A framework for pattern mining and anomaly detection in multi-dimensional time series and event logs

Len Feremans[1], Vincent Vercruyssen[2], Wannes Meert[2],
Boris Cule[1], and Bart Goethals[1,3]

[1] University of Antwerp, Belgium
{firstname.lastname}@uantwerpen.be
[2] KU Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be
[3] Monash University, Melbourne, Australia

Abstract. In the present-day, sensor data and textual logs are generated by many devices. Analysing these time series data leads to the discovery of interesting patterns and anomalies. In recent years, numerous algorithms have been developed to discover interesting patterns in time series data as well as detect periods of anomalous behaviour. However, these algorithms are challenging to apply in real-world settings. We propose a framework, consisting of generic transformations, that allows to combine state-of-the-art time series representation, pattern mining, and pattern-based anomaly detection algorithms. Using an early- or late integration our framework handles a mix of multi-dimensional continuous series and event logs. In addition, we present an open-source, lightweight, interactive tool that assists both pattern mining and domain experts to select algorithms, specify parameters, and visually inspect the results, while shielding them from the underlying technical complexity of implementing our framework.

1 Introduction

Discovering interesting patterns and anomalous periods in heterogeneous time series data is often the main interest of people generating and analyzing these data. In the past decades, the field of pattern mining has developed a large body of algorithms to automatically discover different types of interesting patterns, such as *frequent itemsets* and *sequential patterns* [21]. However, these algorithms are difficult to use for anyone who is not familiar with their inner workings. Moreover, the algorithms require the data to be preprocessed to the proper format and the type and quality of the patterns being found is largely dependent on the choices made in the preprocessing steps. If a dataset consists of multiple time series or dimensions this becomes even more problematic. Recent algorithms for pattern-based anomaly detection in time series suffer from the same drawbacks [7,11].

An example pattern-based anomaly detection pipeline is shown in Figure 1. Here we transform the single continuous signal to a *transaction database* of small

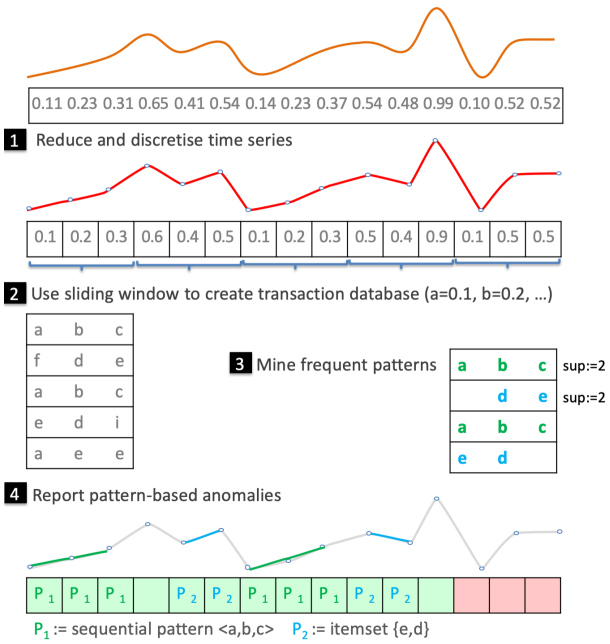

Fig. 1: Example of a pipeline for pattern-based anomaly detection. A window has a high anomaly score if it matches few frequent patterns.

discrete sequences using discretisation and a sliding window. Next, we mine frequent patterns in this transaction database and finally give a high anomaly score to windows that match no (or few) frequent patterns. When a new sequential pattern mining or a pattern-based anomaly detection algorithm is presented, important time series representation choices are often only discussed in the experimental design, and a review of alternative representations is often out of scope. Moreover, a wealth of itemset and sequential pattern mining algorithms has been developed in the past decades [8]. Most of these pattern mining algorithms are optimised towards specific, *built-in* constraints, such as mining closed itemsets or mining sequential patterns satisfying temporal constraints [16]. In the literature, little attention is given to generic *external* constraints for reducing the set of discovered patterns independently from any specific algorithm.

