Transactions on Machine Learning
and Data Mining
Vol. 5, No. 2 (2012) 87-116
©2012, ibai-publishing,
ISSN:1865-6781,
ISBN: 978-3-942952-12-5

ibai Publishing
www.ibai-publishing.org

Forecasting Failures in a Data Stream Context
Application to Vacuum Pumping System Prognosis

Florent Martin[2], Nicolas Méger[1], Sylvie Galichet[1], and Nicolas Bécourt[2]

[1] LISTIC Laboratory, University of Savoie
B.P. 80439, 74944 Annecy-Le-Vieux, France
{nicolas.meger|sylvie.galichet}@univ-savoie.fr
[2] ADIXEN, B.P. 2069, 74009 Annecy Cedex, France
{florent.martin|nicolas.becourt}@adixen.fr

Abstract. This paper presents a local pattern-based method for forecasting failures in a data stream context. It also details a successful application to complex vacuum pumping system prognosis. More precisely, using historical data, the behavior of a set of pumping systems is first modeled by extracting a given type of episode rules, namely the First Local Maximum episode rules (FLM-rules). Each rule comes along with its proper temporal information: its optimal temporal window width. The most reliable FLM-rules are then selected to further forecast system failures in a data stream context. A forecast time interval is supplied for each forecasted failure by merging the temporal information of FLM-rules. The results obtained for production data are very encouraging. Failures are predicted with a good temporal accuracy and precision while very few false alarms are generated. The method presented in this paper is patented and it is being deployed for a customer of the semi-conductor market.

Keywords: Data stream, Episode rules, Forecast, Prognosis, Predictive maintenance, Vacuum pumps.

1 Introduction

In the current economic environment, industries have to minimize production costs. Equipment profitability optimization is a way to meet this objective. The use of Integrated Health Management Systems (IHMS) performing fault prognosis is thus becoming a key practice. These systems allow to better manage production means by anticipating rather than undergoing failures.

Prognosis, from Latin *prognõsis*, is composed of prefix *pro* : *"before"* and *gnõsis* : *"inquiry, investigation, knowing"*. It literally means to know beforehand. Most of prognosis applications come from aerospace (e.g., [1] or [2]) or medicine (e.g, [3]). In [2], fault prognosis is defined as detecting the precursor signs of a system disfunction and estimating how much time is left before a major failure. It is also defined as "the estimation of time to failure and risk for one or more existing and future failure modes" by ISO [4]. An example of prognosis is the forecast of the remaining life time of a gearbox by monitoring the quality of its oil. This clearly differs from class prediction/classification, i.e. finding the value of a categorical variable describing an object for a given time lap by considering others variables. It also differs from *early prediction.* Early prediction indeed deals with identifying the class of an object as soon as possible (e.g. [5]) while prognosis deals with forecasting a failure and its occurrence date.

As explained in [2], most of data-driven approaches are neural network-based. Unfortunately, neural networks are limited by their inability to explain their conclusions [3]. Thus we propose to extract episode rules (e.g., [6–9]) from historical categorical data to model system behavior and to perform fault prognosis using real-time production data. In other words, in a data stream context, we aim to forecast event types, and more specifically failures, by using previously extracted episode rules. These rules are easy to interpret. For example, "$A \rightarrow B \Rightarrow C$" can be read as "if event type B occurs after event type A then event type C is likely to occur". Some recent works show that episode-based techniques can forecast event types in a data stream context. Nevertheless, they all ask end-users to set a temporal window size to extract episodes. Furthermore, either they use this temporal window to forecast event types (e.g., [10]) or forecasted event types are meant to be the next ones to occur (no occurrence date is provided) (e.g., [11]). In this paper, we rely on episode rules having optimal window widths that are automatically extracted and that can differ from one rule to another: the First Local Maximum rules (FLM-rules) [12]. The most reliable FLM-rules are then selected to forecast event types in data stream context. Once an event type is forecasted by some FLM-rules, then a forecast time interval is computed by merging their respective temporal information.

A successful application to vacuum pumping system failure prognosis is detailed in this paper. When dealing with such pumping systems, common sensing technologies (power levels, temperature, pressure and flow rates) are not sufficient to deal efficiently with failure prognosis. Therefore, we use vibratory data, which is more informative about the system status, but also more difficult to handle. Indeed, because of their complex kinematic, vacuum pumping systems may generate high vibration levels even if they are in good running conditions. It is thus difficult to prognose failures using traditional data analysis techniques (e.g., crest level or kurtosis) and to develop monitoring methods based on expert knowledge or physical models.

This paper is structured as follows: Section 2 reviews existing works in data mining applied to prognosis. Section 3 introduces our industrial application and describes the way data are selected and preprocessed. Section 4 presents the notion of FLM-rules. Section 4.3 details the process that is used to select the most reliable FLM-rules. In Section 5, we explain how to proceed to real time forecasting, which includes matching the selected FLM-rules in a data stream and merging their respective temporal information so as to provide a precise forecast time interval. Finally, experimental results are presented in Section 6 while Section 7 concludes this paper and draws perspectives.

2 Related Work

Although maintenance is a manufacturing area that could benefit a lot from data mining solutions, few applications (e.g., [13, 2]) have been identified so far. Most of data mining applications are diagnosis ones (i.e., identifying problems instead of forecasting them) [2]. Prognosis applications often meet difficulties in predicting how much time is left before failure. End-users indeed have to set the width of the temporal window width used to learn a model and forecast failures (e.g., [1, 14, 10, 11]).

Prognosis applications are mainly neural-networks based. For example, in [14], the authors propose a prognosis method that can be split into 2 sub-activities. They first forecast the evolution of a degradation index with an Adaptive Neuro Fuzzy System (ANFIS). This forecast is performed for a date given by the user. Then, using a user-defined distance threshold, they compare this index evolution to a reference one in order to check whether the observed system is about to be affected by a failure or not. This approach has only been tested on synthetic data. In [1], Letourneau and al. present an approach for forecasting aircraft component faults. They build models from historical data before using them to forecast failures. More precisely, for each component, heterogeneous data (numerical, text) originating from measurements and maintenance reports are collected. Learning datasets are then defined according to component replacement occurrences found in maintenance reports by selecting data recorded before and after replacements. Failure periods are defined by considering temporal windows (about 10% of the time scale of the dataset) ending on replacement dates. All temporal aspects are user-defined. Classifiers are finally extracted using decision tree, nearest-neighbor or naive-bayes techniques. A failure is forecasted on the fly if the current production data are classified as 'characterizing a failure period'. It is to notice that the choice of the width of the failure periods significantly impacts results as it is used both to model and forecast failures.

As further expounded in Section 3, the dataset we deal with is a large sequence of events, i.e. a long sequence of time-stamped symbols. Such a context has been identified in [6]. More precisely, in [6], a data mining application, known as the *TASA project*, is presented. It aims to extract symbolic rules describing a network alarm flow, the *episode rules*. As explained in Section 1, these rules syntactically take the temporal aspect into account. They are selected according to a frequency (or support) measure and a confidence measure (for more formal definitions, the reader is referred to [6–9]). In practice, users have to set the maximum temporal window width of episode rule occurrences so as to make extractions tractable. Episode rule occurrences whose window width is greater than this maximum window width are indeed not considered. This approach can still generate a large amount of rules and experts are required to identify interesting rules according to their knowledge. Though this approach has not been designed for prognosis, it inspired some works that actually aim to forecast failures. For example, in [10], the authors propose a method that searches for previously extracted episode rules in data streams. As soon as an episode rule premise is recognized, it is used to forecast future event types. More precisely, they propose to build and to continuously maintain a queue of events which are likely to form the premise occurrences of previously extracted episode rules. Once the premise of a rule is matched in the data stream, they compute the date at which the conclusion is meant to occur by adding the maximum window width to the occurrence date of the first symbol of the premise. This method is interesting as it avoids scanning the entire data stream backward but still, a crucial temporal information, i.e. the maximum window width, has to be set by users. Moreover, it has only been tested on synthetic data. Another episode-based technique can be found in [11]. It aims to forecast, at the current time instant n, the next event types of interest that are about to appear at the time instant $n + 1$. The order matters but no occurrence date is provided. To do so, frequent episode are first extracted using a user-defined window width. Time windows are considered if they end on one of the event type of interest. Each frequent episode is then associated with a specialized Hidden Markov Model (HMM), and a mixture of these HMMs is build for each event type of interest. Finally, the likelihoods of the current window under these mixtures serve as a basis to forecast event types of interest.

Beside asking for temporal parameters, the methods reviewed in this section all impose a same temporal window width for all possible models, for all possible patterns or rules. However, it is quite difficult to justify the use of a same temporal window width for each rule. It has thus been proposed in [12] to extract *First Local Maximum rules (FLM-rules)*. These episode rules indeed come along with their respective *optimal temporal window widths*. In [12], FLM-rules are meant to describe event sequences and not to forecast events in a data stream context. In [15], we proposed a first approach for using FLM-rules and their optimal window widths so as to forecast events. An experiment on synthetic supply chain datasets was also presented. Though the results of this experiment were encouraging, and as explained in Section 5.1, the proposed approach was not consistent with respect to the definition of FLM-rules. Furthermore, dealing with vibratory data is more demanding than dealing with inventory levels. In [16], we thus presented an improved version of this technique. We also described a successful prognosis experiment that had been run on real production datasets acquired from vacuum pumping systems. In this paper, we extend [16] by giving a full insight into the data preparation (see Section 3). We also give more details about the FLM-rule extraction/selection process (see Section 4.3) and our forecast method (see Section 5). Finally, we improve the description of our experimental results and compare our performances with the performances of the technique proposed in [10](see Section 6).

