Extracting Temporal Patterns from Interval-Based Sequences

Thomas Guyet and **René Quiniou**
AGROCAMPUS-OUEST
65 rue de Saint-Brieuc, CS 84215, 35 042 Rennes Cedex, France
thomas.guyet@agrocampus-ouest,
INRIA, Centre de Rennes - Bretagne Atlantique
Campus de Beaulieu, 35 042 Rennes Cedex, France
rene.quiniou@inria.fr

Abstract

Most of the sequential patterns extraction methods proposed so far deal with patterns composed of events linked by temporal relationships based on simple precedence between instants. In many real situations, some quantitative information about event duration or inter-event delay is necessary to discriminate phenomena. We propose the algorithm `QTIPrefixSpan` for extracting temporal patterns composed of events to which temporal intervals describing their position in time and their duration are associated. It extends algorithm PrefixSpan with a multi-dimensional interval clustering step for extracting the representative temporal intervals associated to events in patterns. Experiments on simulated data show that our algorithm is efficient for extracting precise patterns even in noisy contexts and that it improves the performance of a former algorithm which used a clustering method based on the EM algorithm.

1 Introduction

In many application domains data mining methods are used to extract sequential patterns from data (*e.g.* medicine, agronomy, e-commerce, industry, man-machine interaction). Classical sequential data mining aims mainly at extracting patterns where the temporal dimension lies on a simple ordering of items (occurrences in time of items in patterns). However, such an ordering is often insufficient: a quantitative characterization of event durations or delays between event occurrences is needed to refine extracted patterns and produce knowledge enabling the distinction between specific behaviors. For example, transactions separated by a day, a month or a year are not similarly correlated.

Our goal is to propose a method for mining temporal sequences made of events, stamped with date and duration, and for extracting frequent sequential patterns with temporal intervals characterizing event duration and relative event position in time. In this paper, we present an efficient algorithm, `QTIPrefixSpan`, intertwining sequential pattern extraction and temporal interval characterization by clustering. We compare its computation time with existing algorithms. We also compare the efficiency and quality of several distance measures from symbolic data analysis for extracting the representative temporal features of a sequence.

2 Related work

Many efficient methods have been proposed for extracting interesting sequences from large sequential databases. Most of them deal with instantaneous events represented as points on the time axis and linked by symbolic temporal relations based on simple precedence [Srikant and Agrawal, 1996; Pei *et al.*, 2001; Zaki, 2001]. Sometimes, global constraints, on temporal extents or gaps, can be specified to prune the pattern search space [Lee and De Raedt, 2004; Pei *et al.*, 2007; Masseglia *et al.*, 2009].

To cope with events having a non null duration, one must enrich the representation of patterns with temporal relations, either symbolic or numeric, on intervals. Symbolic temporal relations, such as Allen's temporal relations [Allen, 1983], have been widely used to extend "classical" algorithms. Kam and Fu [Kam and Fu, 2000] or Höppner [Höppner, 2002] extends GSP [Srikant and Agrawal, 1996] to take into account symbolic temporal relations. The main difference lies in the candidate pattern generation phase which handles some Allen's temporal relations among interval-based events. Wu and Chen [Wu and Chen, 2007] pointed out that the major drawback of these algorithms is the ambiguity of the representation of pattern temporal relations and proposed algorithm TPrefixSpan to fix it.

Concerning quantitative temporal pattern extraction, several methods have been proposed for processing point-based time stamped event sequences. Most of these methods extract temporal patterns whose constraints correspond to the envelope, the smallest temporal interval, covering all their instantaneous occurrences. In [Dousson and Duong, 1999], Dousson and Duong have proposed to extract temporal patterns called chronicles – graph patterns constrained by quantitative temporal constraints on the delay between point-based events – from time stamped events. The methods in [Giannotti *et al.*, 2006] and [Yoshida *et al.*, 2000] extract delta-patterns, *i.e.* rules having the form $a \xrightarrow{[t_l,t_u]} b$ meaning that the delay between a and b lies between t_l and t_u. Characterizing sets of instants with intervals enables the anti-monotonicity property over the pattern lattice. So, this property can be used to prune candidate patterns as early as possible.

