Expert Systems with Applications 38 (2011) 7003–7015

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Anomaly detection in monitoring sensor data for preventive maintenance

Julien Rabatel [a,b,*], Sandra Bringay [a,c], Pascal Poncelet [a]

[a] LIRMM, Université Montpellier 2, CNRS, 161 rue Ada, 34392 Montpellier Cedex 5, France
[b] Fatronik France Tecnalia Cap Omega, Rond-point Benjamin Franklin - CS 39521, 34960 Montpellier, France
[c] Dpt MIAp, Université Montpellier 3, Route de Mende, 34199 Montpellier Cedex 5, France

ARTICLE INFO

Keywords:
Anomaly detection
Behavior characterization
Sequential patterns
Preventive maintenance

ABSTRACT

Today, many industrial companies must face problems raised by maintenance. In particular, the anomaly detection problem is probably one of the most challenging. In this paper we focus on the railway maintenance task and propose to automatically detect anomalies in order to predict in advance potential failures. We first address the problem of characterizing normal behavior. In order to extract interesting patterns, we have developed a method to take into account the contextual criteria associated to railway data (itinerary, weather conditions, etc.). We then measure the compliance of new data, according to extracted knowledge, and provide information about the seriousness and the exact localization of a detected anomaly.

1. Introduction

Today, many industrial companies must face problems raised by maintenance. The most common solution is called curative maintenance, i.e., equipment is replaced or repaired after the appearance of obvious failures, once the damage is occurred. This solution poses many problems. Curative maintenance is too belated and is particularly costly on several aspects. On the financial side first, for many companies, a few hours of downtime can result in millions of dollars in losses. It is generally much less expensive to make predictive maintenance to prevent a serious breakdown. In addition, the corrective maintenance is also a problem for security aspects. In many sensitive areas, equipment failures can cause death. For example, it is estimated that approximately 5% of motor vehicle accidents are caused by equipment malfunction or a lack of maintenance.[1] Another aspect is related to the environment and energy saving. Indeed, equipment that is worn or subject to malfunctions often consumes more energy than equipment that operates optimally. In addition, a systematic maintenance planning is not a satisfactory solution as too expensive compared to real needs. To reduce the problems of equipment maintenance, to propose ways to make maintenance both faster and more effective by anticipating serious breakdowns represents a particularly critical issue.

Such a preventive maintenance consists in detecting anomalous behavior in order to prevent further damages and avoid more costly maintenance operations. To this end, it is necessary to monitor the working equipment. Usually, monitoring data is available through embedded sensors and provides us with important information such as temperatures, humidity rates, etc. Nevertheless, data collected by sensors are difficult to exploit for several reasons. First, a very large amount of data usually available at a rapid rate must be managed to provide a relevant description of the observed behaviors. Furthermore, they contain many errors: sensor data are very noisy and sensors themselves can become defective. Finally, when considering data transmission, very often lots of information are missing.

In this paper, we focus on the field of train maintenance. Trains monitoring is also ensured by sensors positioned on the main components (wheels, motors, etc.) to provide much information (temperature, acceleration, velocity). In this context, we are subject to the difficulties we have described above: voluminous and noisy data, information transmission problems, etc. Moreover, it is important to take into account the different types of data available. We therefore wish to propose a method to exploit this information in order to assist the development of an effective predictive maintenance.

The needs in the context of train maintenance are twofold. First, it is important to provide a better understanding of monitored systems. Indeed, as they are often complex and contain many components, the experts have little knowledge about their actual behavior. This lack of knowledge makes the problem of maintenance very difficult. From another point of view it could be interesting to get an overview of the normal behavior (e.g., in case of monitoring) and then it is necessary to propose a way for characterizing such normal behaviors from a huge amount of historical data. Another challenging point that we must consider is that

* Corresponding author at: LIRMM, Université Montpellier 2, CNRS, 161 rue Ada, 34392 Montpellier Cedex 5, France.
E-mail addresses: rabatel@lirmm.fr (J. Rabatel), bringay@lirmm.fr (S. Bringay), poncelet@lirmm.fr (P. Poncelet).

[1] http://www.smartmotorist.com.

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.12.014

normal behavior strongly depends on the context. For example, a very low ambient temperature will probably affect a train behavior. Similarly, each itinerary with its own characteristics (slopes, turns, etc.) influences a journey. Consequently it is essential, in order to efficiently characterize the behavior of trains as well as to detect anomalies, to consider the surrounding context. In this paper, we will show how these elements can be directly addressed by data mining techniques and how they can be used to design a system for detecting anomalies in train behavior and help experts so that a detected problem can be dealt as promptly as possible, and the right decisions can be made.

Our approach follows the framework presented in Fig. 1. We address the two issues mentioned above: (i) the knowledge discovery process about normal train behavior and (ii) the anomaly detection in new data.

The characterization of normal behavior is divided into three steps. First, we consider the data describing the normal behavior (i.e., containing no anomalies) that were recorded in the past. These so-called historical data are segmented into different classes, which are defined by the context in which the data were recorded. This organization brings together all trips that were conducted under similar conditions. The criteria used to create the classes are, for example, the outside temperature, the itinerary, etc. Then, we extract the most frequent behaviors to characterize each of these classes. We thus obtain knowledge classes that describe very precisely normal train behavior and also provide us with essential information about the impact of contextual criteria. For example, we can answer questions such as *"What are the behaviors that are specific to a high outside temperature?"*.

After having characterized the train behavior in the first step of our framework, we wish to detect anomalies in newly recorded data. To this end, we use the previously obtained knowledge. We have developed a method to compare new monitoring data with a class of knowledge. Thus, we can detect critical events and notify experts that a maintenance operation may be necessary.

This paper is organized as follows. Section 2 describes the data representation in the context of train maintenance. Section 3 shows the characterization of normal behaviors by discovering sequential patterns. Then we present the anomaly detection for predictive maintenance approach in Section 4. Experiments conducted with real and simulated data are described in Section 5 and related work is presented in section 6. Finally, we conclude in Section 7.

2. Data preprocessing

In this section, we address the problem of preprocessing railway data. From raw data collected by sensors, we design a suitable representation for data mining tasks.

The train monitoring system exploited in this study is such that a large set of sensors is distributed on the main components of each monitored train. The key element of this system is the bogie, because failures and anomalous behaviors are most of the time associated with it. Each of the 8 bogies of a train has 32 sensors collecting information such as temperatures, accelerations and velocity. Every five minutes during a journey, all sensors collect a value stored in a central database. A complete description of these data is available in Carrascal, Díez, and Azpeitia (2009).

2.1. Sensor data for train maintenance

The data resulting from sensors for train maintenance is complex for the two following reasons: (i) very often errors and noisy values pervade the experimental data; (ii) multi-source information must be handled at the same time. For instance, in train maintenance following data must be considered.

2.1.1. Sensors

Each sensor describes one property of the global behavior of a train which can correspond to different information (e.g., temperature, velocity, acceleration).

2.1.2. Measurements

They stand for numerical values recorded by the sensors and could be very noisy for different reasons such as failures, data transfer, etc. Note that the numerical values collected by the sensors are then discretized to obtain a set of data more suited to the step of data mining described in Section 3.

2.1.3. Readings

They are defined as the set of values measured by all the sensors at a given date. The information carried out by a reading could be considered as the state of the global behavior observed at the given moment. Due to the data transfer, some errors may occur and then readings can become incomplete or even missing.

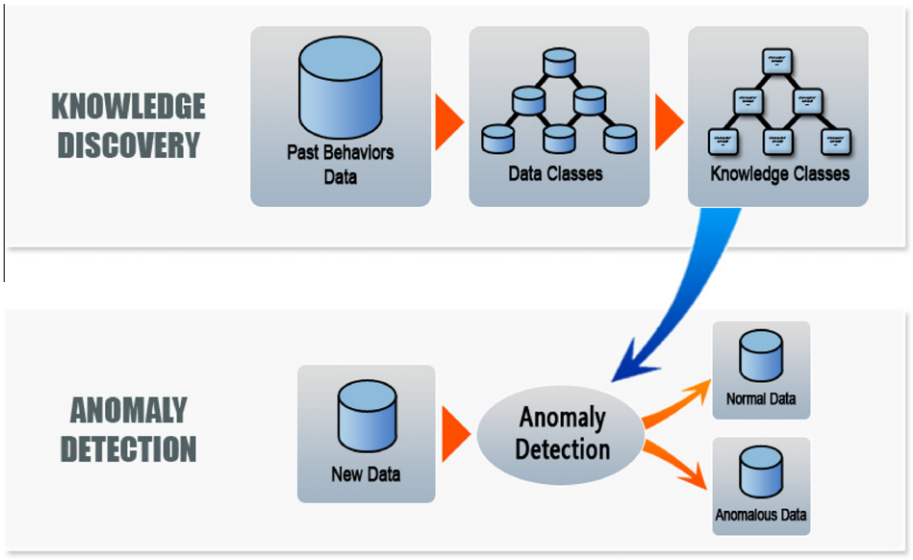

Fig. 1. General framework.

Table 1
Extract from sensor data.

