Journal of Intelligent Manufacturing (2020) 31:1019–1035
https://doi.org/10.1007/s10845-019-01492-x

On mining frequent chronicles for machine failure prediction

Chayma Sellami[1] · Carlos Miranda[2] · Ahmed Samet[3] · Mohamed Anis Bach Tobji[4] · François de Beuvron[3]

Received: 18 March 2019 / Accepted: 19 September 2019 / Published online: 27 September 2019

Abstract
In industry 4.0, machines generate a lot of data about several kinds of events that occur in the production process. This huge quantity of information contains valuable patterns that allow prediction of important events in the appropriate instant. In this paper, we are interested in mining frequent chronicles in the context of industrial data. We introduce a general approach to preprocess, mine, and use frequent chronicles to predict a special event; the failure of a machine. Our approach aims not only to predict the failure, but also the time of its appearance. Our approach is validated through a set of experiments performed on the chronicle mining phase as well as the prediction phase. Experiments were achieved on synthetic data in addition to a real industrial data set.

Keywords Chronicle mining · Predictive maintenance · Industry 4.0

Introduction

The connected factory, also known as *Industry 4.0* (Lasi et al. 2014; Oztemel and Gursev 2018) is a project initiated by German industrialists and supported by the government to make machines connected and intelligent. The term "Industry 4.0" refers to a 4th industrial revolution. The first revolution began with the use of steam power and the first machines. The second introduced electrical energy and the mass production, while the third one started with the use of electronics and computers to automate production. Industry 4.0 uses the concepts of Internet of Things (IoT) (Xia et al. 2012) that offers ubiquitous computing through advanced connectivity of products, systems and services. Due to the ubiquitous nature of objects, a remarkable number of systems will be connected to the Internet. The main aspect of IoT is the use of these connected objects in manufacturing processes. This means that smart sensors and smart tools in general will be available in the factory. These technologies will improve performance of the manufacturing processes in real time by acting in a pro-active manner.

The analysis of collected data allows to model the behaviour of a machine, be it normal or abnormal. When these models are confronted with real-time data, monitoring systems can detect anomalies at appropriate time, and send an alert to humans for a timely intervention (Mobley 1990). This is the principle of *predictive maintenance* (Hashemian and Bean 2011).

Intelligent systems that allow this kind of maintenance are based on analyzing collected signals, that are generally a set of timestamped events. For this aim, data mining techniques are used, particularly sequential data mining (Agrawal and Srikant 1995) and frequent sequential pattern mining (Borgelt 2012).

However, sequential patterns have a main shortcoming (Cram et al. 2011); they inform about the sequentiality of events but nothing about the gap time between events.

Let the data set shown in Table 1 where events are accompanied by instants of occurrence in each tuple.

We can note that sequence $\langle A, B, C \rangle$ is redundant. It shows that events A, B and C occurred frequently in a sequence manner, but without providing any additional information about the gap between them. A richer pattern where time constraints are considered is the *chronicle*, initially

Reviewers are invited to visit this site https://sites.google.com/view/frequent-chronicle-mining/accueil for more information on the development and results of the approach.

✉ Chayma Sellami
chayma.sellami@esen.tn

[1] ESEN, Univ. Manouba, Manouba, Tunisia

[2] INSA Rouen, Rouen, France

[3] ICUBE / SDC Team (UMR CNRS 7357), Pole API BP 10413, 67412 Illkirch, France

[4] ISG, LR01ES02 LARODEC, Université de Tunis, Tunis, Tunisia

Table 1 Data set of sequences (pairs of event/instant)

Sequence id	Events
1	(A,0), (B,5), (C,7)
2	(A,2), (B,3), (C,7)

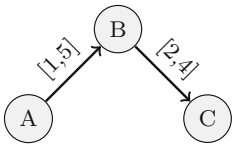


Fig. 1 A chroncile extracted from Table 1

introduced in Dousson and Duong (1999) and developed in Huang et al. (2012) and Cram et al. (2011). In our data set example, we can deduce that A, B and C occur sequentially, and that B occurs after A at least after one instant and at most after 5 instants, while C occurs after B in the interval $[2, 4]$ of instants. We represent our chronicle as $A[1, 5]B$ and $B[2, 4]C$. It is a direct graph where nodes are events and vertices are the instant intervals, denoted by *time constraints* as shown in Fig. 1.

To optimise machines' performance, critical events should be anticipated. Predictive maintenance is based on this principle. Predictive maintenance is of paramount importance and offers considerable potential for innovation to overcome the limitations of traditional maintenance policies (Cho et al. 2018). It consists of collecting and analysing the data of industrial equipments. Then, an alert system learn from previous event sequences and prevent from imminent failures. In opposite to preventive maintenance (Rivera Torres et al. 2018), predictive maintenance avoid a failure before it occurs. To the best of our Knowledge, no research work has been interested in applying frequent chronicle mining to predictive maintenance. This paper introduces a complete data mining approach based on the extraction of frequent chronicles to predict machine failures. In opposite to classic prediction techniques, the aim of our approach is to predict not only the failure event, but also the time interval of its occurrence as time dimension is crucial in predictive maintenance. As chronicle mining algorithms are resource greedy, we are obliged to improve existing works to scale with manufacturing requirements (i.e. large number of sequences, real-time prediction, etc). Thus, we introduce a new algorithm for the extraction of frequent chronicles called *Clasp-CPM*. This algorithm generates only closed failure chronicles in an effective manner. in addition, to handle the huge volume of sequences, several optimization methods and multi-threading coding are brought up. Then, extracted failure chronicles serve as input to a classification algorithm that predict machine failures. Our general approach is validated on a real industrial data set denoted by *SECOM* (McCann et al. 2008) as well as on synthetic data sets. Performance of our algorithms and quality of failures' predictions were investigated against the aforementioned kind of data. To summarize, this paper introduces three contributions: (i) a new approach for mining failure chronicles from a set of sequences that report a machine activity; (ii) a new efficient algorithm called Clasp-CPM introduced to mine failure chronicles; and finally (iii), a new algorithm called FADE that uses the mined chronicles to predict if a new sequence of parameters values will lead to a machine failure, and if yes, in which time interval the crash will occur.

Related works

The literature is plentiful of works that have dealt with the subject of temporal data mining in order to discover interesting patterns (Zhao and Bhowmick 2003; Masseglia et al. 2005; Laxman and Sastry 2006). Sequential Pattern Mining (Srikant and Agrawal 1995) (commonly called SPM) is the discovery of sequences that are frequent in a set of sequences. The process is similar to the frequent itemset mining (Goethals 2003), except that the data set events are ordered and time-stamped. Similarly to the frequent pattern mining problem, SPM algorithms extract frequent subsequences according to a user-defined threshold commonly called minimum support. The pioneering SPM algorithm, called AprioriAll, is introduced in Agrawal and Srikant (1995). It generates the set of frequent sequential pattern in a level-wise manner as does Apriori (Agrawal et al. 1993).

Several algorithms have been introduced to discover sequential patterns from sequences, such as GSP (Srikant and Agrawal 1996), SPADE (Zaki 2001), PrefixSpan (Pei et al. 2001) for frequent sequential patterns based on Apriori, and CloSpan (Yan et al. 2003) and ClaSP (Antonio et al. 2013) for frequent closed sequential patterns. SPM techniques have many uses in several fields of application such as DNA sequence analysis (D'Addona et al. 2017; Zerin and Jeong 2011), Web access models (Fournier-Viger et al. 2012), etc.

It has been proven that sequential patterns are not sufficiently informative in several application fields such as network alarm (Mannila et al. 1997) or analysis of human activity (Mannila et al. 1997). Therefore, the chronicle pattern model, that is an extension of sequential patterns, has been introduced Dousson et al. (1993). Chronicles are rich of information because they add the exact time interval where a sequence events occur. However, this richness raises several drawbacks such as memory consumption and execution time. In Dousson and Duong (1999), made the foundation of what has been later known as chronicle mining. They proposed the first algorithm for chronicle extraction from journal logs of telecommunication alarms. Unfortunately, it has been pointed out that this algorithm also known as Frequency Analyzer for Chronicle Extraction (FACE) is not complete (i.e., does not generate all patterns).

In Cram et al. (2011) introduced the HCDA algorithm, to mine the complete set of chronicles. Frequent chronicles of size 2 are mined and those with the same items are grouped in a tree. Chronicles with the largest interval are located in the root and the tightest are placed in leaves. The CCP-Miner algorithm (Huang et al. 2012), is an extension of HCDA algorithm that searches for chronicles in a set of sequences. It has been applied to extract sequences corresponding to patient pathways in a hospital center. In Dauxais et al. (2017), Dauxais et al. proposed a new approach to extract discriminant chronicles in to the context of pharmaco-epidemiology. The proposed DCM algorithm allows to mine chronicles in a labelled sequential data set that more representative of a single phenomena. Finally, Sellami et al. (2018) introduced a new approach, known as CPM, to mine chronicles from a set of sequential patterns extracted using the CloSpan algorithm (Yan et al. 2003).

Chronicle mining: the basic notions

Extracting frequent chronicles requires discovering sequential patterns taking into consideration temporal information, which is the time of occurrence of the events in the sequences. In this section, we introduce all definitions of the concepts necessary for the task of chronicle mining.

Definition 1 (*Event*) According to Cram et al. (2011), an event is a couple (e, t) where $e \in \mathbb{E}$ is the type of the event and $t \in \mathbb{T}$ is its time.

These events appear together in their order of occurrence, called timestamped events, which allows us to form a sequence.

