An Inductive Database for Mining Temporal Patterns in Event Sequences

Alexandre Vautier[1], Marie-Odile Cordier[1], and René Quiniou[1]

Irisa - DREAM Project Campus de Beaulieu 35042 RENNES Cedex, France
{Alexandre.Vautier,Marie-Odile.Cordier,Rene.Quiniou}@irisa.fr

Abstract. An inductive database formalizes data mining as looking for interesting patterns by querying a database containing raw data and patterns. We propose an inductive database extension for mining temporal patterns in event sequences where few types of event are present and so temporal information is of major importance for information extraction. We are interested in temporal patterns, named chronicles, which are *sets of events* such that the delay between their occurrences is bounded by a numerical interval. We propose a generality relation defined on this kind of pattern and related adaptations to the version space algorithm used for learning. The method can be used to extract temporal patterns that discriminate long event sequences where times of events represent most of the useful information. This extraction process is achieved by answering queries that set (maximum and minimum) frequency thresholds on the temporal patterns in events sequences.

1 Introduction

Most of the temporal data mining methods discover interesting temporal patterns from sequences. All of these sequences are used in the same manner, it means that the extracted patterns represent the potentially useful information of the aggregate of these sequences. Contrary to these methods, in our approach, all the sequences do not play the same role. Some of them are related to *positive* situations where the target phenomenon is known to have occurred and others to *negative* situations where the target phenomenon is known not to have happened. Useful information, i.e, representative of the target phenomenon, should be extracted from the first set of sequences and should not be present in the second set. Basically, we are interested by the extraction of temporal patterns that represent the interesting information contained in the sequences labeled positive but not contained in the sequences labeled negative.

Such labeled sequences may arise in many applications, such as plant or patient monitoring. Given these long sequences describing either the correct or the faulty behavior of some processes, our method aims at finding temporal patterns that discriminate the two kinds of sequences. Moreover, the sequences we are interested in, are characterized by few types of events the occurrence time of which represents most of the useful information.

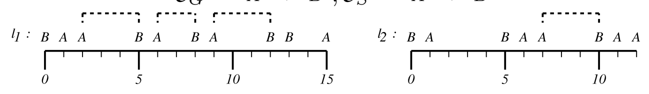
We propose to formalize our work within the framework of inductive database (IDB) [1]. The main goal of IDBs is to manage knowledge discovery applications just as database management systems manage business applications. The data mining process is modeled as an interactive process in which users can query data as well as patterns. Some IDBs implementations [2–4] use sequences of symbols as data and complex sequences of symbols as patterns (sequences with gaps). In our case, numerical temporal information is often necessary to represent interesting phenomena and to discriminate them from uninteresting ones. That is why, we choose the chronicle formalism [5] that makes possible to deal with temporal numerical constraints between events. The complexity added by the management of numerical constraints is the main challenge of the paper.

Section 2 describes our IDB extension characterized by a version space framework based on the generality relation between chronicles and a recognition criterion that is used to define the frequency of chronicles in event sequences. Section 3 presents the query process adapted from the version space algorithm [6], it uses the minimal and frequent chronicles computed by a fuzzy clustering process that was integrated to a temporal data mining tool named FACE [7]. Preliminary results are given in section 4 and section 5 gives an overview of related techniques dealing with temporal patterns. Finally, conclusion and future work can be found in section 6.

2 Inductive database and temporal data

An inductive database $I(\mathcal{D}, \mathcal{P})$ consists of a data component $\mathcal{D}$ and a pattern component $\mathcal{P}$ assumed to be sets of sets. We consider the data domain of strings and the pattern domain of complex strings inspired by [2, 4]. For example, let the data be: $e_1 = ababcc$; $e_2 = abbc$; $e_3 = bb$; $e_4 = abc$; $e_5 = bc$; $D_1 = \{e_1, e_2\}$; $D_2 = \{e_3, e_4, e_5\}$; $\mathcal{D} = \{D_1, D_2\}$ and patterns: $p_1 = ab$; $p_2 = a < c$; $P_1 = \{p_1, p_2\}$; $\mathcal{P} = \{P_1\}$. Pattern p_2 states that c occurs somewhere after a. The query $freq(p, e_1) \geq 2$ asks for all patterns p whose frequency is equal or greater than 2 in e_1. On this database, this query would yield the set $\{a, b, c, ab, a < b, a < c, b < c\}$.

$$\mathcal{C}_G = A \xrightarrow{[0,3]} B \;, \mathcal{C}_S = A \xrightarrow{[2,3]} B$$



Instances of $\mathcal{C}_S$ or $\mathcal{C}_G$ in l_1: $\{((A, 2), (B, 5)), ((A, 6), (B, 8)), ((A, 9), (B, 12))\}$
Instance of $\mathcal{C}_S$ or $\mathcal{C}_G$ in l_2: $\{((A, 7), (B, 10))\}$

Fig. 1. Chronicle discovery under frequency constraints

Numerical temporal information is added to string data and patterns which are called respectively event sequences and chronicles. Our goal, is to discover chronicles that are frequent in some event sequences and infrequent in other ones. For instance, let l_1 and l_2 be the event sequences of Figure 1. A complex query such that $freq(\mathcal{C}, l_1) \geq 3 \wedge freq(\mathcal{C}, l_2) \leq 1$ asks for the set of chronicles $\mathcal{C}$ that are frequent in l_1 and infrequent in l_2 according respectively to thresholds 3 and 1. Chronicles $\mathcal{C}_S$ and