TIME	A	B	C	...
2008/03/27 06:36:39	0	16	16	...
2008/03/27 06:41:39	82.5	16	16	...
2008/03/27 06:46:38	135.6	19	21	...
2008/03/27 06:51:38	105	22	25	...

We consider that handled data are such as those described in Table 1, where a **reading** for a given date (first column) is described by **sensor measurements** (cells of other columns).

The behavior of a sensor is described by the numerical values it measures. The discretization of these sensor values is a preprocessing that, for example, has been used in Chong, Krishnaswamy, Loke, and Gaben (2008), Mihail Halatchev (2005) to group values expressing the same information into classes of values (e.g., a value could be *low*, *medium* or *high*). For example, temperature values $25°C$ and $26°C$ are different but very close, so they can be regarded as carrying the same information. Moreover, for velocity as well as acceleration sensors, we also consider a zero value because it is a particular value that can not be confused with non-zero values. For example, a zero velocity indicates that the train is stationary, while a non-zero velocity, no matter how low it is, concerns a train in motion.

2.2. Granularity in railway data

Data collected from a train constitutes a list of readings describing its behavior over time. As such a representation is not appropriate to extract useful knowledge, we decompose the list of readings at different levels of granularity and then we consider the three following concepts journeys, episodes and episode fragments which are defined as follows.

2.2.1. Journey

The definition of a journey is associated to the railway context. For a train, a journey stands for the list of readings collected during the time interval between the departure and the arrival. Usually, a journey is several hours long and has some interruptions when the train stops in railway stations. We consider the decomposition into journeys as the coarsest granularity of railway data.

Let *minDuration* be a minimum duration threshold, *maxStop* be a maximum stop duration, and J be a list of readings $(r_m, \ldots, r_i, \ldots r_n)$, where r_i is the reading collected at time i. J is a journey if:

1. $(n - m) > minDuration$,
2. $\nexists (r_u, \ldots, r_v, \ldots r_w) \subseteq J \left| \begin{array}{l} (w-u) > maxStop, \\ and\ \forall v \in [u,w],\ velocity(v) = 0. \end{array} \right.$

2.2.2. Episode

The main issue for characterizing train behavior is to compare elements which are similar. However, as trains can have different routes the notion of journey is not sufficient (for instance, between two different journeys, we could have different number of stops as well as a different delay between two railway stations). That is the reason why we segment the journeys into episodes to get a finer level of granularity. Episodes are obtained by relying on the stops of a train (easily recognizable considering the train velocity).

An episode is defined as a list of readings $(r_m, \ldots r_i, \ldots, r_n)$ such as:

- $velocity(m) = 0$ and $velocity(n) = 0$,[2]
- if $m < i < n$, $velocity(i) \neq 0$.

[2] Here, the velocity of the train at time t is denoted as $velocity(t)$.

Fig. 2 describes a segmentation of a journey into episodes by considering the velocity changes. This level of granularity is considered as the most relevant because it provides us with a set of homogeneous data. However, we can segment episodes in order to obtain a more detailed representation and a finer granularity level.

2.2.3. Episode fragment

The level of granularity corresponding to the fragments is based on the fact that the behavior of a train during an episode can easily be divided into three chronological steps. First, the train is stationary (i.e., $velocity\,0$) then an acceleration begins. We call this step the *starting* step. More formally, let $E = (r_m, \ldots, r_n)$ be an episode. The *starting fragment* $E_{starting} = (r_m, \ldots, r_k)$ of this episode is a list of readings such as:

$$\forall i,\ j \in [m,k], \quad i < j \iff velocity(i) < velocity(j).$$

At the end of an episode, the train begins a deceleration ending with a stop. This is the *ending step*. More formally, let $E = (r_m, \ldots, r_n)$ be an episode. The *ending fragment* $E_{ending} = (r_k, \ldots, r_n)$ of this episode is a list of readings such as:

$$\forall i,\ j \in [k,n], \quad i < j \iff velocity(i) > velocity(j).$$

The *traveling fragment* is defined as the sublist of a given episode between the *starting fragment* and the *ending fragment*. During this fragment, there are accelerations or decelerations, but no stop. More formally, let E be an episode, $E_{starting}$ its starting fragment, and E_{ending} its ending fragment. Then, the *traveling fragment* of E, denoted as $E_{traveling}$, is a list of readings defined as:

$$E_{traveling} = E - E_{starting} - E_{ending}.$$

Fig. 2 shows the segmentation of an episode into three fragments: the starting fragment, the traveling fragment and the ending fragment.

From now we thus consider that all the sensor data are stored in a database, containing all information about the different granularity levels. For example, all the sensor readings composing the fragment shown in Fig. 2 are indexed and we know that a particular fragment f is an ending fragment included in an episode e, belonging to the journey J. J is associated with the itinerary I and the index of e in I is 2 (i.e., the second portion of this route).

2.3. Data classes

Previously, we have presented how to process the data in the railway context in order to obtain a relevant representation. However, we have not considered the contextual aspects of such data. Indeed, in the railway field of application as well as other industrial applications, the behavior strongly depends on different exterior parameters.

The nature of such context is inevitably dependant on the field of application. For example, the behavior of a train during a trip depends on contextual criteria such as the weather conditions or its geographical position. We have previously exploited different levels of granularity in data, we now use different existing contexts to consider the variability of train behavior according to them.

Example 1. Table 2 presents a set of fragments, identified by the *id* column. Each fragment is associated with contextual criteria (the humidity rate, the exterior temperature, the global route, and the index of the episode in this route). For example, the fragment f_2 was performed with a low humidity rate and a low exterior temperature. In addition, f_2 is part of the second episode ($E2$) of the itinerary denoted by I_1.

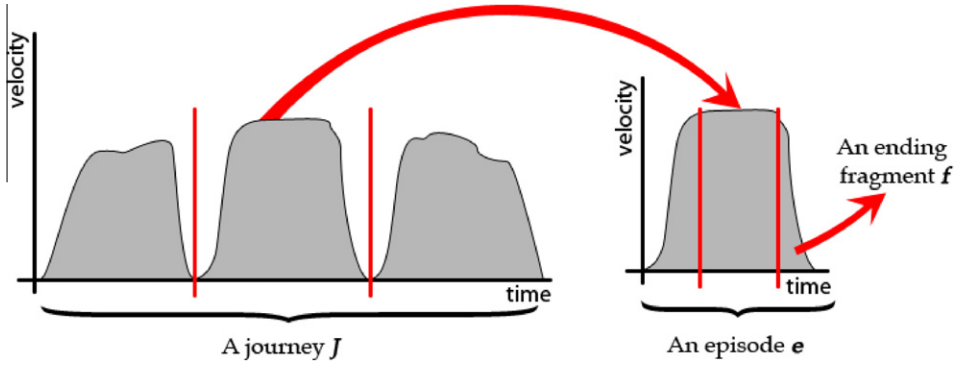

Fig. 2. Segmentation of a journey into episodes.

Table 2
Fragments and contextual information.

ID	Humidity	Exterior temperature	Itinerary	Index
f_1	low	high	I1	E1
f_2	low	low	I1	E2
f_3	high	high	I2	E1
f_4	low	low	I1	E1
f_5	high	low	I1	E2

Table 3
Some contextual classes.

Class	Humidity	Exterior temperature
$[*,*]$	$*$	$*$
$[low,*]$	low	$*$
$[high,*]$	high	$*$
[low,high]	low	high
$[*,high]$	$*$	high
...	...	...

2.3.1. Data classes

Now we present how contextual criteria are handled, by dividing the data into classes according to the different contextual criteria. To this end, we will at first develop a formal representation of the different contexts that can be met.

Based on the general principle of contextualization, here we define how classes are constructed. We consider now a formal description of a context. Each fragment is described in a set of n contextual dimensions (e.g., Duration or Exterior Temperature), denoted by D_C. Let c be a class, defined in D_C. A class c is denoted by $[c_{D_1},\ldots,c_{D_i},\ldots,c_{D_k}]$, where c_{D_i} is the value of c for the dimension D_i, $D_i \in D_C$. We use a *wildcard value*, denoted by $*$, which can substitute any value on each dimension in D_C. In other words, $\forall X \in D_C$, $\forall x \in Dim(X)$, $\{x\} \subset *$.

Thus, an episode e belongs to a class c if the restriction of e on D_C[3] is included in c:

$$\forall X \in D_C, \quad e_X \subseteq c_X.$$

We have identified a lattice structure, called the contextual lattice, to represent the set of contextual classes. We now explain how this structure is defined.

2.3.2. Contextual lattice

The set of contextual classes can be represented in a multidimensional space containing all the combinations of different criteria as well as their possible values.

Example 2. Table 3 shows some of the contextual classes corresponding to two dimensions: Humidity and Exterior Temperature (respectively denoted as *Hum* and *ExtT* below).

A class c is denoted by $[c_{ExtT},c_{Hum}]$. For example, the class denoted by $[low,*]$ is equivalent to the context where the temperature is low (i.e., $c_{ExtT} = low$), for any humidity rate (i.e., $c_{Hum} = *$).

Using the dataset of Table 2, we can see that the set of fragments belonging to the class $[low,*]$ is $\{f_1,f_2,f_4\}$. Similarly, the set of fragments belonging to the class $[low,high]$ is $\{f_1\}$.