Definition 2 (*Sequence*) Let $\mathbb{E}$ be a set of event types, and $\mathbb{T}$ a time domain such that $\mathbb{T} \subseteq \mathbb{R}$. $\mathbb{E}$ is assumed totally ordered and is denoted $\leq_{\mathbb{E}}$. A sequence is a couple $\langle SID, \langle (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)\rangle\rangle$ such that SID is the index of the sequence and $\langle (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)\rangle$ is a sequence of events. For all $i, j \in [1, n], i < j \Rightarrow t_i \leq t_j$. If $t_i = t_j$ then $e_i <_{\mathbb{E}} e_j$.

The appearance of timestamped events in a sequence allows us to define *temporal constraints* between them.

Definition 3 (*Temporal constraint*) A time constraint is a quadruplet (e_1, e_2, t^-, t^+), denoted $e_1[t^-, t^+]e_2$, where $e_1, e_2 \in \mathbb{E}$, $e_1 \leq_{\mathbb{E}} e_2$ and $t^-, t^+ \in \mathbb{T}$.

A time constraint $e_1[t^-, t^+]e_2$ is said satisfied by a couple of events $((e, t), (e', t'))$, $e \leq_{\mathbb{E}} e'$ iff $e = e_1$, $e' = e_2$ and $t' - t \in [t^-, t^+]$.

We say that $e_1[a, b]e_2 \subseteq e'_1[a', b']e'_2$ iff $[a, b] \subseteq [a', b']$.

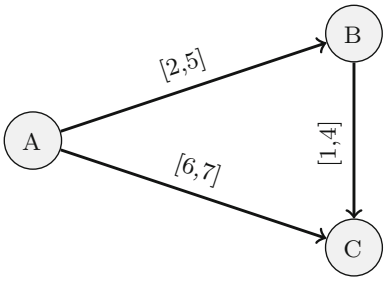


Fig. 2 Example of a chronicle

The extraction of temporal constraints between the events of a sequence leads us to define the concept of chronicles (Dousson and Duong 1999).

Definition 4 (*Chronicle*) A chronicle is a pair $\mathcal{C} = (\mathcal{E}, \mathcal{T})$ such that:

1. $\mathcal{E} = \{e_1 \ldots e_n\}$, where $\forall i, e_i \in \mathcal{E}$ and $e_i \leq_{\mathbb{E}} e_{i+1}$,
2. $\mathcal{T} = \{t_{ij}\}_{1 \leq i < j \leq |\mathcal{E}|}$ is a set of temporal constraints on $\mathcal{E}$ such that for all pairs (i, j) satisfying $i < j$, t_{ij} is denoted by $e_i[t_{ij}^-, t_{ij}^+]e_j$.

$\mathcal{E}$ is called the episode of $\mathcal{C}$, according to the definition of episode's discovery in sequences (Mannila et al. 1997).

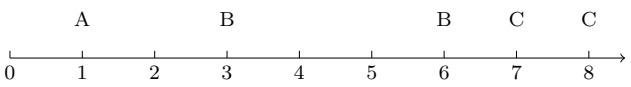
The relevance of a chronicle is based essentially on the value of its support. The support of a chronicle refers to the number of its occurrences in a sequence. It can therefore be formalized by the definition below.

Definition 5 (*Chronicle support*) An occurrence of a chronicle $\mathcal{C}$ in a sequence S is a set $(e_1, t_1)...(e_n, t_n)$ of events of the sequence S that satisfies all temporal constraints defined in $\mathcal{C}$. The support of a chronicle $\mathcal{C}$ in the sequence S is the number of its occurrences in S.

Example 1 Let us illustrate all these basic definitions. Assuming a sequence S of three events $\langle A, B, C\rangle$ represented as follows:

Time constraints that describe the pattern {A, B, C} are noted by A[2,5]B, B[1,4]C and A[6,7]C.

After the generation of temporal constraints, these events can be represented as a graph, as shown in Fig. 2.



Chronicle mining for predictive maintenance

In this work, we seek to develop an approach to detect machine anomalies in advance. Previous works such as Carrault et al. (2003), Fradkin and Mörchen (2015), Vautier et al.

(2005) and Huang et al. (2012), treated the extraction of frequent chronicles, but none put it in the context of predictive maintenance. Our contribution is developed to solve this problem and aims to answer the question, i.e. how can we use temporal constraints between events, and therefore chronicles, to predict anomalies before they occur?

Approach overview

The aim of our approach is to predict anomalies of machines in an industrial context. Our goal is not only to predict the failure, but also the time interval of the occurrence of this failure. For this purpose, we rely on mining the most frequent chronicles describing the events that lead to a machine failure. Our interest in this kind of temporal pattern lies not only in predicting an event, but especially in the time interval in which that event will occur (in our case a machine failure).

Like any knowledge discovery process, our approach starts with a preprocessing step, a mining step and a third step for the interpretation of extracted knowledge. The overall approach is described in Fig. 3.

The data preprocessing step

Chronicle mining algorithms handle data that are discrete and sequential as mentioned in Definition 2. However, the generated data from industrial machines are not necessarily in that format. Raw industrial data are often continuous and not sequential in the sense of Definition 2. Consequently, we discretize continuous values to obtain nominal ones, i.e., the events. Then, we use sequentialization to transform data in a set of sequences in the form of pairs (event, instant) where each sequence finishes with the failure event.

Furthermore, the number of measures in an industrial data set could be tremendous. Analysis of such huge number of data dimensions is on the one hand costly and on the other useless, since not all feature attributes are necessarily relevant to predict the failure event. Therefore, before the discretization and sequentialization steps, we apply a feature selection method, that computes the most relevant attributes in predicting the breakdown[1] event.

At this point, the resulting data set is ready to mine.

The failure chronicle mining step

In this step, we aim at discovering chronicles that represent breakdown. This type of chronicles is called *failure chronicle* and is introduced in Definition 6.

Definition 6 (*Failure chronicle*) Assuming a chronicle $\mathcal{C}_F = (\mathcal{E}, \mathcal{T})$. We say that $\mathcal{C}_F$ is a *failure chronicle* if and only if the events that describe it are set according to their order of occurrence in the sequence, and that the end of the chronicle is the event that represents the failure, i.e. for $\mathcal{E} = \{e_1 \cdots e_n | e_i \leq_{\mathbb{E}} e_{i+1}, i \in [1, n]\}$, e_n is the failure event.

To discover the frequent failure chronicles, we proceed in two steps, namely:

1. **Extraction of the frequent closed sequential patterns from our set of sequences.** We recall here that in our preprocessed data set, all sequences end with a breakdown event. We chose in this step to extract the closed frequent sequential patterns to produce efficiently a minimal set of patterns that describe our sequences. Among the two methods presented in "Related works" section, namely ClosPan and ClaSP, we retained the second one because of its confirmed efficiency (Antonio et al. 2013).
2. **Extraction of the frequent chronicles.** In this step, we scan again the data set to extract the time constraints related to the sequential patterns mined in the previous step. This operation is performed by a new chronicle mining algorithm we introduce in "Clasp-CPM" section.

The failure prediction step

The generated failure chronicles is a set of sequential patterns that describe the most frequent sequence of events that lead to a failure machine. Chronicles provide not only the order of occurrence of those events, but also the interval of time they occur in. Therefore, when a new sequence of timestamped events arrive, we can predict if it favours a breakdown or not by comparing it to the set of mined frequent failure chronicles as well as the time interval it will probably happen. This step is detailed in "Failure detection with chronicles" section.

Clasp-CPM

The first implementation of chronicle mining for predictive maintenance (Sellami et al. 2018) handled few cases of time constraints extraction. It was dependent on the length of the patterns and treated these on a case-by-case basis. Moreover, chronicle extraction needed two steps of time constraints extraction: one for the events on the patterns, and another to extract constraints between regular events and the breakdown event. To overcome these two major setbacks, two notions were used: subsequence graphs and suffix data sets.

First, we introduce a sample data set which will be used for examples in this section.

Definition 7 (*Closed Frequent sequential patterns*) Let D be a sequence data set, i.e. $D = \{s_i\}_{i=1}^{N}$ where $\forall i \in [\![1, N]\!]$, s_i is a sequence.

[1] In this paper, we mean by "breakdown" a failure.

