Proceedings of the ASME 2017 International Design Engineering Technical Conferences & Computers and Information in Engineering IDETC 2017 August 6-9, 2017, Ohio, USA

DETC2017-67910

PATTERN MINING FOR LEARNING TYPICAL TURBINE RESPONSE DURING DYNAMIC WIND TURBINE EVENTS

Len Feremans
Department of Maths & CS
Universiteit Antwerpen
Email: len.feremans@uantwerpen.be

Boris Cule
Department of Maths & CS
Universiteit Antwerpen
Email: boris.cule@uantwerpen.be

Christof Devriendt
Department of Applied Mechanics
Vrije Universiteit Brussel
Email: christof.devriendt@vub.ac.be

Bart Goethals
Department of Maths & CS
Universiteit Antwerpen
Email: bart.goethals@uantwerpen.be

Jan Helsen
Department of Applied Mechanics
Vrije Universiteit Brussel
Email: jahelsen@vub.ac.be

ABSTRACT

Maintenance costs are a main cost driver for offshore wind energy. Prediction of failure and particularly failure understanding can help to bring these costs down significantly. Since the wind turbine is subjected to a large number of dynamic events it is important to fully understand the turbine response to these events. Pattern mining has been used successfully for different applications. We believe it to have large potential for understanding turbine behavior based on turbine status logs. These logs record all turbine actions and can be used as input for pattern mining algorithms. This paper proposes the use of a multi-level pattern mining approach in order to minimize the number of uninteresting patterns and facilitate response understanding. The paper mainly focuses on the extraction of patterns and association rules linked to certain alarms and how they can be annotated for further use in the multi-level pattern mining approach. Several years of wind turbine data is used. The use of the approach is illustrated by detecting the characteristic pattern linked to turbine response to an Extremely High Wind Speed Alert.

1 INTRODUCTION

Wind energy is one of the main renewable energy sources used today. Since open space is limited and in order to maximize wind farm performance most European countries are stimulating wind farm developers to construct new farms offshore. From an energy production point of view this definitely brings added value. However, new logistical challenges arise. Due to bad weather conditions it can happen that turbines are inaccessible for long periods of time. Therefore, predictability of turbine failure is essential for an optimized maintenance strategy.

Dedicated vibration based condition monitoring can predict failures linked to the rotating components of the turbine. Machine learning offers increasingly popular techniques for monitoring the performance of all sorts of machines. These approaches are particularly useful for modeling and anomaly detection on operational machine parameters, such as temperature. If an anomaly is detected using one of these techniques, it is necessary to link it to the operational response of a turbine. In other words, to go beyond failure detection and towards failure prognosis and failure understanding, it is necessary to unveil the link between failure and turbine response to environmental loads. Linking failure initiation to the loads introduced by dynamic events can help in gaining a deeper understanding of the failure mode

and its potential origin. Doing so requires a way to find the underlying mechanisms of typical system responses and allows to embed domain expert knowledge about these events. This paper will investigate the potential of pattern mining for performing this task.

Pattern mining is a core discipline within Data Mining, which is often defined as an activity to extract new nontrivial information contained in large quantities of data. The simplest type of pattern is an itemset, consisting of a frequently co-occurring set of items. A large number of algorithms have been developed for finding frequent itemsets, and extended to other types of more complex patterns, such as sequences (where events are ordered) or episodes (partial orders) in sequential data [1]. Patterns can be used to derive association or prediction rules, but even on their own, unlike the output of many other techniques such as neural networks, they can also be inspected by domain experts, and be used as an intermediate tool for gaining insight into a variety of problems [2]. Pattern mining is used in a wide range of applications, even in, at first sight, less obvious domains, such as image classification [3] or bio-informatics [4]. Association rule mining was first proposed by Agrawal et.al. in 1993 [5]. The original motivation for mining association rules came from market basket analysis, where customer behavior could be described by association rules. A classical example is an association rule that tells us that customers that buy beer, in 80% of cases, also buy chips. Association rules give deeper insight into the correlation between items.

The remainder of this paper is organized as followed. In Section 2 we review existing pattern mining approaches. In Section 3 we outline the main properties of our dataset (status logs of the wind turbines) and describe the necessary pre-processing steps we applied. In Section 4 we propose our method for multi-level pattern mining, while in Section 5 we present and discuss the discovered patterns. Finally, we summarize our conclusions in Section 6.