3 Application and Data Preparation

The aim of the approach described in this paper is to generate a model to forecast a major dysfunction mode of complex pumping systems (i.e. with a complex kinetic). These systems are running under really severe and unpredictable conditions. They basically transfer gas from inlet to outlet. One major mode of dysfunction of such systems is the *seizing* of pump axis. Seizings can be provoked by many causes such as heat expansion or gas condensation. It is thus difficult to establish a preventive maintenance planning. Therefore, ADIXEN initiated a predictive maintenance project. As vibratory analysis is a promising way for monitoring rotating machines [17], the vibration signals originating from 64 pumping systems of a semi-conductor manufacturer have been acquired. The assumption that must be made to further predict seizings is that collected signals are representative of the variety of degradation patterns that emerge from various use conditions.

3.1 Data Acquisition and Selection

In order to measure vibration signals, each one of the 64 pumping systems located in the basement of our customer's buildings is equipped with an accelerometer to measure the vibration speed $s(t)$. According to [18], the standard deviation of $s(t)$ over a period T can be approximated to:

$$\sigma = \sqrt{\frac{1}{T}\int_0^T s(t)^2 dt} \tag{1}$$

This corresponds to the RMS-value (Root Mean Square value) of signal $s(t)$ over period T. Using the RMS-value as a descriptive quantity is interesting as it directly relies to the power content of the vibrations. A common and convenient way to qualify the criticality of the vibrations consists in:

– filtering the vibration signal $s(t)$ using rectangular narrow band filters over central frequencies,
– computing its RMS-value $Vrms$ at the output of each filter, i.e. for each frequency band.

This way, a description of the signal spectrum and the repartition level for each frequency band are obtained. This is important as studying frequencies according to the kinematic of mechanical systems provides information about the fault location. Back to our equipment, an acquisition system collects measurements from accelerometers and directly provides the RMS-value of the vibration speed over a period of 80 seconds for 20 frequency bands. Available data cover more than 2 years for the 64 pumping systems. Pumps may undergo several subsequent failures, and after the first one, systems are damaged. If we succeed in predicting seizing, then we will not monitor damaged systems. Thus data recorded before first seizings are selected. Learning from potentially degraded systems would indeed bias models.

3.2 Data Preprocessing

In order to perform FLM-rule extraction, the RMS-value of the vibration speed of each frequency band has to be encoded into symbols. First, the default severity has to be established. In the sound and vibration analysis domain, severity degree is based on the power ratio [19, 18]:

$$R_n = \frac{Vrms_{(n,T)}}{Vrms_0} \tag{2}$$

where $Vrms_{(n,T)}$ is the n^{th} measured RMS-value of the vibration speed over period $T = 80s$ and $Vrms_0$ is the reference RMS-value of the studied pumping system in good running conditions. $Vrms_0$ is defined as the lowest computed RMS-value/standard deviation over a period P, with $P \gg T$ and such that P corresponds to a whole manufacturing process. This way, $Vrms_0$ is established using a representative distribution of vibrations. However determining $Vrms_0$ over the right period is not easy as it varies from one pumping system to another. According to our experts, $Vrms_0$ has to be computed at least 24 hours after system power-on to get a stable signal. Then, for each frequency band, each time a new measure arises, $Vrms_0$ is computed using a sliding window of size P. If the computed value of $Vrms_0$ is lower than the previous one, $Vrms_0$ is updated. For the selected data (see Section 3.1), $Vrms_0$ generally stops being updated 2 weeks after the setup date.

Any variation of $Vrms_{(n,T)}$ relates to a default. For example, a shock induces large variations of $Vrms_{(n,T)}$ relatively to reference $Vrms_0$. Thus RMS-values are retained from the maximal envelope. Then, for these values, the power ratio R_n (Equation. 2) is computed and discretized using three levels. Level one corresponds to $R_n < \alpha$ which means that $Vrms_{(n,T)}$ is less than $\alpha \times Vrms_0$. Level two corresponds to $\alpha \leq R_n < \beta$ and level three refers to $R_n \geq \beta$. Each time $Vrms_{(n,T)}$ switches from one discrete level to another one, the time spent at the previous discrete level is computed and discretized using four levels ranging from a few minutes to more than ten days. Levels are issued from experiments. Finally, a dictionary of 240 symbols is defined, each symbol being associated to three pieces of information: the frequency band (20 labels), the default severity (3 labels) and the duration at this severity level (4 labels). A specific symbol is also defined to represent seizing occurrences. In the end, 13 sequences containing about 2000 symbols along with their occurrence dates are obtained. Figure 1 illustrates the pre-processing of the signal. The first curve is a sample of one $Vrms_{(n,T)}$ acquisition. The second curve shows its maximal envelope. Finally, the bottom graphic illustrates the coding of that envelope: the two first digits correspond to the signal identification (the frequency band), the third and the fourth digits respectively represent the level and the duration. In Figure 1, symbol 1314 means: signal 13 was low (1) for few weeks (4). Symbol 1322 means: signal 13 was high (2) for few hours (2) and 1333 means that it has been very high (3) for a few days (3). In this paper, when possible and for the sake of fluidity, symbols defined in this section (e.g., 1314 or 1323) are replaced by alphabetical symbols. Figure 2 depicts our data selection and preprocessing process, from vibration speed measurements to symbolic sequences.

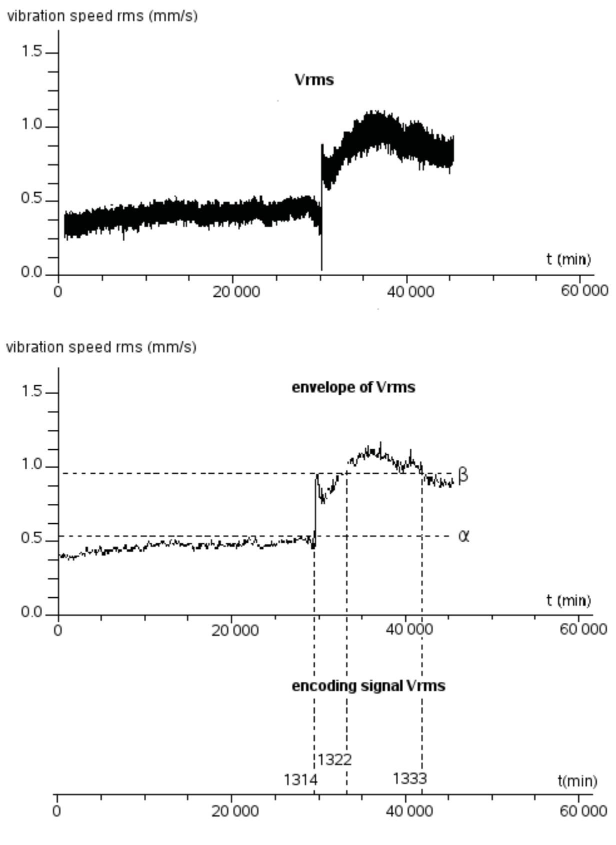


Fig. 1. Signal pre-processing.

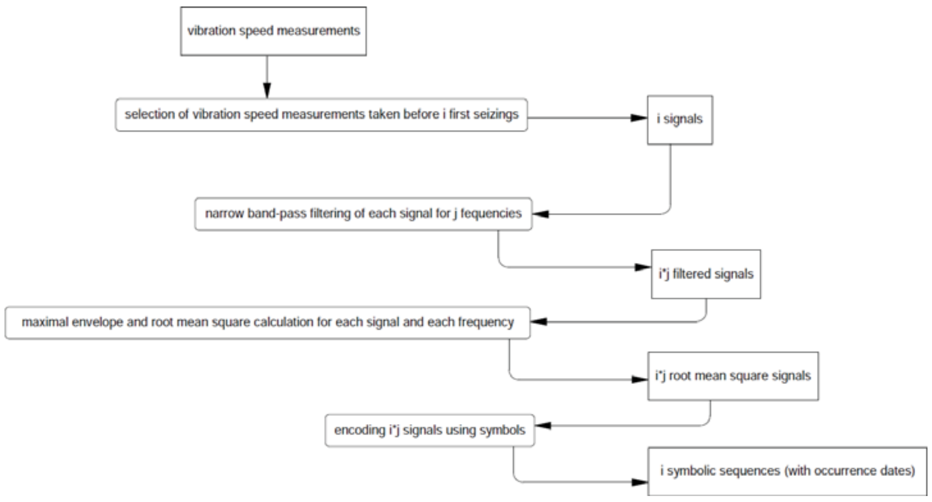

vibration speed measurements
selection of vibration speed measurements taken before i first seizings
i signals
narrow band-pass filtering of each signal for j fequencies
i*j filtered signals
maximal envelope and root mean square calculation for each signal and each frequency
i*j root mean square signals
encoding i*j signals using symbols
i symbolic sequences (with occurrence dates)

Fig. 2. From vibration speed measurements to symbolic sequences: data selection and preprocessing.

4 Learning by Means of FLM-Rules

4.1 Local Pattern Mining in Event Sequences

As presented in Section 3.2, the sequences to be mined are sequences of symbols coming along with their occurrence dates. Such symbol occurrences are termed as *events*. Two pattern extraction contexts can be distinguished. They depend on the type of the dataset: either it is a single *large event sequence* [6, 8, 7, 12, 10, 9] or it is *a collection of short event sequences* namely a *base of sequences* [20–26]. A single large sequence is composed of thousands of events whereas a base of sequences is a set of multiple short sequences (about 500 events as a maximum). As introduced in Section 3.2, we deal with sequences containing 2000 events on average. It is thus not recommended to use standard algorithms extracting patterns from a *base of sequences*. So we focus on pattern extraction from a single *large sequence* and transform our 13 pump sequences into a single large sequence D. A time gap between each subsequence Seq_i is imposed.

A maximum time gap constraint will indeed be further used in order to avoid extracting FLM-rules whose occurrences could spread over the various initial sequences.