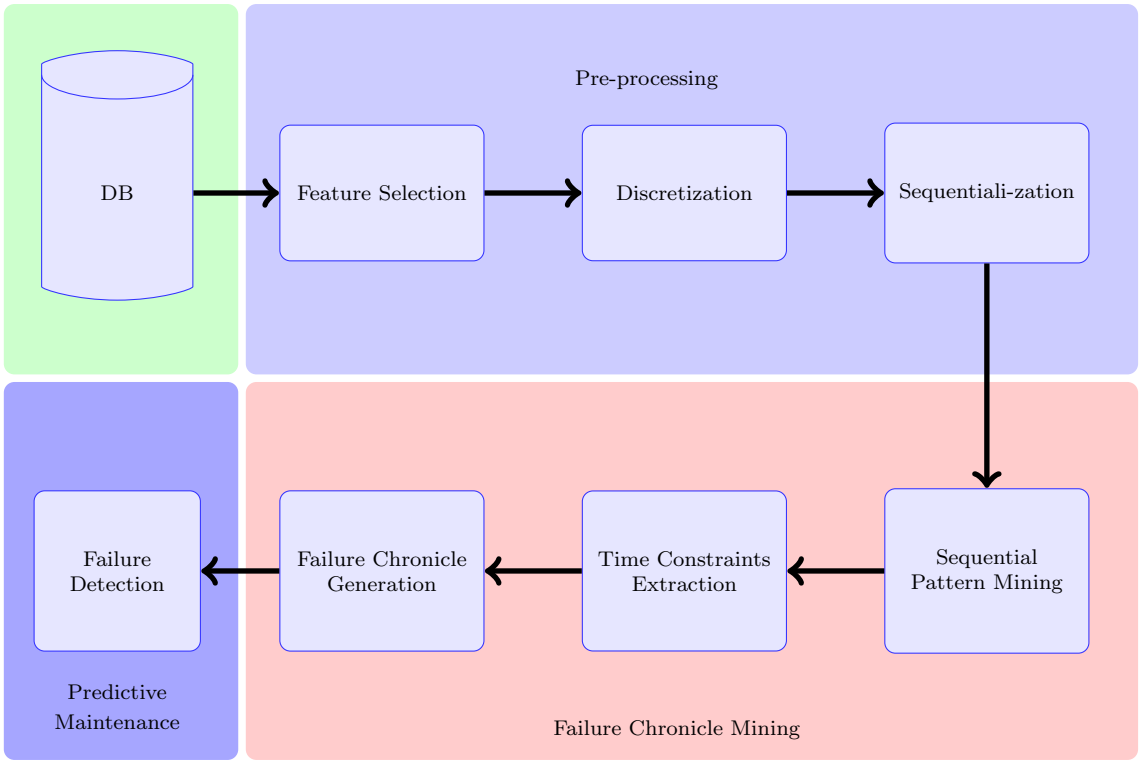


Fig. 3 The different steps describing our approach

Table 2 Sample data set

Seq. id	Events
1	({A, C, E}, 1), ({B, D, F}, 3), ({A, C, F}, 4), ({A, C, E},8), ({B, D, E}, 10)
2	({A, C, F}, 2), ({A, D, F}, 6), ({A, C, E},7), ({B, D, F}, 9)
3	({A, D, F}, 0), ({A, C, F}, 3), ({A, C, F},7), ({B, D, F}, 8),({B, C, E}, 12)
4	({A, C, F}, 1), ({A, D, F}, 3), ({A, C, E},4), ({B, D, F}, 7), ({A, D, E}, 9)
5	({A, D, F}, 2), ({A, C, F}, 4), ({B, D, E}, 7)

Let $\mathcal{FS}$ be the set of frequent sequential patterns for a given minimum support $minsupp$, i.e.

$$\mathcal{FS} := \{s | s \subseteq s_i \in D \wedge \text{supp}(s) \geq minsupp\}$$

Then, we define $\mathcal{CS}$, the set of frequent closed sequences as:

$$\mathcal{CS} := \{s | s \in \mathcal{FS} \wedge (\nexists \beta \in \mathcal{FS} | s \subseteq \beta \wedge \text{supp}(s) = \text{supp}(\beta))\}$$

One can easily notice that $\mathcal{CS}$ is a subset of $\mathcal{FS}$.

Example 2 (*Closed frequent sequences examples*) Referring to Table 2, with a relative $minsupp$ of 0.8, one can see that the sequences $\langle A \rangle$, $\langle A, A, B \rangle$ and $\langle A, A, A, B \rangle$ are frequent sequences (they all appear in at least 4 of the sequences). However, $\langle A \rangle$ is not closed, as it has a support of 1 and is included in $\langle A, A, B \rangle$, which also has a support of 1. On the contrary, $\langle A, A, B \rangle$ is a closed frequent sequence, since, even if it is included in $\langle A, A, A, B \rangle$, the support of the latter is 0.8.

Thus, we see that by keeping closed patterns only, we filter simpler, less informative patterns, but we retain those which are more frequent than other, more complex patterns.

Definition 8 (*Concatenation operators*) Let $s = \langle \alpha_1, \ldots, \alpha_p \rangle$ and $s' = \langle \beta_1, \ldots, \beta_l \rangle$, where α_i and β_i are sets. Let $\Diamond_\bullet$ be the concatenation operator. We define two kind of concatenations (Yin et al. 2012):

- $\Diamond_i$: $s \Diamond_i s' = \langle \alpha_1, \ldots, \alpha_{p-1}, (\alpha_p \bigcup \beta_1), \beta_2, \ldots, \beta_l \rangle$. Here, the first element of s' merges with the last element of s, and then we just append the rest of the second sequence.
- $\Diamond_s$: $s \Diamond_s s' = \langle \alpha_1, \ldots, \alpha_{p-1}, \alpha_p, \beta_1, \beta_2, \ldots, \beta_l \rangle$. This is usual concatenation.

Table 3 Sample suffix database: P is the failure event, s-concatenated at the end of each sequence

Seq. id	Events
1	({A, C, E}, 1), ({B, D, F}, 3), ({A, C, F}, 4), ({A, C, E},8), ({B, D, E}, 10), ({P}, 10)
2	({A, C, F}, 2), ({A, D, F}, 6), ({A, C, E},7), ({B, D, F}, 9), ({P}, 9)
3	({A, D, F}, 0), ({A, C, F}, 3), ({A, C, F},7), ({B, D, F}, 8), ({B, C, E}, 12), ({P}, 12)
4	({A, C, F}, 1), ({A, D, F}, 3), ({A, C, E},4), ({B, D, F}, 7), ({A, D, E}, 9), ({P}, 9)
5	({A, D, F}, 2), ({A, C, F}, 4), ({B, D, E},7), ({P}, 7)

Definition 9 (*Suffix database*) Let ω be a sequence. D_ω is said to be the *suffix database* associated to D if:

$$\forall s \in D_\omega, \exists s' \in D, s = s' \Diamond_s w$$

that is if ω is a suffix for all sequences in D_ω. We will note $\mathcal{FS}_\omega$ and $\mathcal{CS}_\omega$ the set of frequent sequences and the set of closed frequent sequences associated to the suffix database D_ω, respectively.

Remark 1 – We have defined the suffix database with the s-concatenation operator as i-concatenation does **not** preserve the closeness property we need. See B for further details.
- In our application, we will use a sequence of length 1 for ω.
- $\#D = \#D_\omega$, i.e. D and D_ω have both the same number of sequences.
- $\mathcal{FS} \subseteq \mathcal{FS}_\omega$, more precisely $\mathcal{FS}_\omega = \mathcal{FS} \cup \{s \Diamond_s \omega | s \in \mathcal{FS}\}$. This tells us that if a sequence is frequent in D, then it is also frequent in D_ω, but the converse does not hold.
- $s_\omega \in \mathcal{FS}_\omega \wedge s_\omega = s \Diamond_s \omega \Rightarrow s \in \mathcal{FS}$.

Example 3 (*Suffix database*) In Table 3, we have built a suffix database, by s-appending the failure itemset {P} at the end of each sequence. One can choose the timestamp to be different to that of the last element, but the added time should be constant to keep a coherent analysis later.

Proposition 1

$$\mathcal{CS}_\omega = \{s \Diamond_s \omega | s \in \mathcal{CS}\}$$

Lemma 1 *Let ω be the sequence associated to the suffix database D_ω built from the database D. Then*

$$\forall s \in \mathcal{FS}, \exists s' \in \mathcal{FS}_\omega, s' = s \Diamond_s \omega : \text{supp}(s) = \text{supp}(s')$$

Proof Reader may refer to 8 for the detailed proof. □

This proposition is useful to prove that every closed frequent sequence of a suffix database ends with the sequence ω. We use this fact to improve the previous algorithm given, as there will be only one *ExtractTimeConstraints* procedure and there will be no union of chronicles at the end.

Proof Reader may refer to A for the detailed proof of Proposition 1. □

Subsequence graph

A subsequence graph is a data structure built to represent all occurrences of a sequence within another sequence. It was designed in the image of the Knuth–Morris–Pratt algorithm for word matching.

Definition 10 (*Subsequence graph*) Let $s = \langle s_1, \ldots, s_n \rangle$ be a sequence and $p = \langle p_1, \ldots, p_m \rangle$, a pattern. A susbequence graph is a couple (X, U) where X are the vertices and U are the (directed) edges. Vertices $v_{i,j}$ correspond to elements of the pattern p_i, indexed by their position on s, j. There is an edge from $v_{i,j}$ to $v_{k,l}$ iff:

- $k = i$ (both pattern elements are the same) and $v_{k,l}, l > j$, is the first next occurrence (of such pattern element) in s.
- $k = i + 1$ and there is no edge from $v_{i,j}$ to $v_{k,m}$ with $m < l$ ($v_{k,l}$ is the first occurrence of p_{i+1} after $v_{i,j}$).