In addition, we find that in real-world applications anomalies are often *contextual* [1], that is, an outlier value is only anomalous given that it's out of context. For example, a high temperature during a cold winter night is anomalous, while it's normal during a summer day. Likewise, a shopping bill for more than a thousand dollar is anomalous, except during the Christmas period. Note that we do not require defining contextual attributes, but rather mine patterns of normal behaviour in all input dimensions. In contrast, classic outlier detection algorithms assume a single continuous time series that is stationary, meaning

that statistical properties, such as distribution or auto-correlation structure are constant.

We contribute a framework for pattern mining and anomaly detection in time series data. The framework allows its users flexibility regarding the three major steps in the time series analysis workflow: preprocessing, pattern mining, and anomaly detection. First, the framework supports several preprocessing algorithms for representing continuous time series, as well as a generic transformation that creates a transaction or sequence database for both single, multi-dimensional, and mixed continuous and discrete time series data. Second, the framework supports the use of all state-of-the-art pattern mining algorithms for mining itemsets and sequential patterns [8]. In addition, it adds a number of external constraints for reducing the set of discovered patterns independently from any specific algorithm, such as temporal constraints. Third, the framework supports two anomaly detection algorithms [11,7] that are extended to make them compatible with any pattern mining algorithm and multiple dimensions. The framework allows its users to rapidly test various compositions of these three time series analysis building blocks, even new compositions not considered by the original authors of each separate block. For example, instead of frequent sequential patterns, an end-user of our framework can mine a set of sequential patterns using an alternative interestingness measure [6,17], subsequently apply temporal constraints, and then use these patterns as input to an anomaly detection algorithm. Finally. we have created an *open-source tool* for Time series Pattern Mining and anomaly detection (TIPM). The tool enables an iterative, exploratory workflow for preprocessing, finding patterns and discovering anomalies, and visualising data and patterns using our framework.

2 Preliminaries

This section clarifies the important time series and pattern mining terminology used throughout the paper. The concepts are largely adapted from [7].

Time series data. A *continuous time series* is as a sequence of numerical values $(\langle x_1, t_1\rangle, \ldots, \langle x_n, t_n\rangle)$, where each real value x_k is associated with a distinct timestamp t_k. A *discrete event log* is a sequence of discrete events $(\langle e_1, t_1\rangle, \ldots, \langle e_n, t_n\rangle)$ where $e_k \in \Sigma$, with Σ a finite domain of discrete event types. Multiple events can co-occur at the same timestamp. Finally, a *mixed-type time series* $\mathbf{S}$ is a collection of N continuous time series and M event logs and has dimensionality $M + N$. A single time series S^i in $\mathbf{S}$ has only one dimension. It is possible for M or N to be 0. Thus, univariate and multivariate time series are special cases of this definition.

A *time series window* $S^i_{t,l}$ is a contiguous subsequence of a time series S^i containing all measurements for which $\{\langle x_i, t_i\rangle$ or $\langle e_i, t_i\rangle |\ t \leqslant t_i < t + l\}$. A segment of length l can be defined over all dimensions of $\mathbf{S}$ simultaneously.

Pattern mining. The following definitions are adapted from [21]. An *itemset* X consists of one or more items $x_j \in \Omega$, where Ω is a finite domain of discrete

values, that is, $X = \{x_1, \ldots, x_m\} \subseteq 2^{|\Omega|}$. An itemset does not require a temporal order between its items. An itemset X is covered by a window $S^i_{t,l}$ if all items in X occur in that window in any order, denoted as $X \prec S^i_{t,l}$. Given the set of all windows $\mathcal{S}$ of a time series, $cover(X, \mathcal{S})$ is the set of all windows in $\mathcal{S}$ that cover X and $support(X, \mathcal{S})$ is the length of this set.