The patterns used to mine a large event sequence are *episode rules* satisfying a frequency and a confidence constraints. Two approaches have been originally proposed: the *Winepi* [6, 8] and *Minepi* [7, 8] algorithms. Winepi extracts occurrences of episode rules in a sliding window whose width is set using a maximum time span constraint. Minepi searches for episodes rules whose occurrences are minimal, i.e. they do not contain any shorter occurrences. Furthermore, these occurrences can not exceed a maximum window width. Both approaches thus require end-users to set a maximum time span (or window width) constraint which is used to extract all possible rules. As a consequence, these algorithms must be executed each time end-users consider a new and larger maximum time span. In our application, the interesting window width is not known beforehand and can vary from one episode rule to another. We thus decided to use algorithm *WinMiner* [12]. It extracts episode rules having an *optimal window width* and satisfying a frequency, a confidence, a minimal occurrence and a maximum gap constraints. The maximum gap constraint is the maximum elapsed time between two consecutive events forming an episode occurrence. It allows to linearly constrain the span of rule occurrences w.r.t. the number of event types forming the rule instead of setting a unique maximum span for all possible rules. The *optimal window width*, if it exists, is the smallest window width corresponding to a local maximum of confidence for the rule (confidence is locally lower for smaller and larger windows). These rules are termed as *First Local Maximum Rules*: *FLM-Rules*. The following section introduces this concept in more details.

4.2 FLM-Rules

This section defines the main concepts that are necessary to understand the notion of FLM-rules as originally proposed in [12]. As already mentioned in Section 2, FLM-rules were originally designed to capture temporal dependencies between events in event sequences and not to forecast events in data streams. First, we define the dataset itself. As previously explained in Section 3.2, our application context generates a large sequence.

Definition 1 (event, event sequence). *Let E be a set of symbols, namely event types. An* event *is defined by the pair (e, t) where $e \in E$ and t is an integer giving the occurrence date of e. An event sequence is a triple $S = (s, T_s, T_e)$ where s is an* ordered sequence of events $\langle(e_1, t_1), (e_2, t_2), ..., (e_n, t_n)\rangle$ *such that $\forall i \in \{1, ..., n\}$, $e_i \in E \wedge \forall i \in \{1, ..., n-1\}$, $t_i \leq t_{i+1}$. T_s, T_e are integers that denote the starting and ending time of the event sequence.*

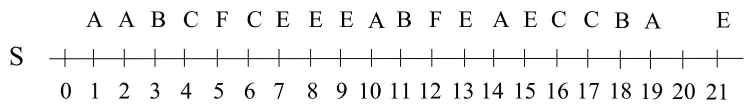


Fig. 3. Event sequence S.

Figure 3 depicts a toy example of such a sequence. FLM-rules are built upon a given kind of episodes, namely serial episodes.

Definition 2 (serial episode, prefix, suffix). *A serial episode is a tuple $\alpha = \langle e_1, e_2, ...e_k\rangle$ such that $\forall i \in \{1, ..., k\}, e_i \in E$ and there exists a total order relation between events types. The prefix of α, denoted by $prefix(\alpha)$ is the tuple $\langle e_1, e_2, ...e_{k-1}\rangle$. The suffix of α, denoted by $suffix(\alpha)$ is the singleton $\{e_k\}$.*

For the sake of simplicity, a serial episode $\alpha = \langle e_1, e_2, ...e_k \rangle$ is also denoted by $e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_k$. As we only consider serial episodes, we will now refer to *episodes* instead of referring to *serial episodes*. For example, $A \rightarrow B \rightarrow C$ is an episode stating that B occurs after A and is followed by C. The prefix of $A \rightarrow B \rightarrow C$ is $A \rightarrow B$ and its suffix is C. Let us now define how an episode is said to occur within an event sequence:

Definition 3 (occurrence). *An episode $\alpha = \langle e_1, e_2, ..., e_k \rangle$ occurs in a sequence $S = (s, T_s, T_e)$ if there exists at least one* ordered sequence of events *s' = $\langle (e_1, t_1), (e_2, t_2), ..., (e_k, t_k) \rangle$ such that s' can be obtained by removing some elements of s or $s' = s$ (which will be denoted by $s' \sqsubseteq s$ in this paper) and $\forall i \in \{1, ..., k-1\}$, $0 < t_{i+1} - t_i \leq maxgap$ with maxgap a user-defined constraint that represents the maximum time gap allowed between two consecutive events. $[t_1, t_k]$ is an occurrence of α. The set of all occurrences of α in S is denoted by $occ(\alpha, S)$.*

The *maxgap* constraint is set both to reduce the search space and to match recurrent application requirements. It has been introduced in [12]. It linearly constrains the time span of episode occurrences in function of the number of symbols that form the episode (instead of having a same maximum time span for all episodes). According to this definition, by considering the example depicted in Figure 3 and by setting *maxgap* to 4 all along this section, $occ(A \rightarrow B, S) = \{[1,3], [2,3], [10,11], [14,18]\}$. Intervals $[1,11], [2,11], [1,18], [2,18], [10,18]$ do not match the *maxgap* constraint. In order to reduce the size of such sets and to consider occurrences that do not already contain another occurrence, *minimal occurrences* are considered, as proposed in [7, 8] and in [12]:

Definition 4 (minimal occurrence). *A minimal occurrence of an episode α in a sequence S is a time interval $[t_s, t_e]$ containing α and such that there is no other occurrence $[t'_s, t'_e]$ verifying $[t'_s, t'_e] \subset [t_s, t_e]$. The set of all minimal occurrences of α in S is denoted by $mo(\alpha, S)$.*

Back to our example, the minimal occurrences of episode $A \rightarrow B$ in S are $mo(A \rightarrow B, S) = \{[2,3], [10,11], [14,18]\}$. Occurrence $[1,3]$ does not belong to $mo(A \rightarrow B, S)$ because it spreads over occurrence $[2,3]$. We here recall that this definition relies on the occurrence definition that includes a *maxgap* constraint. *Minepi* [8] can not handle this constraint as it causes incompleteness. More precisely, if minimal occurrences of episodes of size k (i.e. having k event types/symbols) are considered, then it is not possible to compute episodes of size $k+1$. Let us consider sequence S, episode $A \rightarrow B \rightarrow C$ and episode A. With $maxgap = 4$ time units, $mo(A \rightarrow B \rightarrow C, S) = \{[2,4]\}$ and $mo(A, S) = \{[1,1], [2,2], [10,10], [14,14], [19,19]\}$ can not be used to generate minimal occurrence $\{[2,10]\}$ of episode $A \rightarrow B \rightarrow C \rightarrow A$. Indeed, the minimal occurrence of $(A \rightarrow B \rightarrow C)$ in S occurs too early with respect to the ending date of $A \rightarrow B \rightarrow C \rightarrow A$. Therefore, in [12], algorithm *WinMiner* has been proposed. It extracts all minimal occurrences of the episodes satisfying a *maxgap* constraint. It fully relies on the notion of *minimal prefix occurrence*:

Definition 5 (minimal prefix occurrence). *Let $o = [t_s, t_e]$ the occurrence of episode α in a sequence S. o is a* minimal prefix occurrence *of α iff: $\forall [t_1, t_2] \in mo(prefix(\alpha), S)$, if $t_s < t_1$ then $t_e < t_2$. The set of all mpo of α in S is denoted $mpo(\alpha, S)$.*

Using this definition, $mpo(A \to B \to C, S) = \{[2,4],[2,6]\}$. It is now possible to join $[2,6]$ with $[10,10]$ to build $mpo(A \to B \to C \to A, S) = \{[2,10]\}$. The notion of minimal prefix occurrences will be further used in Section 5.1 to detect FLM-rule premises in data streams. For more details, the reader is referred to [12].

Episode rules are derived from episodes. Let α be an episode. An *episode rule* is the expression $prefix(\alpha) \Rightarrow suffix(\alpha)$. For example if $\alpha = A \to B \to C$, the episode rule built on α is $A \to B \Rightarrow C$. A more generic definition of episode rules can be found in [8]. Episode rules are characterized with two measures :

- *support*: the number of occurrences of an episode rule over the whole sequence. The support of $A \to B \Rightarrow C$, denoted by $support(A \to B \Rightarrow C)$, is equal to the number of occurrences of episode $A \to B \to C$, denoted by $support(A \to B \to C)$.
- *confidence*: the observed conditional probability of observing the conclusion of an episode rule knowing that the premiss already occurred. Confidence of $A \to B \Rightarrow C$ is thus defined as follows: $confidence(A \to B \Rightarrow C) = \frac{support(A \to B \to C)}{support(A \to B)}$.