[3] The restriction of e in D_C is the description of e, limited to the dimensions of D_C.

Contextual classes and their relationships can be represented as a lattice. Nevertheless, we first have to define a generalization/specialization order on the set of environmental classes.

Definition 1. Let c, c' be two classes. $c > c' \iff \forall X \in D_C, c'_X \subset c_X$. If $c > c'$, then c is said to be more general than c' and c' is said to be more specific than c.

Moreover, if there is no class c' such that $c' \neq [\emptyset,\ldots,\emptyset]$ and $c > c'$, then c is said to be a specialized class.

In order to construct contextual classes, we provide a *sum operator* (denoted by +) and a *product operator* (denoted by $\bullet$). The sum of two classes gives us the most specific class generalizing them. The **sum operator** is defined as follows.

Definition 2. Let c, c' be two classes.

$$z = c + c' \iff \forall X \in D_C, z_X = \begin{cases} c_X \text{ if } c_X = c'_X, \\ * \text{ elsewhere.} \end{cases}$$

Example 3. Given the set of classes in Table 3, we note that $[low, low] + [low,high] = [low,*]$, and $[low,high] + [high,low] = [*,*]$.

The product of two classes gives the most general class specializing them. The **product operator** is defined as follows.

Definition 3. Let c, c' be two classes. The class u is defined as follows: $\forall X \in D_C,\ u_X = c_X \cap c'_X$. Then,

$$z = c \bullet c' \iff \begin{cases} z = u \text{ if } \nexists X \in D_C | u_X = \emptyset, \\ [\emptyset,\ldots,\emptyset] \text{ elsewhere.} \end{cases}$$

Example 4. Given the set of classes in Table 3, we note that $[low, high]\bullet[low,*] = [low,high]$, and $[*,high]\bullet[low,low] = [\emptyset,\emptyset]$.

We can now define a lattice, by using the generalization/specialization order between classes and the operators defined above. The ordered set $\langle CS, > \rangle$ is a lattice denoted as CL, in which Meet ($\bigwedge$) and Join ($\bigvee$) elements are given by:

1. $\forall C \subset CL$, $\bigwedge C = +_{c \in C} c$
2. $\forall C \subset CL$, $\bigvee C = \bullet_{c \in C} c$

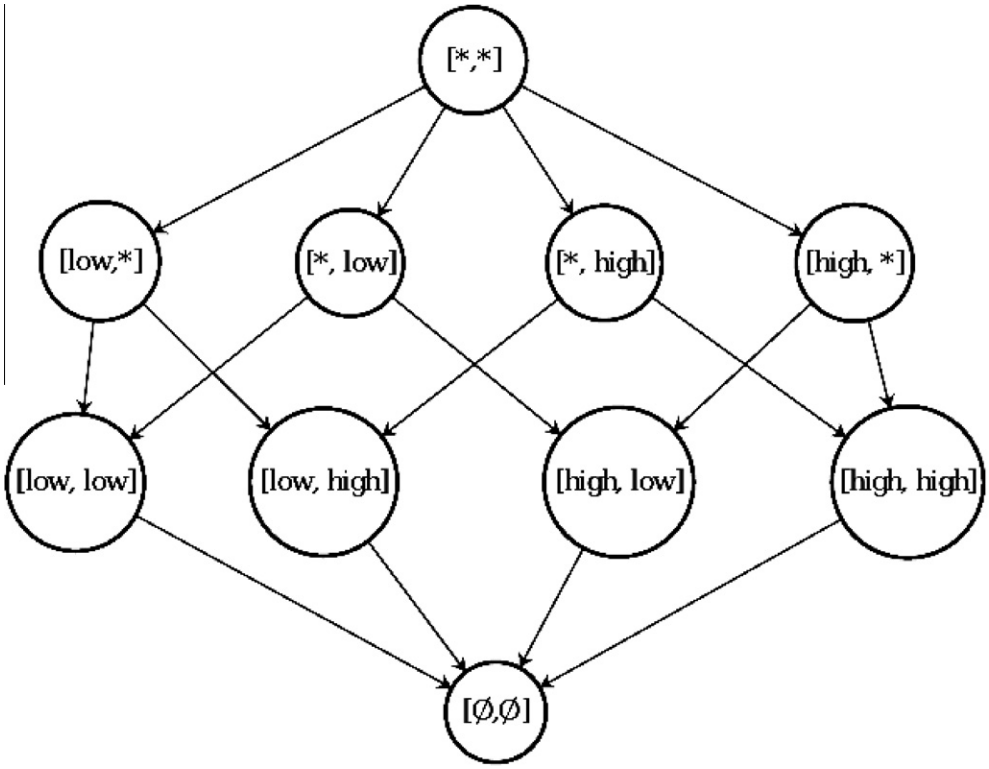

Fig. 3. A contextual lattice.

Fig. 3 illustrates an extract of the lattice of contextual classes of the dataset provided in Table 2. We can now very precisely index historical data in appropriate classes.

3. Normal behavior characterization

In this section, we focus on the data mining step in the knowledge discovery process and more precisely on the extraction of patterns characterizing normal behavior.

3.1. How to extract normal behavior?

The objective of the behavior characterization is, from a database of sensor measurements, to provide a list of patterns depicting normal behavior. We want to answer the following question: *which patterns often appear in the data?* Such a problem, also known as pattern mining, has been extensively addressed by the data mining community in the last decade.

Amongst all the data mining methods, we cite the sequential patterns mining problem. Sequential patterns were introduced in Agrawal and Srikant (1995) and can be considered as an extension of the concept of association rule Agrawal, Imieliński, and Swami (1993) by handling timestamps associated to items. The sequential patterns goal is to extract sets of items commonly associated over time. In the "*basket market* " concern, a sequential pattern can be for example: "*40 % of the customers buy a television, then buy later on a DVD player* ". In the following we adapt the sequential pattern mining problem to our sensor data.

In the following definitions and examples, we will consider a set of sensors, denoted by $\mathbf{\Omega}$, and the set of possible values collected by a sensor $A \in \mathbf{\Omega}$, denoted by $dom(A)$.

Definition 4. An **item** A_v is a pair $\{A, v\}$, where $A \in \mathbf{\Omega}$ and $v \in dom(A)$. It stands for the value v collected by the sensor A at a given time. The item A_v is called an **A-item**. Moreover, we denote by $value(A_v)$ the value v, and by $\mathbf{sensor}(\mathbf{A_v})$ the sensor A.

Definition 5. Let $\omega = \{A^1, A^2, \ldots, A^i, \ldots, A^n\}$ a set of sensors such that $\omega \subseteq \mathbf{\Omega}$. An **itemset** I is a reading or a part of a reading at a given time, i.e., a non-ordered set of items, denoted as:

$$I = \left(A^1_{v_1} A^2_{v_2} \ldots A^i_{v_i} \ldots A^n_{v_n}\right).$$

Note that if $A^i_{v_i} \in I$, $\nexists A^i_{v'_i} | A^i_{v'_i} \in I$. Indeed, a sensor cannot read two different values at the same time.

Definition 6. A **sequence** s is a list of itemsets denoted as:

$$s = \langle I_1 I_2 \ldots I_i \ldots I_n \rangle,$$

where the itemset I_i stands for the i_{th} itemset of s. An empty sequence is denoted as $\emptyset$.

Example 5. Data described in Table 1, are translated into the following sequence:

$$\langle (A_0 B_{16} C_{16})(A_{82.5} B_{16} C_{16})(A_{135.6} B_{19} C_{21})(A_{105} B_{22} C_{25}) \rangle.$$

When data are discretized, the sequence becomes:

$$\langle (A_0 B_{low} C_{low})(A_{avg} B_{low} C_{low})(A_{high} B_{avg} C_{avg})(A_{high} B_{avg} C_{high}) \rangle.$$

Definition 7. Given two sequences $s = \langle I_1, I_2, \ldots, I_m \rangle$ and $s' = \langle I'_1 I'_2 \ldots I'_n \rangle$, if there exist integers $1 \leqslant i1 < i2 < \cdots < im \leqslant n$ such that $I_1 \subseteq I'_{i_1}$, $I_2 \subseteq I'_{i_2}, \ldots, I_m \subseteq I'_{i_m}$, then the sequence s is a **subsequence** of the sequence s', denoted as $s \sqsubseteq s'$, and s' *supports* s.

Table 4
Sample of sequences.

ID	Sequence
s_a	$\langle(A_{high}B_{high})\rangle$
s_b	$\langle(A_{high})(A_{avg}B_{avg})(B_{low})(A_{low})\rangle$
s_c	$\langle(A_{high})(B_{low})\rangle$
s_d	$\langle(A_{high})(B_{avg})\rangle$

Example 6. From Table 4, we can note that $s_c \sqsubseteq s_b$ and $s_d \sqsubseteq s_b$.

Definition 8. If a sequence s is not a subsequence of any other sequences, then we say that s is **maximal**.

Example 7. From the set of sequences contained in Table 4, we can note that s_a and s_b are maximal sequences.

Definition 9. Let s be a sequence. The **length** of s, denoted as $|s|$, is the number of itemsets in s. The **size** of s, denoted as $\|s\|$, is the number of items in s.

Example 8. The length of s_a is $|s_a| = 1$, and its size is $\|s_a\| = 2$.

