],I_1)([m_2,M_2],I_2)\dots([m_n,M_n],I_n)\rangle$ be an ITS, $j < n$ an integer and a window size ws. Then the $\diamondsuit_{ws}$ operator is defined by:

$$\diamondsuit_{ws}(S,j) = S' = \langle([m'_1,M'_1],I'_1)([m'_2,M'_2],I'_2)\dots([m'_n,M'_n],I'_k)\rangle$$

- where $\forall 1 \leq i < j : ([m'_i,M'_i],I'_i) = ([m_i,M_i],I_i)$;
- $\exists\, j \leq l_j \leq l_{j+1},\dots l_i \cdots \leq l_{k-1} \leq n$ such that:
  - $I'_j = \cup_{p=j}^{l_j} I_p$; $\dots$ $I'_i = \cup_{p=l_{i-1}+1}^{l_i} I_p$; $\dots$ $I'_k = \cup_{p=l_{k-1}+1}^{l_n} I_p$,
  - $m'_j = m_j, M'_j = M_{l_j}, \dots, m'_i = m_{l_{i-1}+1}, M'_i = M_{l_i}, \dots m'_k = m_{l_{k-1}+1}, M'_k = M_n$
  - $|m_j - M_{l_j}| \leq ws$; $\dots |m_{l_{i-1}+1} - M_{l_i}| \leq ws$; $\dots |m_{l_{k-1}+1} - M_n| \leq ws$.

Example 3. Consider $SI = \langle([0,2],A)([1,2],B)([3,5],C)([4,6],D)\rangle$ and a window size $ws = 3$. Then $\diamondsuit_3(SI,1) = \langle([0,2],AB)([3,6],CD)\rangle$, it applies the grouping operator $\diamondsuit$ w.r.t ws =3 from the first transaction of SI to the last one. Events from the first (respectively last) couple of transactions are grouped and their intervals merged. Since both transactions are spread into the window size and $(2-0) \leq 3$ (respectively $(6-3) \leq 3$)). We note that $\diamondsuit_3(SI,2) = SI$ it applies the grouping operator $\diamondsuit$ w.r.t ws =3 from the second transaction of SI to the last one. Actually, for the start grouping position 2, the second and third transactions cannot be merged since their unified interval is too large regarding the window size. Finally, $\diamondsuit_3(SI,3) = \langle([0,2],A)([1,2],B)([3,6],CD)\rangle$ and $\diamondsuit_3(SI,4) = SI$.

Remark 1. The resulting sequencing of the application of $\diamondsuit_{ws}$ operator may merge transactions containing the same item. Indeed, it may happen that an item appears in two successive transactions of the initial sequence. If these transactions are merged, then the item will appear only once (transactions are sets). However, because of the window size condition, we know that both occurrences of the same item take place in a time interval at most equal to ws. Hence, the time interval associated to the item in the new sequence does include both initial timestamps. Moreover, the fact that it appears only once does not affect its support in the database since the support is the number of sequences of the database that support an item. Therefore, replacing two occurrences by one it the same sequence does not incur any loss of information regarding the support measure.

Now we define the $\overbrace{\diamondsuit}$ operator which for an n length ITS and a sliding window of size ws, provides a set of ITS's. It is the set of results of all applications of the $\diamondsuit$ operator on a n length sequence (applied by successively starting the merge from the first transaction to the last one , $j \in [1,n-1]$). Intuitively $\overbrace{\diamondsuit}$ merges successive transactions by sliding the window size along the input sequence. It provides the set of all summarized sequences that represent the input one.

Definition 4. Let $S = \langle([m_1,M_1],I_1)\dots([m_n,M_n],I_n)\rangle$, ws be a window size and $S_j = \diamondsuit_{ws}(S,j) \forall 1 \leq j < n$. Then:

$$\overbrace{\diamondsuit}^{ws}(S) = \{S_1, S_2, \dots, S_{n-1}\}$$

Example 4. Let $S = \langle([0,2],A)([1,2],B)([3,4],C)([4,6],D)\rangle$ and $ws = 3$. $\overbrace{\diamondsuit_3}(S) = \{\langle([0,2],AB)([3,6],CD)\rangle, \langle([0,2],A)([1,4],BC)([4,6]D)\rangle, \langle([0,2],A)\ ([1,2],B)\ ([3,6]CD)\rangle\}$.

3.2 Support

Now we define the supporting relationship between ITSs. Intuitively, S supports S' also said S' is a sub-sequence of S iff events of each transaction of S' are contained in one (or successive) transaction(s) of S and transactions interval of S' imply (a combination of) S transaction(s) interval(s). Note that the transactions chronology order must be preserved. More precisely,

Definition 5. Let S and S' be two ITS. Let $S = \langle([m_1,M_1],I_1),\ldots,([m_n,M_n],I_n)\rangle$ and $S' = \langle([m'_1,M'_1],I'_1)\ \ldots\ ([m'_k,M'_k],I'_k)\rangle$. S is a super-sequence of S' denoted by $S \sqsupseteq S'$ (equiv. S' is a sub-sequence of S denoted $S' \sqsubseteq S$) if and only if: $\forall([m'_j,M'_j]I'_j) \in S'$ and $e \in I'_j$ there exists $([m_k,M_k]I_k) \subset S$ such that:

- $e \in I_k$
- $[m_k,M_k] \subseteq [m'_j,M'_j]$ (we say that $[m_k,M_k]$ implies $[m'_j,M'_j]$)

Example 5. Let $SI_1 = \langle([0,2]A)([3,4],B)([5,6]C)\rangle$, $SI_2 = \langle([0,4]AB)\rangle$ and $SI_3 = \langle([0,2]A)([3,6]BC)\rangle$. $SI_1 \sqsupseteq SI_2$ since $[0,4]$ implies $[0,2]$ and $[3,4]$ and $SI_1 \sqsupseteq SI_3$. However, $SI_1 \not\sqsupseteq SI_4 = \langle([0,3]A)([2,6]BC)\rangle$ since $[0,2]$ does not imply $[0,3]$.

The support of an ITS w.r.t. a sequence database is the number of sequences in the collection that support the interval sequence.

Definition 6. The support of a ITS SI in a collection D is defined by:

$$supp_D(SI) = |\{S \in D \,|\, S \sqsupseteq SI\}|$$

Recall that temporal sequences are a special case of interval sequences. Thus, $S \sqsupseteq SI$.

4 ITS Extraction

This section describes the extraction process of ITS patterns from discrete temporal sequences by considering a frequency threshold *minsupp*, and the time constraints: *ws*, *mingap*, *maxgap*, *min_whole_interval* and *max_whole_interval*. We detail the *ITS-PS* (uncertainty interval temporal sequences-PrefixSpan) algorithm. It gradually groups frequent close events into a single transaction by applying a sliding window.