These measures are used for selecting episode rules according to a minimum support threshold σ and a minimum confidence threshold γ. As proposed in [12], support and confidence can be defined for each *window width*, i.e. the maximum time span of episode occurrences. In the example of Figure 3, $mo(A \to B, S) = \{[2,3],[10,11],[14,18]\}$ and $mo(A \to B \to F, S) = \{[2,5],[10,12]\}$. If we consider a window width of 2 time units, then we have $Support(A \to B \Rightarrow F, S, 2) = 1$ and $Confidence(A \to B \Rightarrow F, S, 2) = \frac{1}{2}$. This means that $A \to B \Rightarrow F$ occurs once and has a confidence of 50% for a 2 time units window width. If, for a given episode rule λ, and for the shortest possible window width w,

- the support of λ is greater or equal to σ,
- the confidence c_w of λ is greater or equal to γ,
- there exists a window width $w'|w' > w$ such that confidence of λ for w' is *decreaseRate*% lower than c_w,
- there is no window width between w and w' for which confidence is higher than c_w,

then, episode rule λ is said to be a *First Local Maximum-rule* or *FLM-rule*. Parameter *decreaseRate* is user-defined and allows to select more or less pronounced local maxima of confidence with respect to window widths. The window width w corresponding to a first local maximum is termed as the *optimal window width* of FLM-rule λ. If we set *decreaseRate* to 30%, σ to 2, γ to 100% and *maxgap* to 4, then rule $r = A \to B \Rightarrow F$ (Fig. 4) is a FLM-rule which has a first local maximum of confidence for a 3 time units window. This can be interpreted as: "if I observe the premiss of r at t_0, then its conclusion must appear within t_0 and $t_s + w$". Parameter γ can be of course set to lower values. In this case, the probability of observing the conclusion between t_0 and $t_s + w$ is greater or equal to γ. For more formal definitions, the reader is referred to [12]. Using FLM-rules makes sense when dealing with temporal vibratory data. Indeed, each FLM-rule we extracted in the context of our application has an unique pronounced local maximum that differs from one rule to another. As further explained in Section 5, to forecast failures, most reliable FLM-rules ending on symbols relating to failures are retained. Then optimal window widths are used to establish future failure occurrence dates.

window width	1	2	3	4	5
$mo(A \to B \to F, S)$	$\varnothing$	$\{[10,12]\}$	$\{[2,5],$ $[10,12]\}$	$\{[2,5],$ $[10,12]\}$	$\{[2,5],$ $[10,12]\}$
$mo(A \to B, S)$	$\varnothing$	$\{[2,3],$ $[10,11]\}$	$\{[2,3],$ $[10,11]\}$	$\{[2,3],$ $[10,11],$ $[14,18]\}$	$\{[2,3],$ $[10,11],$ $[14,18]\}$
$support(A \to B \to F, S)$	0	1	2	2	2
$support(A \to B, S)$	0	2	2	3	3
$confidence(A \to B \Rightarrow F, S)$	0	$1/2 = 50\%$	$2/2 = 100\%$	$2/3 = 66\%$	$2/3 = 66\%$

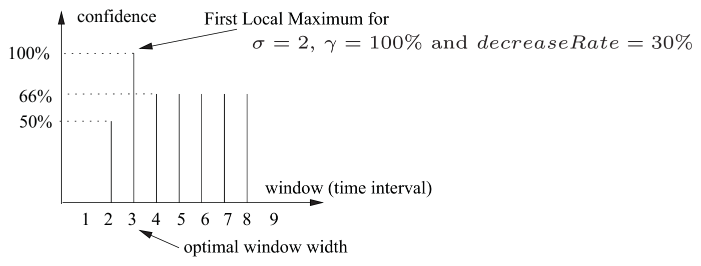


Fig. 4. Confidence and support for rule $A \to B \Rightarrow F$ in sequence S (Fig. 3), for $maxgap = 4$.

4.3 FLM-Rule Extraction and Selection

In order to forecast seizings, FLM-rules concluding on symbol 'seizing' are the only ones to be considered. The most reliable rules are then selected. As few seizing examples are available, the extraction and selection process is inspired from the well known *leave-one-out cross validation* technique. It involves removing a subsequence Seq_i of sequence D, where each subsequence relates to a single failure (see Section 4.1), and using it as a validation dataset. Remaining subsequences form the training dataset. This is repeated so that each subsequence is used once as a validation dataset. We thus alternate FLM-rule extractions (with same parameter values) to get descriptions of training datasets and validations of these descriptions by matching extracted FLM-rules over validation datasets.

A FLM-rule r is matched in a validation dataset if its conclusion occurs after each premise occurrence. If not, then, when used to forecast, rule r could trigger false alarms and it has to be rejected. Temporal constraints are of course taken into account. Let w_r be the optimal window width of r. Firstly, each premise occurrence time span must be lower than w_r. Founding a premise occurrence that does not satisfy this constraint in a validation dataset means that we would not have been able to forecast a failure using this dataset along with r and w_r. Indeed, according to the definition of FLM-rules, for each premise occurrence starting at t_ps and ending at t_pe, the conclusion should appear with a high probability in $]t_pe, t_ps + w_r]$. Rule r has thus to be rejected. Secondly, regarding the conclusion, it must appear in $]t_pe, t_pe + gapmax]$. By definition, $t_ps + w_r \leq t_pe + gapmax$. We thus allow the conclusion to occur after $t_ps + w_r$ which is permissive w.r.t. the definition of FLM-rules. In this case, using r and w_r, we would be able to forecast the failure but we would provide a too narrow forecast time interval. This is better than being unable to forecast any failure, and rule r can be retained. If the conclusion appears after $t_pe + gapmax$, then rule r was learned with a too short *gapmax* to describe and to forecast this case. A rule having more event types might be able to take this case into account. Rule r is thus rejected.

Once our extraction and selection process ends, we get a set of reliable FLM-rules that do not trigger any false alarm on validation datasets (though too narrow forecast time intervals may be provided). A same FLM-rule can be extracted at several iterations, each iteration providing a different optimal window width. In this case, its optimal window width is set to the most observed value. The set of selected FLM-rules is termed as the *FLM-base*. It will be used to prognose seizings. This extraction and selection process, though not fully detailed, was originally proposed in [16]. It is worth noting that if false alarms are permitted (which is not allowed for our application), then this process can be adapted so that a rule is rejected if and only if it triggers a number a false alarms exceeding a user-defined threshold. In that latter case, the minimum confidence threshold γ that is used to extract FLM-rules can thus be set to values lower than 100%. Figure 5 summarizes our extraction and selection process, from symbolic sequences to the FLM-base.

5 Real Time Prognosis

Real time prognosis first relies on matching the FLM-rule premises of the FLM-base in data streams. Then, for each matched premise, a time interval within conclusions/failures/seizings should occur is computed. These aspects are detailed in Section 5.1 which extends the proposition of [16] by giving, for the first time, all necessary algorithms for forecasting events. As many different premises can be matched, we may get different time intervals. We thus detail in Section

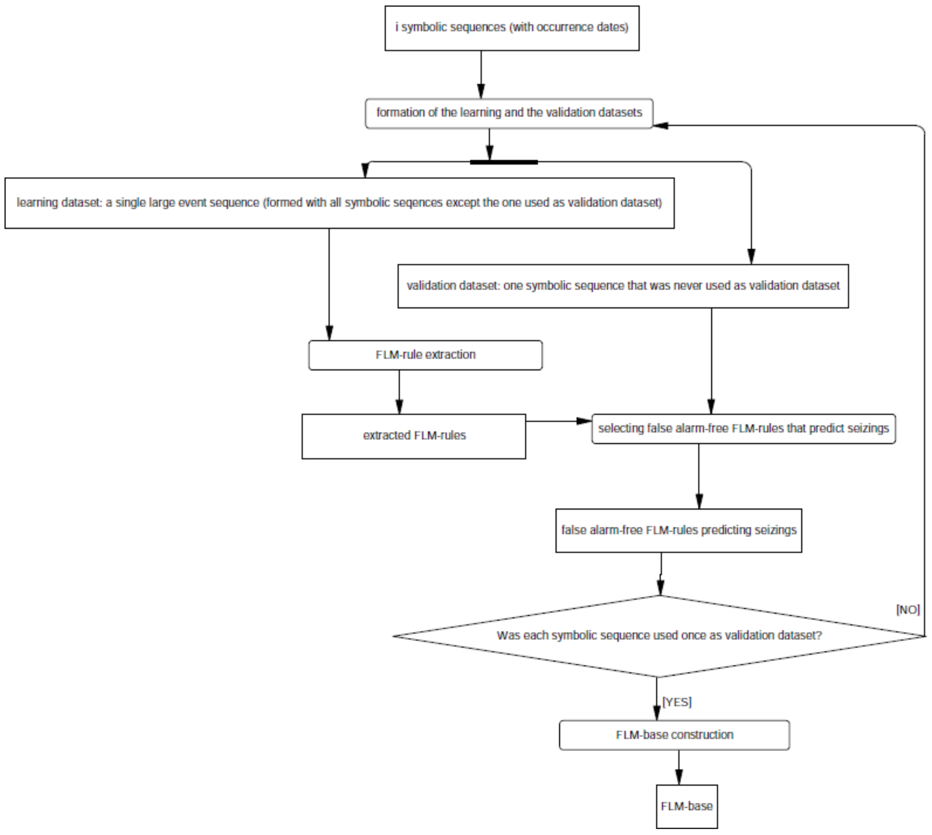

i symbolic sequences (with occurrence dates)
formation of the learning and the validation datasets
learning dataset: a single large event sequence (formed with all symbolic seqences except the one used as validation dataset)
validation dataset: one symbolic sequence that was never used as validation dataset
FLM-rule extraction
extracted FLM-rules
selecting false alarm-free FLM-rules that predict seizings
false alarm-free FLM-rules predicting seizings
Was each symbolic sequence used once as validation dataset?
[NO]
[YES]
FLM-base construction
FLM-base

Fig. 5. From symbolic sequences to the FLM-base: a leave-one-out approach.

5.2 a method, originally proposed in [16], for merging these informations and for providing a single forecast time interval, namely the *forecast window*.

5.1 Matching FLM-Rules Premises in Data Streams

In order to match the premises of the FLM-rules belonging to the FLM-base, we build a queue of event occurrences whose time span is lower than W, the largest optimal window width of the FLM-base. When events occur, they are added to the queue by setting their occurrence date to the current system date t_0. Events whose occurrence date t is lower than $t_0 - W$ are removed. We thus make sure that enough data is kept for being able to identify the premises of all the FLM-rules that form the FLM-base. From now on, a FLM-base is defined as a set of tuples $\langle premise, w_r, tc_r, size \rangle$, each tuple relating to a single FLM-rule. Element *premise* gives the premise of the FLM-rule, w_r records its optimal window width, tc_r is the latest date at which the conclusion is meant to appear (if no conclusion is forecasted then $tc_r = 0$) and *size*, the number of event types forming the rule. The FLM-base is also maintained on the fly by setting tc_r to 0 if $tc_r < t_0$. For each event occurring at time t_0, if its event type corresponds to the last event type of a premise of a FLM-rule such that $tc_r = 0$, then the latest minimal occurrence of the whole premise is searched. In other words, for a given FLM-rule, either the premise has already been matched and tc_r is still valid, or a new premise has to be searched, the latest one. Searching for the latest occurrence is a strategy that is also proposed in [10]: any previous occurrence has indeed already been used. Furthermore, in order to avoid any useless processing, we search for this latest occurrence by reading backward the queue from t_0, the date at which the premise must end. This backward search is also proposed in [10].