Definition 10. The **support** of a sequence is defined as the fraction of total sequences in a sequence database *DB* that support this sequence. A sequence is said to be **frequent** if its support is greater than or equal to a threshold minimum support (*minSupp*) specified by the user.

The sequential pattern mining problem is, for a given threshold *minSupp* and a sequence database *DB*, to find all frequent sequences in *DB*.

3.1.1. Contiguous subsequences mining

To extract interesting patterns in a database of behavioral sequences, it is important to note that a frequent sequence is interesting only if the gap between each itemset is limited. By extracting frequent sequences in a database of behavioral sequences, we want to highlight the frequent interactions and correlations between sensors, but also between readings. However, the interest of those patterns is strongly associated with the temporal gap between each pair of itemsets in a sequence. To take this into consideration, we modify the concept of subsequence in order to consider only the consecutive itemsets in a sequence but first of all we define the notion of concatenation.

Definition 11. The **concatenation** of sequences is denoted as $s + s'$, and the result is the sequence obtained by appending s' to the end of s, so that we have $|s + s'| = |s| + |s'|$ and $\|s + s'\| = \|s\| + \|s'\|$.

Example 9. The sequence corresponding to the concatenation of s_a and s_c is:

$$s_a + s_c = \langle(A_{high}B_{high})(A_{high})(B_{low})\rangle.$$

Definition 12. A sequence s is a **contiguous subsequence** of the sequence s', denoted as $s \sqsubseteq {}^{c}s'$, if there exist three sequences s_1, s_2, and s_3 such that $s' = s_1 + s_2 + s_3$, $|s| = |s_2|$, and $s \sqsubseteq s'$.

Example 10. The sequence s_d is a contiguous subsequence of s_b, i.e., $s_d \sqsubseteq {}^{c}s_b$. Indeed, there exist three sequences $s_1 = \emptyset$, $s_2 = \langle(A_{high})(A_{avg}B_{avg})\rangle$ and $s_3 = \langle(A_{low})\rangle$, such that $s_d = s_1 + s_2 + s_3$, $|s_d| = |s_2|$, and $s_b \sqsubseteq s_2$.

$$s_b = \overbrace{\emptyset}^{s_1} + \overbrace{\langle(A_{high})(A_{avg}B_{avg})\rangle}^{s_2} + \overbrace{\langle(B_{low})(A_{low})\rangle}^{s_3},$$
$$s_d = \langle(A_{high})(B_{avg})\rangle.$$

3.1.2. Aggregated sequences

Handled behavioral data are highly redundant: the behavior described by the sensor measurements are generally stable for a reading to another. This characteristic is particularly visible for data that change slowly over time (e.g., a wheel temperature).

Therefore, we are dealing with very large sequences with consecutive itemsets containing redundant information. Such sequences bring two problems: (i) the mining of this kind of sequences is particularly difficult (the repetition of identical itemsets considerably increases the search space) and (ii) it yields no additional information. We therefore propose the concept of aggregated sequence.

Definition 13. Let $s = \langle I_1, I_2, \ldots, I_n\rangle$ be a sequence. The corresponding **aggregated pattern**, denoted by s^*, is the maximal subsequence of s respecting the following condition:

$$s^* = \langle I_1^* I_2^* \ldots I_i^* \ldots I_m^*\rangle, \quad \textit{such that } \forall I_i^* \in s^*, \quad I_i^* \neq I_{i+1}^*.$$

Note that $s^* \sqsubseteq s$.

Example 11. Let $s = \langle(A_{low}B_{avg})(A_{low})(A_{low})(B_{high})\rangle$. The aggregated pattern of s, denoted by s^*, is:
$s^* = \langle(A_{low}B_{avg})(A_{low})(B_{high})\rangle$.

3.2. Contextualized characterization

We have seen in Section 2 that normal train behavior is related to contextual parameters. We have described the necessary concepts and methodologies to extract knowledge in a set of historical data. However, in Section 2 we have underlined the fact that normal behavior are very dependent on the context. In consequence, to properly characterize such behavior, we take into account the data classes described previously.

3.2.1. Knowledge classes

A class describes a context according to different criteria. To extract knowledge about the influence of these criteria on a train behavior, each class is associated with (i) all behavioral data collected in the associated context (see Section 2), (ii) the patterns characterizing the normal behavior in this context.

In Section 2.3.1, we have seen that the context of a class is provided through the description of this class. We present below how are defined the corresponding set of data and the set of characterizing patterns.

Definition 14. A sequence s is a **c-general sequence** if s is frequent in all child classes of c. If c is a specialized class, then the set of general sequences in c is the set of frequent sequences in c.

A class c, defined in D_C, is associated to the set of behavioral sequences contained in c as well as to the set of c-general sequences. Consequently, a class c is now denoted by a triplet $c = \langle\mathcal{D}, \mathcal{E}, \mathcal{S}\rangle$, where:

- $\mathcal{D} = desc(c)$, is the **description** of c in D_C.
- $\mathcal{E} = data(c)$, is the **set of fragments** contained in c.
- $\mathcal{S} = seq(c)$, is the **set of c-general sequences**.

By using the previous definitions, the set of general sequences for each class in the class lattice is constructed in the following manner:

1. Let c be a specialized class. $seq(c)$ is obtained by extracting all frequent sequences in $data(c)$ (see Section 3.1).
2. Let c be a non-specialized class. $seq(c)$ is defined as follows:

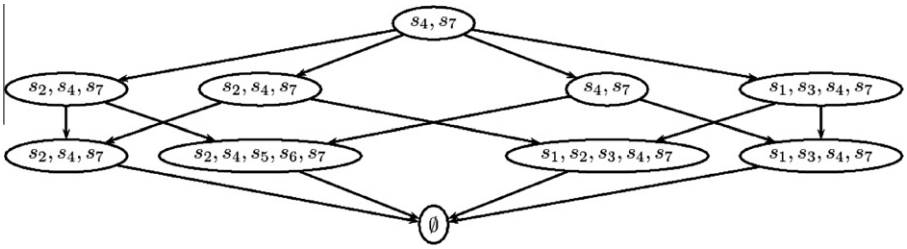

Fig. 4. Sequences spreading in the contextual lattice.

$$seq(c) = \bigcap_{c_i \in \tilde{c}} seq(c_i).$$

Thus, by extracting frequent sequences in specialized classes only, we construct all sets of general sequences. Fig. 4 is depicting the construction of general sequences in a contextual lattice, by following the previously described method.

This method of construction of all general sequences for each class has the advantage of limiting the application of the algorithm for discovering sequential patterns. Indeed, the extraction is performed only on specialized classes, and the remaining sets of general sequences are deducted from them.

We have presented in this section the approach for extracting knowledge from historical behavioral data collected by sensors. In particular, we have seen how to exploit contextual data classes, and thereby obtained results are not associated with all historical data, but with contextual criteria. Therefore, we can precisely describe the impact of these criteria on the normal behavior of a train and answer questions such as:

- What are the behaviors that appear only when the outside temperature is high?
- What behaviors are specific to a given itinerary?
- What behaviors are specific to the ending fragments?
- What are the most general behaviors, which do not depend on the context?

This accuracy greatly improves the lack of knowledge about the normal train behavior. The obtained knowledge can be used besides to detect anomalies in the data collected, as we will see in the next section.

4. Anomaly detection

In this section, we present how anomaly detection is performed. We consider that we are provided with both one database containing normal behavior on which knowledge have been extracted (see Section 3) and data corresponding to one new journey.

During a journey, each component (and each sensor) has a behavior which is related to the behavior of other components. These interactions, both temporal and inter-components, are described by a set of sequential patterns that we have extracted earlier (see Section 3). Therefore, we want to make an evaluation of the behavior of a component based on obtained knowledge. The main idea is relying on the following remarks:

- we can consider that a sensor behaves normally if we find enough patterns in our knowledge base that validate its current state,
- we can say that its behavior is anomalous if we find enough patterns that contradict it,
- if there are not enough patterns to validate or contradict the current sensor state, then the behavior is considered uncertain because we do not have sufficient knowledge to assess its current behavior.

Thus, for each reading and each sensor, we compute two scores: a concordance score and a discordance score. Depending on the value of these scores, we can then indicate whether the behavior is normal, anomalous, or uncertain. Below, we describe how the various needed scores are calculated.

4.1. Preliminary definitions

In order to use the extracted sequential patterns for measuring the compliance of a sequence describing a new journey, first of all we introduce the concept of covering sequences.

Definition 15. Let I be an itemset and $s = \langle I_1, \ldots, I_i, \ldots, I_n \rangle$ a sequence. I is **covering** s if $\forall I_i \in s$, $I \subseteq I_i$.

Example 12. The itemset (A_{low}) is covering the sequence $\langle (A_{low} B_{low})(A_{low}) \rangle$.

Definition 16. Let $p = \langle I_1^* \ldots I_i^* \ldots I_l^* \rangle$ be an aggregated pattern and $s = \langle I_1, \ldots, I_j, \ldots, I_m \rangle$ a sequence. p is **covering** s, denoted by $p \prec s$, if there exists a set of sequences $\{s_1, s_2, \ldots, s_m\}$ such that:

(i) $s = s_1 + s_2 + \cdots + s_m$,
(ii) $\forall i | 1 \leqslant i \leqslant m$, I_i^* is covering I_i. Moreover, I_i^* is called the corresponding itemset of I_i in p.