2 PATTERN MINING APPROACHES

In this section we introduce some preliminary pattern mining terminology, by defining concepts such as frequent itemsets, transactional databases, and explaining the difference between frequent, closed and maximal itemsets. We also briefly explain association rules, and other pattern types, such as sequences and episodes. Finally we provide a brief overview of several algorithms used for discovering frequent patterns.

2.1 Frequent Itemsets

Let $I = \{i_1, i_2, ..., i_n\}$ be a set of all *items*. An *itemset* is a subset of I. In frequent itemset mining a *transaction* is defined as a unique tuple, consisting of a transaction identifier and a set of items. We define a transaction $T_i = (tid, X)$ where tid is a unique transaction identifier (i.e., row number), and $X \subseteq I$. A *transaction database* is the set of all such transactions. We define the transaction database as $D = \{(tid, X) |\ X \subseteq I, 1 \leq tid \leq |D|\}$.

For example, in the context of market basket analysis, one can think of a transaction as the set of items bought by each customer, and the transaction database as the set of all transactions at one supermarket. In the context of turbine event logs, one can consider each type of a logged event as an item, and a transaction can represent a small period (or window) in time, in which several types of events may occur.

The *support* of an itemset is defined as the number of transactions in the database in which the itemset occurs. Formally, $support(Y) = |\{(tid, X) |\ Y \subseteq X, (tid, X) \in D\}|$. An itemset is considered *frequent* if its support is larger than some pre-defined threshold min_sup. Thus the pattern mining process starts with setting the min_sup threshold parameter (defined either relative to the size of the database or in absolute terms) and running an algorithm that returns all frequent itemsets.

An additional problem with pattern mining is that the number of frequent patterns (itemsets, sequences or episodes) can grow extremely high, a problem often described as *pattern explosion*. This is a direct consequence of the definition of support, which has the property that if an itemset is frequent, then all its subsets must also be frequent. Since an itemset of size n has 2^n subsets, it is easy to see how this can lead to massive redundancy in the output. Several techniques have been proposed to reduce the number of redundant patterns. A frequent itemset is *closed*, if no superset has the same support. A frequent itemset is *maximal* if no superset is frequent. For example if itemset $\{i_1, i_2\}$ has a support of 10, itemset $\{i_1, i_2, i_3\}$ has a support of 10, and $\{i_1, i_2, i_3, i_4\}$ has a support of 9, we would filter the first itemset when mining for closed patterns, since it has the same support as the second itemset. When mining for maximal patterns, we would also filter the second itemset, and only keep the third itemset, if the minimum support is set to at least 9.

2.2 Association Rules

Mining frequent itemsets was originally developed as a first step in discovering interesting relationships between items. Mining *association rules* can be described as a second step, which attempts to quantify the strength of such relationships. An association rules is of the form $X \Rightarrow Y$, where X and Y are itemsets. For example, $\{bread, butter\} \Rightarrow \{milk\}$ with $confidence = 80\%$ tells us that 80% of customers that bought both bread and butter also bought milk. Formally, the *confidence* of an association rule is defined as the proportion of transactions that contains X which also contain Y, that is, $conf(X \Rightarrow Y) = support(X \cup Y)/support(X)$.

2.3 Richer Pattern Types: Sequences and Episodes

A *sequence* is an ordered list of itemsets. A sequence s is denoted by $\langle s_1 s_2 ... s_k \rangle$, where each of the k elements s_j is an itemset, thus $s_j \subseteq I$. For each element we write $(i_1 i_2 ... i_m)$, or if the

element consist of a single item, we omit the the brackets. For example $\langle i_1(i_2i_3)i_4i_2\rangle$ is a sequence where i_1 is followed in time by i_2 or i_3, which occur simultaneously, and are then followed by i_4 and finally i_2 again. Frequent sequence discovery aims to find all (sub)sequences that occur frequently in a database of input sequences. Note that, in this context, we consider sequence $\langle i_1i_3i_2\rangle$ to be a subsequence of $\langle i_1(i_2i_3)i_4i_2\rangle$, i.e., gaps are allowed between items making up a pattern. If gaps are not allowed, the data mining literature typically refers to frequent (sub)string mining, and not (sub)sequence discovery.

