Learning chronicles signing multiple scenario instances

Audine Subias, Louise Travé-Massuyès, Euriell Le Corronc

Learning chronicles signing multiple scenario instances

Audine Subias[1,3] and **Louise Travé-Massuyès**[1,2] and **Euriell Le Corronc**[1,4]
[1]CNRS, LAAS, 7, avenue du Colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, LAAS, F-31400 Toulouse, France
[3]Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
[4]Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France
e-mail: {subias, louise, elecorro}@laas.fr

Abstract

Chronicle recognition is an efficient and robust method for fault diagnosis. The knowledge about the underlying system is gathered in a set of chronicles, then the occurrence of a fault is diagnosed by analyzing the flow of observations and matching this flow with a set of available chronicles. The chronicle approach is very efficient as it relies on the direct association of the symptom, which is in this case a complex temporal pattern, to a situation. Another advantage comes from the efficiency of recognition engines which make chronicles suitable for one-line operation. However, there is a real bottleneck for obtaining the chronicles. In this paper, we consider the problem of learning the chronicles. Because a given situation often results in several admissible event sequences, our contribution targets an extension to multiple event sequences of a chronicle discovery algorithm tailored for one single event sequence. The concepts and algorithms are illustrated with representative and easy to understand examples.

1 Introduction

Chronicles are temporal patterns well suited to capture the behavior of dynamic processes at an abstract level based on events. They are among the formalisms that can be used to model timed discrete event systems. Chronicles may represent the signatures of specific situations, and are hence very efficient for diagnosis. They may also be associated to decision rules specifying which actions must be undertaken when reconfiguration is needed.

The chronicle approach has been developed and used in a large spectrum of diagnosis applications [1]: in the medical field for ECG interpretation and cardiac arrhythmia detection [2], in intrusion detection systems [3], in telecommunication networks [4]. More recently, chronicles have been used in the context of web services [5, 6]. Chronicles are also applied on activity recognition in the setting of unmanned aircraft Systems and unmanned aerial vehicles operating over road and traffic networks [7].

Among the challenges raised by the chronicle approach is the problem of acquiring the chronicles. On one hand, model based chronicle generation approaches have been developed. For instance, in [8] the patterns are built from Petri net models of the situation to recognize. Nevertheless, most of the works addressing this problem are data driven. They rely on analyzing logs and extract the significant patterns by temporal data mining techniques [9].

The objective of temporal data mining techniques is to discover all patterns of interest in the input data, by means of an unsupervised approach. There are several ways to define the relevance of a pattern. Among them, the frequency criterium is widely used [10, 11]. One can distinguish two main frameworks: sequential patterns [12] and frequent episodes [13].

- The sequential pattern framework is based on the discovery in a collection of sequences of all possible time ordered sets of event (i.e. sequence of events) with sufficient number of occurrences w.r.t a user-defined threshold. The number of occurrences of a set of events is defined as the number of times the set of events can be observed in the collection. Further, a sequence of events is said to be maximal if it involves the highest possible number of events. Sequential pattern discovery relies on the systematic search of maximal sequences that have a number of occurrences at least equal to a user-defined threshold. Many methods for unearthing sequential patterns are designed along the lines of the Apriori algorithm [12].
- Frequent episode framework uses a single (long) sequence and considers the discovery of temporal patterns, called *episodes*, that occur with sufficient frequence in the sequence. An episode is a partially ordered set of event types. Similarly to the case of sequential patterns, the notion of frequent episode and sub-episode are defined. In [13] episode discovery focusses on two types of episodes: serial episodes when the order between the event types is total and parallel episodes when there is only partial order between the events types.

Chronicles are designed to afford for total order and partial order temporal patterns but also patterns combining the two types. Moreover, chronicles consider temporal constraints between event type occurrences.

One of the main difficulties of chronicle discovery is to guarantee robustness to variations. The chronicle discovery approach proposed in this paper aims at discovering frequent chronicles common to multiple sequences representing variations of a unique situation, that is to say chronicles that are frequent in each sequence of a collection. This is motivated by the fact that the event sequences arising from the same situation generally present variants that must be accounted for.

Our contribution precisely targets an extension to multiple event sequences of the chronicle discovery algorithm proposed by [11], which is tailored for one single event sequence. We target to discover the chronicles not only for a given frequency but for all the possible frequencies higher than a specified threshold but keep the anytime-like strategy of [11] that allows the user to stop the algorithm at any time. This is a way to cope with the algorithm's complexity, which is exponential because of the combinatorial nature of the chronicle discovery problem that heritates the complexity of the episode discovery problem and the one of affecting temporal constraints to each event pair. Clearly the theoretical complexity of our algorithm remains the same as the one of [11] but dealing with multiple sequences tends to reduce it in practice and definetively widens its scope of applicability.