The algorithm applies a *pattern growth* approach [Pei et al., 2001] by performing a depth first extraction based on database projections. First, *ITS-PS* identifies the set of 1-patterns (frequent events) from the initial database *SDB* denoted by $L_1 = \{S; S = \langle([m = 0, M = 0], e)\rangle; support(e) \geq minsupp\}$. Then, recursively $i+1$-patterns are identified by extending an i-pattern. Each recursive step i applies two tasks:

- The first task identifies L_1 the set of frequent 1-ITS from the search space. A 1-ITS is considered frequent if: (1) the event of its transaction appears in a sufficient number of sequences of the search space, and (2) the maximum delay between its occurrences timestamps is at most equal to *ws*. Each 1-ITS is concatenated to the pattern extracted at the $i-1$ iteration to provide a frequent *i*-pattern. Then, a new iteration is executed. This step is detailed in the section 4.1
- The second task computes a new projection of the current data on each frequent 1-ITS computed at iteration *i*. Each new search space is a summary of the initial one such that it resumes each sequence that is a super-sequence of the $i+1$-pattern by selecting only sub-sequences considered as continuity of the $i+1$-pattern. The $i+1$-pattern is the concatenation of the *i*-pattern with the 1-ITS. This procedure is detailed in the section 4.2.

The recursive process continues until one of the two following conditions is satisfied: (1) No frequent 1-ITS is identified or (2) the projection procedure provides an empty search space.

Table 1 Example of sequences database *SDB*

SDB	
S_1	$\langle(0,A)(1,B)(2,CD)\rangle$
S_2	$\langle(0,A)(2,D)(3,B)(4,F)\rangle$

Table 2 SDB_A: the projection of *SDB* over $\langle([0,0]A)\rangle$

SDB_A	
S_1	$\langle(1,B)(2,CD)\rangle$
S_2	$\langle(2,D)(3,B)(4,F)\rangle$

Example 6. Let *SDB* be the data described in Table 1, $minsupp = 2$ and $ws = 2$. First, *ITS-PS* identifies frequent events and associates to each one the null interval. In the sample data, it identifies $L_1 = \{\langle([0,0]A)\rangle, \langle([0,0]B), \langle([0,0]D)\rangle\}$. It is the set of first transactions of all other extended frequent ITS.

Let us consider the frequent 1-ITS $\langle([0,0]A)\rangle$ the projection step summarizes *SDB* by retaining only the continuations of *A*, i.e. sub-sequences with *A* as a prefix. Table 2 shows the resulting projection. The extraction process continues and identifies in the new search space the frequent 1-ITS in order to extract longer patterns. Then, SDB_A is projected over each 1-pattern found frequent in it. When all extensions of $\langle([0,0]A)\rangle$ are identified those extending $\langle([0,0]B)\rangle$ will be explored and finally those extending $\langle([0,0]D)\rangle$.

4.1 Selecting Frequent 1-Sequences

Concerning the 1-ITS identification, the application of the sliding window allows us to associate shifted occurrences of the same event occurring in different sequences. This association is made under the condition that the delay between their two farthest occurrences is less or equal to the window size.

Example 7. Let's consider SDB_A from the previous example. B appears twice. These 2 occurrences are counted in the support because the maximal delay between them is less or equal to ws ($3-1=2\leq 2$). Thus, the 1-ITS $\langle([1,3]B)\rangle$ is frequent. In the same manner $\langle([2,2]D)\rangle$ is frequent because D appears in the two sequences of SDB_A and the time delay between the timestamps of its occurrences is less than ws ($2-2=0\leq ws$).

We now introduce the $\oplus$ operator for the concatenation of an ITS and a 1-ITS. Intuitively, when an i-ITS is extended by an 1-ITS, there exist two possibilities of concatenating the second at the end of the first: a T-extension and a S-extension. We describe them as follows:

- The T-extension merges the 1-ITS with the last transaction of the i-ITS. The timestamp of the resulting transaction is the union of both initial intervals. Considering ws, this kind of concatenation is possible when one of the intervals *implies* the other. It is denoted by $\oplus_T$
- The S-extension adds the 1-ITS as the $(i+1)$ transaction of the i-pattern. This extension is possible if the gap constraints and the coherence of the sequence are satisfied. It is denoted by $\oplus_S$

Finally, $\oplus$ defines the concatenation operator through $\oplus_T$ and $\oplus_S$. Both kinds of concatenation depend on the combination of upper bounds and lower bounds of the concerned intervals. Fig. 1 illustrates intervals relationship with respect to the concatenation type.

Definition 7. Let given ws, $S=\langle([m_1,M_1]I_1)\dots([m_n,M_n]I_n)\rangle$ and $S'=([t_b,t_e]I)$ both satisfying ws. The extension of S by S' is defined as follows:

$$S\oplus S'=\begin{cases} S\oplus_T S' & \text{if } [t_b,t_e] \text{ implies } [m_n,M_n] \\ & or [m_n,M_n] \text{ implies } [t_b,t_e] \\ S\oplus_S S' & \text{if } m_n\leq t_b \text{ and } M_n\leq t_e \\ S & \text{otherwise} \end{cases}$$

Example 8. Let $S=\langle([0,1]A)([2,3]B)\rangle$ and $S'=\langle([4,5]C)\rangle$. $S\oplus S'=S\oplus_S S'=\langle([0,1]A)([2,3]B)([4,5]C)\rangle$. If we consider $ws=3$ and $S''=\langle([1,3]D)\rangle$ then $S\oplus S''=S\oplus_T S''=\langle([0,1]A)([1,3]BD)\rangle$.

4.2 Projection

Concerning search space projection, we extend the classical process by using a restricted (to the window size) backward projection. Such projection allows to take

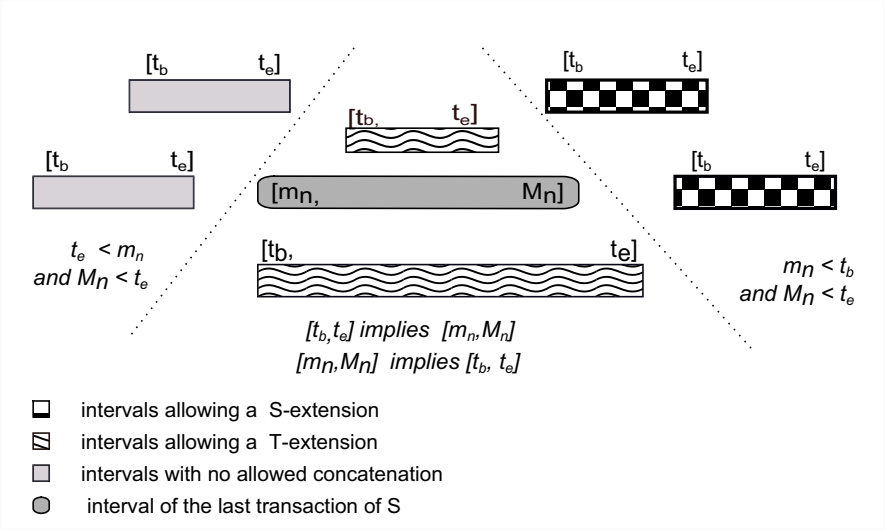


Fig. 1 Illustration of the intervals relationship w.r.t. the extensions types

into account the slide of the window and to consider locally (with regard to the window size) disordered events. This backward exploration permits the selection as an extension of a pattern events occurring frequently around (before or after) its last transaction. Intuitively the new projection holds both of the T-extensions and the S-extensions of the pattern to extend. T-extensions represent close events w.r.t the last transaction of the pattern and to the window size. T-extensions are located in a time delay at most equal to *ws* before and after the last event of the pattern. S-extensions represent events occurring after the pattern in the underlying sequences. The pattern backward analysis has already been used for patterns extension for instance in [Li et al., 2012] to prune the set of close extracted patterns.