In this section, in order not to overwhelm readers with useless outputs, variables are meant to be passed by reference so that they can be directly modified if necessary. Algorithm $ForecastingSeizings$ (Algorithm 1) summarizes this approach. Date t_0 is first set to the current system date (line 3). If event types occur at t_0 then the whole algorithm is executed (line 4 and line 5). The queue S is maintained (line 6, see Algorithm 2) as well as the FLM-base (line 7, see Algorithm 3). The queue is here considered as a set of events representing the sequence of events recorded between $t_0 - W$ and t_0. In order to search the data stream backward and to simply adapt standard algorithms, we invert the whole queue (line 8) w.r.t. t_0, using function *invert*. It is detailed by Algorithm 4. As already mentioned, this inversion is done in order to avoid any useless processing, i.e. we intend to directly search for the latest occurrence: if we were to consider the queue without inverting it, we would have to go through each each occurrence to handle the minimality and the maximum gap constraints, before finding the last occurrence, the one that was initially sought. This strategy has been validated experimentally: seeking times are reduced by 20% in the case of our application. In our case, this speed-up factor was necessary to cope with the refresh rate of the data streams that were monitored. Then, the premises of rules whose last event type occurs at t_0 are searched for with procedure $searchPremisses$ (line 9). If premises are found, then their respective tc_r are updated. The whole FLM-base may be thus updated and, in any case, it is used by procedure $buildForescatIntervall$ (line 10) to provide a forecast window. If the forecast window is set to $]0, 0]$, then no failure/fault is forecasted; otherwise a window of the form $]t_0, tf_e]$ (with $t_0 < tf_e$) is provided. Procedures $searchPremisses$ and $buildForescatIntervall$ will be further detailed.

Searching for premise occurrences that are minimal ones satisfying the maximum gap constraint is not straightforward because of incompleteness issues. Our algorithm thus derives from algorithm $WinMiner$ [12]. As already explained in Section 4.2, $WinMiner$ uses *mpos* to remain complete while handling a maximum gap constraint. Furthermore, as we aim to avoid any useless processing by searching the queue backward for the latest minimal occurrences, we consider an inverted queue, an inverted sequence of events. This requires very few adaptations of algorithm $WinMiner$. Indeed, searching for the minimal occurrence of the premises whose last event type occurs at t_0 in event sequence S is is triv-

Algorithm 1 (ForecastingSeizings)
Input:

- *E, the set of event types that may appear in the data stream*
- *$FLM - base$, the FLM-base*
- *W, the largest optimal window width of the FLM-base*

```
1.  let $S := \emptyset$
2.  while $system\ Is\ ON$ do
3.    let $t_0 := currentSystemDate$
4.    let $O = \{(e_0, t_0) | e_0 \in E \wedge e_0\ occurs\ at\ t_0\}$
5.    if $O \neq \emptyset$
6.      $maintainQueue(W, t_0, S)$
7.      $maintainFLM - base(t_0, FLM - base)$
8.      let $S^{-1} := invert(S)$
9.      $searchPremises(S^{-1}, FLM - base, E, t_0, )$
10.     $buildForecastInterval(FLM - base, t_0)$
11.   fi
12. od
```

ially equivalent to searching for the minimal occurrence of the inverted premises whose first event occurs at time $t = 0$ in inverted event sequence S^-1. For example, in Figure 5.1, if premise $A \rightarrow B \rightarrow C$ and sequence S are considered, then minimal occurrence $[13, 18]$ has to be found. If we now consider S^-1, then the corresponding minimal occurrence $[0, 5]$ of inverted premise $C \rightarrow B \rightarrow A$ shall be found. Algorithm $WinMiner$ is thus adapted to search, in a backward manner, for the latest minimal occurrences, ending at t_0, satisfying a maximum gap constraint. This leads to Algorithm 5 and Algorithm 6. There are detailed hereafter:

We first begin by building an E/O-pair x (E/O stands for episode/occurrence) to store the *mpos* of each event type belonging to a premise of the FLM-base (line 2). To do so, we record the event type itself (line 3) and the occurrence list is obtained using function *scan* (line 4). The occurrence list given by *Occ* is a set of pairs (t_s, T) where t_s is the starting date of a set of *mpos* whose respective ending dates are stored using set T. All E/O-pairs are then collected with set L_1. They will be further used to compute premise occurrences. Basically, the inversion algorithm, Algorithm 4, and the maintenance of L_1 could be directly done in Algorithm 2. The actual presentation was adopted for the sake of clarity. Then, for each premise p that is not activated ($p.tc_r = 0$), if its last event type occurs at t_0 (line 7), then, we search for its first occurrence beginning at t_0 (line 9) with procedure $searchForFirstOccurrenceBeginingAtT_0$. It is described by Algorithm 6.

It recursively builds the *mpos* of the whole inverted premise, if it exists. At recursion level i, it calculates the *mpo* of the inverted subpremise Z composed by the first (last when considering the premise and not the inverted premise) $i + 1$ event types. To do so, the *mpos* of event type $i + 1$, Y, is selected using y (line

Algorithm 2 (maintainQueue)
Input:

- *W, the largest optimal window width of the FLM-base*
- *t_0, the date at which the algorithm is triggered*
- *O, the set of events occurring at time t_0*
- *S, a set of events representing an event sequence, i.e., the queue*

```
1. for all $(e, t) \in S$ do
2.   if $t < t_0 - W$
3.     let $S := S - \{(e, t)\}$
4.   fi
5. od
6. for all $(e_0, t_0) in O$
7.   let $S := S \cup \{(e_0, t_0)\}$
8. od
```

Algorithm 3 (maintainFLM-base)
Input:

- *t_0, the date at which the algorithm is triggered*
- *$FLM - base$, the FLM-base*

```
1. for all $p \in FLM - base$ do
2.   if $p.tc_r \le t_0$
3.     let $p.tc_r := 0$
4.   fi
5. od
```

1). Then, the new subpremise is recorded using mpo z (line 3) and its respective *mpo* occurrence list is initialized to the empty set (line 4). We now consider the earliest (starting at t_0, 0 in the inverted queue) occurrences of subpremise X that have been calculated at level i, that are stored using x (line 5) and that do not end after w_r units of time, i.e. that appear in observation window $W_r^{obs} = [0, w_r]$ (line 7). This way, we avoid building an occurrence exceeding the optimal window width. Then, for each occurrence ending date, we search for the occurrences of event type Y that appears after it while satisfying the maximum gap constraint (line 8). We also make sure that they do not appear right after an occurrence of X starting after t_0 (line 8, second part). This last check is only used at recursion level 1. Indeed, from level 2, by construction, all the occurrences of X start at 0. If needed and if occurrences of Z have been built, we then go the following recursion level (line 11-14) until either the whole premise is build (line 15-18) or there is no occurrence of Z to be able to go deeper. If the whole premise is found, then tc_r is set to $t_0 - min(L) + p.w_r$ (line 17). In other words, knowing that the shortest occurrence of the inverted premise ends at $min(L)$, we can deduce that the first element of the premise occurs, in the non-inverted event sequence, at

Algorithm 4 (invert)
Input :

- *S, a set of events representing an event sequence*
- *t_0, the date at which the algorithm is triggered*

```
1. let $S^{-1} := \emptyset$
2. for all $(e, t) \in S$ do
3.    let $t^{-1} := t_0 - t$
4.    let $S^{-1} := S^{-1} \cup \{(e, t^{-1})\}$
5. od
6. return $S^{-1}$
```

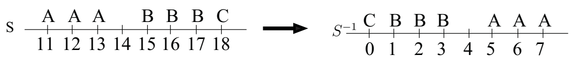


Fig. 6. Event sequence S and inverted event sequence S^-1.

$t_0 - min(L)$. We then add optimal window width w_r to get the latest date at which the conclusion is meant to occur, according to the definition of FLM-rules. An example is given in Figure 7 for rule $A \to B \to C \Rightarrow P$. In other words, the FLM-base is here updated to be further used to forecast failures (see next section): no output is thus to be reported for algorithms 5 and 6. This update relates to the forecasts associated to the rules forming the FLM-base. In [15], for each matched premise of rule r, its conclusion is forecasted at tc_r though it may appear in $]t_0, tc_r]$ by definition of FLM-rules. The prognosis approach proposed in [15] is thus not consistent with respect to the definition of FLM-rules.

5.2 Merging FLM-Rules Predictions

Let t_0 be the date at which a FLM-rule premise is matched. For each matched premise of rule r, its conclusion should occur in $]t_0, tc_r]$ with 100% confidence (if the minimum confidence is set to 100%). Let Tc be the set of all forecast dates tc_r that are active, i.e. that are greater than t_0 (those dates can be computed before and at t_0). Associated failure forecast time interval $]t_s^f, t_e^f]$, also termed as the *forecast window* W^F, is such that $t_s^f = t_0 \wedge t_e^f = min(Tc)$. By choosing the *min* operator to aggregate forecast dates, this forecast window is defined to be the earliest one. Figure 8 provides a forecast window established using rules α, β, δ that have been recognized at t_0^{n-1} and t_0^n.

From a more operative point of view, after having tried to identify rule premises in the queue, the forecast interval is calculated using Algorithm 7. It first selects the earliest tc_r (line 1). If it is different from 0, then interval $]t_0, p.tc_r]$ is provided (line 3), $]0; 0]$ otherwise (line 5, no failure is forecasted).