A *sequential pattern* X_s consists of an ordered list of one or more items, denoted as $X_s = (x_1, \ldots, x_m)$, where $x_j \in \Omega$. A sequential pattern can contain repeating items, and, unlike n-grams, an occurrence of a sequential pattern allows *gaps* between items. A sequential pattern X_s is covered by a window $S^i_{t,l}$ if all items in X occur in that window in the order imposed by the sequential pattern, denoted as $X_s \prec S^i_{t,l}$. The definitions of cover and support are equivalent to those of itemsets. Finally, an itemset or a sequential pattern is *frequent* if its support is higher than a user-defined threshold on *minimal support*.

3 Method

The problem we are trying to solve is defined as follows:

Given: A time series dataset **S** consisting of one or multiple time series.
Do: Find interesting patterns and/or periods of abnormal behaviour in the data.

The general workflow of our framework is shown in Figure 2. Note that in our framework two strategies are available for finding anomalies. In the first strategy, we create a model of normal behaviour and predict anomalies based on deviations from this model. This is the case for the frequent pattern-based anomaly detection technique, where try to find many patterns that occur frequently and are used for *positive* detection of anomalies. A second strategy is to find anomalous patterns directly or use *negative* detection [4]. Which strategy to use, depends on the use case and can be freely chosen by the user.

3.1 Time series representation for pattern mining

Dealing with outliers. If one uses positive detection, it makes sense to remove outlier (extreme) values, that is, cap outlier values that deviate a user-specified number of standard deviations from the mean. If one uses negative detection, it makes sense to keep outlier values and discretise them along with the rest of the data, possibly in a separate bin, as the occurrence of outlier values, is often indicative of contextual anomalies.

Time series dimensionality reduction. A straightforward transformation to reduce time series is piecewise aggregate approximation (PAA) [12]. Given a time series **S**, one sets a window duration l_{PAA} and then replaces each consecutive window in **S** with the mean of the continuous values in the window. This effectively downsamples a time series **S** by a factor $|\mathbf{S}|/l_{\text{PAA}}$. In practice, it is often beneficial to downsample each time series as we are more interested in patterns that span a larger period. Note that PAA allows more flexibility than symbolic aggregate

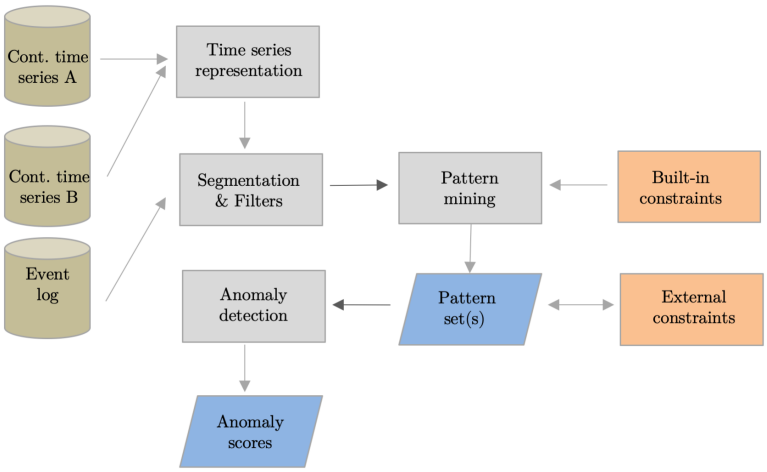

Fig. 2: Workflow of our framework.

approximation [14]. The latter assumes that the time series values are normally distributed, which is rarely the case in a non-stationary time series.

Discretisation. After reducing dimensionality, the continuous time series are discretised using *equal-width* or *equal-length* bins. As a rule-of-thumb, equal-width discretisation is applied if the observations are normally or uniformly distributed over the bins. If this is not the case, equal-length binning with a slightly larger number of bins can be selected by the end-user. The goal of discretisation is to have good coverage of items that occur in at least 5% of segments.