However, these methods cannot extract quantitative temporal patterns from interval-based time stamped event sequences. QTempIntMiner [Guyet and Quiniou, 2008] and QTPSpan [Nakagaito *et al.*, 2009] solve this problem by considering the quantitative duration of events during temporal pattern mining. QTempIntMiner represents temporal interval sequences by hyper-cubes and extends GSP with a temporal sequence clustering algorithm for extracting typical temporal intervals. But QTempIntMiner uses the EM algorithm for clustering which makes it inefficient. QTPSpan is based on QFIMiner [Washio *et al.*, 2007]. This algorithm mines sets of items with attached interval-based quantitative attributes. Extracted patterns are itemsets with associated quantitative constraints elaborated by clustering the quantitative attribute values linked to the itemsets that support the pattern.

3 Notations

In this section we detail the formal notations and definitions used in the paper for representing the temporal pattern extraction problem introduced above.

3.1 Temporal sequence

Definition 1 (Temporal sequence). *A temporal sequence $\mathcal{S}$ is an ordered set of events, where an event $\mathcal{A} = (A, [l, u])$ is composed of a symbol A and a non empty interval $[l, u]$, where $l, u \in \mathbb{R}, l < u$ are dates.*

$$\mathcal{S} = \{(s_i, [l_i, u_i])\}_{i \in \mathbb{N}_n}, \text{ such that } \forall i, j \in \mathbb{N},\ 0 \leq i < j \leq n,$$
$$l_i < l_j \vee (l_i = l_j \wedge (s_i < s_j \vee (s_i = s_j \wedge u_i < u_j)))$$

The size of a sequence $\mathcal{S}$ is the number of events in $\mathcal{S}$: $|\mathcal{S}| = n$. $\oplus$ denotes the concatenation operation on temporal sequences.

l_i (resp. u_i) is the beginning (resp. end) occurrence date of the interval-based event in the temporal sequence. Events in a sequence are ordered by beginning dates and then by lexicographic order. In contrast to a representation using delay between events, the definition of the values l_i et u_i requires a reference instant for each temporal sequence.

Definition 2 (Symbolic signature). *The symbolic signature of a temporal sequence $\mathcal{S} = \{(s_i, [l_i, u_i])\}_{i \in \mathbb{N}_n}$, noted $\underline{\mathcal{S}}$, is the sequence of its symbols: $\underline{\mathcal{S}} = \{s_i\}_{i \in \mathbb{N}_n}$. The order of symbols in the symbolic sequence is the same as in the temporal sequence.*

Definition 3 (Multi-dimensional interval). *A multi-dimensional interval of dimension n is a tuple of n intervals $I = ([l_i, u_i])_{i \in \mathbb{N}_n}$.*

Definition 4 (Projection over a symbolic signature, π). *The projection of a temporal sequence $\mathcal{S} = \{(s_j, [l_j, u_j])\}_{j \in \mathbb{N}_n}$ over a symbolic pattern signature $\underline{\mathcal{M}} = \{m_i\}_{i \in \mathbb{N}_k}$ of size k is a multi-dimensional interval of dimension k.*

$\pi_{\underline{\mathcal{M}}}(S) = ([l_{j_i}, u_{j_i}])_{i \in \mathbb{N}_k}$, with $(j_i)_{i \in \mathbb{N}_k} \in \mathbb{N}_n$, such that $\forall i,\ i < k,\ j_i < j_{i+1}$

If $\underline{\mathcal{M}}$ is not a subsequence of $\underline{\mathcal{S}}$ then $\pi_{\underline{\mathcal{M}}}(S) = \emptyset$.

$\pi(\mathcal{S})$ denotes the projection of the whole sequence over its proper signature: $\pi_{\underline{\mathcal{S}}}(\mathcal{S}) = ([l_i, u_i])_{i \in \mathbb{N}_n}$.

Example 1. *Let $\mathcal{E}$ be the sequence $\{(A, [1, 2]), (C, [1.5, 4]), (B, [3, 4]), (C, [5, 6])\}$ depicted in Figure 1. Then $\underline{\mathcal{E}} = \{A, C, B, C\}$, $\pi_{\{A,B\}}(\mathcal{E}) = ([1, 2], [3, 4])$ and $\pi(\mathcal{E}) = ([1, 2], [1.5, 4], [3, 4], [5, 6])$.*

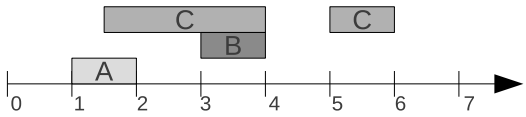

Figure 1: The interval sequence $\mathcal{E}$.