Algorithm 5 (searchPremises)
Input:

- *S, the set of events representing the inverted queue available at t_0*
- *$FLM - base$, the FLM-base*
- *E, the set of events that may appear in S*
- *t_0, the date at which the algorithm is triggered*

```
1.  let $L_1 := \emptyset$
2.  for all $e \in E \mid \exists p \in FLM - base$ such that
        $\exists i \in \{1, ..., p.size\} \mid p.premise[i] = e$ do
3.    let $x.Pattern := e$
4.    let $x.Occ := scan(S, e)$
5.    let $L_1 := L_1 \cup \{x\}$
6.  od
7.  for all $p \in FLM - base \mid \exists (e, t) \in S$ such that
        $t = 0 \wedge p.premise[p.size] = e \wedge p.tc_r = 0$ do
8.    let $x := e \in L_1 \mid e.Pattern = p.premise[p.size]$
9.    $searchForFirstOccurrenceBeginingAtT_0(t_0, p, x, L_1, 1, FLM - base)$
10. od
```

6 Experimental Evaluation

6.1 Rule Selection

As explained in Section 3, seizings can originate from very different causes. In addition, only 13 event sequences relating to seizings are available. Therefore, minimum support threshold σ is set to 2, which is very low. In order to extract the most confident rules, the minimum confidence threshold γ is set to 100%. Parameter *decreaseRate* is set to 30% to select pronounced/singular optimal window widths and the *maximum gap* constraint is set to 1 week to consider very large optimal window widths. Indeed, when searching for the optimal window width of an episode rule, confidence and support measures are computed for window widths that are lower or equal to the number of events of the rule premise multiplied by the *maximum gap* constraint. Finally, FLM-rules containing 4 event types as a maximum are considered to get generic rules and to make extractions tractable.

For each extraction the number of episode rules ranges from 7509368 to 8713574 while the number of FLM-rules ranges from 2760307 to 3554548 rules. Among them, 431 to 486 FLM-rules end on symbol seizing. At the end of the FLM-rule extraction and selection process, we get a FLM-base containing 29 FLM-rules along with their respective optimal window widths. The frequency bands involved in these rules all cope with experts' knowledge except for one band. Using various pump behavior simulations, our experts did validate it: a new frequency band, that was not reported in the literature or in internal reports, has thus been discovered. The optimal window width distribution presents several modes including 2 strong ones: about 3 days (15 FLM-rules)

Algorithm 6 (searchForFirstOccurrenceBeginingAtT_0)
Input:

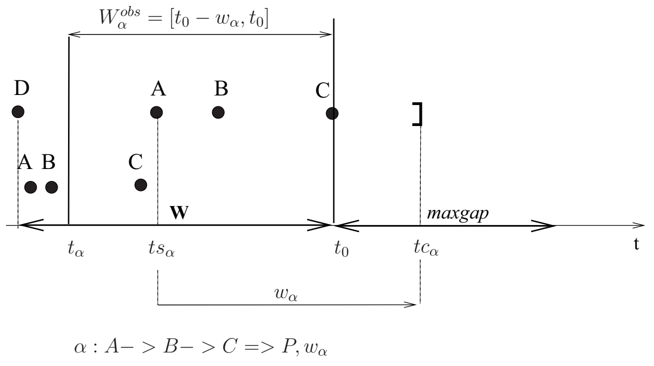
- *t_0, the date at which the algorithm is triggered*
- *p, the tuple $\langle premise, w_r, tc_r, size\rangle$ describing the premise to be searched for*
- *x, a E/O-pair*
- *L_1, the set of E/O-pairs of the event types belonging to the FLM-base premises*
- *i, the recursion level*
- *$FLM - base$, the FLM-base*

```
1.  let $y := l \in L_1 \mid l.Pattern = p.premise[p.size - i]$
2.  let $z$ be a mpo
3.  let $z.Pattern := x.Pattern \rightarrow y.Pattern$
4.  let $z.Occ := \emptyset$
5.  let $(t_s, T) \in x.Occ \mid t_s = 0$
6.  let $L := \emptyset$
7.  for all $t \in T \mid t < p.w_r$ do
8.      let $EndingTimes := \{t'_s \mid \exists(t'_s, T') \in y.Occ$ such that $t'_s > t_s \wedge t'_s - t \leq gapmax$
        $\wedge\forall(t'', T'') \in x.Occ, t_s < t'' \Rightarrow \forall t''' \in T'', t'_s \leq t'''\}$
9.      let $L := L \cup EndingTimes$
10. od
11. if $L \neq \emptyset \wedge i < p.size$
12.     $z.Occ := z.Occ \cup \{(t_s, L)\}$
13.     $searchForFirstOccurrenceBeginingAtT_0(t_0, p, z, L_1, i + 1)$
14. fi
15. if $L \neq \emptyset \wedge i = p.size$
16.     let $p' \in FLM - base \mid p'.premise = p.premise$
17.     let $p'.tc_r := t_0 - min(L) + p.w_r$
18. fi
```

and 10 days (14-rules). This clearly shows that setting a unique window width when extracting such rules is too restrictive. The extracted rules looks like : $(1931- > 1933- > 1933 => 4, 100, 2448, 3)$, which means: if signal over frequency 19 is very high (3) for a few minutes (1) and then very high (3) for few days (3) twice $(1933- > 1933)$, pump has a probability of 100% to seize (event 4) within 2448 minutes. It has been observed 3 times. The extraction and selection process is tractable: execution times does not exceed 3 hours on a standard PC (proc. Intel Xeon CPU 5160 @ 3.00GHz, 4 Gbytes ram, linux kernel 2.26.22.5) and no memory swap is to be reported.

6.2 Real Time Prognosis

When prognosing faults, each time a symbol occurs in the data stream, 2 different informations are provided to end-users. The first one, namely *the fault forecast*, indicates whether the pump that is monitored is about to fail or not. If so, then the second information to be given is the *forecast window*, i.e. the time interval



$W_\alpha^{obs} = [t_0 - w_\alpha, t_0]$
D
A
B
C
A B
C
W
maxgap
t_α
ts_α
t_0
tc_α
t
w_α
$\alpha : A- > B- > C => P, w_\alpha$

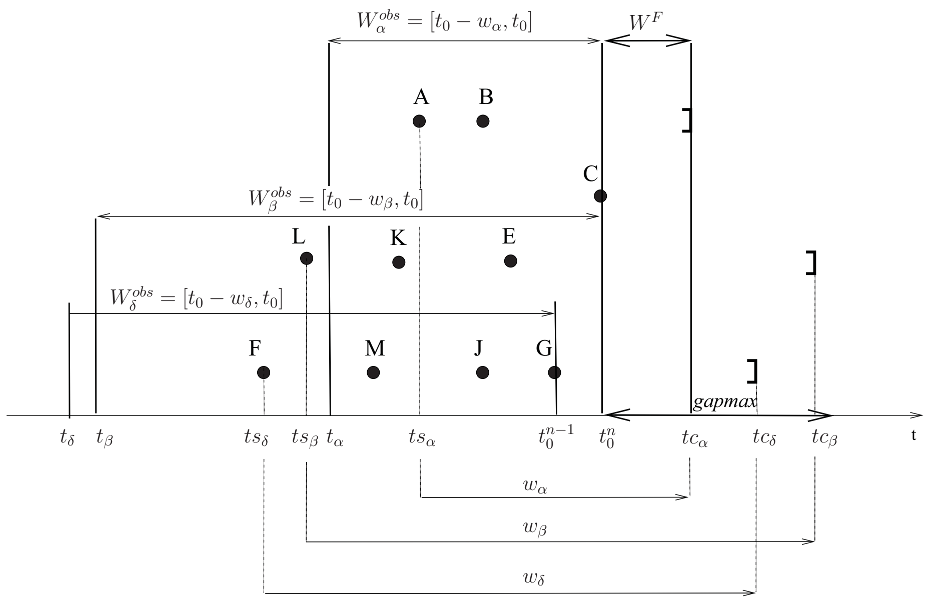
Fig. 7. Projecting the optimal window width of rule $A \to B \to C \Rightarrow P$.

in which the pump is likely to fail. A twofold evaluation is consequently proposed to characterize both informations.

Evaluating the Fault Forecast Each time a symbol occurs, a fault forecast is provided. Let 'a fault is forecasted' and 'no fault is forecasted' be two different classes charactering a pump whose status has just changed (i.e. a symbol just occurred). Having objects and their classes, a classification-based evaluation can be conducted. Nevertheless, temporal aspects must be taken into account. Firstly, the objects are quite unusual: they are defined according to a given system/pump for a given date t_0 (when a symbol occurs). Secondly, the only way to get the class label is to look after t_0. As the FLM-rules having been extracted using a *maximum gap* constraint, no fault occurring after $t_0 + maxgap$ can be forecasted. Therefore, time interval $]t_0, t_0 + maxgap]$ is checked to get the class label. As class 'a fault is forecasted' is the minority one, it is termed the positive class '+' while the other one is termed the negative '-' class. Four different cases, can be distinguished:

- case 1: true positives (TP) - a fault is forecasted and it occurs in $]t_0, t_0 + maxgap]$.
- case 2: false negatives (FN) - no fault is forecasted but a fault occurs in $]t_0, t_0 + maxgap]$.
- case 3: false positives (FP) - a fault is forecasted but no fault occurs in $]t_0, t_0 + maxgap]$.
- case 4: true negatives (TN) - no fault is forecasted and no fault occurs in $]t_0, t_0 + maxgap]$.

The instance number of these cases can be reported in a confusion matrix and can serve as basis for calculating standard classification measures (e.g., accuracy, recall, precision or specificity). The reader is referred to [27] for a full presentation of these measures.



$\alpha : A- > B- > C => P, w_\alpha$

$\beta : L- > K- > E- > C => P, w_\beta$

$\delta : F- > M- > J- > G => P, w_\delta$

Fig. 8. Merging prediction information of FLM-rules.