$\mathcal{C}_G$ (Figure 1) satisfy this query because both cover three instances in l_1 and only one in l_2. Note that $\mathcal{C}_G$ is more general than $\mathcal{C}_S$ and that no solution chronicle is more specific than $\mathcal{C}_S$ or more general than $\mathcal{C}_G$.

$$\mathcal{C}_1 = (\mathcal{E}, \mathcal{D}); \mathcal{E} = \{e_1, e_2, e_3\};$$
$$e_1 = (A, t_1), e_2 = (B, t_2), e_3 = (B, t_3);$$
$$\begin{tikzcd}(\mathcal{E}, \mathcal{D}) : & e_1 \arrow[r, "{[-3;-2]}"'] \arrow[rr, bend left, "{[-1;3]}"] & e_2 \arrow[r, "{[1;5]}"'] & e_3\end{tikzcd}$$

Fig. 2. A chronicle $\mathcal{C}_1$ of size 3. Intervals labeling edges represent bounds on the distance between event times, for instance: $1 \leq t_3 - t_2 \leq 5$.

2.1 Chronicles: definitions

The definitions of events and event sequences that extend strings are taken from Mannila *et al.* definitions [8].

Definition 1 (Event)
Let S be a set of event types. *An* event *is represented by a pair $e = (s, t)$, where $type(e) = s \in S$ is the type of the event and $time(e) = t$ is the* time *of the event.*

Definition 2 (Event sequence)
An event sequence is a sequence containing timed-stamped events, ordered according to their time of occurrence.

Definition 3 (Chronicle)
A chronicle $\mathcal{C}$ is a pair $(\mathcal{E}, \mathcal{D})$, where

- *$\mathcal{E}$ is a set of events, ($|\mathcal{C}| = |\mathcal{E}|$ denotes the size of $\mathcal{C}$).*
- *$(\mathcal{E}, \mathcal{D})$ is a minimum temporal constraint graph where nodes are temporal variables associated to events (noted e), from $\mathcal{E}$ and edges (noted d) from $\mathcal{D}$ represent temporal constraints (noted c_d) between event times. A temporal constraint takes the form of a unique interval that bounds the difference between the connected event times.*

Figure 2 gives an example of chronicle. A chronicle $\mathcal{C}$ occurs in the sequence l if every event of $\mathcal{C}$ is instantiated by an event of l respecting type and temporal constraints. The set of matched events is called a *chronicle instance* of $\mathcal{C}$. For example, the event sequence l_3 (Figure 3) has 11 instances of chronicle $\mathcal{C}_1$.

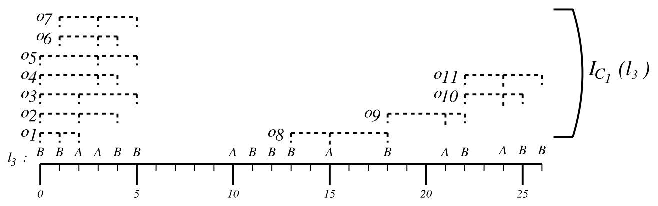


Fig. 3. An event sequence l_3 and the instances $\mathcal{I}_{\mathcal{C}_1}(l_3)$

Definition 4 (Chronicle instance)
Let $\mathcal{C} = (\mathcal{E}, \mathcal{D})$ be a chronicle. A sub-sequence o of the sequence l is an instance of $\mathcal{C}$ in l if and only if there exists a substitution $\sigma : \mathcal{E} \rightarrow o$ such that

$$\left[\forall e \in \mathcal{E}, type(\sigma(e)) = type(e)\right] \wedge \left[\forall d = (e, e') \in \mathcal{D}, time(\sigma(e')) - time(\sigma(e)) \in c_d\right]$$

We note $\mathcal{I}_{\mathcal{C}}(l)$ the set of instances of $\mathcal{C}$ in l.

2.2 Generality relation on chronicles

In IDB, query computation uses the version space algorithm for learning. A generality relation is defined on version space objects, here chronicles. A chronicle $\mathcal{C}$ is *more general* than a chronicle $\mathcal{C}'$ if every event of $\mathcal{C}$ can be matched to an event of $\mathcal{C}'$ and if each temporal constraint of $\mathcal{C}$ is more general than the corresponding constraint in $\mathcal{C}'$(*cf.* example in Figure 4).

Definition 5 (Generality relation on temporal constraints) *Let $c_d = [w, z]$ and $c_{d'} = [x, y]$ be two temporal constraints. c_d is more general than $c_{d'}$ (written $c_{d'} \subseteq c_d$) if and only if $w \leq x \leq y \leq z$.*

Definition 6 (Generality relation on chronicles)
The chronicle $\mathcal{C} = (\mathcal{E}, \mathcal{D})$ is more general than the chronicle $\mathcal{C}' = (\mathcal{E}', \mathcal{D}')$, written $\mathcal{C} \sqsubseteq \mathcal{C}'$, if and only if

$$\begin{aligned}\exists f : \mathcal{E} \rightarrow \mathcal{E}' \wedge \left[\forall e \in \mathcal{E}, type(e) = type(f(e))\right] \wedge \\ \left[\forall (e_1, e_2) \in \mathcal{E} \times \mathcal{E}, d = \mathcal{D}(e_1, e_2), \exists d' = \mathcal{D}'(f(e_1), f(e_2)) \Rightarrow c_{d'} \subseteq c_d\right]\end{aligned}$$

f defines the partial match between every event of $\mathcal{C}$ and some events of $\mathcal{C}'$. Testing whether some chronicle is more general than another requires, at worst, to check every f function. So its complexity is $O(|\mathcal{C}|!)$. The generality relation above is used to define the maximal and minimal elements of a chronicle set.