Example 13. Let s be a sequence, and p an aggregated pattern, such that:

$$s = \langle \overbrace{(A_{low})(A_{low}B_{low})}^{s_1} \overbrace{(A_{avg}B_{avg})(A_{avg}B_{avg})}^{s_2} \overbrace{(B_{high})}^{s_3} \rangle$$
$$p = \langle \underbrace{(A_{low})}_{I_1} \underbrace{(A_{avg}B_{avg})}_{I_2} \underbrace{(B_{high})}_{I_3} \rangle.$$

We can note that s can be broken down into 3 sequences s_1, s_2 and s_3, such that $I_1 \sqsubseteq s_1$, $I_2 \sqsubseteq s_2$, and $I_3 \sqsubseteq s_3$. Thus, $p \prec s$.

On the other hand, p is not covering the sequence

$$s' = \langle (A_{low})(A_{low}B_{low})(A_{avg})(A_{avg}B_{avg})(A_{avg}B_{avg})(B_{high}) \rangle.$$

Using the notion of covering sequence, we can now describe two types of patterns: (1) concordant patterns, validating the behavior of a sensor at a given time, and (2) discordant patterns contradicting this behavior.

Definition 17. Let $A \in \mathbf{\Omega}$, $s = \langle I_1, I_2, \ldots, I_n \rangle$ a sequence, and $p = \langle I_1^* I_2^* \ldots I_m^* \rangle$ an aggregated pattern. p is a concordant pattern for A in the ith itemset of s, i.e., a (A,i)-**concordant pattern** in s, if:

(i) there exist integers h, j such that $1 \leqslant h \leqslant i \leqslant j \leqslant n$, and $p \prec \langle I_h, \ldots, I_i, \ldots I_j \rangle$.
(ii) let I^* be the corresponding itemset of I_i in p, there exists an item $A_v \in I^*$.

Table 5
Concordant patterns.

ID	Aggregated pattern	Support
p_1	$\langle(\mathbf{A_{low}})(A_{avg}B_{avg})\rangle$	25%
p_2	$\langle(\mathbf{A_{low}}B_{avg})\rangle$	70%
p_3	$\langle(\mathbf{A_{low}}B_{avg})(A_{avg})\rangle$	30%
p_4	$\langle(A_{low}B_{low})(\mathbf{A_{low}}B_{avg})\rangle$	55%

Table 6
Discordant patterns.

ID	Aggregated pattern	Support
p_5	$\langle(A_{low}B_{low})(A_{avg}B_{avg})\rangle$	45%
p_6	$\langle(A_{high}B_{avg})\rangle$	20%

Example 14. Let A, B and C be sensors, $p_1 = \langle(A_{avg}B_{low})(B_{avg})\rangle$, $p_2 = \langle(A_{avg})\rangle$ and $p_3 = \langle(A_{low})(A_{avg})\rangle$ three aggregated patterns, and s a sequence such that:

$$s = \langle(A_{avg}B_{low}C_{low})(A_{high}B_{avg}C_{avg})(A_{high}B_{high}C_{high})\rangle.$$

The aggregated patterns p_1 and p_2 are $(A,1)$-*concordant patterns*. On the other hand, p_3 is not a $(A,1)$-*concordant pattern*.

A discordant pattern for the state of a sensor at a given time describes an unexpected behavior.

Definition 18. Let $A \in \mathbf{\Omega}$, $s = \langle I_1,\ldots,I_i,\ldots I_n\rangle$ a sequence such that I_i contains an *A-item*, denoted by i_s, and $p = \langle I_1^* I_2^* \ldots I_j^* \ldots I_m^*\rangle$ an aggregated pattern such that I_j^* contains an A-*item*, denoted by i_p.

p' is the sequence p where i_p has been replaced by i_s in I_j.

p is a discordant pattern for A in the ith itemset of s, i.e., a (A,i)-*discordant pattern* in s if:

(i) p is not a (A,i)-*concordant pattern*,
(ii) p' is a (A,i)-*concordant pattern*.

The items i_s and i_p are called **discordant items**. More precisely, i_s is the discordant item of s, and i_p is the discordant item of p.

Example 15. Let A, B and C be sensors, $p_1 = \langle(A_{low}B_{low})(B_{avg})\rangle$ and $p_2 = \langle(A_{high}B_{avg})\rangle$ two aggregated patterns, and s a sequence such that:

$$s = \langle(A_{avg}B_{low}C_{low})(A_{high}B_{avg}C_{avg})(A_{high}B_{high}C_{high})\rangle.$$

The aggregated pattern p_1 is a $(A,1)$-discordant pattern. On the other hand, p_2 is not a $(A,1)$-discordant pattern.

4.2. Conformity score

In the following examples, we will consider the sequence s, such that:

$$s = \langle(A_{low}B_{low})(A_{low}B_{avg})(\mathbf{A_{low}}B_{avg})(A_{avg}B_{avg})\rangle.$$

Table 5 (respectively Table 6), contains the set of $(A,3)$-concordant patterns (respectively $(A,3)$-discordant patterns) in s as well as their support.

4.2.1. Concordance score

Computing a concordance score for a given sensor A in the ith itemset of a given sequence s consists in distinguishing, in a set of patterns such as that described in Table 5, the (A,i)-concordant pattern in s.

Note that two concordant patterns do not always have the same weight. In Table 5, we can not give the same weight to all the patterns in the overall concordance score of sensor A in the 3th itemset of s. Indeed, the size of the pattern p_2 is too low to consider that it will affect the final score. In addition, we believe that the support of a pattern also influences its weight. Therefore, we define the weight of a sequence as follows:

Definition 19. Let p be an aggregated pattern and s a sequence such that p is an (A,i)-concordant pattern in s. The **weight** of such a concordant pattern is defined as follows:

$$weight(p) = \|p\| \times support(p).$$

Example 16. The weight of p_1 is:

$$weight(p_1) = \|p_1\| \times support(p_1) = 3 \times 0.25 = 0.75.$$

We can now define the concordance score of a sensor A in the ith itemset of a sequence s.

Definition 20. Let $\mathcal{P}^c$ the set of all *(A,i)-concordant patterns*. The **concordance score** of a sensor A in the ith itemset of a sequence s is defined as follows:

$$score_{conc}(A,i) = \sum_{p\in\mathcal{P}^c} weight(p).$$

Example 17. The concordance score of A in the 3rd itemset of s is:

$$score_{conc}(A,i) = \sum_{p\in\mathcal{P}^c} weight(p) = weight(p_1) + \ldots + weight(p_4)$$
$$= 0.75 + 1.4 + 0.9 + 2.2 = 5,25.$$

4.2.2. Discordance score

In this part, we focus on discordant patterns, which tend to invalidate the behavior of a sensor A in the ith itemset of a sequence.

Discordance degree To evaluate the weight of a discordant pattern, it is important to consider the gap between this value and the "expected value".

Let $A \in \mathbf{\Omega}$ a sensor, and $\mathcal{D} = dom(A)$. $\mathcal{D}$ is such that $\mathcal{D} = (v_1,\ldots v_i,\ldots,v_n)$, where v_i is a discrete value.

Definition 21. Let p be a (A,i)-discordant pattern in a sequence s, i_p the discordant item of p, and i_s the discordant item of s. We define the **discordance degree** of p as follows. Let us consider $v_k \in \mathcal{D}$ and $v_l \in \mathcal{D}$, such that $v_k = value(i_p)$ and $v_l = value(i_s)$. The discordance degree of p, denoted by $discDegree(p)$, is:

$$discDegree(p) = \frac{|l-k|}{n}.$$

Example 18. In the previous sequence s, $dom(A) = dom(B) = (i_1,i_2,i_3)$, such that $i_1 = low$, $i_2 = avg$, and $i_3 = high$. By considering the sequence s and the discordant pattern p_5, we can note that the discordant item of s is A_{low} and the discordant item of p_5 is A_{avg}. Thus, the discordance degree of p_5 is:

$$discDegree(p_5) = \frac{|2-1|}{|dom(A)|} = 1/3.$$

We can now define the weight of a discordant pattern. This weight must take into account three features: the discordance degree, the size of a pattern, and its support. Concerning the size of a discordant pattern, we believe that the important part of a discordant pattern is the part covering the tested sequence. Therefore, we consider the size of the sequence s without the discordant item.