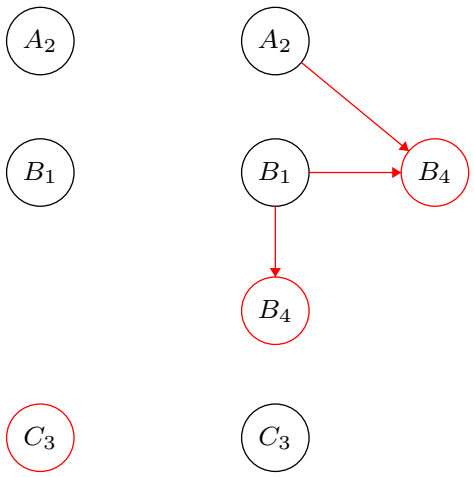
Algorithm 1 Subsequence Graph Construction

Require: S : Sequence, P : Pattern
Ensure: G : Subsequence Graph
1: $minQ \leftarrow 0, maxQ \leftarrow 1$
2: **for** $(i \leftarrow 0; i < length(P); i \leftarrow i + 1)$ **do**
3: $G[i] \leftarrow \emptyset$
4: **for** $(i \leftarrow 0; i < length(S); i \leftarrow i + 1)$ **do**
5: $current \leftarrow S[i]$
6: **for** $(j \leftarrow minQ; j < maxQ; j \leftarrow j + 1)$ **do**
7: **if** $P \subseteq current$ **then**
8: $insertedNode \leftarrow$ addNode(G, i, j)
9: **if** $j > 0$ **then**
10: linkLayer$(j - 1, insertedNode)$
11: **if** $maxQ < length(P)$ **then**
12: $maxQ \leftarrow maxQ + 1$
13: **if** $length(P) + i + 1 > length(S)$ **then**
14: $minQ \leftarrow minQ + 1$

Example 4 (*Graph construction*) Let us take the pattern $\langle A, B, B, C \rangle$ and the sequence $\langle B, A, C, B, B, A, C, A, B, C, B, A, B, C \rangle$. Table 4 includes the enumerated sequence:

Table 4 Sequence enumeration

B	A	C	B	B	A	C	A	B	C	B	A	B	C
1	2	3	4	5	6	7	8	9	10	11	12	13	14



(a) After three steps **(b)** After four steps

Fig. 4 Subsequence graph: third and fourth steps

1. First, we initialize an empty list of lists, such that the first element correspond to the occurrences of A, the second one to the occurrences of B, the third one to the occurrences of B following a B and the fourth one to the occurrences of C.
2. Next, in the first step, B is placed in the second list. Then, A is placed in the first list. In step three, C is placed in the fourth list (Fig. 4a). Notice there are no links yet as the elements, while they are individually in the pattern, have not occurred in the order of the pattern.
3. During the fourth iteration, we find a second B. We proceed in four steps: B is placed in the third list, we add links from the second list elements, without links to a third list element, to this B element, then we add B to the second list and we add links from the first list elements, without links to a second list element, to this B (Fig. 4b).
4. We continue the process. During step seven, the graph allows to extract the first occurrence of the pattern in the sequence: $\langle(A, 2), (B, 4), (B, 5), (C, 7)\rangle$ (Fig. 5a).
5. Last, when the whole sequence has been treated, one can perform pruning steps to remove the nodes without links or that do not allow to extract a whole pattern (Fig. 5b).

We devised this structure to be able to extract all the information needed, namely timestamps and precedence relations for each element of a pattern in a sequence. It is not consuming in space, as each element is only referenced.

Multi-threading

The first step of CPM is frequent closed sequences mining, which we found hard (if possible) to parallelize. Thus, we enchanced the following steps, time constraint extraction and chronicle building, with multi-threading.

Using a Producer–Consumer approach, we defined the following schema :

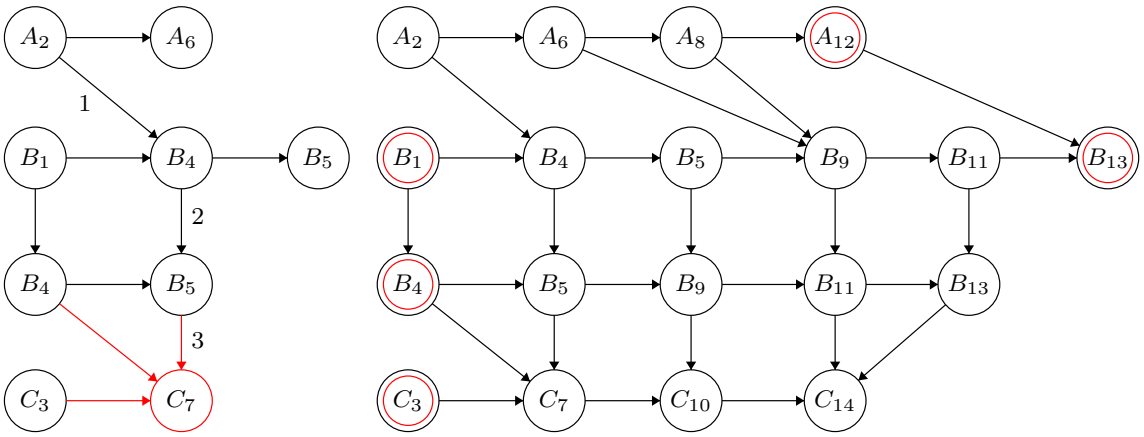
1. Initialize a certain number of workers, a pool of extracted closed patterns, a pool of time constraints sets and a pool of chronicles.
2. (Time constraint extraction) For half the workers:
   (a) Take a pattern from the pool.
   (b) Extract the time constraints associated with it, i.e. build a subsequence graph for each sequence in which the pattern is found and use it to extract the constraints.
   (c) Put the time constraints in the corresponding pool.
   (d) Repeat until the pattern pool is empty.
3. (Chronicle Building) For half the workers:
   – Take the constraints from the pool.
   – Build a chronicle using the time constraints.
   – Put the chronicle in the corresponding pool.
   – Repeat until the constraint pool is empty.

An improvement would be to switch workers tasks when one of the pools is empty, e.g. if the time constraint pool is empty, then all workers are extracting time constraints, and vice versa.

There is no race conditions when accessing the data set as the sequences are not modified but only read. Our implementation uses blocking queues and linked blocking queues (java) for the pools. Here again, each element is a reference only, so we do not significantly increase memory usage beyond the extraction step.

Failure detection with chronicles

Clasp-CPM allows to mine failure chronicle. Semantically, it mines events and indicators of failure. This kind of information is useful to monitor machines and predict the failure. Therefore, we intend to use these mined chronicles for classification and prediction problems. We introduce the algorithm FADE shown in Algorithm 2. The latter uses the set of extracted failure chronicles as input. FADE tries to match failure chronicle to a single sequence. To define chronicle sequence match, we introduce the following definitions.



(a) Subsequence graph : seventh step

(b) Subsequence graph : useless nodes are marked to be pruned

Fig. 5 Subsequence graph: final steps

Table 5 Sequences' data set

Seq. id	Events
1	(A,5), (B,7), (Failure, 9)
2	(A,5) (C,6) (B,7) (Failure, 9)
3	(A,2), (B,6), (Failure, 17)

Definition 11 (*Chronicle cover*) Assuming a sequence $S = \langle (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n) \rangle$ and a failure chronicle c. We say that c covers the sequence SID if and only if the events represented by the chronicle belong to the sequence as well as the time intervals between these events in the sequence belong to the temporal constraints extracted by the chronicle, i.e.,

$$\mathcal{C} < \cdot S \Leftrightarrow \forall e_i [t^-, t^+] e_j \in \mathcal{C},$$
$$\exists ((e, t), (e', t')) \in S \wedge e = e_i, e' = e_j \wedge t' - t \in [t^-, t^+].$$

Definition 12 (*Supported failure chronicle*) Assuming a sequence S and the set of failure covering chronicle ,i.e., $\mathbb{C} = \{c \in \mathcal{C}, c < \cdot S\}$. We say that $\mathcal{C}_f$ is the supported failure chronicle if and only if it has the maximal support among all chronicles of the set $\mathbb{C}$, i.e., $\mathcal{C}_f = c \in \mathcal{C} \wedge \nexists c' \in \mathcal{C} \wedge supp(c') > supp(c)$.

Let explain this principle in the following example.

Example 5 Assuming the following chronicle: A[0,3]B, B[0,7]Failure and the three sequences depicted in Table 5.

For the first sequence, the duration between events A and B is 2 instants,[2] that belongs to [0,3], and between B and the failure is 2 instants belongs to [0,7]. So the occurrence's time of the failure is in the interval illustrated by the chronicle, so we have classified this sequence correctly, likewise for the second sequence. Whereas for the third sequence, the interval between A and B is 4 does not belong to [0,3], and between B and the failure 11 does not belong [0,7]. In this sequence, the failure that has appeared but is not validated by this chronicle. So this failure will not be predicted and the duration of the sequence will be misclassified. Suppose we have two chronicles: $\mathcal{C}_1$ = A[0,3]B, B[0,7]Failure, $\mathcal{C}_2$ = C[0,9]Failure. The second sequence in the Table 5 {(A,5) (C,6) (B,7) (Failure, 9)} is covered by these two chronicles, so to have a relevant prediction, we must choose one of the two chronicles. The chronicle that has the highest support in the data set will be kept.

In this example, the first chronicle covers the first two sequences, so $supp(\mathcal{C}_1) = 2$, and the chronicle $\mathcal{C}_2$ only covers the second sequence, so $supp(\mathcal{C}_2) = 1$, subsequently the supported failure chronicle for this sequence is the chronicle $\mathcal{C}_1$.

Definitions 11 and 12 are implemented in Algorithm 2. First, the algorithm uses the coverage procedure to check whether the processed sequence matches at least one element from the set of chronicles. This procedure takes as parameter a sequence and a set of chronicles. This process ensure that for a single chronicle, only the covering chronicle with the highest support is retained.

Experimentation and results

To validate our approach, we have performed a large set of experiments. Two aspects were subjected to evaluation; the

[2] In this paper, we mean by "instant" a given unit of time.