Definition 7 (minimal and maximal chronicles of a set)
Let F be a set of chronicles. The minimal and maximal chronicles of F are defined by:

$$min(F) = \{\mathcal{C} \in F | \forall q \in F, \mathcal{C} \sqsubseteq q \Rightarrow \mathcal{C} = q\}$$
$$max(F) = \{\mathcal{C} \in F | \forall q \in F, q \sqsubseteq \mathcal{C} \Rightarrow \mathcal{C} = q\}$$

2.3 Union and directed specialization operators

We have adapted the operators $lub(\mathcal{C}_1, \mathcal{C}_2)$ “largest common subfragments” and $mgs(\mathcal{C}_1, \mathcal{C}_2)$ “smallest fragments more specific than $\mathcal{C}_1$ but not more general than $\mathcal{C}_2$” [9] in such a way as to use them on chronicles. The union of two chronicles $\mathcal{C}$ and $\mathcal{C}'$ is the set of chronicles $\mathcal{C}_u$ that are more general than $\mathcal{C}$ and $\mathcal{C}'$ and as specific as possible. This operation corresponds to the least general generalization (lgg) in machine learning. Intuitively each $\mathcal{C}_u$ is such that its event types are event types of $\mathcal{C}$ and $\mathcal{C}'$, and its temporal constraints are the convex hull of matched intervals of $\mathcal{C}$ and $\mathcal{C}'$. The convex hull of two temporal constraints $c_d = [w, z]$ and $c_{d'} = [x, y]$ is $[min(w, x), max(y, z)]$. For instance, in Figure 4, $\mathcal{C}_3 \in \mathcal{C}_2 \bigcup \mathcal{C}_4$. Let n be $min(|\mathcal{C}|, |\mathcal{C}'|)$. Due to the number of possible matches between events of $\mathcal{C}$ and $\mathcal{C}'$, the complexity of computing the union of two chronicles is in $O(n!)$.

Definition 8 (Chronicle union)
Let $\mathcal{C} = (\mathcal{E}, \mathcal{D})$ and $\mathcal{C}' = (\mathcal{E}', \mathcal{D}')$ be two chronicles.

$$\mathcal{C} \bigcup \mathcal{C}' = min\{\mathcal{C}_u | \mathcal{C}_u \sqsubseteq \mathcal{C} \land \mathcal{C}_u \sqsubseteq \mathcal{C}'\}$$

The dso operator is an improved version of mgs. It uses an additional argument, the chronicle $\mathcal{C}_s$, to focus the specialization of $\mathcal{C}_g$ on chronicles that are more general than the chronicle $\mathcal{C}_s$ but not generalized by the chronicle $\mathcal{C}_d$.

Definition 9 (Directed specialization operator)
Let $\mathcal{C}_g$, $\mathcal{C}_d$ and $\mathcal{C}_s$ be three chronicles with $\mathcal{C}_g$ more general than $\mathcal{C}_s$.

$$dso(\mathcal{C}_g, \mathcal{C}_s, \mathcal{C}_d) = max\{\mathcal{C} | \mathcal{C}_g \sqsubseteq \mathcal{C} \land \mathcal{C} \sqsubseteq \mathcal{C}_s \land \mathcal{C} \not\sqsubseteq \mathcal{C}_d\}$$

As $\mathcal{C}_g \sqsubseteq \mathcal{C}_s$, a partial match exists between events of $\mathcal{C}_g$ and events of $\mathcal{C}_s$. $\mathcal{C}_g$ can be specialized by adding events from $\mathcal{C}_s$ (and their associated temporal constraints) that are not matched. This yields chronicles that are more specific than $\mathcal{C}_g$ and more general than $\mathcal{C}_s$ from which $dso(\mathcal{C}_g, \mathcal{C}_s, \mathcal{C}_d)$ can select the maximal chronicles that are not more general than some chronicle $\mathcal{C}_d$.

dso is more efficient than mgs. Indeed, $mgs(\mathcal{C}_g, \mathcal{C}_d)$ would return the maximal chronicles that are more specific than $\mathcal{C}_g$ and not more general than $\mathcal{C}_d$. But without

$$\begin{tikzcd}
(A, t_1) \arrow[d, "{[-4,-3]}"] \arrow[dd, bend right, "{[-3,0]}"'] & (A, t'_1) \arrow[l, dotted, "f_2"'] \arrow[d, "{[-3,4]}"] \arrow[dr, dotted, "f_4"] & (A, t''_1) \arrow[d, "{[-1,8]}"'] \arrow[dd, bend left, "{[2,5]}"] \\
(B, t_2) \arrow[d, "{[1,3]}"] & (B, t'_2) \arrow[dl, dotted, "f_2"'] \arrow[dr, dotted, "f_4"] & (A, t''_2) \arrow[d, "{[-2,4]}"'] \\
(B, t_3) & & (B, t''_3) \\
\mathcal{C}_2 \arrow[r, phantom, "\sqsupseteq"] & \mathcal{C}_3 \arrow[r, phantom, "\sqsubseteq"] & \mathcal{C}_4
\end{tikzcd}$$

Fig. 4. Generality relation between chronicles: $\mathcal{C}_3 \sqsubseteq \mathcal{C}_2$ and $\mathcal{C}_3 \sqsubseteq \mathcal{C}_4$

constraining the specialization of $\mathcal{C}_g$, $mgs(\mathcal{C}_g, \mathcal{C}_d)$ can be very large. dso requires that $\mathcal{C}_s \neq \bot$. Indeed, if $\mathcal{C}_s = \bot$ then the specialization of $\mathcal{C}_g$ is not directed and dso is equivalent to $mgs(\mathcal{C}_g, \mathcal{C}_d)$. In section 3, we will see that this case never occurs in our algorithm.