The paper is organizes as follows. Section 2 provides the concepts and definitions underlying the proposed approach. Section 3 first presents the algorithm for building a database representing the sequences at hand. It then presents the chronicle learning algorithm that uses the constructed database. Section 4 summarizes and concludes the work.

2 Concepts and definitions

In this section, the concepts that underly our chronicle mining algorithm are presented and formalized. Chronicles have been introduced as a way to express temporal information about a domain [14].

The domain is assumed to be described through a set of *features* whose values change over time with the evolutions of the domain.

The data samples are hence given in terms of the set of features $\{\chi^1, \ldots, \chi^n\}$. Every χ^j takes its value in the set $\mathcal{U}^j$, called the *domain* of χ^j. The *universe* $\mathcal{U} = \mathcal{U}^1 \times \cdots \times \mathcal{U}^n$ is defined as the Cartesian product of the feature domains. Therefore any sample can be represented by a vector $\vec{x} = (x^1, \ldots, x^n)^T$ of $\mathcal{U}$, so that every component x^j corresponds to the feature value χ^j qualifying the object $\vec{x}$. The subset of $\mathcal{U}$ formed by these vectors is called the *database*.

When samples need to be indexed by time, a sample taken at time t_i is represented by a vector $\vec{x}_{t_i} = (x^1_{t_i}, \ldots, x^n_{t_i})^T$. The value taken by the feature χ^j across time can be considered as a random variable $x^j_t, t \in \mathbb{Z}$. The corresponding time series, taken from time t_i to time t_f is noted $X_{t_i - t_f} = \{\vec{x}_t, t = t_i, \ldots, t_f\} = \langle \vec{x}_{t_i}, \ldots, \vec{x}_{t_f} \rangle$.

The concept of *event type* expresses a change in the value of a given domain feature or set of features. Let us define E as the totally ordered set of all event types. The order relation is denoted by $\leq_E$.

Definition 1 (Event)**.** *An* event *is defined as a pair (e_i, t_i), where $e_i \in E$ is an event type and t_i is a variable of integer type called the event date.*

Time representation relies on the time point algebra and time is considered as a linearly ordered discrete set of instants whose resolution is sufficient to capture the environment dynamics.

Definition 2 (Temporal sequence)**.** *A* temporal sequence *on E is an ordered set of events denoted $\mathcal{S} = \langle (e_i, t_i)_j \rangle_{j \in \mathbb{N}_l}$ where $\mathbb{N}_l$ is a finite set of linearly ordered time points of cardinal l and $l = |\mathcal{S}|$ is the size of the temporal sequence, i.e. the number of events in $\mathcal{S}$.*

The temporal sequence *typology* is the set of event types $E' \in E$ that occur in $\mathcal{S}$. Moreover, let us point out that in such temporal sequences, the index i refers to the event type e_i whereas the index j refers to the chronology of the event dates.

Example: Let us consider the following temporal sequence $\mathcal{S} = \langle (e_1, t_1)_1, (e_2, t_2)_2, (e_1, t_1)_3 \rangle$ where $l = 3$. The event date of $(e_1, t_1)_1$ is lower than the event date of $(e_2, t_2)_2$, itself lower than the event date of $(e_1, t_1)_3$.

To represent specific domain evolutions, we consider that event dates may be constrained. Consider two events (e_i, t_i) and (e_j, t_j). We define I as the time interval expressed as the pair $I_{ij} = [t^-, t^+]$ corresponding to the lower and upper bounds on the temporal distance between the two event dates t_i and t_j, $t^- \leq t^+$. We denote by τ_{ij} the temporal constraints defined by $\tau_{ij} = t_j - t_i \in [t^-, t^+]$. If the event dates t_i and t_j satisfy the temporal constraint τ_{ij}, we write $e_i[t^-, t^+]e_j$ and say that the events are temporally constrained by τ_{ij}. $I_{ij} = [t^-, t^+]$ is called the *support* of τ_{ij}.

Several events may have the same event type and hence a pair (e_i, e_j) may not be unically temporally constrained. Consequently, temporal constraints τ_{ij} and their corresponding supports I_{ij} are indexed by k as τ^k_{ij} and I^k_{ij}. For sake of simplicity, the index k is avoided when not necessary.