We applied the real time forecast method on 2 datasets: the 13 sequences used to build our FLM-base (dataset 1) and 21 new sequences of production data (dataset 2). Using these datasets, we simulated 2 data streams and made respectively 24125 and 32525 forecasts. Results of evaluations are given in Table 1 and in Table 2, using confusion matrices

Table 1. Dataset 1 confusion matrix.

		Predicted classes	
		+	−
Actual	+	492	262
classes	−	20	23351

Table 2. Dataset 2 confusion matrix.

		Predicted classes	
		+	−
Actual	+	300	0
Classes	−	404	31821

Using these counts, accuracy, recall, precision, and specificity measures are calculated for dataset 1 and dataset 2 (see Table 3).

Algorithm 7 (buildForecastInterval)
Input:

- t_0*, the date at which the algorithm is triggered*
- $FLM - base$*, the FLM-base*

```
1. let $p \in FLM - base \mid p.tc_r \neq 0 \wedge \nexists p' \in FLM - base$ such that $p'.tc_r < p.tc_r$
2. if $p'.tc_r \neq 0$
3.   output $\langle t_0, p.tc_r \rangle$
4. else
5.   output $\langle 0, 0 \rangle$
6. fi
```

Table 3. Classification measures for dataset 1 and dataset 2.

dataset	accuracy	recall	precision	specificity
dataset 1	0.98	0.64	0.96	0.99
dataset 2	0.98	1	0.43	0.99

Though we could access few data relating to failures so far, results are encouraging as we forecast 10 seizings out of 13 with 98% of accuracy on dataset 1 and as we foresee 2 upcoming seizings with 98% of accuracy as well on dataset 2. There are very few false alarms, which is very important in our production context: specificity measures reach 0.99 for both datasets. Furthermore, the 20 and 404 false alarms (Tables 1 and 2) stating that a pump will seize are generated for a pump that seizes few days later. Thus, the precision measures (96% for dataset 1, 43% for dataset 2) should be balanced. A typical illustration can be found on Figure 9. More precisely, Figure 9 shows the evolution of the forecast window for seizing #11. It is worth notice that the first forecast window, provided 7 days before seizing, did not include the failure. However, all forecast windows provided between 5 days before the failure and the failure date contain the date of occurrence of the forthcoming seizing. This figure also illustrates that the more we are close to the failure occurrence, the more the forecast window is precise (the largest one covers 7.5 days). As the low precision level observed for dataset 2 is due to too early warnings, the optimal window widths that have been learned on dataset 2 might be to short ones. This can be explained by the small number of training examples, only 13 seizings, which imposes low support thresholds. From an applicative point of view, this does not represent a major problem because the pumps all seized a few days later. Finally, the 262 false negatives on dataset 1 all relate to the 3 failures that are not forecasted. The associated recall measure is thus 64% while it rises up to 100% for dataset 2. A so low recall measure for the training dataset indicates that the extraction and selection process presented in Section 4.3 tends to focus on very generic rules. It is indeed the case: this process has been designed to select rules with high confidence ratios that do not trigger any false alarm. Very specific rules, that only hold for a pump or so, are thus discarded. This inhibits us from forecasting 3 seizings. This has to be balanced, because extracted rules are generic enough to forecast seizings using dataset 2, which has been built by monitoring pumps that were subjected to different manufacturing processes (e.g., different gases).

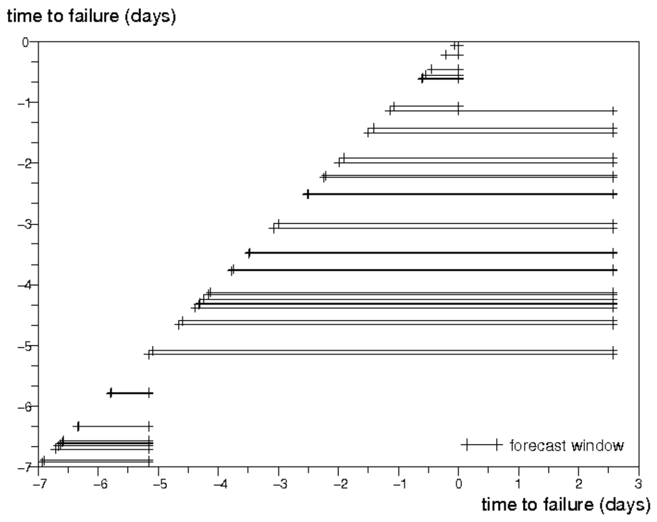


Fig. 9. The forecast window evolution of seizing #11.

Evaluating the Forecast Window As previously mentioned in this section, by construction, a failure can not be forecasted after $t_0 + maxgap$. A rough solution would be therefore to provide end-users with constant forecast window $]t_0, t_0 + maxgap]$. Nevertheless, as explained in Section 5.2, a narrower forecast window can be calculated: $]t_0, t_e^f]$, with $t_e^f \leq t_0 + maxgap]$. Is it correct to propose such a reduced forecast window? This section tries answering that question. In [28], the authors propose different measures for evaluating prognosis techniques. Among them, the accuracy and the precision are to be mentioned. They differ from the ones used in the previous paragraph and can be expressed as follows:

– accuracy: the average distance between the seizing occurrence date and its forecasted occurrence date.
– precision: the standard deviation of the distance between the seizing occurrence date and its forecasted occurrence date.

These measures holds for techniques providing a single occurrence while our method outputs time interval $]t_0, t_e^f]$, the forecast window. We thus propose to adapt these measures to our framework by calculating the mean μ_{error} (cf. Equation 3) and the standard deviation σ_{error} (cf. Equation 4) of the distances/errors between dates t_e^f, t_0 and $t_0 + maxgap$ w.r.t. to t_r, the date at which the seizing occurs. Dates t_e^f, t_0 and $t_0 + maxgap$ are denoted t_i in equations 4 and 3 while N is the number of forecast windows.

$$\mu_{error} = \frac{1}{N}\sum_{i=1}^{N}(t_r - t_i) \tag{3}$$

$$\sigma_{error} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_r - t_i)^2} \tag{4}$$

In order to check whether the forecasted occurrence dates are, in average, before or after the seizing occurrence dates, $(t_r - t_i)$ is preferred to $|t_r - t_i|$ in Equation 3. These measures are reported in Table 4 and Table 5 for each one of the datasets. For dataset 1, the lowest average distance is obtained for date t_e^f: -0.85 days. The same holds for the standard deviation with a value of 1.48 days. Date t_e^f is thus the most accurate and the most precise estimation of t_r. For dataset 2, date t_e^f remains the most precise (2.15 days) but it is not the most accurate. In this case, date t_0 is indeed better with an accuracy of 2.27 days. In any case, for both datasets, date t_e^f and date t_0 are far better than $t_0 + maxgap$. The proposed interval reduction thus makes sense.

Table 4. Accuracy and precision for dataset 1.

	Date to which t_r is compared.		
	$t_s^f = t_0$	t_e^f	$t_0 + maxgap$
μ_{error} (days)	2.06	-0.85	-4.88
σ_{error} (days)	2.26	1.48	2.26

Table 5. Accuracy and precision for dataset 2.

	Date to which t_r is compared.		
	t_s^f	t_e^f	$t_0 + maxgap$
μ_{error} (days)	2.27	-3.14	-4.68
σ_{error} (days)	2.69	2.15	2.69

In more details, for dataset 1, the distance w.r.t. t_0 is always positive and, most of the time, it ranges from 0 to 4 days. Sizings thus occur after t_0 in 100%

of the case, while in 90% of the cases, they occur right before t_e^f. Furthermore, as already seen, in most cases, the distance w.r.t. t_e^f is below 1 day. On the contrary, in 97% of the case, $t_0 + maxgap > t_r$. The forecast interval is thus a precise and accurate estimation of seizing occurrence dates. For dataset 2, the distribution remain similar for date $t_0 + maxgap$. In most of the cases, seizings occur between 5 and 7 days after $t_0 + maxgap$. They also appear mainly between 2 and 4 days (even 6 days in some cases) after t_e^f. In that case, the interval reduction is too strong but still, t_e^f remains more accurate and precise than $t_0 + maxgap$. It thus better expresses how much time is left before a failure/seizing. Finally, latest failure predictions provided by our software prototype arise at least 3 hours before failure really occurs and, most of the time, more than 2 days before. This is enough to plan maintenance tasks.

6.3 Performances

In Section 2, two techniques for forecasting event types in a data stream context are presented. The one proposed in [11] aims to forecast the next event types of interest that are about to arise. Given that this technique does not provide any occurrence date, it can not be applied in our context. Indeed, technical teams need this information to plan maintenance tasks.

The second technique is proposed in [10]. It searches for previously extracted episode rule premises in data streams. As soon as an episode rule premise is recognized, it is used to forecast future event types. The latest date at which the corresponding conclusion is meant to occur is computed by adding the maximum window width of episode rule occurrences, β, to the occurrence date of the first symbol of the premise. If several dates are computed and active, then the lastest one is retained. This is contrary to our method: we select the earliest one. A crucial temporal information, i.e. β, has to be set by users. It is indeed used both to learn episode rules and to forecast event types. Another maximum window width, termed as α, must be set by users to learn episode rules: the premise maximum window width, i.e., the maximum time gap between the first and the last event of the premise occurrences. These two time constraints, originally proposed in [6, 8, 7], have to be satisfied by all rule occurrences, whatever the size of rules, i.e., the number of event types. We here remind the reader that we propose to use FLM-rules whose occurrences are linearly constrained w.r.t. the size of rules by using a maximum gap constraint between events. Our technique also selects automatically the most reliable FLM-rules, i.e. episode rules having optimal window widths, which are, in turn, used to forecast event types. Unlike the method proposed in [10], users do not have to set the width of forecast windows and different window widths, depending on the rules that are found in data streams, can be used.