2.4 Frequency of chronicles in an event sequence and recognition criterion

The most used interest measure for patterns mining is based on frequency. The use of a generality relation makes chronicle search based on frequency easier, but only on the condition that constraints on frequency satisfy monotonicity or anti-monotonicity properties. Let $\mathcal{C}$ be more general than $\mathcal{C}'$. The definition of frequency satisfies these properties if the frequency of $\mathcal{C}$ is equal or greater than the frequency of $\mathcal{C}'$ in an event sequence. Depending on the *recognition criterion* that specifies how instances of some chronicle are selected in a sequence to compute the frequency of this chronicle, these properties are or are not satisfied. That is why a *monotonic criterion* is defined to ensure these properties.

Definition 10 (Recognition criterion)
Let $2^{\mathbb{I}}$ be the power set of a set of chronicle instances $\mathbb{I}$ and $\mathcal{I}_{\mathcal{C}}(l)$ be the set of instances of a chronicle $\mathcal{C}$ in an event sequence l. $Q : 2^{\mathbb{I}} \rightarrow \{true, false\}$ is a recognition criterion if and only if:

$$\forall \mathcal{C}\ \forall l\ \exists!\ E \subseteq \mathcal{I}_{\mathcal{C}}(l), Q(E)$$

Definition 11 (Frequency of a chronicle in an event sequence)
The frequency $freq^Q(\mathcal{C}, l)$ of a chronicle $\mathcal{C}$ in an event sequence l, according to a recognition criterion Q, is the cardinal of the set of instances of $\mathcal{C}$ in l satisfying Q:

$$E \subseteq \mathcal{I}_{\mathcal{C}}(l) \wedge Q(E) \Rightarrow freq^Q(\mathcal{C}, l) = |E|$$

Definition 12 (Monotonic criterion)
Q is monotonic criterion if and only if:

$$Q \textit{ is a recognition criterion and}$$
$$\forall \mathcal{C}\ \forall \mathcal{C}', \mathcal{C} \sqsubseteq \mathcal{C}' \Rightarrow freq^Q(\mathcal{C}, l) \geq freq^Q(\mathcal{C}', l).$$

Different criteria[1] have been defined in the literature. The minimal occurrences criterion Q_m [8] selects all the shortest instances of a temporal pattern. As pointed by [10], this criterion is a recognition criterion but not a monotonic criterion and is not acceptable.

The earliest distinct instances criterion, $Q_{e\&d}$ [7], selects all instances such that two instances of a same chronicle have no common events and occur as early as possible in the event sequence according to a total order on instances. In the set $\mathcal{I}_{\mathcal{C}_1}(l_3)$ (Figure 3), the criterion $Q_{e\&d}$ recognizes the set of instances $E = \{o_1, o_8, o_{10}\}$ and $freq^{Q_{e\&d}}(\mathcal{C}_1, l_3) = |E| = 3$. $Q_{e\&d}$ is a monotonic criterion but it is too restrictive because it selects instances arbitrarily (it favors earliest instances).

[1] The choice of the criterion by an user depends on the application domain

We introduce the monotonic criterion Q_d that selects the largest set of distinct instances in $\mathcal{I}_{\mathcal{C}}(l)$. The instances of $\mathcal{I}_{\mathcal{C}}(l)$ can be viewed as nodes of a graph whose edges represent *distinctness* (no common events) between instances. Evaluating Q_d[2] consists in finding one of the maximal cliques of such a graph. The criterion Q_d recognizes the set of instances $E = \{o_2, o_7, o_8, o_{10}\}$ and $freq^{Q_d}(\mathcal{C}_1, l_3) = |E| = 4$.

3 Queries on event sequences

Let us now return to the problem definition that consists in finding temporal patterns that support minimum or maximum frequency thresholds in events sequences. Let P and N be two sets of event sequences and T be a set of frequency thresholds. Every event sequence l is associated to a sub-query and has a threshold T_l. If l is an element of P (resp. N) then T_l is a minimum (resp. maximum) frequency threshold. The purpose of this section is to show how to extract some phenomenon in the form of chronicles which are frequent in at least one event sequence of P and infrequent in every event sequence of N. Formally, this can be described as the task of finding the set of chronicles $\{\mathcal{C}|Qu(P, N, T, \mathcal{C})\}$. Here the query Qu is a predicate or constraint that determines whether a chronicle $\mathcal{C}$ is a solution. The query Qu has the general form:

$$Qu(P, N, T, \mathcal{C}) = \big(\exists l \in P, freq^{Q}(\mathcal{C}, l) \geq T_l\big) \wedge \big(\forall l \in N, freq^{Q}(\mathcal{C}, l) < T_l\big) \quad (1)$$

Two ways can be followed to compute solutions. The first one considers the request globally and makes clever uses of its different parts[11]. The second one consists in computing all the frequent chronicles from the different event sequences separately, memorizing them, and merging them to get the solutions of the query. If we consider only one query, the first method is more efficient because many computations are avoided. But, if we consider a set of queries that use the same sets T and $P \cup N$ then the second method can be more efficient because only merging is different from a query to another. As we favor a database point of view, we have developed the second method. The result of a query computation is represented as bounds of version spaces. First, we present an adaptation of Mitchell's algorithm that computes those bounds on chronicles. Then, we introduce the notion of frequent minimal chronicle (FMC).