Definition 3 (Chronicle)**.** *A* chronicle *is a triplet $\mathcal{C} = (\mathcal{E}, \mathcal{T}, \mathcal{G})$ such that $\mathcal{E} \subseteq E$ with $\forall e_i \in \mathcal{E}, e_i \leqslant_E e_{i+1}$, $\mathcal{T} = \{\tau_{ij}\}_{1 \leqslant i \leqslant j \leqslant |\mathcal{E}|}$, and $\mathcal{G} = (V, A)$ is a directed graph where the nodes V represent event types of $\mathcal{E}$ and the arcs A represent the constraints between the event dates. $\mathcal{E}$ is called the* typology *of the chronicle, $\mathcal{T}$ is the set of* temporal constraints *of the chronicle, and $\mathcal{G}$ is the precedence graph.*

Example: The chronicle concept can be illustrated by the chronicle $\mathcal{C}$ defined by $\mathcal{E} = \{e_1, e_2, e_3\}$, $\mathcal{T} = \{\tau^1_{12}, \tau^2_{12}, \tau_{23}\}$, and $\mathcal{G}$ given in Figure 1. Moreover, we have $I^1_{12} = [1, 3]$, $I^2_{12} = [2, 10]$ and $I_{23} = [4, 6]$.

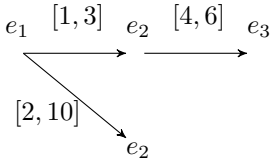

Figure 1: Chronicle precedence graph $\mathcal{G}$

A chronicle $\mathcal{C}$ represents an evolution pattern involving a subset of event types $\mathcal{E}$, a set of temporal constraints $\mathcal{T}$ linking event dates, and a directed graph $\mathcal{G}$ linking event occurences. Chronicles define temporal patterns similar to temporal constraint networks [15], where the temporal order of events is quantified with numerical bounds and reflects the represented piece of temporal evolution.

The *episodes* of [13] are a particular type of chronicle. A parallel episode is a collection of event types that occur in a given partial order whereas the event types of a serial episode are totally ordered. In the case of episodes, temporal constraints do not specify numerical temporal bounds but are just precedence constraints of the form $t_j - t_i \in [0, +\infty[$ or $t_j - t_i \in]-\infty, 0]$. An episode can hence be viewed

as a degenerated chronicle defined by the pair $(\mathcal{E}, \mathcal{G})$. Conversely, chronicles are parallel episodes with additional temporal information.

The occurrences of a given chronicle $\mathcal{C}$ in a temporal sequence $\mathcal{S}$ are denoted by subsequences called chronicle *instances*.

Definition 4 (Chronicle instance)**.** *An instance of $\mathcal{C} = (\mathcal{E}, \mathcal{T}, \mathcal{G})$ in a temporal sequence $\mathcal{S}$ is a subset of event types $\mathcal{E}'$ of $\mathcal{S}$ such that $\mathcal{E}'$ is isomorphic to $\mathcal{E}$, in other words $|\mathcal{E}'| = |\mathcal{E}|$, and the event types of $\mathcal{E}'$ satisfy all temporal constraints $\mathcal{T}$ of the chronicle $\mathcal{C}$ according to the graph $\mathcal{G}$.*

Definition 5 (Frequency of a chronicle)**.** *The* frequency *of a chronicle $\mathcal{C}$ in a temporal sequence $\mathcal{S}$, noted $f(\mathcal{C}|\mathcal{S})$, is the number of instances of $\mathcal{C}$ in $\mathcal{S}$.*

In the literature, it is hardly the case that all the instances of a given chronicle are returned by a chronicle recognition engine. Additional constraints are generally used to form a *recognition criterion* γ. For example, [13] returns the *shortest* instances in the sense of the instance duration in the sequence. [10] returns all the recognized at the earliest disjoint instances, i.e. all the instances that do not overlap in the sequence and that occur earliest in the sequence. The frequency of a chronicle $\mathcal{C}$ in a temporal sequence $\mathcal{S}$ can be defined according to the recognition criterion γ and it is then noted $f_\gamma(\mathcal{C}|\mathcal{S})$.

Given a set of event types E, the space of possible chronicles can be structured by a *generality relation*.

Definition 6 (Generality relation among chronicles)**.** *A chronicle $\mathcal{C} = (\mathcal{E}, \mathcal{T}, \mathcal{G})$ is* more general than *a chronicle $\mathcal{C}' = (\mathcal{E}', \mathcal{T}', \mathcal{G}')$, denoted $\mathcal{C} \sqsubseteq \mathcal{C}'$, if $\mathcal{E} \subseteq \mathcal{E}'$ or $\forall \tau_{ij} \in \mathcal{T}, \tau_{ij} \supseteq \tau'_{ij}$ or $\mathcal{G}'$ is a subgraph of the transitive closure of $\mathcal{G}$. Equivalently, $\mathcal{C}'$ is said stricter than $\mathcal{C}$.*

Definition 7 (Monotonicity)**.** *A frequency f_γ is said to be* monotonic *for the relation $\sqsubseteq$ if and only if $\mathcal{C} \sqsubseteq \mathcal{C}'$ implies that $f_\gamma(\mathcal{C}|\mathcal{S}) \geqslant f_\gamma(\mathcal{C}'|\mathcal{S})$ for any temporal sequence $\mathcal{S}$ of events.*