Definition 5 (Temporal sequence dissimilarity). *Let $\mathcal{S}^1$ and $\mathcal{S}^2$ be two temporal sequences. $d(\mathcal{S}^1, \mathcal{S}^2)$, the dissimilarity between $\mathcal{S}^1$ and $\mathcal{S}^2$, is defined by:*

$$d(\mathcal{S}^1, \mathcal{S}^2) = \begin{cases} \infty \text{ if } \underline{\mathcal{S}^1} \neq \underline{\mathcal{S}^2}, & \text{(different symbolic signature)} \\ \delta\left(\pi(\mathcal{S}^1), \pi(\mathcal{S}^2)\right) \end{cases}$$

*where δ is a distance measure on multi-dimensional intervals (*cf. *Section 4.3).*

3.2 Frequent temporal pattern extraction

Definition 6 (Temporal pattern). *A temporal pattern is a representative temporal sequence of a set of temporal sequences (instances of the pattern).*

As [Giannotti *et al.*, 2006], our pattern definition relies on the notion of sub-sequence representativity. A temporal pattern statistically represents a set of temporal sequences, but the notion of representativity is not given explicitly. This should be distinguished from patterns such as chronicles or delta-patterns whose temporal information is built from the temporal enveloppe of covered events in the sequence.

Exact matching of temporal intervals is too constraining to extract interesting patterns from a sequence. ϵ-covering relaxes the matching operation between intervals.

Definition 7 (Temporal pattern ϵ-covering). *A temporal pattern $\mathcal{P} = \{(p_i, [l_i^p, u_i^p])\}_{i \in \mathbb{N}_n}$ ϵ-covers a sequence $\mathcal{S} = \{(s_i, [l_i^s, u_i^s])\}_{i \in \mathbb{N}_m}$, noted $\mathcal{S} \prec_\epsilon \mathcal{P}$, if and only if there exists a projection of $\mathcal{S}$ over $\underline{\mathcal{P}}$ such that its dissimilarity with $\mathcal{P}$ is less than ϵ:*

$$\exists J \subset \mathbb{N}_m, \quad d\left(\mathcal{P}, \left\{\left(s_j, [l_j^s, u_j^s]\right)\right\}_{j \in J}\right) < \epsilon$$

Example 2. *Let consider the three patterns $p_1 = \{(C, [1, 2]), (A, [2, 4])\}$, $p_2 = \{(A, [1, 2]), (C, [2, 4])\}$ and $p_3 = \{(A, [1, 2]), (C, [4, 5])\}$. Let $\epsilon = 1$ and δ be the interval distance CityBlock (sum of absolute difference values between corresponding interval bounds,* cf. *Section 4.3). p_1 does not ϵ-cover $\mathcal{E}$ (*cf. *Example 1) because $\mathcal{E}$ does not contain the sub-sequence $\{C, A\}$. p_2 ϵ-covers $\mathcal{E}$ because δ ($([1, 2], [2, 4]), ([1, 2], [1.5, 4])$) $= 0.5 < \epsilon$. p_3 does not ϵ-cover $\mathcal{E}$ becauses the dissimilarity of each occurrence of p_3 in $\mathcal{E}$ is greater than ϵ: δ ($([1, 2], [4, 5]),\ ([1, 2], [1.5, 4])$) $= 3.5$ and δ ($([1, 2], [4, 5]),\ ([1, 2], [5, 6])$) $= 2$.*

Definition 8 (Frequent temporal pattern extraction). *Given a frequency threshold $f_{min} \in [0, 1]$, a maximal dissimilarity $\epsilon \in \mathbb{R}^+$ and a database of temporal sequences $\mathcal{D}$,*

the frequent temporal pattern extraction consists in building temporal patterns (P_i) such that the frequency of sequences that ϵ-cover a temporal pattern P_i is greater than f_{min}.

A temporal pattern is representative of a set of ϵ-covered temporal sequences. ϵ represents the acceptable temporal distance between a sequence and its representative. This is an additional parameter compared to the classical setting of sequential pattern mining.

4 Algorithm presentation

In this section we will detail algorithm `QTIPrefixSpan` for extracting interval-based sequential patterns. First of all, we introduce a data structure used by this algorithm.

4.1 Data structure

In frequent itemset or sequence mining, the computation of the support or frequency is very time consuming. To avoid systematic scans of the sequence database [Srikant and Agrawal, 1996] have proposed to use a data structure that associates a pattern to the set of its supporting sequences. We have adapted this data structure for counting the sequences which support a temporal pattern m.