3.1 Version space computation: Mitchell's algorithm

Mitchell's algorithm is extensively used to compute version spaces of symbol sequences [9]. Algorithm 1 is a specialization of this algorithm to chronicle search. It computes the bounds S and G of a version space corresponding to solutions of a conjunction of constraints $c = c_0 \wedge c_1 \wedge \ldots \wedge c_n$. We can define S and G from the operators min and max (definition 7) and from the solutions of c: $S(c) = min(sol(c))$, $G(c) = max(sol(c))$. In our case, two kinds of constraints c_i are valid for algorithm 1: $\mathcal{C} \sqsubseteq \mathcal{C}'$ and $\mathcal{C} \not\sqsubseteq \mathcal{C}'$ where $\mathcal{C}'$ is known a priori and where $\mathcal{C}$ is some solution chronicle. The chronicle union operator (definition 8) is used to solve a constraint of the form $\mathcal{C} \sqsubseteq \mathcal{C}'$.

[2] Note that $freq^{Q_{e\&d}}(\mathcal{C}, l)$ is a lower bound of $freq^{Q_d}(\mathcal{C}, l)$

Algorithm 1: Mitchell's algorithm

```
Input: $c$: a constraint conjunction
Output: $S = min(Sol(c))$
        $G = max(Sol(c))$
$S = \{\bot\}$ $G = \{\top\}$
foreach $c_i \in c$ do
    case $c_i$ of kind $\mathcal{C} \sqsubseteq \mathcal{C}'$
        $G = \{g \in G | g \sqsubseteq \mathcal{C}'\}$
        $S = min\{\mathcal{C} | \exists s \in S, \mathcal{C} \in (\mathcal{C}' \cup s) \wedge \exists g \in G, g \sqsubseteq \mathcal{C}\}$
    case $c_i$ of kind $\mathcal{C} \not\sqsubseteq \mathcal{C}'$
        $S = \{s \in S | s \not\sqsubseteq \mathcal{C}'\}$
        $G = max\{\mathcal{C} | \exists g \in G, \exists s \in S, \mathcal{C} \in dso(g, s, \mathcal{C}')\}$
```

In the case of a constraint of the form $\mathcal{C} \not\sqsubseteq \mathcal{C}'$, the algorithm uses the operator $dso(\mathcal{C}_g, \mathcal{C}_s, \mathcal{C}_d)$. As seen in section 2.3, if $\mathcal{C}_s = \bot$ causes problems of non computability. This case occurs in algorithm 1 if $S = \{\bot\}$. If no constraint of kind $\mathcal{C} \sqsubseteq \mathcal{C}'$ is processed before a constraint of kind $\mathcal{C} \not\sqsubseteq \mathcal{C}'$ then $S = \{\bot\}$. This can be avoided by processing a constraint of kind $\mathcal{C} \sqsubseteq \mathcal{C}'$ before any other constraint. In the sequel, we prove that such a constraint exists for every version space computation.

3.2 Computing queries

Query (1) is rewritten in such a way as to use Mitchell's algorithm. Let $\mathit{Fmc}^Q(l, T_l) = min\{\mathcal{C} | freq^Q(\mathcal{C}, l) \geq T_l\}$ be the set of frequent and minimal chronicles (FMCs) in an event sequence l according to a minimum frequency threshold T_l and a recognition criterion Q. Sub-queries that impose a minimum frequency threshold are anti-monotonic, so each solution of query (1) is more general than at least one FMC of one event sequence from P. Sub-queries that impose a maximum frequency threshold are monotonic, so each solution of query (1) is not more general than every FMC of all event sequences from N. So, query (1) is rewritten:

$$Qu(P, N, T, \mathcal{C}) = \Big(\bigvee_{\mathcal{B} \in \mathit{Fmc}^Q(P,T)} \mathcal{C} \sqsubseteq \mathcal{B}\Big) \wedge \Big(\bigwedge_{\overline{\mathcal{B}} \in \mathit{Fmc}^Q(N,T)} \mathcal{C} \not\sqsubseteq \overline{\mathcal{B}}\Big) \tag{2}$$

where $\mathit{Fmc}^Q(E, T) = \bigcup_{l \in E} \mathit{Fmc}^Q(l, T_l)$.