An *episode* is a further specialization of an itemset. Here we consider frequent occurrences, with a *partial* sequential order imposed on the items. For example, we might discover the frequent episode $\{i_1,i_2\} \rightarrow i_3$, meaning i_3 occurs both after i_1 and i_2, but the order between i_1 and i_2 is unspecified (this could mean that i_1 sometimes occurs before i_2, sometimes after i_2, or sometimes even simultaneously). A corresponding itemset $\{i_1,i_2,i_3\}$ does not capture any order, while frequent sequences $\langle i_1i_3\rangle$ and $\langle i_2i_3\rangle$ would not be able to represent the fact that all three items occur together. Both sequential and episodal patterns have similar definitions for support, and allow for the mining of both closed and maximal patterns.

In the remainder of this paper, we will focus on *closed itemset mining*, as this limits the search-space considerably. However, in our turbine event log, some itemsets might be better represented using sequences or episodes.

2.4 Frequent Pattern Algorithms

Frequent Itemset Mining (FIM) algorithms discover all itemsets that occur in at least *min_sup* transactions in the database. This is *not* a trivial computation, since a brute-force algorithm that would enumerate all possible subsets, and then check how many times these subsets occur in the database, would result in enumerating $2^{|I|}$ subsets, which is clearly infeasible.

Most FIM algorithms therefore exploit the *anti-monotonic* property of itemset support, in the design of a *branch-and-bound* algorithm. We will briefly explain the intuition behind the anti-monotonic property by an illustrative example.

Assume item a occurs in 10 transactions, item b occurs 5 times, and item c occurs 2 times. Suppose we want to find all frequent itemsets, where $min\_sup = 4$. Itemsets of size 1 that are frequent are $\{a\}$ and $\{b\}$. Item $\{c\}$ is not frequent. Next we enumerate the itemsets of size 2: $\{a,b\}$, $\{b,c\}$ and $\{a,c\}$. To compute the support for $\{a,b\}$, we must count the number of transactions (tid,X) where $a \in X \wedge b \in X$. Obviously the support for $\{a,b\}$ is smaller than or equal to both $support(\{a\})$ and $support(\{b\})$, but since b occurs in 5 transactions, it could be the case that a is also in each 5 of those transaction, thus making $\{a,b\}$ frequent. However, already at this stage, it is clear that the subsets of length 2 containing c do not have to be investigated any further. In general terms, it holds that for any superset Z of Y where $support(Y) < min\_sup$, it holds that $support(Z) < min\_sup$.

This key ingredient is implemented in most types of FIM algorithms. Three of the most popular itemset mining algorithms are `Apriori` [6], `Eclat` [7] and `FP-Growth` [8]. Specialized algorithms exist to mine only closed or maximal patterns. The `Apriori` algorithm was introduced by Agrawal et al. and uses a breadth-first approach similar to our illustrative example. `Eclat` was introduced by Zaki et al. and uses a depth-first approach, by first re-organizing the transactional databases and computing the set of transactions for each item. Using this representation, support can be computed by computing set intersections, in a depth-first manner. Finally, `FP-Growth` by Han et al., uses a divide-and-conquer strategy by maintaining an efficient data structure, that represents the database. At each recursive step this data structure represents the database filtered by the current itemset prefix, thus shrinking in size, and speeding up the support computation.

Specialized algorithms for mining sequences, such as `PrefixSpan` by Han et al. [9], and episodes, such as `Winepi` by Mannila et al. [10] are often based on FIM algorithms.

3 DATA DESCRIPTION AND PRE-PROCESSING

The dataset contains status log data registered by the Supervisory Control and Data Acquisition (SCADA) system of one wind turbine, collected over a period of 3 years. A status log typically consists of a characteristic numerical id, a detection time and a duration for which the code was active. The messages contained in the status logs are mostly *operational* parameters of the turbine, but also *warning*, *alarm* and *system* events. The status logs linked to the *warning*, *alarm* and *system* events of the turbine are collected at the time of occurrence of each individual event. This results in unequal spacing of the data. In addition, the *operational* status of the turbine, containing *wind resource parameters* and *energy conversion*, are typically logged every 30 minutes. Therefore, most of the registered status logs are related to the operational status of the turbine. The details of the preprocessing are discussed in the following sections.

3.1 SCADA Data

An example of the cleaned input data is shown in Table 1. In the dataset, 114 distinct event codes were found. As illustrated in Table 1 a single status log can contain multiple operational parameters. In total, 41 event codes contained multiple descriptions, and thus might be considered as separate events. An example is a status log that contains a value both for the wind speed and the rotor speed. Such a log event should be considered as two separate events characterized by numerical values. The frequency of occurrences of values of the different status codes forms a *long tail* distribution. Some operational codes make up 99% of the data, while some alarm/warning/system events may occur only 10 times (or even only once).