For every m, $\mathcal{L}(m) = \{stid_i\}$ gathers the set of its representative instances. Each element of $\mathcal{L}(m)$ is a 3-tuple $stid = \langle tid, pos, is\rangle$ where tid is the identifier of a temporal sequence of $\mathcal{D}$, is is a representative instance of pattern m, *i.e.* a sub-sequence of the sequence tid ϵ-covered by m ($is \prec_\epsilon m$), and pos is the position of the last symbol of is in sequence tid.

$\mathcal{L}(m)$ represents the sequence database projected on m in the sense of PrefixSpan, *i.e.* the set of suffixes (located by pos) of every sequence tid having an instance of m as prefix.

Example 3. *Let $\epsilon = 4$ and $tid = 1$ be the identifier of $\mathcal{E}$ in $\mathcal{D}$). The temporal pattern p_3 (*cf. *Example 2) ϵ-covers two sub-sequences of $\mathcal{E}$ (*cf. *Example 1):*

$$\mathcal{L}(p_3) = \{\langle 1,\ 2,\ \{(A,[1,2]),(C,[1.5,4])\}\rangle,\\ \langle 1,\ 4,\ \{(A,[1,2]),(C,[5,6])\}\rangle\}$$

4.2 Algorithm

`QTIPrefixSpan` (see Algorithm 1) is a recursive depth-first algorithm based on PrefixSpan [Pei *et al.*, 2001]. It explores the extensions of a temporal pattern p, given as input. It makes successive projections of the sequence database to reduce the complexity of computing the support and of clustering the extended temporal patterns.

In the first step (lines 3-5), the algorithm extends the prefix p with one symbol e from S_1 (S_1 is built in a preliminary step). Next, the algorithm performs the projection of the sequence database on the pattern $np = p \oplus e$ (lines 6-10). As the list $\mathcal{L}(np)$ represents the projected database (*cf.* Section 4.1), this step just builds the $stid$ list associated to np. At this stage, the temporal dimension of e is ignored.

In the second step (lines 11-18), the function `TemporalPatternConstruction` is called to build a list of temporal patterns $\{Mr\}$ having the same signature as np and whose temporal features are computed by classifying multi-dimensional intervals (*cf.* Section 4.3). In addition,

Algorithm 1: `QTIPrefixSpan`

input : $\mathcal{D}$: temporal sequence database, f_{min}: frequency threshold, ϵ: pattern similarity threshold, p : temporal pattern
output: FP: set of frequent temporal patterns

```
1  Construction of S_1 // Frequent symbols in D
2  foreach e ∈ S_1 do
3      //Extension of temporal pattern p
4      np = p ⊕ e
5      L(np) = ∅
6      //Projection over the sequence database
7      foreach stid ∈ L(p) do
8          s = D(stid.tid)
9          foreach i: i < stid.pos + 1 ∧ s_i = e do
10             L(np) = L(np) ∪ ⟨stid.tid, i, stid.is ⊕ s_i⟩
11     //Construction of temporal constraints and recursion
12     {Mr} = TemporalPatternConstruction(np)
13     foreach q ∈ {Mr} do
14         if q ∉ already_explored then
15             already_explored = already_explored ∪ q
16             if |L(q)| > f_min * |D| then
17                 ExtentedPatterns =
                   QTIPrefixSpan( D, f_min, ε, q )
18                 FP = FP ∪ ExtentedPatterns
```

this function updates the data structure for each built pattern. Next, the non-frequent temporal patterns are pruned (line 16). Each frequent pattern $q \in Mr$ that has not been already explored is added to the list of explored patterns and `QTIPrefixSpan` is called recursively to extend q.

4.3 Multi-dimensional interval clustering

The algorithm `TemporalPatternConstruction` (*cf.* Algorithm 2) is used in algorithm `QTIPrefixSpan` (line 12) for building the representative temporal patterns of a set of temporal sub-sequences having the same symbolic signature. The method consists in 1) clustering the set of multi-dimensional intervals resulting from the projection of $stid.is$ contained in $\mathcal{L}(p)$, 2) building, for each obtained cluster, a new pattern q having the same symbolic signature as p and the characteristic temporal intervals of the multi-dimensional intervals in the cluster, 3) building $\mathcal{L}(q)$ from $\mathcal{L}(p)$.