Table 3 Projection of the sequences database SDB_A by $\langle([1,3]B)\rangle$

$SDB_{A,B}$	
S_1	$\langle(1,CD)\rangle$
S_2	$\langle(-1,D)(1,F)\rangle$

Example 9. Let us continue the execution of Example 6. Continuations of $\langle([0,0]A)([1,3]B)\rangle$ are identified in the projection of SDB_A by $\langle([1,3]B)\rangle$ denoted by $SDB_{A,B}$ (illustrated in Table 3). The first sequence represents only events appearing after B because there is no event occurring before it (their timestamps w.r.t B are positive). However, in the second sequence, D appears close to and before B (its timestamp w.r.t B is negative). So it is considered as one of its T-extensions and is time stamped

with -1: its time delay w.r.t B. In this new search space, D appears twice and the time delay between its occurrences is less than ws $(1-(-1)=2\leq 2)$. Therefore, the 1-ITS $\langle([-1,1]D)\rangle$ is frequent. Actually, the backwardness of the projection allows us to consider this event frequent despite the fact that in the two sequences it appears on both sides of B. In order to concatenate it to the last extracted pattern, we have first to adjust the temporal reference of $\langle([-1,1]D)\rangle$ w.r.t A. D appears earlier 1 temporal unit after B which turns to appear 3 units after A $(3-1=2)$. It appears at most 1 unit after B which itself appears 1 $(1--1=2)$ units after A. Hence, D is referenced by $[2,2]$ w.r.t the occurrences of A. $\langle([0,0]A)([1,3]B)\rangle \oplus \langle([2,2]D)\rangle = \langle([0,0]A)([1,3]B)\rangle \oplus_T \langle([2,2]D)\rangle = \langle([0,0]A)([1,3]BD)\rangle$.

Let us now consider the extension of $\langle([0,0]A)\rangle$ by the 1-ITS $\langle([2,2]D)\rangle$ frequent in the sequences presented in table 2. $\langle([0,0]A)\rangle \oplus \langle([2,2]D)\rangle = \langle([0,0]A)\rangle \oplus_S \langle([2,2]D)\rangle = \langle([0,0]A)([2,2]D)\rangle$. The projection of SBD_A by $\langle([2,2]D)\rangle$ is illustrated in Table 4. The 1-ITS $\langle([-1,1]B)\rangle$ is frequent and extends the last extracted pattern. We have first to adjust the time reference of the interval associated to B, (-1 +2 =1) for the lower bound and (1+2=3) for the upper bound, then $\langle([0,0]A)([2,2]D)\rangle \oplus \langle([1,3]B)\rangle = \langle([0,0]A)([2,2]D)\rangle \oplus_T \langle([1,3]B)\rangle = \langle([0,0]A)([1,3]BD)\rangle$

Table 4 Projection of SDB_A by $\langle([2,2]D)\rangle$

$SDB_{A,D}$	
S_1	$\langle(-1,B)(0,C)\rangle$
S_2	$\langle(1,B)(2,F)\rangle$

Using the simple backward projection provides in some cases the problem of multiple extractions of the same pattern. This case happens when the proximity between two events is analyzed several times. Actually, when close events can be merged on the same transaction, the order by which the events are considered does not matter because the result is always the same.

Property 1. Let $S=\langle([m_1,M_1]I_1)\dots([m_n,M_n]I_n)\rangle$, ws and $\alpha=\{([m_1,M_1]I_1),\dots,([m_p,M_p]I_p)\}$ the set of the T-extensions of S. Let $m=min(m_1\dots m_p)$ and $M=max(M_1\dots M_p)$ such that $M-m\leq ws$. The ITS provided by successive concatenations of S with all 1-ITS from α in any order are equivalent.

Proof. Let $S_p=([m_p,M_p]I_p)$ and $S_k=([m_k,M_k]I_k)$ two T-extensions of S. S_p and S_k are close to the last transaction of S and then close also from each other. We propose to evaluate the patterns $S\oplus([m_p,M_p]I_p)\oplus([m_k,M_k]I_k)=S\oplus_T([m_p,M_p]I_p)\oplus_T([m_k,M_k]I_k)$ and $S\oplus_T([m_k,M_k]I_k)\oplus_T([m_p,M_p]I_p)$

- Let us first consider the extension of S by $S_p=([m_p,M_p]I_p)$ and then by S_k to obtain: $S\oplus S_p=S\oplus_T S_p=\langle([m_1,M_1]I_1)\dots([m'_n,M'_n]I'_n)$ such that

$$\begin{cases} m'_n=min(m_n,m_p)\\ M'_n=max(M_n,M_p)\\ I'_n=I_n\cup\{I_p\}\end{cases}$$

Then the second concatenation provides: $S \oplus S_p \oplus S_k = S \oplus_T S_p \oplus_T S_k = \langle([m_1, M_1] I_1) \ldots ([m''_n, M''_n] I''_n)$ such that:

$$\begin{cases} m''_n = min(m'_n, m_k) = min(m_n, m_p, m_k) \\ M''_n = max(M'_n, M_k) = max(M_n, M_p, M_k) \\ I''_n = I'_n \cup \{e_k\} = I_n \cup \{I_p, I_k\} \end{cases}$$

- Now we consider the concatenation of S first with S_k and then with S_p: $S \oplus S_k = S \oplus_T S_k = \langle([m_1, M_1] I_1) \ldots ([m'''_n, M'''_n] I'''_n)$ such that:

$$\begin{cases} m'''_n = min(m_n, m_k) \\ M'''_n = max(M_n, M_k) \\ I'''_n = I_n \cup \{I_k\} \end{cases}$$

Then the second concatenation provides: $S \oplus S_k \oplus S_p = S \oplus_T S_k \oplus_T S_p = \langle([m_1, M_1] I_1) \ldots ([m^1_n, M^1_n] I^1_n)$ such that

$$\begin{cases} m^1_n = min(m'''_n, m_k) = min(m_n, m_p, m_k) \\ M^1_n = max(M'''_n, M_k) = max(M_n, M_p, M_k) \\ I^1_n = I'''_n \cup \{e_p\} = I_n \cup \{I_p, I_k\} \end{cases}$$

We can then conclude that $S \oplus_T S_p \oplus_T S_k = S \oplus_T S_k \oplus_T S_p$.