Algorithm 2 Failure Detection with Chronicles

Require: S: Sequence, $\mathcal{C}$: Chronicles set
Ensure: $\mathcal{C}_f$
1: $\mathcal{C}_f \leftarrow \emptyset$, Struct Tab[]
2: **if** $(COVERAGE(S, \mathcal{C}) \neq 0)$ **then**
3: **for** (i=0; i<Tab.length, i++)
4: **for** (j=0; j< $\mathcal{C}$.length, j++)
5: Tab[i] ← C[j]
6: **end for**
7: **end for**
8: $\mathcal{C}_f \leftarrow Tab[0]$
9: **for** (i=1; i<Tab.length, i++)
10: **if** $supp(Tab[i]) > supp(\mathcal{C}_f)$ **then** $\mathcal{C}_f \leftarrow Tab[i]$
11: **if** $\mathcal{C}_f \neq \emptyset$ **then**
12: **return** $\mathcal{C}_f$

performance in term of resources' cost, as well as the prediction quality of the machine failures. As the costly part of our approach is mining the frequent chronicles, we experimented *Clasp-CPM* on both synthetic data and a real industrial data set. Then, we experiment the quality of failure prediction, i.e., the *FADE* algorithm on the real data set only. To this aim, we used the cross validation principle (Stone 1974) as it is the most used in literature. We compared FADE with the state-of-the-art chronicle mining based approaches. Our approach not only predicts failure, but also its time interval. Classical classification approaches like K-Nearest Neighbours or Long Short Term Memory (LSTM) (Malhotra et al. 2015) to cite a few are unable to consider the time dimension neither mined pattern for training, that's why they are ignored in this experimental study. All experiments were performed on a personal computer equipped with a 2.5 GHz processor and 16 GB main memory under the Microsoft Windows OS.

Experimental data sets

As mentioned above, synthetic data sets were generated to evaluate the scalability of *Clasp-CPM* according to several parameters we identified. These parameters are the number of sequences in the data set (denoted DB), the size of a single sequence (denoted seq_size), the maximum number of items (also called the dictionary size, denoted dic_size) and the maximum number of items per event (denoted $max\_items/event$). Our generator produces randomly a set of time-stamped sequences according to the parameters above. The experiments presented in the next subsection consists in comparing the performance of *Clasp-CPM* to existing algorithms over several generated data sets where the parameters in question were varied.

In addition, our experiments were performed on a real industrial data set; the SECOM data set (McCann et al. 2008). It consists on a set of measurement data captured from senors installed on the machines of a manufacturing production line. Each row contains a set of measures and signals produced by the machine, its status (1 for normal running and −1 for a failure) and a time-stamp (the instant where the machine measures and status were observed). The SECOM data set includes 590 attributes and 1567 records. To monitor the semi-conductor manufacturing process, these data are mined. However, to benefit from the concept of *chronicles*, data have to be pre-processed, hence the discretization and the sequentialisation performed of the raw SECOM data. Feature selection is also achieved to reduce dimensionality of data, and to keep only the relevant measures that affect at most the machine status.

Evaluation of the chronicle mining phase

In this first part, we confront *Clasp-CPM* to other algorithms of the state-of-the-art using synthetic data sets. We compared it to a previous brute-force algorithm called *CPM* (Sellami et al. 2018), as well as DCM (Dauxais et al. 2015) and FACE (Dousson and Duong 1999). In our experiments, we used different synthetic data sets to test the effect of several parameters on the results. Table 6 shows the number of generated chronicles by the aforementioned algorithms for a range of support threshold that goes from 0.4 to 1 for three different synthetic data sets where we change seq_size, dic_size and $max\_items/event$.

As Clasp and Clospan extract the same number of closed patterns (Antonio et al. 2013), the experiments show that both Clasp-CPM and CPM generate the same number of frequent chronicles. This number depends on the different parameters used when generating the data set. Obviously, it increases considerably while increasing the parameters' values. These same experiments done on FACE algorithm (Dousson and Duong 1999) show that this algorithm generates the highest number of chronicles, since it is based on Apriori algorithm. In fact, Apriori extracts all the frequent patterns, not only the closed frequent ones. In our introduced approach, we bypassed the Apriori-like methods to avoid redundant chronicles, and to optimise the performance of our chronicle mining step. On the other hand, DCM (Dauxais et al. 2015) was designed to consider discriminancy in data, so the comparison with our algorithm is not straightforward.

Extracting chronicles was only possible for threshold values greater than 0.7 as shown in Table 6. One hypothesis for these results is, as already mentioned, the use of the discriminance constraint. This parameter is used in the epidemiology algorithm to distinguish two populations (positive and negative) and to extract patterns that are frequently present in the positive base, which is not really the case in our approach since we only process a single data set (population) at a time.

Table 6 shows that the maximum size of a sequence is the parameter that affects the most the number of obtained chron-

Table 6 Number of chronicles of Clasp-CPM, CPM, DCM and FACE w.r.t data set size, sequence's size, dictionary size, maximum events/item and support threshold

Threshold	Clasp-CPM	CPM	DCM	FACE
DB 500 ; seq_size = 10 ; dic_size = 5 ; max_items/event = 3				
0.4	70,103	70,103	N/A	74,531
0.5	61,403	61,403	N/A	65,312
0.6	8156	8156	N/A	9703
0.7	7180	7180	8336	8854
0.8	471	471	514	630
0.9	380	380	483	520
DB 500 ; seq_size = 10 ; dic_size = 25 ; max_items/event = 7				
0.4	86,723	86,723	N/A	89,641
0.5	81,641	81,641	N/A	85,644
0.6	71,187	71,187	N/A	74,290
0.7	11,280	11,280	13,002	13,383
0.8	6521	6521	6701	6824
0.9	4236	4236	4812	5148
DB 500 ; seq_size = 15 ; dic_size = 30 ; max_items/event = 7				
0.4	135,681	135,681	N/A	140,023
0.5	126,550	126,550	N/A	127,670
0.6	27,930	27,930	N/A	28,354
0.7	20,998	20,998	22,601	21,564
0.8	930	930	941	950
0.9	784	784	846	897

icles. On the other hand, increasing the dictionary size and the maximum number of items per event leads to a small increase of the number of chronicles. This behaviour is explained by the fact that small sequences generate fewer closed patterns. A larger sequence size induces more frequent closed sequential patterns, which increases the number of generated frequent chronicles. This experiment result supports our choice to use an attribute selection method in the preprocessing step of our approach. Indeed, the more attributes (measures) we consider, the more our sequences are long. That's why we consider only relevant attributes, avoiding a huge number of "irrelevant" chronicles that would make our approach more costly, without a significant impact on the prediction step.

The results shown in Tables 7 and 8 confirm the effect of the sequence sizes on the performance of the tested algorithms. Indeed, this parameter increases the number of mined patterns which increases time execution and memory consumption to compute the temporal constraints between the different pairs of events in each chronicle. We can see that varying parameters DB, dic_size and $max\_items/event$ do not change considerably the performance of the algorithms, in opposite to the variation of seq_size. Another main observation, Clasp-CPM outperforms all its competitors in term of time execution, especially CPM and FACE (where comparison is fair in term of resulted chronicles). About the memory consumption, Clasp-CPM uses clearly less memory especially when the support threshold exceeds 0.7. We note that FACE consumes more memory than the other algorithms since it generates more chronicles than the other approaches. We note also that used memory increases lightly each time the frequency threshold decreases for all approaches. This is explained by the fact that a lower frequency threshold generates more frequent sequences and thus all algorithms need more memory space to store them.

The performance evaluation was also done on the SECOM data set. The interpretations done on the synthetic data sets are confirmed with our real data set as shown in Figs. 6 and 7.

Tables 9 and 10 show the number of generated chronicles with respect to the variation of the support threshold. As remarked with the synthetic data experiments, the number of chronicles extracted by FACE is huge. This is explained by the kind of patterns mined by FACE that considers all frequent ones while Clasp-CPM and CPM consider only the close ferquent chronicles.

From Table 10, we notice that CPM, Clasp-CPM and FACE generate chronicles with the same maximum size. Clasp-CPM outperforms FACE in the sense that it decreases the number of extracted chronicles as well as the execution time, but it generates the same larger patterns, also extracted by the FACE algorithm.

Choosing a high frequency threshold may be interesting in the sense that it extracts the patterns that have a higher

Table 7 Time consumption of Clasp-CPM, CPM, DCM and FACE w.r.t data set size, sequence's size, dictionary size and threshold

Threshold	Time (s)			
	DB 1000 ; seq_size 15 ; dic_size 20 ; max_items/event = 10			
	Clasp-CPM	CPM	DCM	FACE
0.4	52.325	0.24978×10^5	N/A	0.25645×10^5
0.5	10.876	0.228×10^5	N/A	0.2361×10^5
0.6	3.298	0.185×10^5	N/A	0.22284×10^5
0.7	1.388	0.16×10^5	10.89	0.1836×10^5
0.8	0.575	0.4×10^4	2.154	0.432×10^4
0.9	0.194	0.3516×10^4	1.36	0.3684×10^4
	DB 5000 ; seq_size20 ; dic_size 40 ; max_items/event = 20			
	Clasp-CPM	CPM	DCM	FACE
0.4	91.421	0.30456×10^5	N/A	0.30888×10^5
0.5	16.191	0.29268×10^5	N/A	0.30096×10^5
0.6	5.294	0.21475×10^5	N/A	0.2264×10^5
0.7	2.3	0.19348×10^5	18.297	0.21636×10^5
0.8	0.94	0.41032×10^4	7.486	0.5113×10^4
0.9	0.297	0.41068×10^4	5.31	0.43×10^4
	DB 5000 ; seq_size 30 ; dic_size 40 ; max_items/event = 20			
	Clasp-CPM	CPM	DCM	FACE
0.4	194.4	0.32154×10^5	N/A	0.3316×10^5
0.5	42.567	0.30741×10^5	N/A	0.31145×10^5
0.6	15.478	0.24156×10^5	N/A	0.2548×10^5
0.7	3.567	0.21735×10^5	35.64	0.224×10^5
0.8	2.731	0.588×10^4	16.751	0.67×10^4
0.9	0.935	0.496×10^4	8.241	0.549×10^4

relevance in the data set, but on the other hand it extracts a smaller number of chronicles with a small size, which does not help to evaluate the performance of our approach. In addition, in predictive maintenance, we need to have a sufficient number of chronicles with an interesting size to be able to efficiently predict the failures caused by the occurrence of a long sequence of events.