Definition 8 (Minimal and maximal chronicles of a set)**.** *Given a set of chronicles C, the subset of minimal and maximal chronicles of C are defined by:*

$$Min(C) = \{\mathcal{C} \in C | \forall \mathcal{C}' \in C, \mathcal{C} \sqsubseteq \mathcal{C}' \Rightarrow \mathcal{C} = \mathcal{C}'\},$$
$$Max(C) = \{\mathcal{C} \in C | \forall \mathcal{C}' \in C, \mathcal{C}' \sqsubseteq \mathcal{C} \Rightarrow \mathcal{C} = \mathcal{C}'\}.$$

3 An algorithm for learning general chronicles from multiple temporal sequences

The chronicle mining process consists in discovering all chronicles $\mathcal{C}$ whose instances occur in a given temporal sequence $\mathcal{S}$. However, it is often the case that the same situation does not result in perfectly identical temporal sequences. In this case, we are interested in learning the chronicles whose instances occur in all the temporal sequences. This problem can be stated as: given a set of temporal sequence $\mathbb{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ and a minimum frequency threshold f_{th}, find all minimal frequent chronicles $\mathcal{C}$ such that $f_\gamma(\mathcal{C})$ is at least f_{th} in all temporal sequences of $\mathbb{S}$.

This paper builds on the chronicle learning algorithm proposed by [11] and presents an extension to the case of multiple temporal sequences. The chronicle learning algorithm by [11] has two phases:

1. It builds a constraint database $\mathbb{D}$ representing the temporal sequence $\mathcal{S}$. This is performed with the *Complete Constraint-Database Construction algorithm* (CCDC algorithm).
2. It generates a set of candidate chronicles starting with a set of chronicles that were proved to be frequent and using $\mathbb{D}$ to explore the chronicle space. This is implemented by the *Heuristic Chronicle Discovery Algorithm* (HCDA algorithm).

Extending this algorithm to multiple temporal sequences requires to redesign the first phase so that the constraint database not only represents one temporal sequence but all the temporal sequences in the set $\mathbb{S}$.

3.1 Constraint database representing multiple temporal sequences

The constraint database $\mathbb{D}$ is an object in which every temporal constraint $\tau_{ij} = e_i[t^-, t^+]e_j$ that is frequent in all the sequences of $\mathbb{S}$ is stored. It is organized as a set of trees T_{ij}^α for each pair of event types (e_i, e_j) with $i, j = 1, \ldots, |E|, i \leqslant j$ and $\alpha = 1, \ldots, n_{ij}$. The nodes of the trees are temporal constraints and arrows represent *is_parent_of* relations.

Definition 9 (*is_parent_of* relation)**.** $e_i[t^-, t^+]e_j$ is_parent_of $e_i[t^{-'}, t^{+'}]e_j$ *iff* $[t^{-'}, t^{+'}] \subset [t^-, t^+]$ *and* $\nexists e_i[t^{-''}, t^{+''}]e_j$ *such that* $[t^{-'}, t^{+'}] \subset [t^{-''}, t^{+''}] \subset [t^-, t^+]$.

The root of a tree T_{ij}^α is hence a temporal constraint $e_i[t^-, t^+]e_j$ such that the occurrences of $\langle (e_i, t_i), (e_j, t_j) \rangle$ in all temporal sequences of $\mathbb{S}$ are maximal. Unlike in [11], there may be a number n_{ij} of such temporal constraints for the same pair (e_i, e_j), hence n_{ij} trees. Let us notice that a temporal constraint $e_i[t^-, t^+]e_j$ actually defines a 2-length chronicle $\mathcal{C} = (\mathcal{E}, \mathcal{T}, \mathcal{G})$ for which $\mathcal{E} = \{e_i, e_j\}$ $\mathcal{T} = \tau_{ij}$ and $\mathcal{G}$ is the reduced graph with one edge labeled by τ_{ij} between two nodes. Then, the root of T_{ij}^α is the 2-length chronicle with topology $\mathcal{E} = \{e_i, e_j\}$ that is the most general for all temporal sequences of $\mathbb{S}$ and the child nodes are stricter 2-lengh chronicles with the same typology. $\mathbb{D}^T$ is defined as the set of all tree roots.

As we consider multiple temporal sequences, only the pairs of event types (e_i, e_j) shared by all the temporal sequences $\mathcal{S}_i \in \mathbb{S}$ are examined.