TABLE 1. EXAMPLE OF WIND TURBINE STATUS LOG

Code	Description	Detected	Type
572	Wind Speed: 4.4 Generator Speed:1048	03/01 07:40	Op
672	Pitch Angle: -1.3 Power Production: 113.4	03/01 07:40	Op
068	Set Point: 750	03/01 08:07	Op
572	Wind Speed: 5.7 Generator Speed:1074	03/01 08:10	Op
672	Pitch Angle: -2.6 Power Production: 332.0	03/01 08:10	Op
132	ERROR Occurred:	03/01 08:25	Op
432	Turbine in Pause State:	03/01 08:25	Op
913	System in Pause State: C= 13	03/01 08:25	Alarm
521	Generator Configuration Changed:	03/01 08:26	Op

TABLE 2. EXAMPLE WIND TURBINE STATUS LOG TRANSFORMED TO A SEQUENCE OF ITEMSETS

Detected	Itemset
03-01 07:40	{speed-vlow, pitch-low, pow-low rpm-median}
03-01 08:07	{remote-command}
03-01 08:10	{speed-vlow, pitch-low, pow-median, rpm-median}
03-01 08:25	{error, pause, continuous-pause}
03-01 08:26	{generator-configuration-changed}

3.2 Alarm Codes

We are particularly interested in the alarm related events, since these potentially give insights in the operational state that has resulted in the failure. The dataset contains 29 distinct alarm codes. We neglect those alarm codes that only occur once or twice, leaving us with 11 alarm codes. The reason we neglect them is to first gain confidence in the approach. If an alarm has multiple occurrences it is possible to validate consistency of the discovered patterns. This can help support our conclusions and boost confidence in our approach. For each of these codes, we are interested in finding *frequent patterns* occurring *near* these alarm codes, that can characterize each failure.

3.3 Extracting Items From Events

As shown in Table 1 input data is stored in a *semi-structured* way where the description for certain event codes contains a template string filled with certain values, such as wind speed and turbine rpm for code 572. As such, the log consists of a mix of both single and multi-valued *categorical* events, such as 'Error' or 'Generator configuration 2' and events with associated *continuous* values, of which *wind speed*, *rotor rpm*, *power output*, and *blade pitch angle* are the most frequent. The latter is the angle over which the blades are rotated in order to optimize the angle of attack. The values shown in the description are numerical values. For pattern mining these absolute values for the operational parameters are less useful. Since there is a very low probability that exactly the same value will be recorded for, e.g., the wind speed at different time points, the pattern-mining algorithm would not be able to detect re-occurring operating ranges. To overcome this, the continuous operational parameter values were discretized into five equal intervals. For example, the power values were discretized into five equal bins between zero and nominal turbine power.

For other events, that might be split into several items, according to additional parameters found in the template string, we try to balance the support of each item on one hand, and cardinality of the total itemset on the other. We made an empirical choice: for events that occur fewer than 10 times, we do not generate multiple items for each description, as the support then becomes too low to be relevant for extracting patterns around alarms. On the other hand, for events with more than 20 different descriptions, we do not add sub-items, since this would possibly also result in having too many additional (possibly low-support) items. We considered this technique as sufficient in this case. We remark that more advanced techniques could also be applicable here, such as automatic extraction of variables for each template, and enumerating all possible sub-items up to a certain minimum support threshold.

During *pre-processing* the status events are transformed into items. In this step, each event is converted into an itemset. This conversion results in a sequence of tuples of the form $\langle datetime, X \rangle$ where X is an itemset, containing more than 1 item if two or more items occur at the same time. For example, the input sample shown in Table 1 is transformed into the sequence shown in Table 2.

4 PROPOSED METHODOLOGY

In this section we first outline our general *multi-level* approach towards characterizing turbine behavior, and then explain how we can create an appropriate transactional database, mine all patterns, and, in particular, mine patterns near alarms.