Let S_1 be the result pattern, consider two other *T-extension* $S_u = ([m_u, M_u] e_u)$ and $S_v([m_v, M_v] I_v)$ such that $\{S_v, S_u\} \in \alpha$. By the same manner, if we extend twice (1) by concatenating first S_u to S_1 and then S_v to the obtained sequence and (2) by concatenating first S_v to S_1 and then S_u to the obtained sequence. Then, it is clear that $S \oplus S_v \oplus S_u = S \oplus_T S_v \oplus_T S_u$.

We can finally conclude that whatever the number of T-extensions is, if we extend a sequence S with the same set of T-extensions by considering different orders, the result is always the same. □

In order to avoid multiple extractions of the same ITS k-pattern from an ITS $(i-1)$-pattern, Property 1 is useful. Indeed, we assume that backward exploration does not take into account events already processed as the last element of a i-pattern from the same $(i-1)$-pattern. To cope with this consideration, we suppose a total order $\lhd$ between events such that event e_1 is lower than e_2 w.r.t $\lhd$ (noted $e_1 \lhd e_2$). Let $I = \{e_1, \ldots, e_n\}$. For notation convenience, we note $e \lhd I$ iff $e \lhd e_i$ for $1 \leq i \leq n$ and generalize it to sets of events, i.e., $I_1 \lhd I_2$ iff $\exists e_j \in I_2$ such that $\forall e_i \in I_1$ $e_i \lhd e_j$.

Example 10. Let $\omega = \{A, B, C, D, E, F\}$ and $A \lhd B \lhd C \lhd D \lhd E \lhd F$, then $A \lhd EF$

Considering $\lhd$, we define the prefix and suffix of a sequence. Intuitively, the prefix of S w.r.t S' is the set of sub-sequences of S starting at the beginning of S and supporting S'. The suffix of S w.r.t S' is the set of sub-sequences of S containing the possible continuations of S'.

Definition 8 (*Prefix*). Let $S = \langle (t_1, I_1) \dots (t_n, I_n) \rangle$ and $S' = \langle ([m,M], I) \rangle$. The subsequence $\langle (t_1, I_1), (t_2, I_2) \dots (t_j, I_j) \rangle$ is a Prefix of S w.r.t S' iff $I_j \supseteq I$ and $t_j \subseteq [m,M]$. We denote by Prefix(S,S') the set of prefixes of S w.r.t S'.

We define the $wsufix_{\triangleleft}$ that represents the possible continuations (T-extensions and S-extensions) of a sequence S' on a sequence S by taking into account the window size backward. We use property 1, and the $\triangleleft$ order to avoid the extraction of patterns already discovered. In this way, the $\triangleleft$ order selection is applied on T-extensions which extend on an area span equal to $2ws$ and centered on S. Formally,

Definition 9 ($wsufix_{\triangleleft}$). Let $\omega = \{e_1, e_2 \dots e_m\}$, $S = \langle (t_1, I_1) \dots (t_n, I_n) \rangle$ and $S' = \langle ([m,M], I) \rangle$.

- For $(1 \leq j \leq n)$ such that $I \in I_j$ and $t_j \in [m,M]$ $\langle (t'_k, I'_k) \dots (t'_j, I'_j \setminus \{I\}) \dots (t'_n, I'_n) \rangle$ is a suffix of S w.r.t S' iff:
  1. $\forall i, k \leq i \leq n, t'_i = t_i - t_j$
  2. $t'_k \leqslant (t'_j - ws)$ and $t'_{k-1} > (t_j - ws)$
  3. $\forall i, k \leq i \leq n$ and $t'_i \leq ws$ then $I'_i = I_i \setminus \{e_u | e_u \triangleleft I\}$
- Otherwise, the empty sequence $\langle \emptyset \rangle$ is the suffix of S w.r.t S'.

Fig. 2 illustrates the Prefix and Suffix concepts.

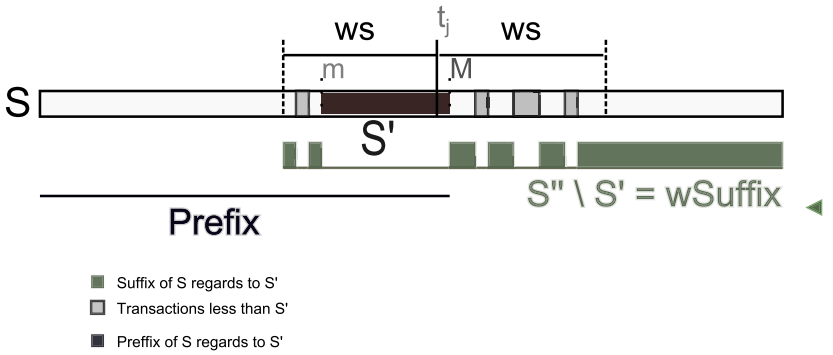


Fig. 2 Illustration of Prefix and Suffix of S w.r.t S'

We denote by $wsuffix_{\triangleleft}(S_1, S')$ the set of suffix of S regards to S'.

We define now the projection considering the new definition of suffix. It resumes a sequence database w.r.t an ITS by calculating the possible continuations of this ITS in the data sequences.

Definition 10 ($wprojection_{\triangleleft}$). Let BDS and $S' = ([m,M]I)$. We define the projection $wprojection_{\triangleleft}$ of BDS by SI as follows:

$$wprojection_{\triangleleft}(BDS|S') = \{S''|S'' = wsuffixe_{\triangleleft}(S,S'), S \in BDS\}$$

Example 11. Let us reconsider Example 3 by applying the $wprojection_{\lhd}$. Let $\lhd$ be the lexicographic order. In SDB_A the following 1-ITS are frequent: $\langle([1,3]B)\rangle$ and $\langle([2,2]D)\rangle$. Considering $\lhd$, $\langle([0,0]A)\rangle$ is first extended by $\langle([1,3]B)\rangle$ and the pattern $\langle([0,0]A)\rangle \oplus \langle([1,3]B)\rangle = \langle([0,0]A)\rangle \oplus_S \langle([1,3]B)\rangle = \langle([0,0]A)([1,3]B)\rangle$ is identified. The projection of SDB_A by $\langle([1,3]B)\rangle$ is then calculated. The result is denoted by $SDB_{A,B}$ and is represented in table (5) of Fig. 3. In $SDB_{A,B}$ the 1-ITS $\langle([-1,1]D)\rangle$ is frequent and allows to identify the extended pattern $\langle([0,0]A)([1,3]BD)\rangle$. Then, $SDB_{A,B}$ is projected by $\langle([-1,1]D)\rangle$ and the result is represented in table (8) of Fig. 3. It does not contain any frequent event. The extraction process extends then the pattern $\langle([0,0]A)\rangle$ by $\langle([2,2]D)\rangle$.