In a first stage, for each sequence $\mathcal{S}_k \in \mathbb{S}$ and for each pair $(e_i, e_j) \in E^2$ such that $(e_i, t_i) \in \mathcal{S}_k$ and $(e_j, t_j) \in \mathcal{S}_k$, the set of all the occurrences of the pair in the sequence $\mathcal{S}_k$ (noted $\mathcal{O}_{ij}^k$) is determined. The set of time interval durations between the two event types of the occurrences of $\mathcal{O}_{ij}^k$ is given by $\mathcal{D}_{ij}^k = \{d_{ij}^k = (t_j - t_i) | \langle (e_i, t_i), (e_j, t_j) \rangle \in \mathcal{O}_{ij}^k\}$. From the frequency f_{ij}^k of (e_i, e_j) in each temporal sequence $\mathcal{S}_k$, we introduce $f_{max} = min_k(f_{ij}^k)$. f_{max} is the *maximal number of occurrences* for (e_i, e_j) present in *all* the sequences of $\mathbb{S}$.

Example: Let us consider the three temporal sequences of Figure 2. $\mathbb{S} = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$. For the pair (e_1, e_2), $\mathcal{O}_{e_1,e_2}^1 = \{\langle (e_1, 3), (e_2, 1) \rangle, \langle (e_1, 3), (e_2, 4) \rangle, \langle (e_1, 3), (e_2, 5) \rangle\}$, $\mathcal{O}_{e_1,e_2}^2 = \{\langle (e_1, 2), (e_2, 1) \rangle, \langle (e_1, 2), (e_2, 3) \rangle\}$ and finally $\mathcal{O}_{e_1,e_2}^3 = \{\langle (e_1, 2), (e_2, 1) \rangle, \langle (e_1, 2), (e_2, 3) \rangle, \langle (e_1, 2), (e_2, 5) \rangle\}$.

Additionally, $\mathcal{D}_{12}^1 = \{-2, 1, 2\}$, $\mathcal{D}_{12}^2 = \{-1, 1\}$ and $\mathcal{D}_{12}^3 = \{-1, 1, 3\}$. Finally the frequencies of the pair

(e_1, e_2) for each sequence are given by $f_{12}^1 = 3$, $f_{12}^2 = 2$, $f_{12}^3 = 3$, hence $f_{max} = 2$.

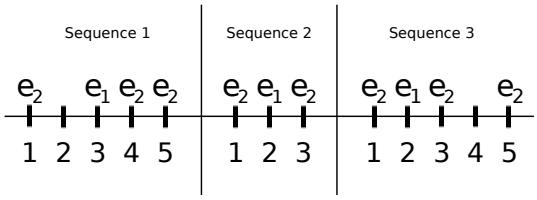

Figure 2: Example of multiple sequences

The roots of the trees for the pair (e_i, e_j) must hence be such that the number of occurrences $\langle (e_i, t_i), (e_j, t_j) \rangle$ in all temporal sequences of $\mathbb{S}$ is equal to f_{max}. The following explains how to obtain these roots.

Two sets of supports are considered for each sequence $\mathcal{S}_k$: the set of *minimal supports* $\underline{\mathbb{I}}_{ij}^k$ and the set of *maximal supports* $\overline{\mathbb{I}}_{ij}^k$ that guaranty exactly f_{max} occurrences of the pair (e_i, e_j) in $\mathcal{S}_k$. Whereas minimal supports are used in the algorithm of [11], maximal supports are a new concept required by our method to deal with multiple sequences. These sets are defined as follows:

$$\underline{\mathbb{I}}_{ij}^k = \{\underline{I}_{ij}^k = [\underline{t}^-, \underline{t}^+] | f_{ij}^k = f_{max} \text{ and } \forall [t^-, t^+] \subseteq [\underline{t}^-, \underline{t}^+] \; f_{ij}^k < f_{max}\}.$$

$$\overline{\mathbb{I}}_{ij}^k = \{\overline{I}_{ij}^k = [\,\overline{t}^-, \overline{t}^+] | f_{ij}^k = f_{max} \text{ and } \forall [t^-, t^+] \supseteq [\overline{t}^-, \overline{t}^+] \; f_{ij}^k > f_{max}\}.$$

Example: On the example of Figure 2, the minimal and maximal supports that guaranty exactly $f_{max} = 2$ for each sequence are $\underline{\mathbb{I}}_{12}^1 = \{[-2, 1], [1, 2]\}$ and $\overline{\mathbb{I}}_{12}^1 = \{] - \infty, 1], [-1, +\infty[\}$, $\underline{\mathbb{I}}_{12}^2 = \{[-1, 1]\}$ and $\overline{\mathbb{I}}_{12}^2 = \{] - \infty, +\infty[\}$, $\underline{\mathbb{I}}_{12}^3 = \{[-1, 1], [1, 3]\}$ and $\overline{\mathbb{I}}_{12}^3 = \{] - \infty, 2], [1, +\infty[\}$.