Segmentation. Before pattern mining, the time series need to be transformed into a *transaction* database. This is done by sliding a fixed-size window over the data and storing each time series window separately as a transaction. The window duration l_{segm} and increment i_{segm} are specified in time units or steps. Setting segmentation parameters is largely domain-specific. For instance, if the length of the datasets is two hours, but measurements (or events) are sampled every second, then finding patterns within 1 minute makes sense. The window duration and increment are important parameters towards pattern mining since they directly determine which patterns will found as well as their length. In practice, useful patterns are limited in length so one must ensure that windows are of moderate size by either setting a relatively small value for the duration or by reducing the time series dimensionality.

Filters and aggregation. Finally, our framework supports basic filtering and aggregation on the time series, as well as generic SQL queries. Filtering is useful if the goal is to model only a part of the dataset. For instance, an end-user can filter the time series on time, on periods where certain warning or error codes occur, or periods where some continuous variable exceeds a certain threshold. This has the advantage that end-users can mine and discover interesting patterns

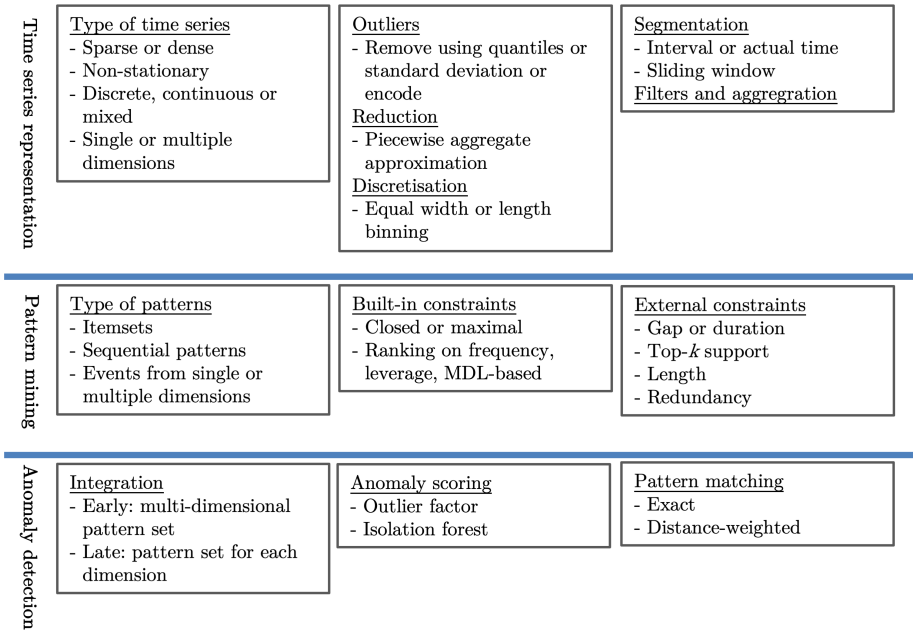

Fig. 3: Detailed overview of our framework.

local to certain events or conditions. Finally, the framework provides options to aggregate values within each window and compute summary statistics such as min, mean, max, count and unique.

Automatically selecting parameters. What constitutes a good time series representation depends strongly on the specific application. Good parameters are either selected using domain knowledge or set interactively in a trial-and-error way. However, for the anomaly detection algorithms, it is possible to select parameters using a wrapped approach. Let l_{PAA}, l_{segm}, i_{segm}, and b be the PAA window duration, segmentation length, window increment, and the number of bins respectively. The optimal parameters are selected from the parameter space $\Omega = \{l_{\text{PAA}}, l_{segm}, i_{segm}, b\}$ through optimization of an evaluation metric on the anomaly scores (e.g., AUROC).

3.2 Pattern mining

After the time series data are discretised and segmented, we can mine patterns. A more detailed overview of our framework is shown in Figure 3.