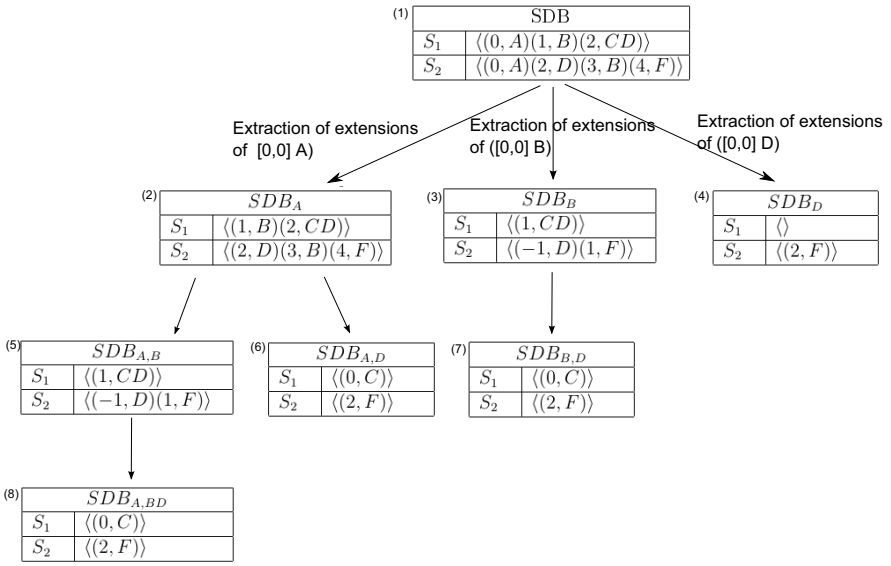


Fig. 3 Extraction steps during the processing of ITS-PS over *SDB*

$\langle([0,0]A)\rangle \oplus \langle([2,2]D)\rangle = \langle([0,0]A)\rangle \oplus_T \langle([2,2]D)\rangle = \langle([0,0]A)([2,2]D)\rangle$ is identified and the projection of SDB_A by $\langle([-1,1]D)\rangle$ is calculated. The result of this last projection is illustrated on Table (6) of Fig. 3. In this sequences database, B doesn't appear because it precedes D wrt the lexicographic order and all its occurrences are close to D in the sequences of the projected database. So in $SDB_{A,D}$ there is no frequent ITS. At this stage of the extraction process, all patterns extending $\langle([0,0]A)\rangle$ are identified. Now the extraction process extends the pattern $([0,0]B)$. First, the initial database (table (1) of Fig. 3) is projected by the pattern, the resulting search space denoted by SDB_B is illustrated in table (3). In this search space the item A does not appear because its close to B w.r.t *ws* and $A \lhd B$. In SDB_B, the 1-ITS $([-1,1]D)$ is frequent and is used to extend $([0,0]B)$. It identifies the pattern $([0,2]BD)$ by adjusting the time reference of both patterns to the smallest timestamp. The extraction projects SDB_B by $([-1,1]D)$, the resulting search space $SDB_{B,D}$ is illustrated in

the table(7) of Fig. 3. $SDB_{B,D}$ does not contain any frequent 1-ITS. The extraction process extends the $[0,0]D)$ pattern. *SDB* is projected by $([0,0]D)$ to obtain SDB_D illustrated in table(4) Fig. 3. It does not contain any frequent 1-ITS to extend the pattern. Here the extraction process is done.

This section presented our algorithm *ITS-PS*. It extracts $(i+1)$-sequences from an i-sequence by progressive reductions of the search database following a pattern growth procedure. The temporal intervals are built by using the sliding window on two levels of the extraction process: the identification of frequent events and the search space projection.

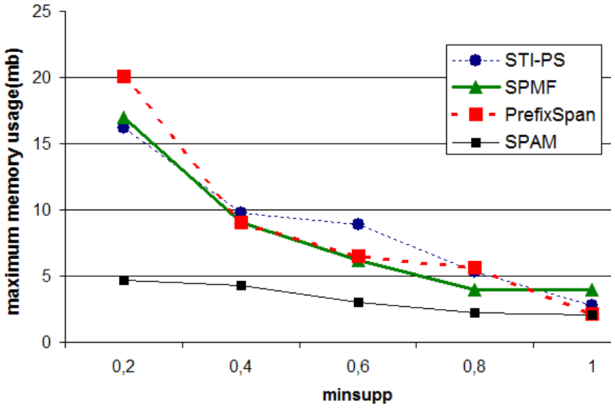
5 Experiments

In this section we evaluate the performances of the *ITS-PS* algorithm. In a first paragraph we analyze its computation time and memory consumption. They are compared with those of other FP-growth algorithms: *PrefixSpan*, *SPMF* and *SPAM*. *PrefixSpan* [Pei et al., 2001] is the pathfinder algorithm of the FP-growth extraction approach which perform a divide and conquer extraction. The *SPMF* algorithm [Fournier-Viger et al., 2008] is an Fp-growth algorithm that applies a time grouping constraint and *SPAM* algorithm [Ayres et al., 2002] enforce bitmap representation for sequences database. In a second paragraph we study the relevance of the *ITS* extracted patterns and compare them with patterns extracted by the *GSPM* algorithm.

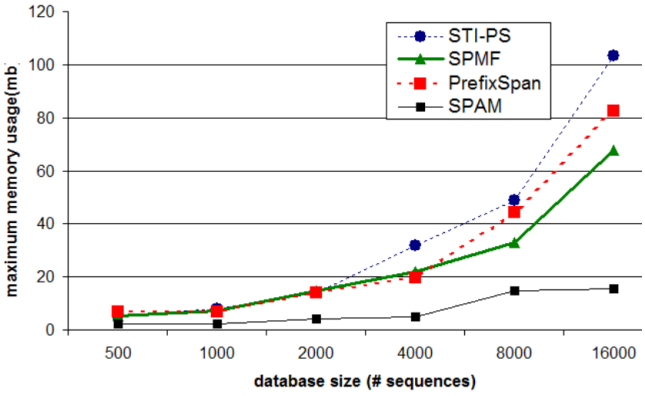
From a theoretical point of view, there is no hope to come up with an extraction algorithm having a worst case complexity less than exponential w.r.t. the number of events appearing in the mined data since the number of returned sequences may itself be in exponential size. Therefore, our algorithm has an exponential worst case complexity.

In order to work over the performance evaluation of the *ITS-PS* algorithm, we analyze in the following its computation time and memory consumption. For this proposal, we compare both criterion to those of other *FP-growth* algorithms (*PrefixSpan*, *SPMF* and *SPAM*) by varying the support threshold and the sequences database size. We use randomly generated data.

Fig. 4 shows the maximum memory consumption of *ITS-PS* compared to those of the algorithms SPMF [Fournier-Viger et al., 2008], PrefixSpan [Pei et al., 2001] and SPAM [Ayres et al., 2002]. Considering the support threshold variation (Fig. 4a), the memory consumed by ITS-PS is similar to that of SPMF and PrefixSpan unless for a 0.6 support value since data events are especially close and frequent. Considering the database size variation (Fig. 4b), the memory consumption behavior of *ITS-PS* algorithm is similar to memory consumed by the other studied algorithms, except that for larger database the memory consumption of SPAM is especially important because of the high cost of database bitmap transformation. So, Fig. 4a and Fig. 4b show that despite the extended projection of *ITS-PS*, the memory consumption of our algorithm has the same trend that the other "classical" *FP-growth* algorithms (except *SPAM* algorithm that use bitmap representation).