For the clustering, line 5, we choose to evaluate two methods: KMeans and Affinity Propagation (AP) [Frey and Dueck, 2007] which we have adapted to the clustering of multi-dimensional intervals. For this purpose, we have selected two similarity measures for clustering objects, here n-dimensional intervals: the Haussdorf distance, noted d_H, the CityBlock distance, noted d_{CB}. Other distance measures between multi-dimensional intervals are given in [Irpino and Verde, 2008]. Given $I = ([l_k^I, u_k^I])_{k\in\mathbb{N}_n}$ and $J = ([l_k^J, u_k^J])_{k\in\mathbb{N}_n}$, we have:

$$d_H(I,J) = \sum_{k=1}^{n} \max\left(|l_k^I - l_k^J|, |u_k^I - u_k^J|\right),$$
$$d_{CB}(I,J) = \sum_{k=1}^{n} |l_k^I - l_k^J| + |u_k^I - u_k^J|$$

Algorithm 2: `TemporalPatternConstruction`

input : p: temporal pattern, $\mathcal{L}(p)$: p sub-pattern instances
output: $\{Mr\}$: set of temporal pattern with symbolic signature $\underline{p}$ temporally representative of instances in $\mathcal{L}(p)$

1 *//Construction of the set of temporal intervals to be clustered*
2 $Ex = \emptyset$
3 **foreach** $stid \in \mathcal{L}(p)$ **do**
4 $\quad Ex = Ex \cup \pi(stid.is)$
5 $\mathcal{C} = clustering(Ex)$
6 *//Construction of temporal patterns*
7 $\{Mr\} = \emptyset$
8 **foreach** $C \in \mathcal{C}$ **do**
9 $\quad q = \left\{ \left(\underline{p}_i, c_i \right) \right\}_i$
10 $\quad \mathcal{L}(q) = \{stid \in \mathcal{L}(p),\ d\left(stid.is, q\right) < \epsilon\}$
11 $\quad \{Mr\} = \{Mr\} \cup q$

In line 10, the list of instances of the temporal pattern q is updated. As with projected databases in PrefixSpan, we avoid an exhaustive scan of the sequence database by searching the instances of the temporal pattern q starting only from its sub-patterns instances. As a precondition of the algorithm, we notice that $\mathcal{L}(p)$ contains the instances of sub-patterns of p: $p(1..|p|-1)$. Thus we build $\mathcal{L}(q)$ from this $stid$ list.

To find the optimal number of clusters for the interval distribution, we test different values of the parameter k for K-Means or s for AP, and choose the most adequate to the data using the Bayesian Information Criterion (BIC).

5 Experiments

In the following experiments, we evaluate the computing performances of the proposed algorithm and assess the quality of extracted patterns on simulated data. We compare the results with those of QTempIntMiner [Guyet and Quiniou, 2008] and of a GSP-like algorithm, so called `QTIAPriori`. This last algorithm uses the same principles as `QTIPrefixSpan` (intertwining sequence mining and interval clustering) but uses an APriori-based breadth-first exploration of the sequence lattice. The algorithms have been implemented in Matlab[1].

5.1 Simulated data generation

We have implemented a temporal sequence generator for producing datasets that can be tuned with respect to the particular features of patterns to be searched (particularly their numerical temporal features) and of the sequence database (number of sequences, noise level). The principle used to generate a temporal sequence database consists in building sequences from temporal pattern prototypes and, next, in adding random sequences (noise) in the sequence database at a rate rP, $0 \leq rP \leq 1$.

A temporal pattern prototype is a set of 5-tuples $\{(E_i, \mu_{b_i}, \sigma_{b_i}, \mu_{d_i}, \sigma_{d_i})\}_{i \in \mathbb{N}_n}$ where E_i is a symbol, μ_{b_i}, μ_{d_i} (resp. $\sigma_{b_i}, \sigma_{d_i}$) are the means (resp. standard deviation) of the beginning date and of the duration of E_i. A prototype specifies a temporal pattern ($\{(E_i, [\mu_{b_i}, \mu_{b_i} + \mu_{d_i}])\}_{i \in \mathbb{N}_n}$) to be discovered. The instances of such a pattern are generated from a gaussian distribution of dates (resp. durations) centered on μ_b (resp. μ_d) with a standard deviation σ_b (resp. σ_d). More the standard deviation is large, more the temporal variations are important and more the extraction of the original pattern will be difficult. To simplify the result presentation, we fix a unique parameter $tN \in \mathbb{R}^+$ for quantifying all the standard deviation values. This parameter quantifies the *temporal noise* of a dataset.