Frequent pattern mining. An end-user can discover patterns for each dimension of time series **S** that is either discrete or has been transformed into a discrete representation. Our current framework integrates with the SPMF library containing more than 40 algorithms for itemset and sequential pattern mining,

covering efficient algorithms for mining frequent, closed, and maximal itemsets and sequential patterns, top-k sequential patterns ranked on leverage and a set of sequential patterns compressed using minimal description length [8,17,13]. For the brevity of this paper, we will not discuss the details of these algorithms and refer to existing work [8,21]. Itemset and sequential pattern mining algorithms require a suitable representation for enumerating patterns and computing support. Itemset mining algorithms require a *transaction database*. This database is created by generating a transaction, or unordered set of *items*, for each window. Likewise, sequential pattern mining algorithms require a *sequence database* where for each window, we create a chronologically ordered list of items (if two events happen at the same time, this is also encoded). Each item is encoded using an integer identifier and either represents an event code or discretised continuous value. We decode item identifiers to report human-readable patterns. An example of maximal itemset mining is shown in Figure 4.

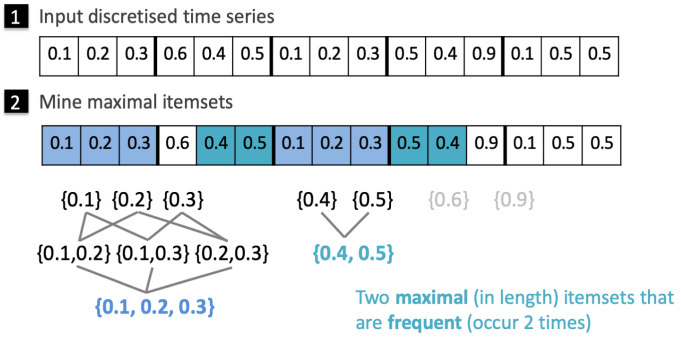

Fig. 4: Example of maximal itemset mining.

External constraints. A recent benchmark study found that temporal constraints for pattern mining in time series are of high importance [22]. Our framework computes occurrences of itemsets and sequential patterns, reported by any algorithm, and computes the occurrences that have a minimum duration in each window, by looking at the raw dataset. If the minimal occurrence does not satisfy *temporal constraints* on maximal duration and maximal gap (time between two pattern items in one occurrence), we remove the occurrence and re-compute the support for each pattern. In addition, we provide basic external constraints for filtering patterns on the minimum and maximal *length*, filtering the top-k patterns on *support*, and removing *redundant* patterns using a threshold on Jaccard similarity, i.e., if two patterns cover mostly the same transactions, filter out the pattern with the lowest support.

Multi-dimensional pattern mining. Thus far, pattern mining algorithms only work on a *single*-dimensional event log or continuous time series, after preprocessing. Our framework makes it possible to uncover patterns with events

from multiple dimensions of a time series **S**. Under this *early integration* strategy, a transaction (or sequence) database is created from the time series by adding events from multiple dimensions of the time series to a transaction. Similarly, sequence transactions (necessary for sequential pattern mining) are created by adding events from multiple dimensions in a chronological fashion. We differentiate between events from different dimensions by encoding the item identifier to reflect the source dimension. An example pipeline is shown in Figure 5.