Evaluation of the prediction phase

To evaluate the quality of prediction, we used four measures. The first is to compute the *True Positive Rate* of the extracted chronicles. It is used to measure the number of positive sequences that are correctly classified, i.e., the sequences for which there is at least one chronicle allowing to predict the appearance of the failure without taking into consideration the temporal constraints. Indeed, for each sequence, if its events are described by the chronicle, we have correctly classified the sequence's failure and the chronicle could have predicted what are the events which caused this breakdown. Otherwise, if there is no chronicle that could describe the failure for a given sequence, therefore there is no prediction of failure so the sequence was misclassified. The idea is to bring the approach to a classification problem to apply the cross-validation method (Stone 1974).

For each value of fq_{min}, the chronicles are extracted from the training sequences. Then, for the test set, we check for each sequence, its membership in at least one chronicle among those extracted. The number of sequences validated by the chronicles is computed to estimate its percentage with respect to the sequence set. This procedure is repeated 10 times to validate all the sequences of the data set. This is the same principle used to compute the recall rate (Davis and Goadrich 2006), which is defined by the number of relevant instances found in relation to the number of relevant instances in the data set.

The True Positive Rate is computed according to this formula:

$$\frac{TP}{TP + FN} \tag{1}$$

Table 8 Memory consumption of Clasp-CPM, CPM, DCM and FACE w.r.t data set size, sequence's size, dictionary size and threshold

Threshold	Memory usage (GB)			
	DB 1000 ; seq_size 15 ; dic_size 20 ; max_items/event = 10			
	Clasp-CPM	CPM	DCM	FACE
0.4	1.102	1.34	N/A	1.41
0.5	0.6	0.7	N/A	0.81
0.6	0.23	0.64	N/A	0.72
0.7	0.06	0.4	0.16	0.68
0.8	0.026	0.05	0.053	0.07
0.9	0.0055	0.03	0.0071	0.05
	DB 5000 ; seq_size 20 ; dic_size 40 ; max_items/event = 20			
	Clasp-CPM	CPM	DCM	FACE
0.4	2.24	2.67	N/A	2.89
0.5	0.62	0.8	N/A	0.95
0.6	0.45	0.72	N/A	0.83
0.7	0.068	0.65	0.086	0.78
0.8	0.041	0.07	0.0541	0.087
0.9	0.012	0.07	0.046	0.084
	DB 5000 ; seq_size 30 ; dic_size 40 ; max_items/event = 20			
	Clasp-CPM	CPM	DCM	FACE
0.4	2.86	3.01	N/A	3.24
0.5	1.8	2.1	N/A	2.56
0.6	1.26	1.54	N/A	2.04
0.7	0.16	0.89	0.23	0.93
0.8	0.11	0.36	0.184	0.38
0.9	0.052	0.31	0.078	0.35

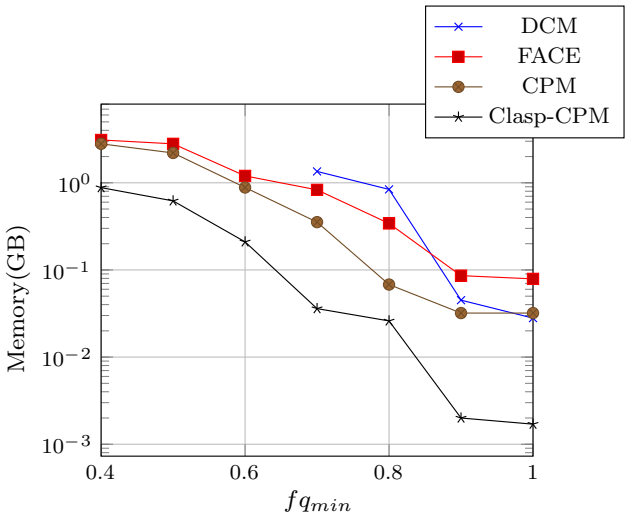


Fig. 6 Memory consumption of Clasp-CPM, CPM, DCM and FACE

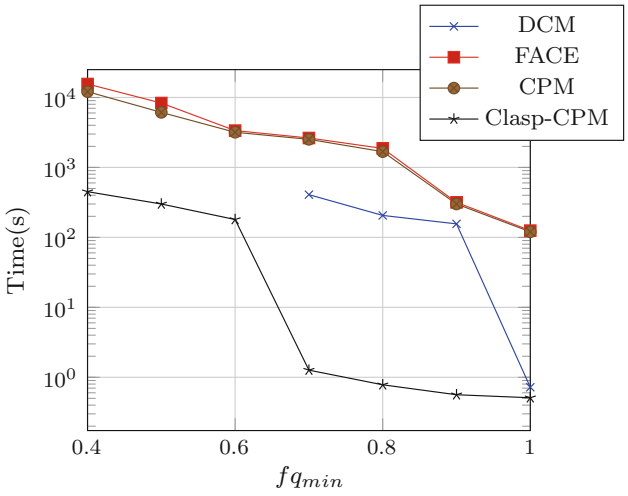


Fig. 7 Execution Time of Clasp-CPM, CPM, DCM and FACE

where TP (the true positive results) is the number of validated sequences, for which we found at least one chronicle that could have predicts the failure, and FN (the false negative results) is the number of sequences for which no chronicle could predict the failure.

Indeed, if there is no chronicle that describe a sequence S, then we cannot classify the sequence as a "failure", and

Table 9 Comparison of the three algorithms according to the number of generated chronicles

min_{sup}	Clasp-CPM	CPM (Sellami et al. 2018)	DCM (Dauxais et al. 2015)	FACE (Dousson and Duong 1999)
0.4	3429	3429	N/A	400,147
0.5	911	911	N/A	2605
0.6	206	206	N/A	348
0.7	81	81	154	94
0.8	37	37	68	41
0.9	11	11	31	11
1.0	3	3	13	3

Table 10 Maximum size of chronicles according to fq_{min}

min_{sup}	Clasp-CPM	CPM (Sellami et al. 2018)	DCM (Dauxais et al. 2015)	FACE (Dousson and Duong 1999)
0.4	6	6	N/A	6
0.5	4	4	N/A	4
0.6	4	4	N/A	4
0.7	3	3	16	3
0.8	2	2	13	2
0.9	2	2	12	2
1.0	2	2	4	2

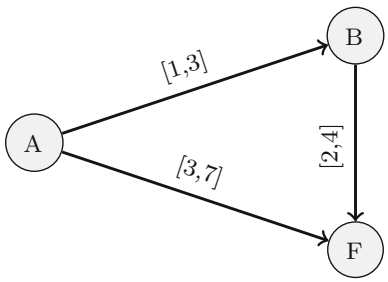


Fig. 8 Example of a chronicle

we state the "normal" case. However, all the sequences of our data set lead to a machine failure, that's why we consider such a sequence classified as false negative.

Example 6 Let's take the chronicle shown in Fig. 8 and the four sequences depicted in Table 11.

In this example, the first three sequences are described by the given chronicle, since according to this chronicle an event A followed by a B causes a failure, whereas this is not the case for the fourth sequence since the event D is not described by the chronicle. So the true positive rate for this example is: $\frac{3}{3+1} = 75\%$.

The second measure we used evaluates the precision of the results with consideration of the failure time, i.e., it estimates the percentage of sequences for which time constraints are extracted correctly. Indeed for each sequence, if the moment predicted by the extracted chronicles is outside the failure

Table 11 Sequences' data set

Seq. id	Events
1	(A,1), (B,2), (F,4)
2	(A,0), (B,3), (F,7)
3	(A,2), (B,6), (F,17)
4	(A,3), (D,5), (F,5)

appearance interval in the sequence, so the chronicle could not extract the temporal constraints of this failure, and the failure is classified as false positive. These interpretations allow us to apply the following precision formula:

$$\frac{TP}{TP + FP} \tag{2}$$

Example 7 With the same data from the Example 6, the first two sequences are classified as TP since the temporal constraints between events belong to those extracted by the chronicle, whereas the third sequence is classified as FP since the events are described by the chronicle but the temporal constraints are not checked. So the precision rate is: $\frac{2}{2+1} = 66.66\%$.