4.1 Multi-level Pattern Mining

From a wind turbine application point of view, we intend to use status logs to understand how different turbine response sequences can be linked to failure events. We ultimately attempt to identify turbine response to external triggers by means of a multi-level approach, shown in Figure 1. A turbine response pattern can be defined as a consistent way of turbine response action to an external event or trigger. Such a trigger is, for example, an extremely high wind alarm. After the triggering of such an alarm

the turbine will perform actions to bring the turbine back to a safe state. For example, in the extremely high wind case, the turbine will perform a stop event. The different turbine actions the turbine performs during the stop should be reproducible, since the turbine is a machine. In our multi-level approach this is located at Level 1, the *Turbine State Event Stream*. On the time signal of the event logs pattern mining techniques are used to identify the different turbine state patterns. These patterns describe a consistent turbine response. Only patterns with a sufficiently high support, possibly relative to a local alarm event, are kept. A domain expert annotates each of the identified patterns in order to link them to a turbine action. The different turbine events are given specific names. These are the *turbine state events* shown on Level 1.

At Level 2, the *Event stream*, these annotated turbine state events are combined with the *External triggers* to form a new time series of annotated events. These time series will be significantly reduced in complexity compared to the original *Raw status log stream* (Level 0). The goal is to reduce the number of present items by an order of magnitude. One important challenge in pattern mining is to be able to distinguish between interesting and less interesting patterns. Pattern mining techniques will deliver a significant amount of potential patterns that can be bigger than the original dataset. Therefore, a reduction in the absolute number of status logs to process can help in making this filtering step easier without the potential loss of important information. A second advantage of using these annotated patterns is readability. The *Turbine Event Stream* is a lot easier to understand and more logical to a domain expert.

Similar to the *Raw status log stream*, these annotated events time series can serve as the input for a pattern mining algorithm. This will allow to link turbine responses to external triggers. If maintenance data is used as external trigger information, then the patterns in turbine state sequences prior to failure will indicate the sequence of turbine events that has potentially led to the failure. Again, the driving patterns can be determined. If rules based on these patterns occur with high confidence it would even be possible to use them as a failure predictor.

This work takes the first steps in this approach and focuses on Level 0, and as discussed in the next sections. The approach will be illustrated by an example: turbine response to high wind trigger.

4.2 Creating the Transaction Database

Given the *Raw status log stream*, we apply a *sliding window* to convert our sequence of itemsets into a sequence of itemsets in *overlapping* periods. This step is crucial, since we are interested in finding items that co-occur in transactions, within a given period, not at exactly the same time. We define a parameter *par_window* that defines how far apart two events may occur and still be considered correlated. In our experiments we used $par\_window = 60m$, since crucial parameters such as *windspeed* or *power* are logged every 30 minutes.

Remark that when computing the support of an itemset, after the sliding window transformation, we *over-count* each itemset, as windows overlap. For example, the itemset at 8:10 in our example will be in at least in 5 windows. We added a parameter *is_overlapping* to our sliding window algorithm that can be used to re-compute support using *non-overlapping* windows. This parameter is set to $false$ when computing the support of itemsets or association rules, but set to $true$ when used for reporting.

4.3 Mining Itemsets in the Entire Sequence

Now that we have our transaction database, we can start searching for patterns. We use the publicly available *PyFim* implementation of `Eclat` by Borgelt[1]. PyFim includes all mentioned FIM algorithms, implemented in C++, with an interface available to python. In our experiments, we first set the relative support threshold *min_sup* to 0.01, meaning that each discovered itemset must occur in at least 1% of the 98350 transactions of turbine 1 (or in at least 983 instances). We vary the parameters to produce either *closed* or *maximal* itemsets that occur frequently in 60-minute transactions.

A selection of discovered patterns is shown in Table 3. We found 921 closed itemsets, and 110 maximal itemsets. We selected some of the most frequent itemsets, as well as some less frequent itemsets that contain an alarm item. Maximal itemsets condense the pattern set the most, however more frequent subsets, such as the top 4 patterns in Table 3, are pruned. The discovered frequent patterns provide an overview of the event log, since there are 230 distinct events in 98350 transactions, it is not trivial to inspect the relations between all of these events.

4.4 Mining Patterns near Alarms

One drawback of mining *all* patterns is that frequency only is often not a good proxy for *interestingness*. Several metrics have been introduced in pattern mining research to rank patterns on different criteria.

In our setting, we want to discover events, possibly (and often likely) with *low* support, but that occur consistently *near* each type of *alarm*. One straightforward strategy would be to first lower the absolute minimum support parameter to almost 1 when using `Eclat`, and then, as a post-processing step, filter out all itemsets that contain a certain type of alarm code. However, this would be extremely *inefficient*, as lowering the support to extremely low values would take *exponentially* more time to compute.