Then, the minimal and maximal supports obtained for each $\mathcal{S}_k$ are combined to obtain all the possibilities for the whole set of sequences $\mathbb{S}$. Let us denote by $\underline{\mathbb{I}}_{ij}^{comb}$ and $\overline{\mathbb{I}}_{ij}^{comb}$ the set of minimal and maximal support combinations, respectively:

$$\underline{\mathbb{I}}_{ij}^{comb} = \{\underline{I}_{ij}^{comb} = \{\underline{I}_{ij}^1, \cdots, \underline{I}_{ij}^n\} | \underline{I}_{ij}^k \in \underline{\mathbb{I}}_{ij}^k, k = 1, \cdots, n\},$$

$$\overline{\mathbb{I}}_{ij}^{comb} = \{\overline{I}_{ij}^{comb} = \{\overline{I}_{ij}^1, \cdots, \overline{I}_{ij}^n\} | \overline{I}_{ij}^k \in \overline{\mathbb{I}}_{ij}^k, k = 1, \cdots, n\}.$$

The union of the minimal supports of every combination of $\underline{\mathbb{I}}_{ij}^{comb}$ is a candidate for being a tree root for (e_i, e_j). The set of tree root candidates for (e_i, e_j) is given by:

$$\mathbb{RC}_{ij} = \{r_{ij}^\alpha = \bigcup_k \underline{I}_{ij}^k, \underline{I}_{ij}^k \in \underline{\mathbb{I}}_{ij}^k, k = 1, \cdots, n, \; \alpha = 1, \ldots, \text{card}(\underline{\mathbb{I}}_{ij}^{comb})\}.$$

However, minimal supports are determined independently for each sequence. As a consequence, a candidate may violate the f_{max} occurrences rule in some of the sequences, in which case it is not valid.

The validity of a candidate can be assessed thanks to the maximal supports. Indeed, the intersection of the maximal supports of every combination of $\overline{\mathbb{I}}_{ij}^{comb}$ provides a *maximal common interval*, denoted MCI, that guaranties exactly f_{max} occurrences of the pair (e_i, e_j) in all the sequences $\mathcal{S}_k \in \mathbb{S}$:

$$\mathbb{MCI}_{ij} = \{MCI_{ij}^\beta = \bigcap_k \overline{I}_{ij}^k, \overline{I}_{ij}^k \in \overline{\mathbb{I}}_{ij}^k, k = 1, \cdots, n, \; \beta = 1, \ldots, \text{card}(\overline{\mathbb{I}}_{ij}^{comb})\}.$$

Property 1. *A tree root candidate r_{ij}^α of $\mathbb{RC}_{ij}$ is valid if $\exists \beta$ such that $r_{ij}^\alpha \subseteq MCI_{ij}^\beta$, where $MCI_{ij}^\beta \in \mathbb{MCI}_{ij}$.*

A valid tree root candidate for (e_i, e_j) is denoted R_{ij}^α and the set of such roots is denoted $\mathcal{R}_{ij}$. The cardinal of $\mathcal{R}_{ij}$ is n_{ij}.

Example: Let us build the first following combinations by taking the first elements of the minimal and maximal supports for each sequence: $\{[-2, 1], [-1, 1], [-1, 1]\} \in \underline{\mathbb{I}}_{12}^{comb}$ and $\{] - \infty, 1],] - \infty, +\infty[,] - \infty, 2]\} \in \overline{\mathbb{I}}_{12}^{comb}$. The intersection of the maximal supports provides a maximal common interval MCI_{12}^1 that validates the union of the minimal supports r_{12}^1, that is:

$$r_{12}^1 = [-2, 1] \subseteq MCI_{12}^1 =] - \infty, 1].$$

Hence, a first valid tree root for pair (e_1, e_2) is given by $R_{12}^1 = r_{12}^1 = [-2, 1]$. However, about the second obtained combinations $\{[-2, 1], [-1, 1], [1, 3]\} \in \underline{\mathbb{I}}_{12}^{comb}$ and $\{] - \infty, 1],] - \infty, +\infty[, [1, +\infty[\} \in \overline{\mathbb{I}}_{12}^{comb}$, the intersection of the maximal supports provides this maximal common interval $MCI_{12}^2 = [1, 1]$ that do not validate the union of the minimal supports $r_{12}^2 = [-2, 3]$. Hence, r_{12}^2 is not a good candidate to be a tree root, it authorizes 3 occurences in sequences 1 and 3 so it violates the $f_{max} = 2$ occurences rule.

The same procedure is applied for $f = f_{max} - 1$ and so on until $f = 1$ to find the tree roots in case no candidate is valid for $f + 1$ or to find the child nodes of the lower levels of the trees. The lowest level always corresponds to $f = 1$ and is never empty because the considered pairs have been taken as pairs appearing in all the sequences. The trees for the different pairs (e_i, e_j) may have a root corresponding to a different frequency. The root frequency for the trees of (e_i, e_j) is denoted f_{ij}^{root}.