Definition 22. Let p be an aggregated pattern and s a sequence such that p is an (A,i)-discordant pattern. The **weight** of such a discordant pattern is defined as follows:

$$weight(p) = (\|p\| - 1) \times support(p) \times discDegree(p).$$

Example 19. The weight of p_5 is:

$$weight(p_5) = (\|p_5\| - 1) \times support(p_5) \times discDegree(p_5) = (4-1) \times 0.45 \times \frac{1}{3} = 0.45.$$

Definition 23. Let $\mathcal{P}^d$ the set of all (A,i)-*discordant patterns*. The **discordance score** of a sensor A in the ith itemset of a sequence s is:

$$score_{disc}(A,i) = \sum_{p \in \mathcal{P}^d} weight(p).$$

Example 20. The discordance score of A in the 3rd itemset of s is calculated as follows:

$$score_{disc}(A,3) = \sum_{p \in \mathcal{P}^d} weight(p) = weight(p_5) + weight(p_6) = 0.45 + 0.13 = 0.58.$$

4.2.3. Conformity score

For each sensor and each itemset in a sequence, we defined how to calculate a concordance score and a discordance score. We can now address the problem of defining a global *conformity score* of a sensor A in the ith itemset of a sequence, denoted as *score* (A,i). This score, defined between -1 and 1, must meet the following requirements:

- if $score(c,t)$ is close to 1, the state of A in i is considered as normal,
- if $score(c,t)$ is close to -1, the state of A in i is considered as abnormal,
- if $score(c,t)$ is close to 0, the state of A in i is considered as uncertain.

Definition 24. Let $A \in \mathbf{\Omega}$ be a sensor and s a sequence. The **conformity score** of A in the ith itemset of s, denoted by $score(A,i)$ is defined as follows:

$$score(A,i) = \frac{score_{conc}(A,i) - score_{disc}(A,i)}{max(score_{conc}(A,i), score_{disc}(A,i))}.$$

Example 21. By considering the previous examples, we can now calculate the **conformity score** of A in the 3rd itemset of s as follows:

$$score(A,3) = \frac{score_{conc}(A,3) - score_{disc}(A,3)}{max(score_{conc}(A,3), score_{disc}(A,3))} = \frac{5,25 - 0.58}{5,25} = 0.89.$$

4.2.4. Smoothing window

One important issue when addressing the problem of detecting anomalies concerns false alarms, especially when data are noisy. Thus, it is possible that a sensor has a bad score on a reading, which do not correspond to a real problem. In this case, the score of the sensor then quickly goes back to normal values. To avoid this, it is preferable to take into account the surrounding score values of the same sensor.

We have thus developed the possibity to set a parameter called *smoothing window* and noted w. The score of a sensor for a given date is the average score on the smoothing window.

Fig. 5 shows the evolution of the score of a sensor during a fragment, with and without smoothing (with $w = 3$). Without smoothing, we can see a decrease of the value that could be interpreted as an anomalous behavior. However, the immediate increasing of the score indicates that it is not a real anomaly. The smoothed score is an efficient method for restricting this phenomenon.

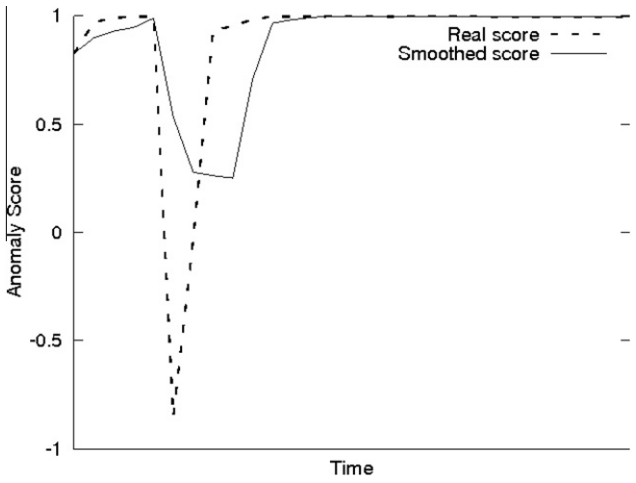

Fig. 5. Influence of the smoothing window on the anomaly score.

We have described throughout this section how to use a list of sequential patterns, illustrating the normal behavior of the monitored system, to detect abnormal behavior among new collected data. This approach is particularly suited to the problem of monitoring complex industrial systems because it does not only declare a data sequence as normal or abnormal, but provides also very precise information about the anomaly detected: the localization of the fault (which sensor, which component, etc.), the precise moment when this anomaly occurred, and a score quantifying the seriousness of the detected problem.

Moreover, the described approach can also obtain information about the nature of the detected anomaly. We can in particular distinguish the case where the detected anomaly corresponds to a sensor failure (which means that the behavior of the component on which the sensor is located is normal, but the measurements are corrupted) from the more serious case where the behavior of the component is actually involved in the anomaly. To differentiate those cases, we exploit the fact that several sensors are generally installed on a same component (e.g., 4 temperature sensors are installed on each of the wheels of a train). We can therefore infer that if only one sensor from a group of sensors has a low score, then the corresponding anomaly relates to that sensor only. We can therefore analyze the data according to several levels of granularity: the sensor level (by considering only the score of each sensor), the level of a component (e.g. by interpreting the average score of sensors installed on it), the level a car (by calculating the average score of components in the car), etc.

5. Experimental results

In order to evaluate our proposal, several experiments were conducted on a real dataset. They correspond to the railway data collected on 12 trains where each train has 249 sensors, over a one-year period. A reading is collected every five minutes. 232 temperature sensors and 16 acceleration sensors are distributed on the different components (e.g., wheels, motors, etc.) and a sensor measures the overall speed of the train.

The experimental protocol follows the organization of the proposals:

1. **Characterization of normal behavior** (see Section 3). We have studied the impact of contextualization on the characterization of normal behavior, and the benefit of the search for specific patterns in various contexts.

2. **Anomaly detection** (see Section 4). We have evaluated the *conformity score* by applying it on normal and abnormal data.

5.1. Normal behavior characterization

The discovery of frequent sequences has been performed with the PSP algorithm described in Masseglia, Cathala, and Poncelet (1998) with the minimum support set to 0.3.

To build a contextual lattice such as that presented in Section 3.2, we used four criteria:

- the fragment type (denoted as FT): *beginning*, *middle*, or *ending* fragments.
- the exterior temperature (denoted as T): *low* or *high*.
- The itinerary (denoted as I): I_1, I_2, I_3.
- The episode in an itinerary (denoted as E): 1st, 2nd, 3rd, or 4th episode.

Table 7 shows, for some classes extracted from the obtained hierarchy, the respective number of general sequences, as well as the number of aggregated patterns among them. For example, the class of *middle* fragments which have been achieved with a *low* exterior temperature for any itinerary and any episode, denoted as [T = *low*; FT = *middle*; I = ∗; E = ∗], contains 123 general sequences. 89 (72%) among them are aggregated patterns.

We can note that the more a class is general, the less it contains general sequences. This shows the necessity of contextualizing the behavior characterization. Indeed, very few behaviors are common to all contexts: a normal behavior is actually dependent on the context in which it operates. Moreover, aggregated patterns have reduced the number of extracted sequential patterns, by removing all very long and redundant sequences.

5.2. Anomaly detection

We described in Section 4 the approach used to detect anomalies in new data. Thus, we can use it to classify new fragments of journeys into two categories: normal data and abnormal data. To evaluate our approach in an appropriate manner, it is necessary to conduct experiments on both types of data. However, if it is easy to find data which do not contain faults, it is often more difficult to obtain a large data set containing anomalies. For this reason, we have simulated a set of anomalous data, on which we have conducted our experiments.

To this end, we have used the real normal data set, and we have corrupted the data in order to reproduce classic behavioral anomalies on the values collected by sensors. Anomalies generated are threefold:

- **Blocked values** (see Fig. 6): the value collected by a sensor is not refreshing. This anomaly is usually related to a faulty sensor, or a problem of transmitting information between the sensor and the database, but very rarely describes a real problem on a component.
- **Shifted values** (see Fig. 7): a value that is shifted in comparison with the normal behavior of a sensor (or group of sensors): a constant value is added (or substracted) to the real collected value. This type of anomaly may, for example, describe an abnormal overheating of a component.
- **Random values** (see Fig. 8): in this case, collected values are randomized. They describe an aberrant behavior without any link with the expected behavior.

We simulated these anomalies in order to build a data set containing about 600 fragments of episodes (i.e., 200 episodes) composed as follows:

Table 7
Number of general sequences and aggregated patterns, according to the contextual class.

Class	General sequences	Aggregated patterns
[T = ∗; FT = ∗; I = ∗; E = ∗]	5	5
[T = *low*; FT = ∗; I = ∗; E = ∗]	20	17
[T = *low*; FT = *middle*; I = ∗; E = ∗]	123	89
[T = *low*; FT = *middle*; I = 1; E = ∗]	270	189
[T = *low*; FT = *middle*; I = 1; E = 2]	542	455

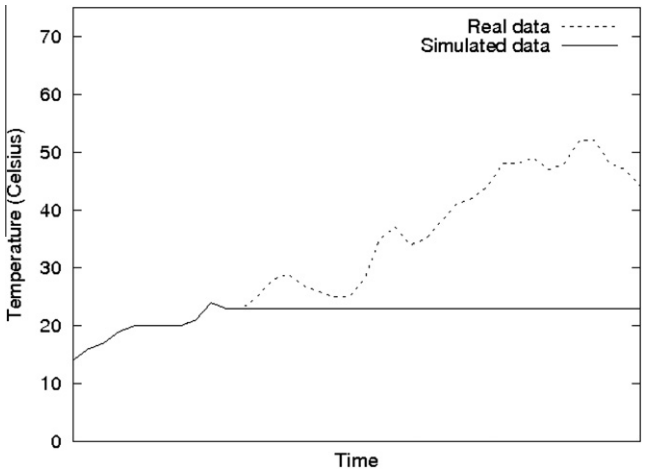

Fig. 6. Simulated anomaly: blocked values.