5.2 Results

All the curves presented in the sequel were obtained by averaging the results on several (from 5 to 10) different datasets generated from the same parameters. Where not stated otherwise, the following default parameter values were used: $|\mathcal{D}| = 100$, $rP = 0.4$, $tN = 0.2$, $f_{min} = 0.1$ and $\epsilon = \infty$. We used the Hausdorff distance and looked for the temporal pattern $\{(A, [2, 3]), (B, [5, 5.5]), (C, [8, 10]), (B, [12, 12.5])\}$.

Figures 2-(a) and 2-(b) compare the computation time of algorithms QTempIntMiner, `QTIAPriori` and `QTIPrefixSpan` with respect to the size of the sequence database, on the one hand, and to the size of searched patterns, on the other hand. In both case, we can see that the computation times are exponential. In addition, `QTIAPriori` and `QTIPrefixSpan` are significantly faster than QTempIntMiner.

More precisely (*cf.* Figure 2-(c)), `QTIPrefixSpan` runs twice as fast as `QTIAPriori`, whether it is associated to KMeans or to AP. The KMeans clustering leads to better performances than the AP clustering. `QTIAPriori`-KMeans is even faster than `QTIPrefixSpan`-AP for patterns of size greater than 4. Figure 2-(d) illustrates the ability of `QTIPrefixSpan`-KMeans to cope with complexity. The computation time rises exponentially with respect to the number of sequences but the rate is equal to 1.26, so it is quite close to 1. The mean time for extracting temporal patterns of size 4 from a database of 100000 sequences, corresponding to 600000 symbols, is about 15 min.

The quality of patterns extracted by `QTIAPriori` and `QTIPrefixSpan` can be evaluated with respect to several criteria: i) the presence or absence among the extracted patterns of some pattern that ϵ-covers the prototype; ii) the dissimilarity between the temporal intervals in the prototype and in the extracted patterns, *i.e.* the precision of temporal intervals associated to extracted patterns; iii) the number of extracted patterns having the same symbolic signature as the prototype's, so called "clones"[2]. This last criterion concerns the results readibility: the lower the number of clones, the easier the results analysis will be to the user.

The main factors that impact these criteria are the temporal noise level tN, the dissimilarity threshold ϵ and the minimal frequency threshold f_{min}. These factors are related: if the data are very noisy then a high ϵ will be needed to insure that

[1] The experiments were done on a PC with a Xeon processor (only one core was used), with 8Go of RAM. The source code of the algorithms and of the data generator can be downloaded from `http://www.irisa.fr/dream/QTempIntMiner`

[2] Our method may extract several frequent patterns with the same symbolic signature but with discriminative temporal intervals.