Therefore, we propose an alternative method. We assume that the alarms trigger the turbine to respond. We are interested in this response. Therefore we apply pattern mining in the time window around the alarm, as schematically shown in Figure 2. Assuming a target alarm item i_{target}, we filter the transaction

[1] http://www.borgelt.net/pyfim.html

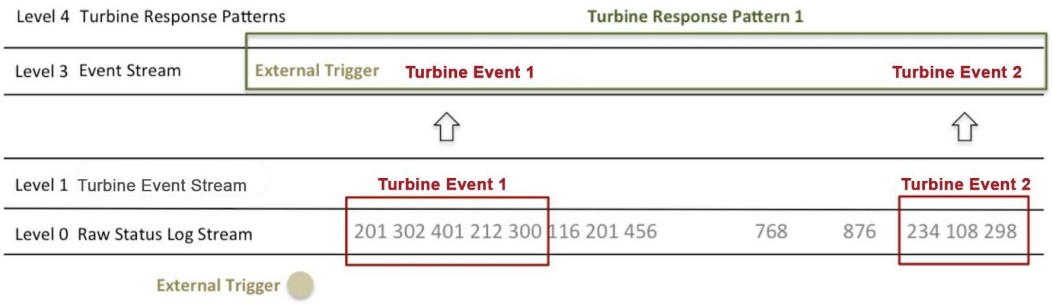

FIGURE 1. SCHEMA OF SUGGESTED MULTI-LEVEL PATTERN MINING APPROACH

FIGURE 2. SCHEMA OF PATTERN MINING APPROACH AT TIME WINDOWS AROUND TRIGGER EVENT

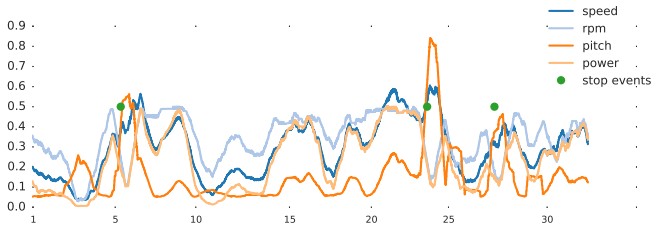

FIGURE 3. OVERVIEW FOR 1 MONTH. PARAMETERS ARE: WIND SPEED (speed), GENERATOR RPM (rpm), PITCH ANGLE (pitch) AND POWER PRODUCTION (power)

database, leaving only the transactions that contain this item, and then we run `Eclat` with min_sup relatively high on such a *projected* database. The projected databases will be very small, compared to the original database, since even the most frequent alarm code occurs just 144 times for turbine 1, and occurs in about ten times as many transactions. In our experiments we set $min\_sup = 0.5$. The result is that we find all frequent itemsets Z occurring together with i_{target}.

Next, we mine all *association rules* and keep rules that have confidence higher than some threshold min_conf. We are interested in two types of association rules, namely $Z \Rightarrow i_{target}$ and $i_{target} \Rightarrow Z$. The first type of rule is useful to predict alarm events. Note that the first type requires computing the support of Z using a scan of the entire database. Also note that itemsets that are frequent in the entire database, can never be on the left-hand side, as the confidence would be extremely low. In turbine 1, we did *not* find many rules useful in predicting an alarm code directly. The second type of association rules can be computed using *only* the projected database. We found this second type of association rules useful to characterize different response patterns, as discussed in the next section. Note that in our experiments we set $min\_conf = 0.9$ for both types of rules.

5 RESULTS

In this section we illustrate the utility of our method by showing patterns and association rules linked to turbine response during an Extremely High Wind Speed Alert.

5.1 Event Description

An example of an alarm is *Extremely High Wind Speed Alert*. When the wind speed exceeds the speed of normal operation the turbine will stop producing electricity and turn itself out of the wind. Since this is an intuitive turbine state sequence we use

TABLE 3. FREQUENT ITEMSETS FOUND IN 3 YEAR STATUS LOG OF TURBINE 1

Support	Itemset
52%	{speed-vlow, pitch-vlow, pow-vlow}
28%	{speed-vlow, pitch-vlow, pow-vlow, rpm-low}
27%	{speed-low, pitch-vlow, pow-median, rpm-median }
16%	{speed-vlow, pitch-vlow, pow-vlow, generator-configuration-1, generator-configuration-2}
1.5%	{speed-median, pitch-vlow, pow-median rpm-median }
1.1%	{speed-vlow, pow-vlow, manual-stop, error, emergency, pause }
1.0%	{extreme-high-windspeed-alarm, error auto-restart}