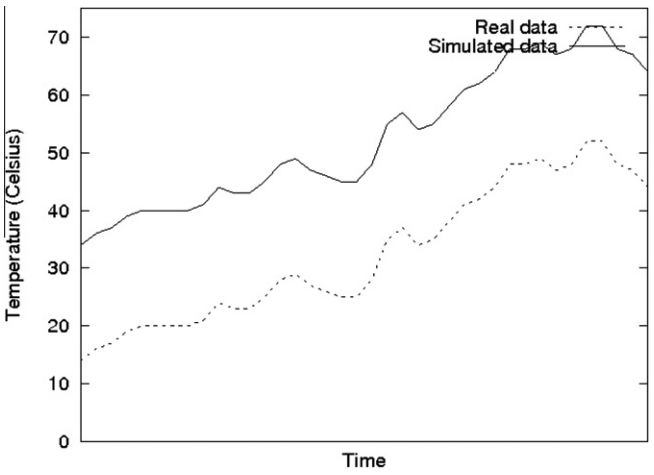

Fig. 7. Simulated anomaly: shifted values.

- 300 are fragments of real data validated as normal (i.e., without anomalies),
- 300 fragments containing anomalies were generated from normal data. The three types of anomalies described above are distributed equally between these fragments.

The approach was tested on the basis of cross-validation, by segmenting the total set of data into 10 equal parts. Thus, each part contains 600 fragments, of which 300 are normal data. Note that the learning step (i.e., characterization of normal behavior) is performed on normal data only. To quantify the number of anomalous fragments in our data set, we consider a fragment is abnormal if its anomaly score falls below −0.5.

Thus, we have obtained the confusion matrices presented in Table 8, containing the average results obtained by cross validation

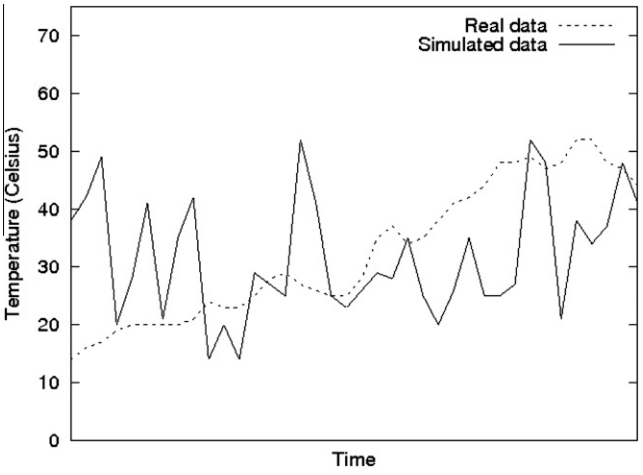

Fig. 8. Simulated anomaly: random values.

Table 8
Global confusion matrix.

	Predict. normal	Predict. anomalous	Recall	Precision
Real normal	272	28	90.67%	91.58%
Real anomalous	25	275	91.67%	90.76%
Global results	297	303	**91.17%**	**91.17%**

Table 9
Confusion matrix for random values anomalies.

	Predict. anomalous	Predict. normal
Real anomalous	98	2

Table 10
Confusion matrix for blocked values anomalies.

	Predict. anomalous	Predict. normal
Real anomalous	85	15

Table 11
Confusion matrix for shifted values anomalies.

	Predict. anomalous	Predict. normal
Real anomalous	92	8

on the entire test set. We can note that the system is well balanced and has similar qualities to both recognize the normal behaviors, but also to identify anomalies.

In order to evaluate the results, we have calculated two measures of quality widely used: the precision and the recall. Moreover, recall and precision is in all cases above 90%, so the approach limit both the number of false alarms and the number of undetected anomalies.

Tables 9–11 contain the results for each type of anomaly. We note that the results are very good to recognize a "*random values*" or a "*shifted values*" anomaly. This is due to the fact that in this case the simulated behavior is very different from a normal behavior. In contrast, "*blocked values*" anomalies are different: the anomalous value may not be sufficiently distant from "expected" values for detecting the problem.

5.2.1. Additional information about detected anomalies

We have seen how to detect anomalies in new collected data. Moreover, it is possible to obtain additional information about detected anomalies. In particular, we use the lattice of classes to better understand the anomalies.

When an anomaly is detected in a specialized class it may be useful, for users taking maintenance decisions, to better understand anomalies sources in order to assess its seriousness. In particular, the user can recalculate the conformity score in other classes of the contextual lattice, and particularly in the superclasses of its specialized class. Several cases are then possible.

- The score is low in the more specific classes of the lattice, but is improved in the more general classes. In this case, the tested journey is anomalous, but the anomaly is not serious. It does not meet the most specific behaviors expected in its class, but nonetheless meets the most general behaviors.
- The score is low in all classes of the lattice, including the general classes. Here, the observed journey is problematic. The behavior of the train does not correspond to what is expected.
- The score is low in a branch of the lattice, but still good in another branch. For example, consider a journey that contains an anomaly in its specialized class [*high*,*low*] (i.e., high exterior temperature and low humidity rate). So it is particularly helpful for decision makers to know the scores obtained by this journey in the superclasses [*high*,∗] or [∗*low*]. If the anomaly is related to one of these classes, i.e., if the score is low in [*high*,∗] and correct in [∗,*low*] for example, then the user gets additional information, which will allow him to investigate more effectively the cause of the anomaly.

This information, combined with the fact that user may at any time consult the list of concordant or discordant patterns when an anomaly is detected, can provide a better understanding of the trains behavior. Therefore, the approach not only detects problems but also allows decision makers to investigate the causes and potential seriousness of an anomaly, and help them to make the right decisions.

6. Related work

The needs to develop our approach are twofold: (i) extract knowledge from sensor data, and (ii) detect anomalies in such data. Recently, the problem of mining sensor data has been addressed by the data mining community. Different approaches focusing either on the data representation by performing for example sensor clustering Ci, Guizani, and Sharif (2007), Rodrigues and Gama (2006), or knowledge extraction by mining association rules Boukerche and Samarah (2007), Boukerche and Samarah (2008), Chong et al. (2008), Ma et al. (2004), Mihail Halatchev (2005), Yairi, Kato, and Hori (2001), or sequential patterns Cook et al. (2003), Guralnik and Haigh (2002), Tseng and Lu (2009), Wu, Peng, and Chen (2001) were proposed.

The second problem we address in this paper concerns the detection of anomalous behavior from data collected by the sensors. This topic, usually known as the anomaly detection problem, has received considerable attention in recent years. It can be viewed as a classification problem in which a system behavior can be classified as normal or anomalous. Defined in Grubbs (1969), an anomaly or outlier is an observation that appears to deviate markedly from other members of the sample in which it occurs. More recently, different definitions have been proposed in literature (Barnett & Lewis, 1994; Douglas M., 1980; Ramaswamy, Rastogi, & Shim, 2000). More generally, we will consider an anomaly as

behaving differently from the norm. A lot of approaches have been developed in order to meet the requirements of very different application domains, such as network intrusion detection (Jayakumar et al., 2008; Siaterlis & Maglaris, 2004; Wang, Guan, & Zhang, 2008; Wu and, 2006), industrial systems monitoring (Keogh, Lin, Lee, & Van Herle, 2006; Yairi et al., 2001), medical condition monitoring (Lin, Keogh, Fu, & Van Herle, 2005; Wong, Moore, Cooper, & Wagner, 2003), fraud detection (Brause, Langsdorf, & Hepp, 1999, Phua, Alahakoon, & Lee, 2004), etc.

Of course, the techniques employed are strongly related to the nature of input data. In particular, numerous anomaly detection methods have been developed considering transaction data sets. However, a lot of real-world domains have to handle temporal data that can be stored as sequential data sets. In particular, the behavioral data collected by sensors in the complex systems monitoring problem belongs to this category. The temporal nature of sequential data requires new techniques to detect anomalies within. Several methods for detecting anomalies in sequential data have been developed for different application domains (Lee, Stolfo, & Chan, 1997; Michael & Ghosh, 2000; Sun, Chawla, & Arunasalam, 2006), but they cannot meet all our requirements. We have previously seen that an anomaly behave differently from the norm. Consequently, in order to properly detect anomalies in a data set, we need to define what is the norm, i.e., to characterize normal behavior. Some application domains can benefit from *a priori* knowledge of experts to construct this characterization. However, in other domains such as train monitoring for example, *a priori* knowledge is difficult to obtain. In consequence, we have to develop a system for exploiting the extracted results to characterize normal behavior. These patterns are describing the expected norm.

We also have to manage several difficulties inherent to the problem that we address. For example, developing a hierarchy of contexts is necessary to (i) take into account variations in the train behavior according to various criteria, and (ii) provide more explanations to maintenance experts when an anomaly is detected. In addition, we handle sequences particularly large and redundant (see Section 2). Therefore, we propose the concept of aggregated sequences to accomplish our goals.