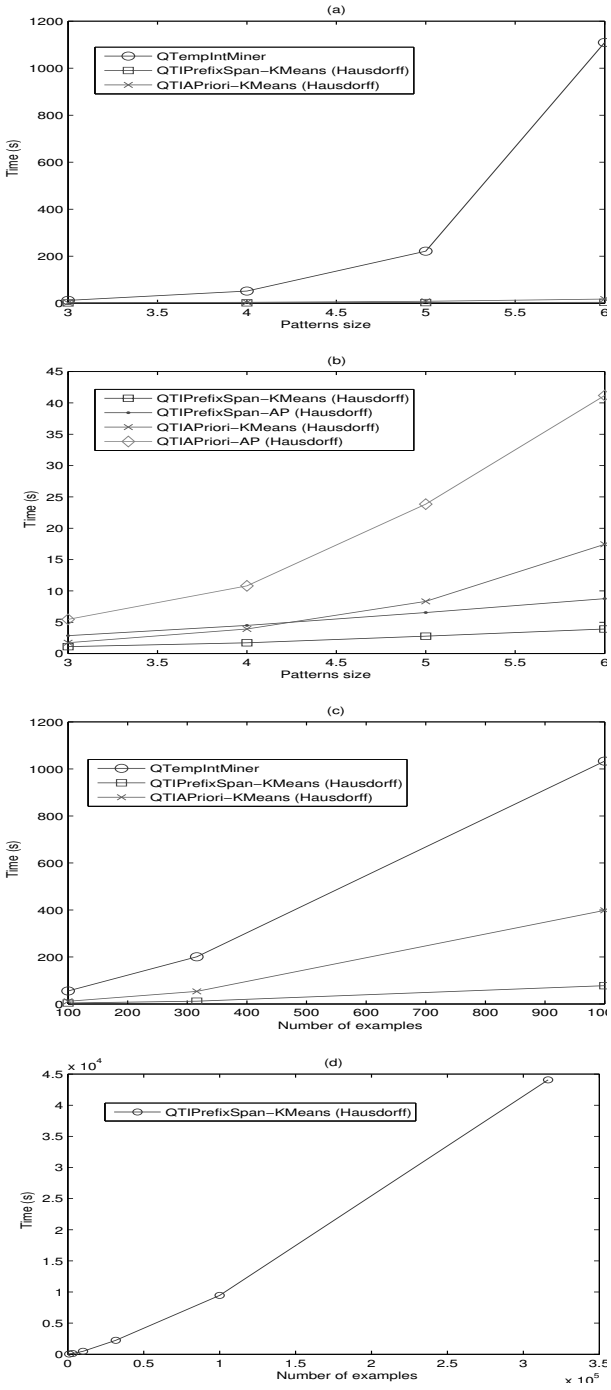

Figure 2: Mean computing time (in seconds) with respect to the size of searched temporal patterns (a and b) and to the number of sequences (c and d).

noisy instances are classified in the same cluster. But if ϵ is too high (with respect to tN) then the clusters will be larger and pruning using f_{min} will be less efficient and this will affect the overall algorithm performance.

For sufficiently high values of ϵ and low values of f_{min} the three algorithms can extract the prototype pattern from the sequence dataset. Moreover, when we add a second prototype having the same symbolic signature but different temporal intervals the two patterns are still correctly extracted. In the latter case, GSP or PrefixSpan can extract only one pattern.

Figure 3 illustrates the quality of patterns that are extracted by the algorithms from simulated data that contains only one prototype among random sequences. Figure 3-(a) gives, for `QTIPrefixSpan`, the number of patterns FP (line 18 of algorithm 1) and the number of prototype clones, with respect to ϵ. When ϵ is small ($\epsilon < .1$), few patterns having the size of the prototype are generated: the expected similarity of pattern instances imposes a too strong constraint with respect to noise. For $.1 < \epsilon < .8$, the number of generated patterns grows very much. Due to the strong similarity that is still required by the value of ϵ, the temporal clustering induces many clusters but with a number of instances that remains below f_{min}. For $.8 < \epsilon < 1.7$, several frequent patterns are extracted. For $1.7 < \epsilon$, the number of extracted frequent patterns stabilizes to 1: ϵ is sufficiently large for considering that the dissimilarity between temporal intervals is quite reasonable.

We can see on Figure 3-(b) that even with noise, the best temporal pattern extracted by the three algorithms is close to the searched prototype. A Hausdorff distance equal to 1 is very low for a temporal pattern where temporal intervals may last 12 time units. For `QTIAPriori` and `QTIPrefixSpan`, when the temporal noise moves up, the precision logically moves down. We can see also, that the choice of the clustering method has more impact on the quality of results than the choice of the search method (GSP or PrefixSpan). KMeans can cope with harder temporal noise than AP: the mean dissimilarity obtained with KMeans is less than that obtained with AP. To finish, QTempIntMiner is as precise, even more precise, than the two other algorithms for big patterns: the quality induced by the clustering based on EM is better than that of KMeans or AP.

6 Conclusion

We have presented the problem of quantitative temporal pattern extraction from temporal interval sequences where events are qualified by a type and a numerical date and duration, *i.e.* associated with numerical temporal intervals. Such patterns are particularly expressive: they can discriminate input sequences not only on the succession of events but also on their relative position in time and on their respective duration. We have proposed the algorithm `QTIPrefixSpan` for extracting temporal patterns that contains quantitative temporal information from temporal interval sequences. The comparison with QTempIntMiner shows that our algorithm is more efficient in runtime and can extract more precise temporal patterns on simulated datasets.

The improvements brought to interval-based sequence mining enable an efficient computation of large datasets. Applications, such as the characterization of cardiac diseases from electrocardiograms or faults in monitored industrial plants from time series, are investigated now to illustrate the practical interest of mining such temporal patterns. The clustering step is the major source of complexity of the proposed