Algorithm 1 computes the bounds of the version space from a conjunction of constraints. To this end, expression (2) is rewritten as a disjunction of conjunctions:

$$Qu(P, N, T, \mathcal{C}) = \bigvee_{\mathcal{B} \in \mathit{Fmc}^Q(P,T)} \Big(\mathcal{C} \sqsubseteq \mathcal{B} \bigwedge_{\overline{\mathcal{B}} \in \mathit{Fmc}^Q(N,T)} \mathcal{C} \not\sqsubseteq \overline{\mathcal{B}}\Big) \tag{3}$$

For each chronicle from $\mathit{Fmc}^Q(P, T)$, the version space of a conjunction of constraints is computed by Mitchell's algorithm. Then, the union of these version spaces gives the solutions to query (3). Algorithm 2 describes this process. One of the main benefits of this method is that FMCs are computed only once in each event sequence (line 1 of algorithm 2).

```
Algorithm 2: General algorithm
  Input: $P, N$: Sets of event sequences
         $T$: Sets of thresholds associated to event sequences
  Output: $(S, G)$: Version space corresponding to solution $\mathcal{C}$ of $Qu(P, N, T, \mathcal{C})$
1 Computation of $Fmc^Q(P, T)$ and $Fmc^Q(N, T)$
  $S = \top$ $G = \bot$ $c = \emptyset$
  foreach $\overline{\mathcal{B}} \in Fmc^Q(N, T)$ do  $c = c \cup \{\mathcal{C} \not\sqsubseteq \overline{\mathcal{B}}\}$
  foreach $\mathcal{B} \in Fmc^Q(P, T)$ do
      $\{S_\mathcal{B}, G_\mathcal{B}\} = mitchell(\{\mathcal{C} \sqsubseteq \mathcal{B}\} \cup c)$
      $S = min(S \cup S_\mathcal{B})$
      $G = max(G \cup G_\mathcal{B})$
```

3.3 FMCs computation

The aim of this section is to show how to extract $Fmc^Q(l, T_l)$. The extracted FMCs must respect three conditions: (1) be as specific as possible, (2) generalize at least T_l instances such that (3) they respect the recognition criterion Q. The first step of the method consists in extracting from l frequent chronicles $\mathcal{C}$ such that every temporal constraint of $\mathcal{C}$ is of the form $[-W, W]$ (parameter W is the window-size). This is achieved by the Chronicle Recognizer System of FACE [7] which gives also the set of instances $\mathcal{I}_\mathcal{C}(l)$ of every such chronicle $\mathcal{C}$.

From such a set $\mathcal{I}_\mathcal{C}(l)$, our bottom-up-like method should generate every maximally specific chronicles[3] that covers T_l instances of $\mathcal{I}_\mathcal{C}(l)$ with respect to the recognition criterion Q. Unfortunately, this problem is NP-complete. This is why we propose to use an heuristic search that favors FMCs built from *close* instances. This requires to define a distance on instances. An instance can be represented by the unique maximally specific chronicle which generalizes it. For example, the instance $\{(A, 1), (A, 3), (B, 8)\}$ is related to the chronicle $\begin{tikzcd} A \arrow[r, "{[2,2]}"'] \arrow[rr, bend left, "{[7,7]}"] & A \arrow[r, "{[5,5]}"'] & B \end{tikzcd}$. Thus, to an instance can be associated a point (here [2,5,7]) in a space of dimension $|\mathcal{C}| \times (|\mathcal{C}| - 1)/2$. Many distances can be used in this vector space, in the experiments we used the euclidean distance.

Finding a set of close instances is handled by a clustering method in the so-defined space. To satisfy conditions (2) and (3), a cluster must contain at least T_l instances respecting criterion Q. Moreover, an instance can be covered by several FMCs which means that a point is allowed to belong to several clusters. Thus, fuzzy-clustering is preferred to hard-clustering. A numerical value in $[0, 1]$ denotes the degree to which an instance, more exactly the point related to the instance, belongs to a cluster. The instances are sorted in the decreasing order of their membership degree and only the T_l first instances (condition (2)) that respect the Q criterion (condition (3)) are retained to construct the chronicle.

Finally, we apply this clustering to every sets $\mathcal{I}_\mathcal{C}(l)$ corresponding to a frequent chronicle $\mathcal{C}$ discovered by FACE. The union of these clustering results is computed and the most specific chronicles (condition (1)) are retained to give $Fmc^Q(l, T_l)$.