Though there exist significant differences between our method and the one proposed in [10], especially when it comes to define the window widths that are used to forecast event types, we checked whether the later would work in our context. To do so, we used the same data sets and the same preprocessing to learn and to forecast seizings. In order to be as close as possible in terms of learning parameters, we also asked for rules formed by 4 event types as a maximum and ending on event type 'seizing'. We used the same support and confidence thresholds ($\sigma = 2, \gamma = 100\%$). Knowing that the maximum gap constraint was set to 10 days to learn FLM-rules, we set α to 20 days and β to 30 days to learn episode rules. Thus, as for FLM-rules:

– the largest occurrences to be considered do not exceed 20 days when considering rule premises and 30 days when considering episode rules,
– there is no more than 10 days between the last event type of premises and rules conclusions.

In order to learn episode rules by taking into account parameters α, β and all other parameters, we used the prototype of C. Rigotti, DMT4SP [29] : 611 episode rules ending on event type 'seizing' were extracted.

About forecasts, both methods add a temporal window width to the occurrence date of the first event type of the found premises to compute the latest date at which the conclusion of the rule is forecasted. Our method adds the optimal window widths of FLM-rules while the method proposed in [10] adds β, the unique maximum window width of episode rules. Nevertheless, by construction and because of our parameter choice strategy, for both methods, the largest possible forecast interval is $]t_0, t_0 + 10\ days] =]t_0, t_0 + maxgap]$. Therefore, the same measures, as defined in Section 6.2, are considered to evaluate the performance of the method proposed in [10]. About fault forecasts, results are as follows:

Table 6. Classification measures for dataset 1 and dataset 2.

dataset	accuracy	recall	precision	specificity
dataset 1	0.86	0.12	0.14	0.93
dataset 2	0.92	0.18	0.07	0.94

As it can be observed, all our measures are better, especially when considering recall and precisions ones. Furthermore, on dataset 1, only 7 out of 13 were predicted while 1 out of 2 seizings are predicted on dataset 2. We remind the reader that we were able to predict 10 seizings on dataset 1 and 2 on dataset 2. Our forecast windows are also far better:

Table 7. Accuracy and precision for dataset 1.

	Date to which t_r is compared.		
	t_s^f	t_e^f	$t_0 + maxgap$
μ_{error} (days)	26.87	14.34	21.96
σ_{error} (days)	36	26	37.18

Table 8. Accuracy and precision for dataset 2.

	Date to which t_r is compared.		
	t_s^f	t_e^f	$t_0 + maxgap$
μ_{error} (days)	43.28	-19.25	34.73
σ_{error} (days)	76	27.49	71.49

These results show that, even if some seizings are forecasted, forecast windows can not be trusted for both datasets. As a conclusion, when considering our datasets, the concept of FLM-rules and optimal window widths (originally proposed in [12]), along with a leave-one-out-based FLM-rule selection process (see Section 4.3), leads to better forecasts: our method indeed outperforms the method proposed in [10].

7 Conclusion and Perspectives

In this paper, we present an approach for modeling pumping systems by means of FLM-rules and for forecasting failures (namely seizings) in a data stream context using these rules along with their respective temporal information. This information is used to define a temporal forecast window. The developed approach is applied in an industrial context in which systems are running under severe and unpredictable conditions. Although a small statistical population of seizings resulting from different causes (heat expansion or gas condensation) is available, results are encouraging. Failures are forecasted with a good temporal accuracy and precision on both the learning dataset and a new dataset, while the false alarm level remains low. In addition, our forecasts provide enough time to technical teams to plan interventions. This approach is now patented [30]. Future works include introducing fuzzy logic in order to provide a gradual warning within a forecast window and studying loose matching of episode rules in data streams. Loose matching could help us in generalizing our technique to monitor other systems where temporal aspects are not always prevalent. Similarity measures capturing both ordered and non-ordered dependencies, such as the one proposed in [31], are good candidates.

References

1. Letourneau, S., Famili, F., Matwin, S.: Data mining for prediction of aircraft component replacement. IEEE Intelligent Systems and their Applications 14(6), 59–66 (december 1999)
2. Schwabacher, M., Goebel, K.: A Survey of Artificial Intelligence for Prognostics. In: Working Notes of 2007 American Institute in Aeronautics and Astronautics Fall Symposium: AI for Prognostics (2007), http://www.aaai.org/Library/Symposia/Fall/2007/fs07-02-016.php
3. Magoulas, G., Prentza, A.: Machine learning in medical applications. Machine Learning and Its Applications 2049, 300–307 (2001)

4. I.S.O.: Condition monitoring and diagnostics of machines prognostics - part1 : General guidelines, iso 13381-1. Geneva, Switzerland (2004)
5. Xing, Z., Pei, J., Yu, P.S.: Early prediction on time series: a nearest neighbor approach. In: Proceedings of the 21st international jont conference on Artifical intelligence. pp. 1297–1302. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2009)
6. Hatonen, K., Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H.: TASA: Telecomunications Alarm Sequence Analyzer or: How to enjoy faults in your network. In: 1996 IEEE Network Operations and Management Symposium (NOMS'96). pp. 520–529. Kyoto, Japan (April 1996)
7. Mannila, H., Toivonen, H.: Discovering generalized episodes using minimal occurrences. In: In Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining (KDD'96). pp. 146–151. AAAI Press (1996)
8. Mannila, H., Toivonen, H., Verkamo, A.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3), 259–298 (November 1997)
9. Achar, A., Laxman, S., Sastry, P.: A unified view of the apriori-based algorithms for frequent episode discovery. Knowledge and Information Systems pp. 1–28 (2011), http://dx.doi.org/10.1007/s10115-011-0408-2
10. Cho, C.W., Zheng, Y., Chen, A.L.P.: Continuously matching episode rules for predicting future events over event streams. In: Dong, G., Lin, X., Wang, W., Yang, Y., Yu, J. (eds.) APWeb/WAIM 2007. LNCS, vol. 4505, pp. 884–891. Springer-Verlag, Berlin, Heidelberg (2007)
11. Laxman, S., Tankasali, V., White, R.W.: Stream prediction using a generative model based on frequent episodes in event sequences. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08), Las Vegas, Nevada, USA, August 24-27, 2008. pp. 453–461 (2008)
12. Méger, N., Rigotti, C.: Constraint-based mining of episode rules and optimal window sizes. In: Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS, vol. 3202, pp. 313–324. Springer (2004)
13. Choudhary, A., Harding, J., Tiwari, M.: Data mining in manufacturing: a review based on the kind of knowledge. Journal of Intelligent Manufacturing 20, 501–521 (2009)
14. El Koujok, M., Gouriveau, R., Zerhouni, N.: Towards a Neuro-Fuzzy System for time series forecasting in Maintenance Applications. In: 17th Triennal World Congress of the International Federation of Automatic Control, (IFAC'08). Elsevier, Seoul, Korea (2008)
15. Le Normand, N., Boissiere, J., Méger, N., Valet, L.: Supply chain management by means of FLM-rules. In: 12th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'08). pp. 29–36 (september 2008)
16. Martin, F., Méger, N., Galichet, S., Becourt, N.: Episode rule-based prognosis applied to complex vacuum pumping systems using vibratory data. In: Perner, P. (ed.) ICDM 2010. LNCS, vol. 6171, pp. 376–389. Springer-Verlag, Berlin, Heidelberg (2010)
17. Doebling, S., Farrar, C., Prime, M.: A summary review of vibration-based damage identification methods. Shock and Vibration Digest 30(2), 91–105 (1998)
18. Broch, J.T.: Application of B and K Equipment to Mechanical Vibraton and Shock Measurement. Brel & Kjr (1973)
19. Lalanne, C.: Vibrations aléatoires. Hermes Science (1999)

20. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering. pp. 3–14. ICDE '95, IEEE Computer Society, Washington, DC, USA (1995)
21. Zaki, M.J.: Efficient enumeration of frequent sequences. In: Proceedings of the seventh international conference on Information and knowledge management. pp. 68–75. CIKM '98, ACM, New York, NY, USA (1998)
22. Zaki, M.J.: Sequence mining in categorical domains: incorporating constraints. In: Proceedings of the ninth international conference on Information and knowledge management. pp. 422–429. CIKM '00, ACM, New York, NY, USA (2000)
23. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning 42, 31–60 (January 2001)
24. Leleu, M., Rigotti, C., Boulicaut, J.F., Euvrard, G.: Go-spade: mining sequential patterns over datasets with consecutive repetitions. In: Perner, P., Rosenfeld, A. (eds.) MLDM 2003. LNCS, vol. 2734, pp. 293–306. Springer-Verlag, Berlin, Heidelberg (2003)
25. Senkul, P., Salin, S.: Improving pattern quality in web usage mining by using semantic information. Knowledge and Information Systems pp. 1–15 (2011), http://dx.doi.org/10.1007/s10115-011-0386-4
26. Loekito, E., Bailey, J., Pei, J.: A binary decision diagram based approach for mining frequent subsequences. Knowledge and Information Systems 24, 235–268 (2010)
27. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, (First Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2005)
28. Zemouri, R., Gouriveau, R., Patic, P.: Combining a recurrent neural network and a PID controller for prognostic purpose: a way to improve the accuracy of predictions. WSEAS Transactions on Systems and Control 5(5), 353–371 (2010)
29. Rigotti, C.: Data mining tool 4 sequential patterns - dmt4sp (2011), http://liris.cnrs.fr/ crigotti/dmt4sp.html
30. Bécourt, N., Martin, F., Pariset, C., Galichet, S., Méger, N.: Method for predicting a rotation fault in the rotor of a vacuum pump and associated pumping devices, alcatel-lucent (2010)
31. Kelil, A., Wang, S., Jiang, Q., Brzezinski, R.: A general measure of similarity for categorical sequences. Knowledge and Information Systems 24, 197–220 (2010)