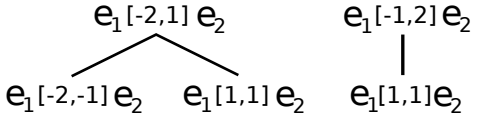

Figure 3: Set of trees for pair (e_1, e_2)

Example: For the example of Figure 2, two valid tree roots are found for pair (e_1, e_2): $R_{12}^1 = [-2, 1]$ and $R_{12}^2 = [-1, 2]$. The set of trees is illustrated in Figure 3 with $f_{12}^{root} = 2$ and $n_{12} = 2$.

3.2 Chronicle discovery algorithm

Once the constraint database $\mathbb{D}$ has been built to account for all the sequences in $\mathbb{S}$ as presented in section 3.1, the algorithm for discovering all minimal frequent chronicles for $\mathbb{D}$, given an input frequency threshold f_{th}, is the same as the HCDA algorithm of [11] but the counting step. This

algorithm is hence called HCDA-modified (HCDAM). The counting step is slightly different because we want HCDAM to provide *all* the frequent chronicles whose frequency is above the specified threshold. Like in [11], the candidate chronicles are called $\mathbb{D}$-chronicles.

The principle of HCDAM is to generate a set of candidate $\mathbb{D}$-chronicles from a chronicle that was proved to be frequent. The set *Candidates* is initiated with the set of root trees $\mathbb{D}^T$. The algorithm maintains two lists:

- the list *Frequent* includes the candidate $\mathbb{D}$-chronicles that have been proved frequent, i.e. whose frequence is higher or equal to f_{th}, and strictest,
- the list *NotFrequent* includes the chronicles that have been proved not frequent.

Instead of immediately counting a candidate $\mathcal{C}$, i.e. determining the minimal number of occurences in the sequences of $\mathbb{S}$, the algorithm first makes use of the generality relation and monotonicity property to discard or accept the candidate without counting:

- if there exists a chronicle $\mathcal{C}'$ more general than $\mathcal{C}$ in *NotFrequent*, then $\mathcal{C}$ is not frequent as well,
- if there exists a chronicle $\mathcal{C}'$ stricter than $\mathcal{C}$ in *Frequent*, then $\mathcal{C}$ is also frequent.

If none of the two above situations apply, then $\mathcal{C}$ is counted, which is performed with CRS (*Chronicle Recognition System*) [14].

In our algorithm, the candidate is also counted in the second situation to determine its actual frequency, which is necessarily higher than f_{th} and higher than the frequency of the stricter chronicle $\mathcal{C}'$. Obviously, only maximal chronicles are saved in *NotFrequent* because this set is used to search for more general chronicles. Conversely, only minimal chronicles are saved in *Frequent*.

Example: Consider the following sequences of events.

Sequence 1	Sequence 2	Sequence 3
$(e_1, 1.049432)$	$(e_1, 13.354919)$	$(e_5, 7.207688)$
$(e_2, 1.606904)$	$(e_2, 14.1784)$	$(e_6, 7.36308)$
$(e_3, 1.873512)$	$(e_3, 14.377672)$	$(e_2, 8.00252)$
$(e_4, 2.18784)$	$(e_4, 14.706472)$	$(e_3, 8.273512)$
$(e_5, 3.441056)$	$(e_5, 15.196873)$	$(e_4, 8.482312)$
$(e_4, 5.871024)$	$(e_4, 18.395527)$	$(e_5, 9.8347435)$
		$(e_4, 12.974768)$

The algorithm HCDA-modified has been used to learn the two following chronicles:

- $\mathcal{C}_1 = (\mathcal{E}_1, \mathcal{T}_1, \mathcal{G}_1)$, with $\mathcal{E}_1 = \{e_2, e_3, e_4, e_5\}$, $\mathcal{T}_1 = \{\tau_{52}, \tau_{54}, \tau_{24}, \tau_{53}, \tau_{23}\}$ and $\mathcal{G}_1$ given in Figure 4.

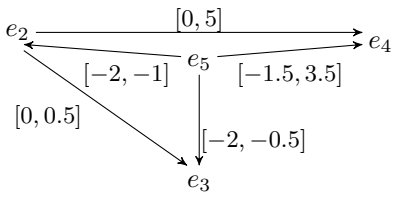

Figure 4: Graph $\mathcal{G}_1$ of chronicle $\mathcal{C}_1$

- $\mathcal{C}_2 = (\mathcal{E}_2, \mathcal{T}_2, \mathcal{G}_2)$, with $\mathcal{E}_2 = \{e_4, e_5\}$, $\mathcal{T}_2 = \{\tau_{54}\}$ and $\mathcal{G}_2$ given Figure 5.