[3] Note that these chronicles are more specific than $\mathcal{C}$

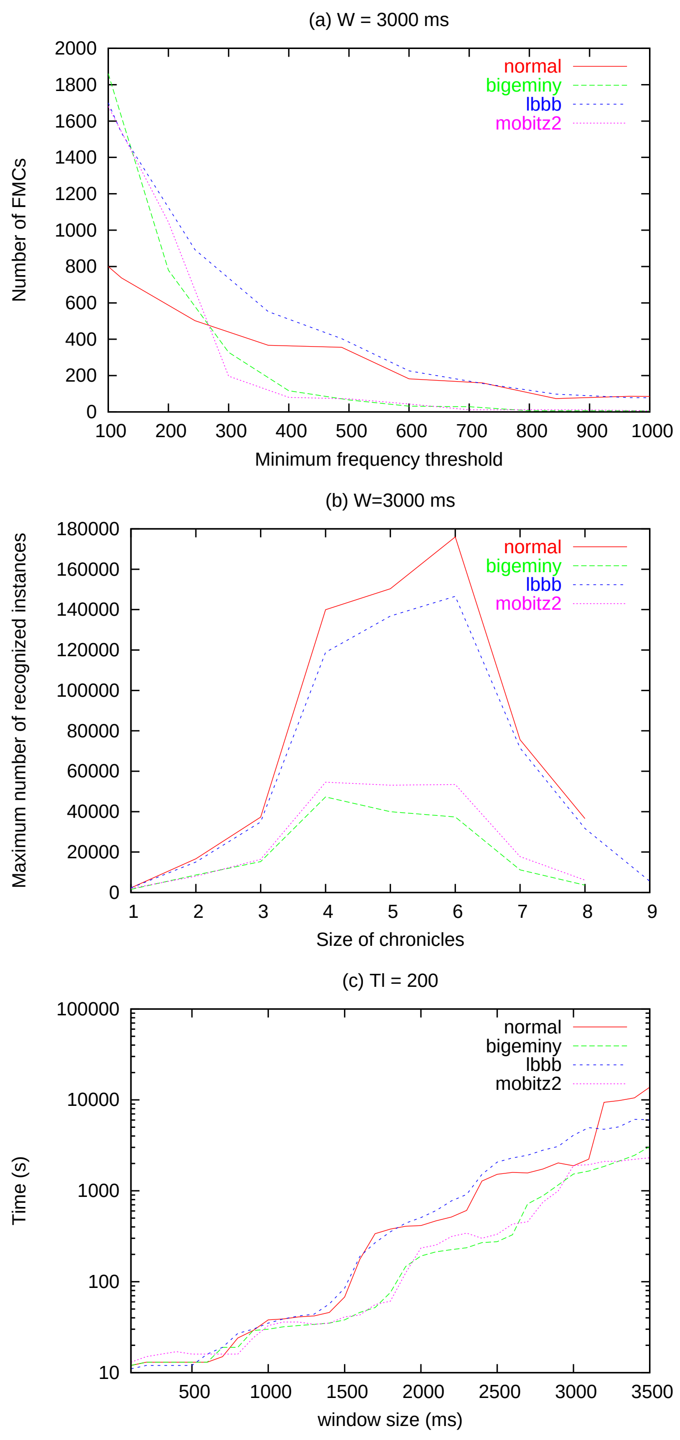


Fig. 5. Experiments on four sequences of ECG, (a) $|Fmc^{Q}(l, T_l)|$ as a function of T_l, (b) $max(\mathcal{I}_{\mathcal{C}}(l))$ as a function of $|\mathcal{C}|$, (c) Computation time as a function of the window-size W.

4 Experiments

The first experiments concerns the characterization of cardiac arrhythmia. The related IDB contains sequences of cardiac events elaborated from electrocardiograms (ECG)[12]. Every sequence is labeled by an expert according to the arrhythmia observed on the associated ECG: normal, lbbb, bigeminy and mobitz2. The goal is to find chronicles that are frequent in event sequences labeled by some arrhythmia and infrequent in other event sequences. Each sequence describes a 30 min ECG and contains around 4000 events of three types (P waves, normal QRSs and abnormal QRSs). Figure 5 shows quantitative results of experiments.

The first experiments concern the FMCs computation. Figure 5(b) shows the maximum cardinal of $\mathcal{I}_{\mathcal{C}}(l)$ during a FMC search. It can reach 180,000 instances and so requires to use a fuzzy clustering method based on k-means. The number of clusters is simply set to $2 \times \mathcal{I}_{\mathcal{C}}(l)/T_l$ and is limited by the size of the memory (1 Gb).

The l_{normal} and l_{lbbb} sequences correspond to regular rythm (l_{normal} is extremely regular), and so the associated clusters contain many distinct instances. Indeed, for l_{lbbb} and l_{normal} and a threshold $T_l \geq 500$, instance clustering computes still one hundred FMCs (Figure 5(a)). For l_{normal}, less FMCs are generated for low minimum frequency (Figure 5(a)) because of the memory size limitation. The number of FMCs falls down for the sequences $l_{bigeminy}$ and $l_{mobitz2}$ when the minimum frequency is greater than 300. As these sequences are less regular, fewer chronicles are discovered by FACE for high frequency thresholds.

Graph 5(c) shows that the FMC computation time increases dramatically with the window size. Furthermore, a stairs shape is observed. This corresponds to the introduction of new events in chronicles.

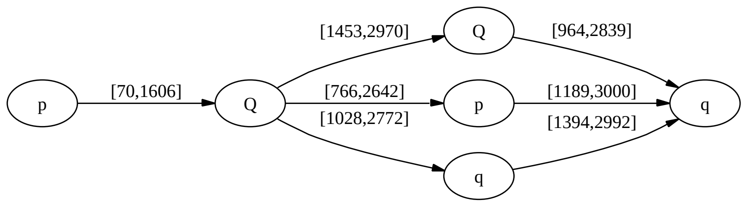


Fig. 6. A characteristic chronicle of $l_{bigeminy}$ (p=P Waves, Q=abnormal Qrs and q=normal Qrs)

The second experiments concern the algorithm 2. Many chronicles are generated during version space construction. This slows down computation dramatically for low frequency thresholds. So, the results of the general algorithm are only available, in acceptable waiting time, for high frequency thresholds (≥ 500). Since the set of FMCs corresponding to the *negative* part of expression (2), *Fmc*$^Q(N,T)$, is not complete, some incorrect frequent chronicles are not *filtered out* by more specific *negative* chronicles that were missed by FMC computation. From 0 to 25% of the discovered chronicles have to be removed to obtain a correct solution.

5 Related work

Several solutions have been proposed to tackle the problem of time integration in data mining. In the domain of temporal data mining, we can distinguish two different tasks: mining patterns in a sequence database [13] and mining patterns in a unique long sequence [8]. The method proposed in this paper is closer to long sequence mining than to sequence database mining despite we manage more than one sequence.