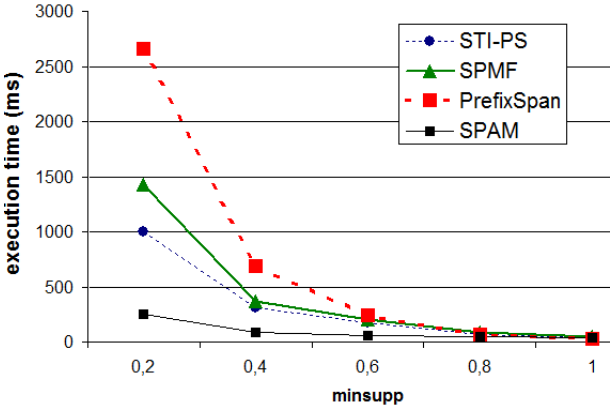
(a) Maximum memory consumption vs support variation



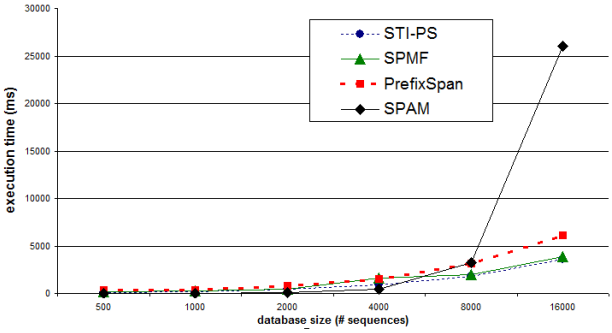
(b) Maximum memory consumption vs database size

Fig. 4 Maximum memory consumption

Fig. 5 displays computation time behaviors of *ITS-PS* compared to those of the algorithms *SPMF*, *PrefixSpan* and SPAM with regards to support threshold variation (Fig. 5a) and database size variation (Fig. 5b). Considering the support variation, Fig. 5a shows that time consumption of *ITS-PS* outperforms those of "classical" *FP-growth* algorithms. However, it is higher than the time consumption of SPAM algorithm, since bitmap representation reduces considerably time consumption. Considering the database variation, Fig. 5b shows that for larger databases ITS-PS needs big execution time since larger projection computation is costly. We can say that the time consumption of the *ITS-PS* algorithm is regular regards to *SPMF* and PrefixSpan ones. We can conclude that execution performances of *ITS-PS* algorithms are regular with respect to the performance of classical *FP-growth* algorithms. Since the extension of projections space results does not affect the cost of the algorithm.



(a) computation time vs support variation



(b) computation time vs database size

Fig. 5 Computation time

In the rest of this section, we evaluate the relevance of the extracted patterns. In the following, we compare patterns extracted by our method with those extracted by *GSPM* presented in [Hirate and Yamana, 2006]. Both algorithms are based on the *PrefixSpan* method. They are different because of the application of distinctive grouping methods: *GSPM* is based on the application of an increment function unlike *ITS-PS* which uses a sliding window. For a meaningful comparison, when the sliding window is fixed to a *ws* value, the *GSPM* step function is set to $f(t) = \lfloor 1/ws \rfloor$. The following example explains the *GSPM* process. More details can be found in the original paper.

Both *GSPM* and *ITS-PS* look for frequent 1-pattern associated with time intervals on the projected databases. However, while time intervals identified by *GSPM* are defined by a step wise function, those looked by *ITS-PS* are defined by merging close occurrences of a frequent item. Thus, for *GSPM* an occurrence of an event e at a timestamp t denoted by (t, e) can be associated with a single 1-pattern.

This timestamp is equal to $f(t)$. However, for *ITS-PS*, a such event may be associated with as many intervals as possible in the margin $[t-ws, t+ws]$ as item e is fairly frequent in this period. A 1-patterns selection such that of *ITS-PS* allows us to broaden the number of continuation possibilities of a pattern by associating a frequent item to a full pallet of intervals.

Example 12. Consider the database $\{S_1 = \langle(0,A)(1,B)\ (2,C)(3,F)(4,B)\ (6,G)\rangle$, $S_2 = \langle(0,A)(1,C)(2,B)(3,D)(4,F)(5,G)\rangle\}$, a threshold support $minsupp = 2$, a sliding window $ws = 2$ and a step function $f(t) = \lfloor t/2 \rfloor$. Timestamps interval provided by *GSPM* are in the form $[2 \times f(t), 2 \times (f(t)+1)[$. The extraction algorithm first identifies the frequent 1-sequences A, B, C, F and G (They are timestamped with null intervals). If we consider the frequent B, the projection provides: $\{S'_1 = \langle(1,C)(2,F)(3,B)(5,G)\rangle$, $S''_1 = \langle(2,G)\rangle$, $S'_2 = \langle(2,F)(3,G)\rangle\}$. In this search space, the pattern $([2,4[,F)$ is identified as frequent since (1) F appears twice: in S'_1 and in S'_2 and (2) for the both occurrences, $f(t) = \lfloor t/2 \rfloor = 1$. Then, in order to identify the interval timestamps to be associated with F, we apply $[2.f(t), 2.(f(t)+1)[$ which provides $[2,4[$. In the same projection, G appears in 3 sequences. In S''_1 and S'_2 with $f(t) = 1$, while in S'_1 its function step value corresponds to $f(t) = 2$. So, only the 1-sequence $([2,4[,G)$ is extracted and $([4,6[,G)$ is not considered so.

Both algorithms are implemented in JAVA using a *PrefixSpan* version[1] proposed in [Fournier-Viger et al., 2008]. The implementation is done on a Windows 7(64) machine, Intel(R) Core(TM) 3 CPU 2.40 GHz with 3 GB RAM.

We compare both extraction results using synthetic data. Data sequences have 7 different events, the average deviation between strictly successive transactions is equal to 3 time units and a sequence average length is equal to 15 transactions. During extraction executions the time constraints *mingap* (respectively *maxgap*, *min_whole_interval* and *max_whole_interval*) are fixed to 0 (resp. 1, 0 and 15). Synthetic sequences database contains 12 sequences since we focus our experimentation on the nature and the number of results that is why we choose a small sequences database. In the following, our goal is focused on validating our algorithm by checking the relevance of its extracted frequent patterns regards to our interested application domain. Considering the time constraints relaxation employed by our approach, we expect that the *ITS-PS* algorithms provide more information because of two point: (1) first, we use the extended projection which is larger than the projection used on *GSPM* and than offer a larger range of continuities by considering the backward projection. (2)Second, the *ws* relaxation allows to group a larger palette of timestamps to identify the *1-ITS*. A such frequent selection provides more options of time intervals and more patterns and the extraction process is deeper.