These two previous measurements allow to compute the F-measure as follows:

$$\frac{2TP}{2TP + FP + FN} \tag{3}$$

Table 12 True positive rate of FADE and FACE approaches

fq_{min}	FADE	FACE
1	83.63% ± 6.43%	**84.6**% ± 6.1%
0.9	85.45% ± 4.98%	**87.81**% ± 3.48%
0.8	87.27% ± 7.50%	**89.22**% ± 5.25%
0.7	89.09% ± 6.68%	**89.89**% ± 6.3%
0.6	90.82% ± 7.93%	**90.90**% ± 7.2%
0.5	90.82% ± 7.93%	**90.90**% ± 7.2%
0.4	90.94 ± 5.12%	**91.2** ± 6.04%

Table 13 Precision of FADE and FACE approaches

fq_{min}	FADE	FACE
1	**84.58**% ± 6.55%	78.53% ± 5.41%
0.9	**84.62**% ± 6.16%	79.49% ± 6.18%
0.8	**86.22**% ± 6.55%	80.15% ± 7.23%
0.7	**86.83**% ± 6.43%	83.49% ± 6.25%
0.6	**87.49**% ± 5.26%	84.5% ± 6.31%
0.5	**88.71**% ± 4.41%	84.9% ± 5.46%
0.4	**89.61**% ± 3.71%	86.52 ± 4.12%

Table 14 F-measure of FADE and FACE approaches

fq_{min}	FADE	FACE
1	**84.1**% ± 6.48%	81.45% ± 5.73%
0.9	**85.03**% ± 5.5%	83.44% ± 4.45%
0.8	**86.74**% ± 6.9%	84.44% ± 6.08%
0.7	**87.95**% ± 6.55%	86.57% ± 6.27%
0.6	**89.12**% ± 6.32%	87.58% ± 6.73%
0.5	**89.75**% ± 5.66%	87.79% ± 6.21%
0.4	**90.27**% ± 4.3%	88.8 ± 4.9%

Taking always the same example, the F-measure value will be equal to: $\frac{2 \times 2}{2 \times 2 + 1 + 1} = 66.6\%$.

The literature is plentiful of other measures that use the number of true negatives such as specificity measure (also called the true negative rate). It measures the proportion of negatives that are correctly identified. In our application case, we can not apply them because a sequence is classified as a true negative if the chronicles are able to predict normal cases for a sequence where there is no failure. This class can not be used in our case since all the available sequences lead to failures. Furthermore, a chronicle is made to predict a failure and not the normal operation of a machine.

Tables 12, 13 and 14 show the results obtained for the three aforementioned measures. The computed values vary between 80 and 90% which are encouraging results for our approach. We also note that the values decrease by increasing the minimum frequency threshold. Indeed, increasing the frequency threshold will produce a set of the most relevant patterns, but on the other hand, it affects our measures since the number of chronicles will decrease, so the test sequences will have less chance of being validated by extracted chronicles. Therefore, we can use small values of the frequency threshold. It will produce a best quality of prediction. However, as shown in the previous section, small values of the frequency threshold will decrease the performance of the system in term of running time and memory consumption. That's why, we should look for a trade-off between performance and quality by decreasing the frequency threshold until our mining algorithm finds scaling difficulties.

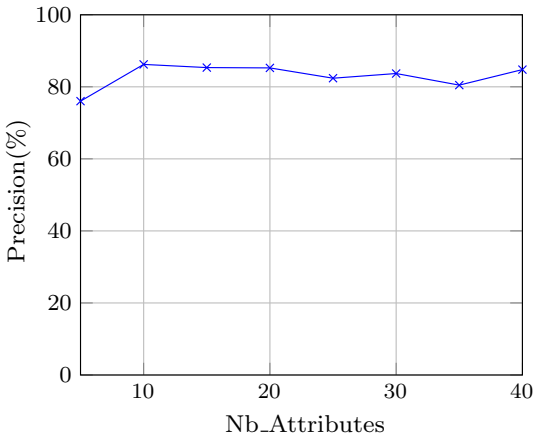


Fig. 9 Precision of results of Clasp-CPM according to the number of selected attributes for $fq_{min} = 0.8$

These same experiments are performed on the FACE algorithm. Note that the TPR values are slightly larger than those found by the FADE algorithm. This is due to the fact that FACE is based on Apriori so it extracts more frequent chronicles. But on the other hand, the precision will decrease, since the time constraints computed by FACE are as close as possible to 0, so some chronicles will not be extracted. Unlike our approach, where Clasp-CPM generates the largest time constraints, and therefore the test sequences are more likely to be validated, which increases the precision of the results. We performed another experiment to evaluate the impact of the feature selection processing on the precision of the results. As shown in Fig. 9, the number of attributes that lead to an optimum value of the precision is 10 (86.22%). This fact argue our choice to perform such a pre-processing method before applying Clasp-CPM. With a such small number of attributes, we can produce a "precise" classifier and ensure at the same time a very acceptable performance of our approach, which is affected by the number of attributes we consider.

Conclusion

This article extends an existing work (Sellami et al. 2018), where we dealt with the problem of frequent chronicle mining for predictive maintenance. We have discussed the techniques of frequent chronicle mining whose purpose is to extract from frequent sequences, the events that trigger machine failures. This process considers the time constraints between the different events, which allow the prediction of the failure event. In this article we improved our approach. We applied the Clasp principle (Antonio et al. 2013) to extract closed patterns from which we mine the frequent chronicles using a new algorithm we called *Clasp-CPM*. We implemented it using a multi-thread environment. These improvements have considerably impacted the performance of our method as shown in the experiments. We also improved the evaluation of our approach especially by using configurable synthetic data sets. The main finding is that performance is greatly impacted by sequences length in the mined data set. When dealing with the real data set, whose attributes number is huge, we applied an attribute selection method to reduce the number of measures to consider. This pre-processing task has reduced the length of sequences, which impacts the performance of the mining phase, without affecting the quality of prediction as shown in the experiments.

In future work, we will focus on the probability of occurrence of a failure in given temporal constraints using specific techniques of data mining, i.e. uncertain data, which allows to evaluate the trust of the data and subsequently use uncertain ones along with the techniques of chronicle mining. We are investigating extending deep learning algorithm like LSTM to predict failure and time to failure as well.

Acknowledgements This work has received funding from INTERREG Upper Rhine (European Regional Development Fund) and the Ministries for Research of Baden- Wrttemberg, Rheinland-Pfalz (Germany) and from the Grand Est French Region in the framework of the Science Offensive Upper Rhine HALFBACK project.

Appendix A Proofs

Proof of Lemma 1 Let us prove $\text{supp}(s') \leq \text{supp}(s)$ and $\text{supp}(s') \geq \text{supp}(s)$:

- Since s is contained within s', any sequence containing s' also contains s. Thus, $\text{supp}(s') \leq \text{supp}(s)$.
- We are to prove that concatenation with ω as a suffix does not reduce the support of a given sequence in FS_ω, i.e. $\text{supp}(s') \geq \text{supp}(s)$.
  Let $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{\text{supp}(s)}\}$ be the set of sequences of D_ω containing s. Then, we can write any γ_i as follows:

  $$\gamma_k = \alpha_1 \Diamond_i \langle s_1 \rangle \Diamond_i \alpha_2 \Diamond_i \cdots \Diamond_i \langle s_p \rangle \Diamond_i \alpha_{p+1} \Diamond_i \omega$$

  where $\alpha_i, i \in [\![1, p+1]\!]$ are the remaining sequences needed to build γ_k.
  One can easily notice that s' is contained within γ_k. Thus, $\forall k \in [\![1, \text{supp}(s)]\!]$, $s' \subseteq \gamma_k$. Ultimately, $\text{supp}(s') \geq \text{supp}(s)$.

□

Proof of proposition 1 Let us prove $\mathcal{CS}_\omega \subseteq \{s \Diamond_s \omega | s \in \mathcal{CS}\}$ and $\{s \Diamond_s \omega | s \in \mathcal{CS}\} \subseteq \mathcal{CS}_\omega$.

- Let us use a reduction ad absurd argument.
  Assume $\exists s_\omega \in \mathcal{CS}_\omega, \nexists s \in \mathcal{CS}, s_\omega = s \Diamond \omega$. Let $s_\omega = \langle s_{\omega,1}, s_{\omega,2}, \ldots, s_{\omega,p} \rangle$. Let us consider $\gamma = \{\gamma_i\}_{i=1}^{\text{supp}(s_\omega)}$ the set of sequences of D_ω containing s_ω. Then, as $\forall i \in [\![1, \text{supp}(s_\omega)]\!]$, $\gamma_i \in D_\omega$, we can write γ_i as follows:

  $$\gamma_i = \alpha_1 \Diamond \langle s_1 \rangle \Diamond \alpha_2 \Diamond \ldots \Diamond \langle s_p \rangle \Diamond \alpha_{p+1} \Diamond \omega$$

  where $\alpha_i, i \in [\![1, p+1]\!]$ are the remaining sequences needed to build γ_i.
  Then, let $s'_\omega = s_\omega \Diamond \omega$. This sequence contains s_ω, i.e. s'_ω is a super-sequence of s_ω. Moreover, as per Lemma 1, $\text{supp}(s_\omega) = \text{supp}(s_\omega \Diamond \omega)$. Thus, s_ω is not closed, so $s_\omega \notin \mathcal{CS}_\omega$, which is absurd. Therefore, $\forall s_\omega \in \mathcal{CS}_\omega, \exists s \in \mathcal{CS}, s_\omega = s \Diamond \omega$.
- Let us consider $s_\omega = s \Diamond \omega$ with $s \in \mathcal{CS}$. Let us show that $s_\omega \in \mathcal{CS}_\omega$.
  - $s \in \mathcal{CS} \Rightarrow s \in \mathcal{FS}$, so from lemma 1, $\text{supp}(s_\omega) = \text{supp}(s)$. Moreover, let $\{\gamma_i\}$ be the sequences of D where s occurs, then s_ω occurs in every sequence of $\{\gamma_i \Diamond \omega\} \subseteq D_\omega$. So $s_\omega \in \mathcal{FS}_\omega$.
  - Let us use a reductio ad absurdum argument. Let us assume $\exists \beta_\omega \in \mathcal{FS}_\omega, \text{supp}(s_\omega) = \text{supp}(\beta_\omega) \wedge s_\omega \subseteq \beta_\omega$. Let β_ω be of the form $\beta \Diamond_s \omega$, so $\beta \in \mathcal{FS}$, without loss of generality. Lemma 1 gives us that $\text{supp}(s_\omega) = \text{supp}(s)$ and $\text{supp}(\beta_\omega) = \text{supp}(\beta)$, so $\text{supp}(s) = \text{supp}(\beta)$. In addition, we have $s \subseteq \beta$. Therefore, $s \notin \mathcal{CS}$, which is absurd. Hence, there exist no β_ω in $\mathcal{FS}_\omega$ with same support as and which contains s_ω. □

Appendix B Suffix database with i-concatenation

In Table 15, we consider a version of the suffix database presented in Table 3 but with i-concatenation.