Most of long sequence mining solutions use numerical event times to constrain learning for extracting patterns respecting time constraints (min-gap, max-gap and window-size) and to order symbols in extracted pattern. For instance, Winepi and Minepi [8] extract *episodes* that are patterns represented by serial or parallel symbol sequences and SeqLog [2] extracts ordered symbol sequences with gaps. The abstraction level of time is more accurate in chronicles since they are based on numerical constraints on event times.

In sequence database mining, chronicle-like patterns are studied by Yoshida et al.[14]. Frequent itemsets containing a temporal feature are extracted. The purchases of each client are contained in a unique sequence. A purchase is an itemset associated to a purchase time. Their algorithm extracts “delta-patterns” that are ordered lists of itemsets with time intervals between two successive itemsets ($\{b\} \xrightarrow{[17,19]} \{a,c\} \xrightarrow{[5,12]} \{b,c\}$, for instance) which specifies the numerical temporal constraints between the itemsets. Interval bounds of delta-patterns are always positive whereas interval bounds of chronicles can be negative. Thus, delta-patterns are limited to serial phenomena whereas chronicles can express both *serial and parallel phenomena*. Most of sequence mining solutions do not extract simultaneously parallel and serial phenomena contrary to our proposal that uses the fully expressiveness of chronicles.

As in Lin et al. [15], many time series mining methods start by discretizing the signal and then use a method relatively close to ours to extract frequent patterns (named “motifs” in [15]). The signal is decomposed into signal chunks of equal length. A motif is a set of small series of adjacent chunks that are closed enough given a distance measure on such series. Compared to Lin et al.’s method, we discover frequent patterns that represent discontinuous signal sections without any distance measure. These patterns can be viewed as sets of non ordered chunks of signal.

Extracted chronicles are accurate if event time intervals (numerical constraints) are as small as possible. Lin [16] provides a method for mining maximally specific (i.e minimal according to version space) frequent intervals from an interval set given a generality relation (for instance, the inclusion). A frequent interval is maximally specific if and only if there exists no more specific frequent interval. In our approach, we look for maximally specific frequent multi-intervals which is much more complex.

6 Conclusion and future work

We have presented an original method that extracts temporal information in the form of patterns named chronicles. These patterns use numerical temporal constraints on events. Furthermore, they provide a way to express sequentiality and parallelism between events and they generalize temporal patterns used by Mannila and De Raedt [8, 2], among others. Chronicles are extracted by querying an IDB that contains event sequences and chronicles. The user sets the minimum or maximum frequency thresholds of searched chronicles in event sequences.

Our contribution introduces the notion of recognition criterion that generalizes the specification of frequent pattern computation on temporal data. A generality relation is defined on chronicles that enables us to reuse and to adapt version space algorithms to manage numerical temporal constraints. These algorithms require the prior computation of frequent and maximal chronicles for each event sequence used in the query. This computation is performed by a data mining tool that we have adapted to the task.

The analyzes of the FMCs computation show the efficiency of the method. The next step is to control the clustering in such a way that only FMCs that are needed by the version space algorithm are computed. Furthermore, in order to gain in efficiency, the version space algorithm itself should be adapted to provide an approximate solution whose quality is user-defined.

References

1. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Communications of the ACM **39** (1996) 58–64
2. Lee, S.D., De Raedt, L.: Constraint based mining of first-order sequences in SeqLog, University of Alberta, Edmonton, Canada (2002)
3. Kramer, S., De Raedt, L., Helma, C.: Molecular feature mining in HIV data. In: Proc. of the seventh ACM SIGKDD, ACM Press (2001) 136–143
4. De Raedt, L.: A perspective on inductive databases. SIGKDD Explor. Newsl. **4** (2002) 69–77
5. Dousson, C., Gaborit, P., Ghallab, M.: Situation recognition: Representation and algorithms. In: IJCAI. (1993) 166–174
6. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
7. Dousson, C., Duong, T.V.: Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In: Proc. of IJCAI 1999. (1999) 620–626
8. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Mining and KD **1** (1997)
9. De Raedt, L., Kramer, S.: The levelwise version space algorithm and its application to molecular fragment finding. In: Proc. of IJCAI. (2001)
10. Sun, X., Orlowska, M.E., Li, X.: Finding negative event-oriented patterns in long temporal sequences. In Dai, H., Srikant, R., Zhang, C., eds.: PAKDD. Volume 3056 of Lecture Notes in Computer Science., Springer (2004) 212–221
11. Fischer, J., De Raedt, L.: Towards optimizing conjunctive inductive queries. In: PAKDD. (2004) 625–637
12. Carrault, G., Cordier, M.O., Quiniou, R., Wang, F.: Temporal abstraction and inductive logic programming for arrhythmia recognition from electrocardiograms. AI in Medicine **28** (2003)
13. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In Apers, P.M.G., Bouzeghoub, M., Gardarin, G., eds.: Proc. 5th Int. Conf. Extending Database Technology, EDBT. Volume 1057., Springer-Verlag (1996) 3–17
14. Yoshida, M., Iizuka, T., Shiohara, H., Ishiguro, M.: Mining sequential patterns including time intervals. In: Data Mining and KD. (2000)
15. Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: Proceedings of the 2nd Workshop on Temporal Data Mining, Canada (2002)
16. Lin, J.L.: Mining maximal frequent intervals. In: Proc. ACM symposium on Applied computing. (2003)