Fig. 6, Fig. 7, Fig. 8 and Fig. 9 show the number of returned patterns by both algorithms regards to support variation for different values of merging parameters (*ws* and $f(t)$). for each parameter configuration, we measure the number of returned patterns by each algorithm and compute the maximal ones. Fig. 6 (respectively Fig. 7, Fig. 8 and Fig. 9) show that the amount of $ITS-PS$ results is greater than the

[1] `http://www.philippe-fournier-viger.com/spmf/index.php`

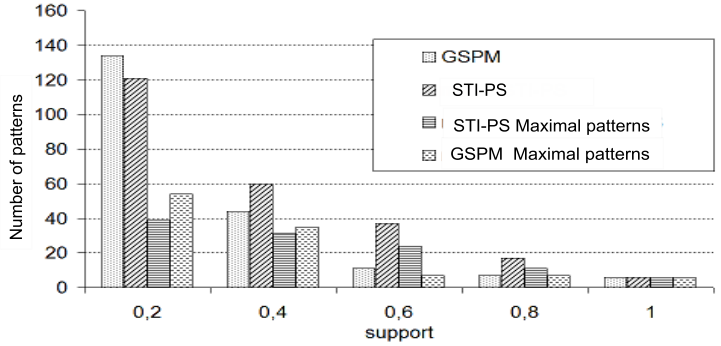


Fig. 6 Number of extracted sequences by varying *minsupp*, *ws* and the step function (WS=1, $f(t) = \lfloor t/1 \rfloor$)

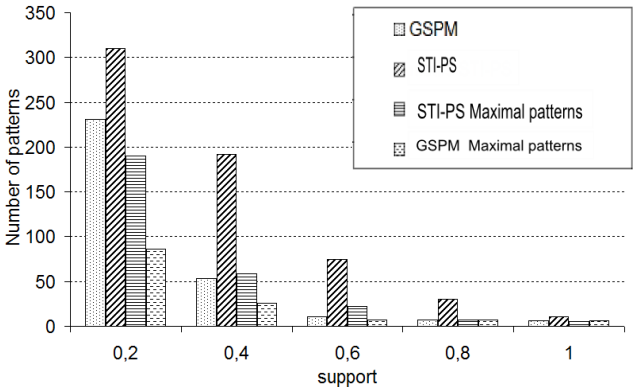


Fig. 7 Number of extracted sequences by varying *minsupp*, *ws* and the step function (WS=3, $f(t) = \lfloor t/3 \rfloor$)

amount of *GSPM* result. Actually, the application of the sliding window gradually groups successive transactions and considers all possible merging combinations. It also allows longer sequences extraction since more events combination are considered frequent from the data sequences and the extraction process stop 'later'. On the other hand, the backward projection employed by *ITS-PS* takes into account more continuation possibilities and so some events see their support growing up.

Considering an aeronautics's historical sequences where each transaction relates flight parameters by indicating (1) the hauled distance done which can be *high.haul*, *med.haul* and *low.haul*.(2) the filling degree of the plane: $Pfull.fill$, $Pmed.fill$, and $Plow.fill$ indicate how full is the plane (3) the third flight parameter is the crossed environment which can be: salt, sand. If this last parameter is normal no information are mentioned. Table 5 presents patterns extracted from such sequences database by both algorithms *GSPM* and *ITS-PS*. The outstanding difference between patterns

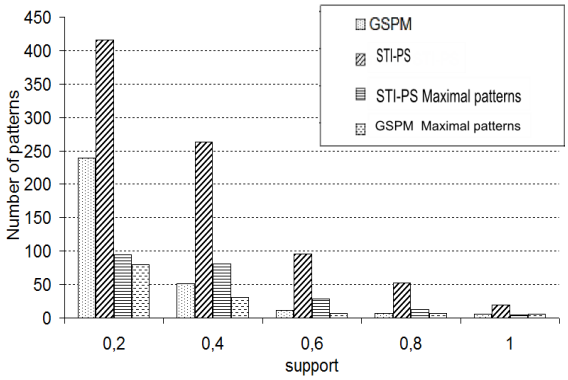


Fig. 8 Number of extracted sequences by varying *minsupp*, *ws* and the step function (WS=5, f(t)= $\lfloor t/5 \rfloor$)

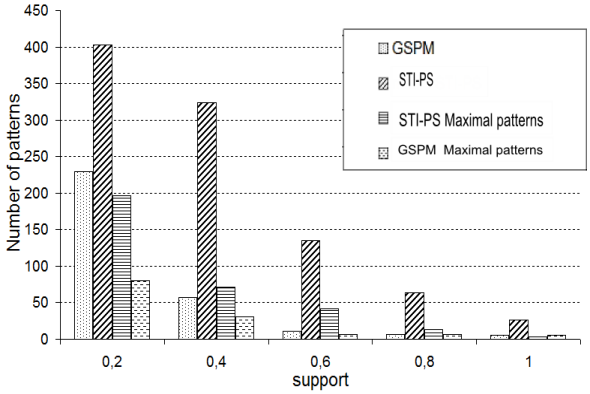


Fig. 9 Number of extracted sequences by varying *minsupp*, *ws* and the step function (WS=7, f(t)= $\lfloor t/7 \rfloor$)

extracted by both algorithms is the timestamp representation. Actually, patterns extracted by *GSPM* are timestamped with discrete values that represent predefined intervals (by the step wise function). However, Patterns extracted by *ITS-PS* convey more flexible temporal interval information where intervals may be narrower than *ws* and cover sliding span. This last point helps us to perform the accuracy of events prediction and provides more precise time laps occurrence.

In Table 5, transactions of the *ITS-Ps* patterns have overlapped intervals, such a representation is made since the *mingap* constraint is aborted. Also, we choose to not apply merging for $\oplus_T$ when intervals are not equal in order to preserve time precision and reduce uncertainty.

The *ITS* patterns convey more realistic time information as regards to data behavior. This information can be handled in order to regulate time precision and uncertainty.

Table 5 Example of patterns extracted from an aeronautical data sequence

GSPM patterns	$\langle(0, sand)(1, verification.mot)\rangle$ $\langle(0, Plow.fill\ med.haut)(1, sand)\rangle$
ITS-PS patterns	$\langle([0,0], Plow.fill)([1,4], long.haul)([4,4], sand)([4,7], verification.mot)\rangle$ $\langle([0,3], Plow.fill)([2,2], med.haut)([5,5], verification.mot)\rangle$ $\langle([0,1], Plow.fill)([3,6], med.haut)([5,7], verification.mot)\rangle$

Table 6 Number of extracted i-sequences (L_i) by varying the window size, the step function depth and fixing *minsupp* to 0.4

	maximal GSPM patterns			ITS-PS maximal						ITS-PS patterns				
ws	L_1	L_2	L_3	L_1	L_2	L_3	L_4	L_5	L_6	L_2	L_3	L_4	L_5	L_6
1	13	21	1	17	39	14	0	0	0	21	0	14	0	0
2	11	16	5	7	47	44	3	0	0	3	19	3	0	0
3	9	12	5	7	53	96	26	3	0	0	30	26	3	0
4	8	12	6	7	53	108	62	9	1	0	23	34	8	1
5	9	13	2	7	55	133	75	9	1	0	26	42	13	1
6	9	14	4	7	52	98	88	38	7	0	26	20	27	7
7	9	19	3	7	51	115	88	21	4	0	24	29	12	4

Table 6 details the number of k-patterns extracted by *ITS-PS* and *GSPM* for a fixed *minsupp* value (equal to 0.4) and different grouping values. We notice that when both methods provide the same patterns length results (correspondence between Fig.6 and Table 6), maximal sequences extracted by our approach are fewer than maximal patterns obtained by *GSPM*. Such situation is illustrated in Example 13. However, when the sequences returned by *ITS-PS* are longer than those provided by *GSPM*, *ITS-PS* maximal sequences are more than *GSPM*'s ones and majority represent longer patterns then those from maximal *GSPM* result. Finally, notice that the number of maximal sequences extracted by our approach is still similar to those extracted by *GSPM*.