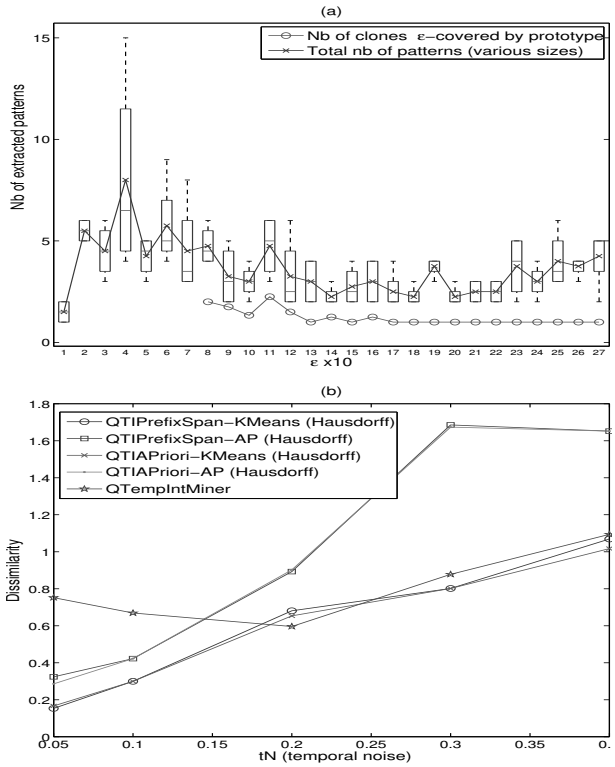

Figure 3: (a) mean number of extracted patterns with respect to ϵ for `QTIPrefixSpan`-KMeans. (b) mean dissimilarity between the best extracted pattern and the prototype with respect to the temporal noise.

algorithms. We are investigating different ways to improve its efficiency such as defining heuristics to choose the number of clusters a priori or introducing incrementality into clustering methods.

References

[Allen, 1983] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Dousson and Duong, 1999] C. Dousson and T.V. Duong. Discovering Chronicles with Numerical Time Constraints from Alarm Logs for Monitoring Dynamic Systems. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 620–626, 1999.

[Frey and Dueck, 2007] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[Giannotti *et al.*, 2006] F. Giannotti, M. Nanni, D. Pedreschi, and F. Pinelli. Mining sequences with temporal annotations. In *Proceedings of the Symposium on Applied Computing*, pages 593–597, 2006.

[Guyet and Quiniou, 2008] T. Guyet and R. Quiniou. Mining temporal patterns with quantitative intervals. In *Proceedings of the International Workshop on Mining Complex Data*, 2008.

[Höppner, 2002] F. Höppner. Learning dependencies in multivariate time series. In *Proceedings of the Workshop on Knowledge Discovery in (Spatio-)Temporal Data*, pages 25–31, 2002.

[Irpino and Verde, 2008] A. Irpino and R. Verde. Dynamic clustering of interval data using a wasserstein-based distance. *Pattern Recognition Letters*, 29:1648–1658, 2008.

[Kam and Fu, 2000] P.-S. Kam and A.W. Fu. Discovering temporal patterns for interval-based events. In *Proceegins of data Warehousing and Knowledge Discovery (DaWaK)*, pages 317–326, 2000.

[Lee and De Raedt, 2004] S. D. Lee and L. De Raedt. *Database support for data mining applications : discovering knowledge with inductive queries*, chapter Constraint based mining of first order sequences in SeqLog, pages 154–173. 2004.

[Masseglia *et al.*, 2009] F. Masseglia, P. Poncelet, and M. Teisseire. Efficient mining of sequential patterns with time constraints: reducing the combinations. *Expert Systems with Applications*, 36(2):2677–2690, 2009.

[Nakagaito *et al.*, 2009] F. Nakagaito, T. Ozaki, and T. Ohkawa. Discovery of quantitative sequential patterns from event sequences. In *Proceedings of the International Conference on Data Mining Workshops*, pages 31–36, 2009.

[Pei *et al.*, 2001] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan mining sequential patterns efficiently by prefix projected pattern growth. In *Proceedings of International Conference on Data Engineering*, pages 215–226, 2001.

[Pei *et al.*, 2007] J. Pei, J. Han, and W. Wang. Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems*, 28(2):133–160, 2007.

[Srikant and Agrawal, 1996] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the Fifth International Conference on Extending Database Technology*, pages 3–17, 1996.

[Washio *et al.*, 2007] T. Washio, K. Nakanishi, and H. Motoda. A classification method based on subspace clustering and association rules. *New Generation Computing*, 25:235–245, 2007.

[Wu and Chen, 2007] S.Y. Wu and Y.-L. Chen. Mining nonambiguous temporal patterns for interval-based events. *Transactions on Knowledge and Data Engineering*, 19(6):742–758, 2007.

[Yoshida *et al.*, 2000] M. Yoshida, T. Iizuka, H. Shiohara, and M. Ishiguro. Mining sequential patterns including time intervals. In *Proceedings of the conference on Data Mining and Knowledge Discovery: theory, tools, and technology*, pages 213–220, 2000.

[Zaki, 2001] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.