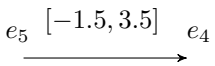

Figure 5: Graph $\mathcal{G}_2$ of chronicle $\mathcal{C}_2$

The second chronicle is a chronicle of frequency 2, which means that it must be recognized twice to sign the situation captured by the three sequences of the exemple. The first chronicle is the stricter chronicle. It involves the larger set of event types with the tightest temporal constraints. It is of frequency 1.

4 Conclusion

This paper deals with the problem of discovering temporal patterns in the form of chronicles that are common to a set of temporal sequences issued from the same situation. The obtained chronicles then sign the situation and can be used for situation assessment or diagnosis purposes. The paper builds on work by [11] and extends the proposed algorithm that targets learning the frequent chronicles for one single sequence to multiple temporal sequences that represent variants of a unique situation. This requires to deeply revise the algorithm to generate the constraint database representing the temporal sequences. The revised algorithm is illustrated by a simple example which helps understand the different steps of the method.

Future research includes theoretical as well as applied work. The complexity of the modified algorithm, in particular the way to build the constraint database, should be carefully analyzed and ways to improve efficiency should be studied. The theoretical complexity of HCDAM is clearly the same as the one of [11] but dealing with multiple sequences reduces the number of constraint graphs in the database as well as the number of possible temporal constraints between each pair of events, resulting in increased tractability. Actually, we believe that improving the algorithm's complexity is rather to be expected from a more efficient way to generate the constraint database and this is one of our goal for the near future.

On the other hand, it is planned to use this work for a real prognostic problem, applying the algorithm HCDAM to the observable signals of a pressure regulation valve of a modern aircraft in different wear situations.

References

[1] MO Cordier and C Dousson. Alarm driven monitoring based on chronicles. In *4th Sumposium on Fault Detection Supervision and Safety for Technical Processes (SafeProcess)*, pages 286–291, Budapest, Hungary, june 2000.

[2] G. Carrault, M.-O. Cordier, R. Quiniou, M. Garreau, J.-J. Bellanger, and A. Bardou. A model-based approach for learning to identify cardiac arrhythmias. In W. Horn, Y. Shahar, G. Lindberg, S. Andreassen, and J. Wyatt, editors, *Proceedings of AIMDM-99 : Artificial Intelligence in Medicine and Medical Decision Making*, volume 1620 of LNAI, pages 165–174, Aalborg, Denmark, june 1999. Springer Verlag.

[3] Benjamin Morin and Hervé Debar. Correltaion on intrusion: an application od chronicles. In *6th International Conference on recent Advances in Intrusion Detection RAID*, Pittsburgh, USA, september 2003.

[4] P. Laborie and J.-P. Krivine. Automatic generation of chronicles and its application to alarm processing in power distribution systems. In *8th international workshop of diagnosis (DX97)*, Mont Saint-Michel, France, 1997.

[5] M.-O. Cordier, X. Le Guillou, S. Robin, L. Rozé, and T. Vidal. Distributed chronicles for on-line diagnosis of web services. In G. Biswas, X. Koutsoukos, and S. Abdelwahed, editors, *18th International Workshop on Principles of Diagnosis*, pages 37–44, May 2007.

[6] Y. Pencolé and A. Subias. A chronicle-based diagnosability approach for discrete timed-event systems: Application to web-services. *Journal of Universal Computer Science*, 15(17):3246–3272, 2009.

[7] F. Fessant, F. Clérot, and C. Dousson. Mining of an alarm log to improve the discovery of frequent patterns. *Lecture Note on Artificial Intelligence*, 3275:144–152, 2004.

[8] B. Guerraz and C. Dousson. Chronicles construction starting from the fault model of the system to diagnose. In *International Workshop on Principles of Diagnosis (DX04)*, pages 51–56, Carcassonne, France, 2004.

[9] Theophano Mitsa. *Temporal data mining*. CRC Press, 2010.

[10] C. Dousson and T. Vu Duong. Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *IJCAI 99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 620–626, San Francisco, CA, USA, June 1999.

[11] D. Cram, B. Mathern, and A. Mille. A complete chronicle discovery approach: application to activity analysis. *Expert Systems*, 29(4):321–346, 2012.

[12] R Agrawal and R Srikant. Fast algorithms for mining association rules. *Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile.*, pages 487–499, Jan 1994.

[13] H. Mannila, H. Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[14] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: representation and algorithms. In *IJCAI: International Joint Conference on Artificial Intelligence*, pages 166–172, Chambéry, France, august 1993.

[15] R. Dechter, Meiri I., and J. Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1):61–95, 1991.