Example 13. If we consider Example 12 then the longest maximal sequences extracted by *GSPM* are: $\langle([0,0[, B)([2,4[, F)\rangle$, $\langle([0,0[, G)\rangle$, $\langle([0,0[, A)\rangle$, $\langle([0,0[, C)\rangle$. The only one extracted by *ITS-PS* is $\langle([0,2], ABC)([3,4], F)([5,6], G)\rangle$. The sequence $\langle([0,0[, B)([2,4[, F)\rangle$ extracted by *GSPM* means that "*F appears randomly in* $[2,4[$ *after B*". However, the sequence $\langle([0,2], ABC)([3,4], F)([5,6], G)\rangle$ provided by *ITS_PS* means, among others, that *F appears in the interval* $[3-2=1, 4-0=4]$ *after B*. Given that $[1,4]$ contains $[2,4[$, we can say that the maximal sequence

provided by our approach includes all maximal sequences extracted by *GSPM* by tolerating more uncertainty.

6 Conclusion

This paper presents *ITS-PS*, a sequences extraction algorithm based on the sliding window principle allowing time constraints relaxation. The sliding window gradually merges close transactions (co-occurring events) by considering several merging combinations. The algorithm extracts interval temporal sequences from a collection of discrete temporal ones. The interval timestamps express an uncertainty of the exact moment when transaction events occur. The uncertainty magnitude is managed by the size of the sliding window fixed by the user. The implementation of our algorithm is inspired by that of [Pei et al., 2001]. We compared qualitatively the results of our method to those provided by the *GSPM* algorithm proposed in [Hirate and Yamana, 2006]. It turns that our algorithm provides more and longer sequences than *GSPM* since result patterns convey more information from input data. Actually, when "local" events appear (in the data sequences) with an alternate order are met in the data sequences, *GSPM* (and other extraction algorithms) stops extension of the pattern. However, the *ITS-PS* algorithm extract the same kind of information as frequent and continues its extension. This makes this latter comparing to *GSPM* providing such additional patterns.

Future works will concern the optimization of maximal patterns extraction process. Indeed, due to our relaxation of the chronological sequence of event occurrences, we extract more sequences than other approaches. However, when we restrict our result to the maximal sequences, not only the amount of result is lower than that of the other approaches but it encompasses it. From a practical viewpoint, it is not relevant to first extract all patterns and then select the maximal ones. In order to cope with the huge data manipulated by our targeted industrial application (aeronautic maintenance tasks prediction), we are currently optimizing the present proof-of-concept implementation.

References

[Agrawal and Srikant, 1995] Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceeding of ICDE Conference, Taipei, Taiwan, pp. 3–15. IEEE Computer Society Press (1995)

[Agrawal and Srikant, 1996] Agrawal, R., Srikant, R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 1–17. Springer, Heidelberg (1996)

[Agrawal et al., 1994] Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, vol. 1215, pp. 487–499 (1994)

[Allen, 1983] Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of ACM 26 (1983)

[Ayres et al., 2002] Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, pp. 429–435. ACM (2002)

[Ceci et al., 2011] Ceci, M., Loglisci, C., Salvemini, E., D'Elia, D., Malerba, D.: Mining spatial association rules for composite motif discovery. In: Mathematical Approaches to Polymer Sequence Analysis and Related Problems, pp. 87–109 (2011)

[Fournier-Viger et al., 2008] Fournier-Viger, P., Nkambou, R., Nguifo, E.M.: A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. In: Gelbukh, A., Morales, E.F. (eds.) MICAI 2008. LNCS (LNAI), vol. 5317, pp. 765–778. Springer, Heidelberg (2008)

[Giannotti et al., 2006] Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F.: Mining sequences with temporal annotations. In: Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), pp. 593–597. ACM (2006)

[Guyet and Quiniou, 2008] Guyet, T., Quiniou, R.: Mining temporal patterns with quantitative intervals. In: Workshops Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), Pisa, Italy, pp. 218–227. IEEE Computer Society (2008)

[Guyet and Quiniou, 2011] Guyet, T., Quiniou, R.: Extracting temporal patterns from interval-based sequences. In: IJCAI, Barcelona, Catalonia, Spain, pp. 1306–1311 (2011)

[Hirate and Yamana, 2006] Hirate, Y., Yamana, H.: Generalized sequential pattern mining with item intervals. JCP 1(3), 51–60 (2006)

[Li et al., 2012] Li, C., Yang, Q., Wang, J., Li, M.: Efficient mining of gap-constrained subsequences and its various applications. ACM Trans. Knowl. Discov. Data 6(1), 2:1–2:39 (2012)

[Pei et al., 2001] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns by prefix-projected growth. In: Proceedings of the 17th International Conference on Data Engineering ICDE, pp. 215–224 (2001)

[Pham et al., 2009] Pham, Q., Raschia, G., Mouaddib, N., Saint-Paul, R., Benatallah, B.: Time sequence summarization to scale up chronology-dependent applications. In: EDBT 2008, 11th International Conference on Extending Database Technology, Hong Kong, China, pp. 1137–1146 (2009)

[Plantevit et al., 2007] Plantevit, M., Laurent, A., Teisseire, M., et al.: Extraction de motifs séquentiels multidimensionnels clos sans gestion d'ensemble de candidats. In: EGC 2007: Extraction et Gestion des Connaissances, p. 6 (2007)

[Rabatel et al., 2009] Rabatel, J., Bringay, S., Poncelet, P.: So_mad: Sensor mining for anomaly detection in railway data. In: Perner, P. (ed.) ICDM 2009. LNCS (LNAI), vol. 5633, pp. 191–205. Springer, Heidelberg (2009)

[Srinivasulu et al., 2010] Srinivasulu, P., Rao, J.R., Babu, I.R.: Network intrusion detection using fp tree rules. CoRR, abs/1006.2689 (2010)

[Wu and Chen, 2007] Wu, S., Chen, Y.: Mining nonambiguous temporal patterns for interval-based events. IEEE Trans. on Knowl. and Data Eng. 19, 742–758 (2007)

[Yi-Cheng et al., 2010] Yi-Cheng, C., Ji-Chiang, J., Wen-Chih, P., Suh-Yin, L.: An efficient algorithm for mining time interval-based patterns in large database. In: ACM, Proceedings of CIKM Conference, Hong Kong, China, pp. 49–58 (2010)