this one to illustrate our approach. Figure 3 shows the visualization of the operational status codes for one month. Operational parameters are: wind speed, power, generator speed, and pitch angle. We extracted the values from the operational status log messages as was explained in Section 3.3. Continuous parameters where scaled to 0.0-1.0 range. During this month in the winter period the turbine experienced several very high wind speed periods. Stops after alarm events are indicated by means of green dots and are triggered by *Extremely High Wind Speed Alerts*, where duplicate events within a window are removed. Data was smoothed using an *average window* of size 100 for visualization purposes. This month was chosen because several stops were seen. Intuitively the user can see that in case the wind speed (dark blue curve) exceeds the threshold, the turbine will start a stop event. At that moment the power produced (light orange curve) is brought down. At the same time the turbine is decelerating towards zero. This is seen in the rpm signal (light blue curve). Deceleration is achieved by pitching the blades towards a 90 degree angle, as can be seen in the dark orange signal. This sequence of events characterizes the turbine response to high wind and is the one we want to characterize by a corresponding pattern. This pattern could then be annotated with the label *Stop after extremely high wind speed alarm* and added as event to Level 1 in our multi-level mining approach shown in Figure 1.

TABLE 4. FREQUENT ITEMS NEAR HIGH WIND SPEED ALARM

no in fig.	Status code description
1	Extremely High Wind Speed Alert
2	ERROR Occurred
3	Paused State
4	Power drop from Value A to Value B
5	Generator configuration changed
6	Attempt to restart

5.2 Event Pattern Characterization

Now the approach discussed in Section 4.4 will be used to try to characterize the turbine state pattern linked to the turbine response to the *Extremely High Wind Speed Alert*. This alarm was located within the whole dataset. For clarity reasons, however, we only show one month of data (the month of data shown in Figure 3). Zooms about these three *Extremely High Wind Speed Alert* related stop events are shown in Figures 4, 5 and 6.

We mined frequent itemsets in a window between -30m and +30m around the alarm. We have set 0.5 as threshold for the minimal relative support. We found in total 49 closed itemsets. In these different itemsets the majority is highly similar and a variation of a general theme. This theme contains the most important status logs linked to stop events and are intuitively the ones to expect for this type of event. Table 4 shows the 6 status codes that we generally saw in the closed itemsets and that we expect to be characteristic for the event. The numbers in Table 4 are also shown in Figures 4, 5 and 6. Since we mine itemsets, the order in which the events occur is not considered. Based on the figures we see that the order changes a bit. However, these codes are always linked together for the response to the *Extremely High Wind Speed Alert*. We can therefore annotate this itemset with the label *Stop after extremely high wind speed alarm* and use it as input for Level 1 pattern mining. These codes are logical in the characterization of the turbine response. After the *Extremely High Wind Speed Alert* is triggered the turbine will trigger an error response (ERROR Occurred). The general response will be to perform a shutdown. Similar to the response discussed in Section 5.1, this will be done by changing the pitch such that the turbine drops in speed, reducing the power production (Power drop from Value A to Value B), decoupling the generator from the grid (Generator configuration changed) and, after the stop, attempting to restart the turbine (Attempt to restart). These sequences are always present in the discovered itemsets. From these itemsets we derived association rules where the confidence should exceed 0.5. These rules can be used to predict turbine response and define a confidence level for the prediction. In our case we found 11 association rules. The association rules with a