Moreover, we must attach importance to the interpretability of results in the normal behavior characterization process as well as in the anomaly detection. Indeed, the problem is very sensitive (errors can lead to heavy financial costs), the experts need to understand the results to make and justify the right decisions. For all these reasons, we must develop a new appropriate method.

7. Conclusion

In this paper, we have addressed a problem involved in many areas: the maintenance of complex systems. There are several solutions to perform maintenance of such systems. Firstly, corrective maintenance, consisting in making the necessary maintenance operations after the occurrence of a failure, is not suited to this contexte as too costly and too dangerous. A planned and systematic maintenance is too expensive, although it can usually avoid serious failures. Thus, we addressed the problem of developing an approach to allow preventive maintenance, which is a good compromise between the two previous solutions. Preventive maintenance consists in detecting abnormal behavior that may be harbingers of major failures, to perform only the necessary preventive maintenance operations.

However, the complexity of such systems (e.g., trains, industrial machinery, etc.) makes their behavior difficult to understand and interpret. In these circumstances it is particularly difficult to detect abnormal behavior that often herald significant and costly failures.

The problem that we address is particularly challenging. Indeed, errors in diagnosis may cause many inconveniences. We have therefore developed an approach to use data collected by sensors in order to analyze behaviors and allow the detection of anomalies. Our contribution is divided into three parts. First, we have proposed an adequate representation of data in order to extract knowledge based on sensor data describing the past normal behavior of systems. Secondly, we studied the possibility to extract sequential patterns in historical data both to improve the understanding of systems for experts, but also to provide a knowledge database used to develop a method for detecting anomalies. To characterize normal behavior adequately, we also took into account the context. Indeed, the context in which a system is running often has an influence on its behavior, and on the definition of an anomaly. Finally, we proposed an approach to compare new patterns with all sequential patterns describing normal behavior. We provide experts with an anomaly score for each sensor and each component of the systems studied. The approach allows to locate precisely the anomalies and to quantify the extent to which the anomaly seems problematic.

One main advantage of the presented approach is that all obtained results are easily interpretable for decision makers. This is particularly important because users must be able to make decisions with certainty.

Although the presented work meets our expectations in terms of results, it opens interesting perspectives. In particular, the development of a graphical user interface will allow users to access results quickly and efficiently. Another important aspect may be to adapt the approach to a real-time context. This will detect the anomalies on the running trains. Furthermore, an interesting question may be addressed: how to manage the evolution of normal behavior over time? This will conduct the knowledge database to be incrementally updated in order to ensure handled knowledge to be valid over time.

References

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD Record, 22*(2).

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In P. S. Yu & A. S. P. Chen (Eds.), *Eleventh International Conference on Data Engineering*. IEEE Computer Society Press.

Barnett, V., & Lewis, T. (1994). *Outliers in statistical data*. *Wiley series in probability & statistics*. Wiley. April.

Boukerche, A., & Samarah, S. (2007). A new in-network data reduction mechanism to gather data for mining wireless sensor networks. In *MSWiM '07: Proceedings of the 10th ACM symposium on modeling, analysis, and simulation of wireless and mobile systems*. New York, NY, USA: ACM.

Boukerche, A., & Samarah, S. (2008). A novel algorithm for mining association rules in wireless ad hoc sensor networks. *IEEE Transactions on Parallel Distributed Systems, 19*(7), 865–877.

Brause, R., Langsdorf, T., & Hepp, M. (1999). Neural data mining for credit card fraud detection. In *ICTAI '99: Proceedings of the 11th IEEE international conference on tools with artificial intelligence* (pp. 103). Washington, DC, USA: IEEE Computer Society.

Carrascal, A., Díez, A., & Azpeitia, A. (2009). Unsupervised methods for anomalies detection through intelligent monitoring systems. In *Proceedings of the fourth international conference on hybrid artificial intelligence systems* (pp. 144). Springer.

Chong, S. K., Krishnaswamy, S., Loke, S. W., & Gaben, M. M. (2008). Using association rules for energy conservation in wireless sensor networks. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*. ACM.

Ci, S., Guizani, M., & Sharif, H. (2007). Adaptive clustering in wireless sensor networks by mining sensor energy data. *Computer Communications, 30*(14–15), 2968–2975.

Cook, D. J., Youngblood, M., Heierman, E. O., III, Gopalratnam, K., Rao, S., Litvin, A., et al. (2003). Mavhome: An agent-based smart home. In *PERCOM '03: Proceedings of the first IEEE international conference on pervasive computing and communications*. Springer.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics, 11*, 1–21.

Gruenwald, L., & Halatchev, M. (2005). Estimating missing values in related sensor data streams. In J. R. Haritsa & T. M. Vijayaraman (Eds.), *Proceedings of the 11th international conference on management of data (COMAD '05)*. Computer Society of India.

Guralnik, V., & Haigh, K. Z. (2002). *Learning models of human behaviour with sequential patterns. In Proceedings of the AAAI-02 workshop "Automation as Caregiver"*.

Hawkins Douglas, M. (1980). *Identification of outliers*. Chapman and Hall.

Keogh, E., Lin, J., Lee, S.-H., & Van Herle, H. (2006). Finding the most unusual time series subsequence: algorithms and applications. *Knowledge Information Systems, 11*(1), 1–27.

Lee, W., Stolfo, S. J., & Chan, P. K. (1997). Learning patterns from unix process execution traces for intrusion detection. In *In AAAI workshop on AI approaches to fraud detection and risk management* (pp. 50–56). AAAI Press.

Lin, J., Keogh, E., Fu, A., & Van Herle, H. (2005). Approximations to magic: Finding unusual medical time series. In *CBMS '05: Proceedings of the 18th IEEE symposium on computer-based medical systems* (pp. 329–334). Washington, DC, USA: IEEE Computer Society.

Masseglia, F., Cathala, F., & Poncelet, P. (1998). The psp approach for mining sequential patterns. In J. M. Zytkow & M. Quafafou (Eds.), *PKDD*. *Lecture Notes in Computer Science* (Vol. 1510). Springer.

Ma, X., Yang, D., Tang, S., Luo, Q., Li, S., & Zhang, D. (2004). Online mining in sensor networks. In H. Jin, G. R. Gao, Z. Xu, & H. Chen (Eds.), *NPC*. *Lecture Notes in Computer Science* (Vol. 3222). Springer.

Michael, C. C., & Ghosh, A. (2000). Two state-based approaches to program-based anomaly detection. In *ACSAC '00: Proceedings of the 16th annual computer security applications conference* (pp. 21). Washington, DC, USA: IEEE Computer Society.

Phua, C., Alahakoon, D., & Lee, V. (2004). Minority report in fraud detection: Classification of skewed data. *SIGKDD Explorations Newsletter, 6*(1), 50–59.

Ramaswamy, S., Rastogi, R., & Shim, K. Efficient algorithms for mining outliers from large data sets. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (pp. 427–438). New York, USA: ACM.

Rodrigues, P. P., & Gama, J. (2006). Online prediction of streaming sensor data. In: Jesus, S., Gama, A. -R. J., & Roure, J., (Eds.), *Proceedings of the 3rd international workshop on knowledge discovery from data streams (IWKDDS 2006), in conjuntion with the 23rd international conference on machine learning*.

Shaikh, R. A., Jameel, H., d'Auriol, B. J., Lee, S., Song, Y.-J., & Lee, H. (2008). Trusting anomaly and intrusion claims for cooperative distributed intrusion detection schemes of wireless sensor networks. In *ICYCS '08: Proceedings of the 2008 the 9th international conference for young computer scientists* (pp. 2038–2043). Washington, DC, USA: IEEE Computer Society.

Siaterlis, C., & Maglaris, B. (2004). Towards multisensor data fusion for dos detection. In *SAC '04: Proceedings of the 2004 ACM symposium on applied computing* (pp. 439–446). New York, NY, USA: ACM.

Sun, P., Chawla, S., & Arunasalam, B. (2006). Mining for outliers in sequential databases. In D. Lambert, J. Ghosh, J. Srivastava, & D. B. Skillicorn (Eds.), *SDM*. SIAM.

Tseng, V. S., & Lu, E. H.-C. (2009). Energy-efficient real-time object tracking in multi-level sensor networks by mining and predicting movement patterns. *Journal of Systems and Software, 82*(4), 697–706.

Wang, W., Guan, X., & Zhang, X. (2008). Processing of massive audit data streams for real-time anomaly intrusion detection. *Computer Communications, 31*(1), 58–72.

Wong, W.-K., Moore, A., Cooper, G., & Wagner, M. (2003). Bayesian network anomaly pattern detection for disease outbreaks. In *Proceedings of the twentieth international conference on machine learning* (pp. 808–815). AAAI Press.

Wu, P.-H., Peng, W.-C., & Chen, M.-S. (2001). Mining sequential alarm patterns in a telecommunication database. In *DBTel '01: Proceedings of the VLDB 2001 international workshop on databases in telecommunications II*. Springer-Verlag.

Wu, N., & Zhang, J. (2006). Factor-analysis based anomaly detection and clustering. *Decision Support Systems, 42*(1), 375–389.

Yairi, T., Kato, Y., Hori, K. (2001). Fault detection by mining association rules from house-keeping data. In *Proceedings of the sixth international symposium on artificial intelligence, robotics and automation in space*.