Let us recall the motivation to build a suffix database is to always extract the last event of a sequence (corresponding to a failure in our application), so that we do not have to re-check the database to extract such event or do case-based approaches for each sequence. Moreover, we would like to

Table 15 Sample suffix database with i-concatenation: P is the failure event, s-append at the end of each sequence

Seq. id	Events
1	({A, C, E}, 1), ({B, D, F}, 3), ({A, C, F}, 4), ({A, C, E}, 8), ({B, D, E, P}, 10)
2	({A, C, F}, 2), ({A, D, F}, 6), ({A, C, E}, 7), ({B, D, F, P}, 9)
3	({A, D, F}, 0), ({A, C, F}, 3), ({A, C, F}, 7), ({B, D,F}, 8), ({B, C, E, P}, 12)
4	({A, C, F}, 1), ({A, D, F}, 3), ({A, C, E}, 4), ({B, D,F}, 7), ({A, D, E, P}, 9)
5	({A, D, F}, 2), ({A, C, F}, 4), ({B, D, E, P}, 7)

keep the same number of closed frequent sequences, as this number is an important part of their discrimination role.

The problem with i-concatenation is that the resulting suffix database can produce a closed frequent sequences set with P missing from some patterns, and can eventually produce more sequences. Using the i-concatenation suffix database, and still considering a relative threshold of 0.8, we can see that $\langle A, A, A, B\rangle$ is closed (support of 0.8). It is contained within $\langle A, A, A, \{B, P\}\rangle$ (support of 0.6) and $\langle A, A, A, \{A, P\}\rangle$ (support of 0.2), but support of both of these sequences fall short to the one of $\langle A, A, A, B\rangle$. Thus, equality of Proposition 1 does not hold, and one cannot ensure we will have the same number of closed frequent sequences, nor that the failure event will be contained within every extracted pattern.

References

Agrawal, R., Imieliski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD international conference on management of data, Washington, DC, USA—May 25–28* (pp. 207–216).

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the international conference on data engineering, Taipei, Taiwan, March 06–10* (pp. 3–14).

Antonio, G., Manuel, C., Roque, M., & Bart, G. (2013). Clasp: An efficient algorithm for mining frequent closed sequences. In *Proceedings of the Pacific-Asia conference on advances in knowledge discovery and data mining, Gold Coast, Australia, April 14–17* (pp. 50–61).

Borgelt, C. (2012). Frequent itemset mining. *WIREs Data Mining Knowledge Discovery*, *2*, 437–456. https://doi.org/10.1002/widm.1074.

Carrault, G., Cordier, M. O., Quiniou, R., & Wang, F. (2003). Temporal abstraction and inductive logic programming for arrhythmia recognition from electrocardiograms. *Artificial Intelligence in Medicine*, *28*, 231–263.

Cho, S., May, G., Tourkogiorgis, I., Perez, R., Lazaro, O., de la Maza, B., et al. (2018). A hybrid machine learning approach for predictive maintenance in smart factories of the future. In I. Moon, G. M. Lee, J. Park, D. Kiritsis, & G. von Cieminski (Eds.), *Advances in production management systems. Smart manufacturing for industry 4.0* (pp. 311–317). Cham: Springer.

Cram, D., Mathern, B., & Mille, A. (2011). A complete chronicle discovery approach: Application to activity analysis. *Expert Systems*, *29*, 321–346.

D'Addona, D. M., Ullah, A. M. M. S., & Matarazzo, D. (2017). Tool-wear prediction and pattern-recognition using artificial neural network and DNA-based computing. *Journal of Intelligent Manufacturing*, *28*(6), 1285–1301.

Dauxais, Y., Gross-Amblard, D., Guyet, T., & Happe, A. (2015). Chronicles mining in a database of drugs exposures. In: *Proceedings of the ECML doctoral consortium, Porto, Portugal, September 7–11*.

Dauxais, Y., Guyet, T., Gross-Amblard, D., & Happe, A. (2017). Discriminant chronicles mining: Application to care pathways analytics. In: *Proceedings of the conference on artificial intelligence in medicine, Vienna, Austria, June 21–24*.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the international conference on machine learning, Pittsburgh, Pennsylvania, USA—June 25–29* (pp. 233–240).

Dousson, C., & Duong, T. V. (1999). Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *Proceedings of the international joint conference on artificial intelligence, San Francisco, CA, USA, July 31–August 06* (pp. 620–626).

Dousson, C., Gaborit, P., & Ghallab, M. (1993). Situation recognition: Representation and algorithms. In *Proceedings of the international joint conference on artificial intelligence. Chambry, France, August 28–September 3* (pp. 166–174).

Fournier-Viger, P., Gueniche, T., & Tseng, V. S. (2012). Using partially-ordered sequential rules to generate more accurate sequence prediction. In *Proceedings of the international conference on advanced data mining and applications, Nanjing, China, December 15–18* (pp. 431–442).

Fradkin, D., & Mörchen, F. (2015). Mining sequential patterns for classification. *Knowledge and Information Systems*, *45*, 731–749.

Goethals, B. (2003). *Survey on frequent pattern mining*. Helsinki: University of Helsinki.

Hashemian, H. M., & Bean, W. C. (2011). State-of-the-art predictive maintenance techniques. *IEEE Transactions on Instrumentation and Measurement*, *60*(10), 3480–3492.

Huang, Z., Lu, X., & Duan, H. (2012). On mining clinical pathway patterns from medical behaviors. *Artificial Intelligence in Medicine*, *56*, 35–50.

Lasi, H., Fettke, P., Kemper, H. G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, *6*, 239–242.

Laxman, S., & Sastry, P. S. (2006). A survey of temporal data mining. *Sadhana*, *31*(2), 173–198.

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. In *ESANN*.

Mannila, H., Toivonen, H., & Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, *1*, 259–289.

Masseglia, F., Teisseire, M., & Poncelet, P. (2005). Sequential pattern mining: A survey on issues and approaches. In *Encyclopedia of data warehousing and mining* (pp. 3–29).

McCann, M., Li, Y., Maguire, L., & Johnston, A. (2008). Causality challenge: Benchmarking relevant signal components for effective monitoring and process control. In *Proceedings of the international*

conference on causality: objectives and assessment, Whistler, Canada (pp. 277–288).
Mobley, R. K. (1990). *An introduction to predictive maintenance*. Amsterdam: Elsevier Science.
Oztemel, E., & Gursev, S. (2018). Literature review of industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-018-1433-8.
Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, Mc. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the international conference on data engineering—Heidelberg, Germany, April 2–6* (pp. 215–224).
Rivera Torres, P. J., Serrano Mercado, E. I., Llanes Santiago, O., & Anido Rifón, L. (2018). Modeling preventive maintenance of manufacturing processes with probabilistic Boolean networks with interventions. *Journal of Intelligent Manufacturing*, *29*(8), 1941–1952. https://doi.org/10.1007/s10845-016-1226-x.
Sellami, C., Samet, A., & Bach Tobji, MA. (2018). Frequent chronicle mining: Application on predictive maintenance. In *Proceedings of the IEEE international conference on machine learning and applications, Orlando, Florida, USA, December 17–20*.
Srikant, R., & Agrawal, R. (1995). Mining sequential patterns. In *Proceedings of the international conference on data engineering, Taipei, Taiwan, March 6–10* (pp. 3–14).
Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the international conference on extending database technology, Avignon, France, March 25–29* (pp. 3–17).
Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society Series B (Methodological)*, *36*, 111–147.
Vautier, A., Cordier, MO., & Quiniou, R. (2005). An inductive database for mining temporal patterns in event sequences. In *Proceedings of the international joint conference on artificial intelligence, Edinburgh, Scotland, July 30–August 05* (pp. 1640–1641).
Xia, F., Yang, L. T., Wang, L., & Vinel, A. (2012). Internet of things. *International Journal of Communication Systems*, *25*, 1101–1102.
Yan, X., Han, J., & Afshar, R. (2003), Clospan: Mining closed sequential patterns in large datasets. In *Proceedings of the SIAM international conference on data mining, San Francisco, CA, USA, May 1–3* (pp. 166–177).
Yin, J., Zheng, Z., & Cao, L. (2012). Uspan: An efficient algorithm for mining high utility sequential patterns. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining, Beijing, China, August 12–16* (pp. 660–668).
Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, *42*, 31–60.
Zerin, S. F., & Jeong, B. S. (2011). A fast contiguous sequential pattern mining technique in DNA data sequences using position information. *IETE Technical Review*, *28*, 511–519.
Zhao, Q., & Bhowmick, SS. (2003). Sequential pattern mining: A survey. In *Communications of The Ais—CAIS*.