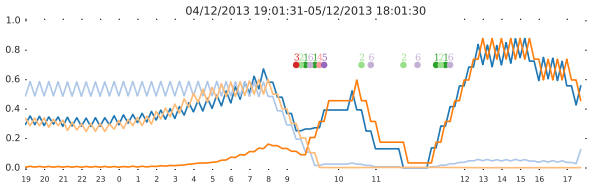

FIGURE 4. DETAILED VIEW OF HIGH WIND SPEED PATTERN AT DAY 4

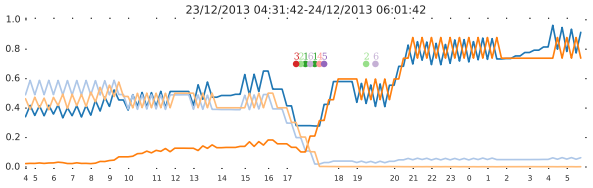

FIGURE 5. DETAILED VIEW OF HIGH WIND SPEED PATTERN AT DAY 24

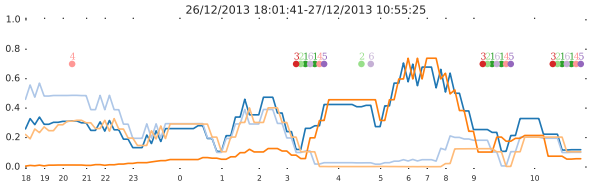

FIGURE 6. DETAILED VIEW OF HIGH WIND SPEED PATTERN AT DAY 26

confidence level close to one are linked to shorter patterns: e.g., 1-2-3 from Table 4. This is logical since these codes are linked to the most basic response of the wind turbine. The most logical association rule we found is this one: Extremely High Wind Speed $\rightarrow$ { Alert, ERROR Occurred, Paused State, Attempt to restart, rpm very low, pitch median, power very low }. It has a confidence level of 70 percent. This rule perfectly describes the sequence described by the domain expert in Section 5.1. It can therefore be concluded that the pattern mining approach can discover itemsets of logical items for describing the turbine response of a turbine to a dynamic event. In the next level of our multi-level pattern mining approach this itemset can be annotated with the turbine state action name. Since pattern mining techniques are used it is possible to automatically detect these itemsets from the raw data stream in future and perform the annotation automatically.

6 CONCLUSIONS

We investigated the use of pattern mining techniques for identifying the wind turbine response to dynamic events the turbine might encounter. Both the use of frequent itemsets and association rules approaches were tested. In a first step the complete dataset was taken into account. However, this resulted in a massive amount of itemsets and correspondingly an even bigger set of association rules. In theory, these association rules could predict certain failures of interest, however by doing this directly on all data the prediction of failure is doubtful.

To overcome this we suggested the use of a multi-level pattern mining approach. Events were used as a trigger for performing pattern mining in a window around the specific event. We illustrated the example of the *Extremely High Wind Speed Alert*. The mined itemsets found a consistently re-occurring set of codes after the triggering of this alert. The derived association rule for this itemset showed high confidence and a prediction level of 70 percent. The set corresponded to the domain expert description of the event. Therefore, this set could be recognized automatically and annotated for use on Level 1 of our multi-level pattern mining approach. In the future, we intend to extend this approach for other types of alarms. Additional service logs, and domain-expert review of existing data, is needed in order to have more accurate labels, that might be used for inferring more accurate rules to predict different classes of failures, as mentioned in related work [11].

7 ACKNOWLEDGEMENTS

The authors would like to thank the VLAIO SBO HYMOP project for funding this research and OWI-lab for its support in facilitating the study.

REFERENCES

[1] Goethals, B., 2003. "Survey on frequent pattern mining". *Univ. of Helsinki*, p. 19.

[2] Aggarwal, C. C., and Han, J., 2014. *Frequent pattern mining*. Springer.

[3] Fernando, B., Fromont, E., and Tuytelaars, T., 2012. "Effective use of frequent itemset mining for image classification". In European conference on computer vision, Springer, pp. 214–227.

[4] Naulaerts, S., Meysman, P., Bittremieux, W., Vu, T. N., Berghe, W. V., Goethals, B., and Laukens, K., 2015. "A primer to frequent itemset mining for bioinformatics". *Briefings in bioinformatics,* ***16***(2), pp. 216–231.

[5] Agrawal, R., Imieliński, T., and Swami, A., 1993. "Mining association rules between sets of items in large databases". In Acm sigmod record, Vol. 22, ACM, pp. 207–216.

[6] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A. I., et al., 1996. "Fast discovery of association

rules.". *Advances in knowledge discovery and data mining,* ***12***(1), pp. 307–328.

[7] Zaki, M. J., 2000. "Scalable algorithms for association mining". *IEEE Transactions on Knowledge and Data Engineering,* ***12***(3), pp. 372–390.

[8] Han, J., Pei, J., and Yin, Y., 2000. "Mining frequent patterns without candidate generation". In ACM Sigmod Record, Vol. 29, ACM, pp. 1–12.

[9] Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M., 2001. "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth". In proceedings of the 17th international conference on data engineering, pp. 215–224.

[10] Mannila, H., Toivonen, H., and Verkamo, A. I., 1997. "Discovery of frequent episodes in event sequences". *Data mining and knowledge discovery,* ***1***(3), pp. 259–289.

[11] Kusiak, A., and Li, W., 2011. "The prediction and diagnosis of wind turbine faults". *Renewable Energy,* ***36***(1), pp. 16–23.