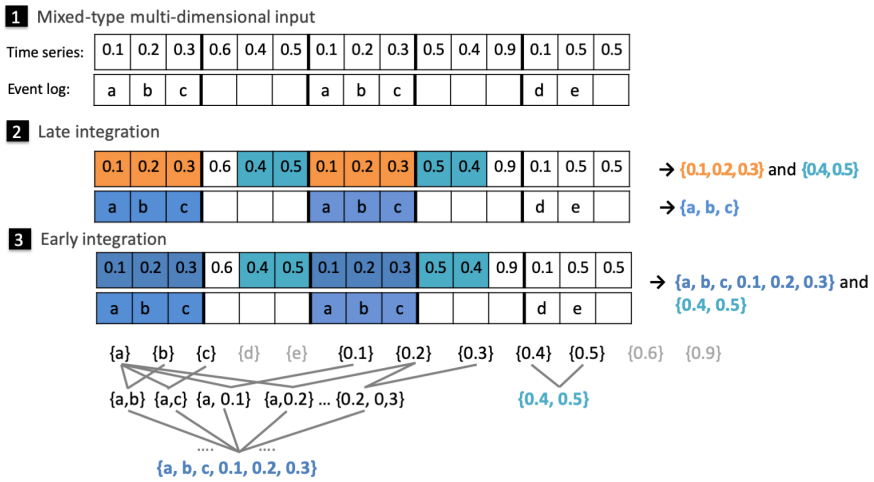

Fig. 5: Example of mining pattern in multi-dimensional time series. Under the early integration strategy, events from both dimensions are considered simultaneously. Under the late integration strategy, patterns are mined in each dimension separately.

Pattern explosion in time series. While the pattern mining community has gone through great lengths in creating efficient algorithms for different tasks, time series remain a difficult data source for efficient pattern mining. For example, imagine a time series that contains a sequence of 20 values and occurs frequently. Because this series is frequent, any subsequence will also be frequent, thereby generating an exponential number of patterns. In general, time series generate a lot of patterns due to naturally occurring autocorrelation. The problem becomes even worse when two or more dimensions are added, especially if different time series dimensions are highly correlated. In practice, we prefer mining maximal patterns with relatively high support *in each dimension separately*. This strategy, dubbed the *late integration* strategy, is illustrated in Figure 5. Alternatively, we can change the representation of the time series. In our experience, we find that using pattern sets with more than a few thousand of patterns rarely results in higher accuracy.

3.3 Pattern-based anomaly detection

Our framework supports two algorithms for anomaly detection: a generalised version of frequent pattern-based outlier factor (FPOF) and a generalised version of pattern-based anomaly detection (PBAD) [11,7]. Both methods take a set (or sets) of patterns as input and compute an anomaly score between 0.0 (normal) and 1.0 (abnormal) for each time series segment. By setting the window increment i_{segm} equal to a single time step, it is possible to compute the anomaly score at each timestamp. Figure 6 shows an example of both anomaly detection approaches.

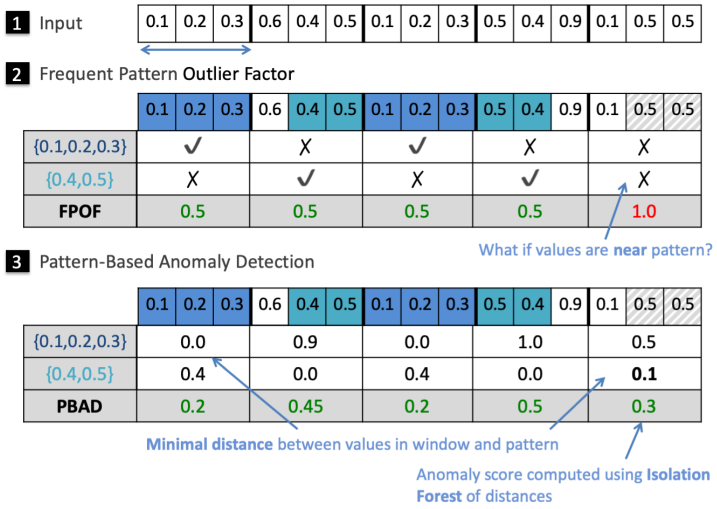

Fig. 6: Example of FPOF and PBAD for computing anomalies based on a previously discovered pattern set. In FPOF the anomaly score is based on the number of exactly matching patterns. In PBAD we compute the distance between each window and pattern and compute scores using an isolation forest.

Generic outlier factor. FPOF [11] computes an anomaly score a for each segment $S^i_{t,l}$ in time series $\mathbf{S}$, given a pattern set $\mathbf{P}$, based on the total number of patterns matching each segment, denoted by $P_k \prec S_i$:

$$a(S^i_{t,l}, \mathbf{P}) = 1.0 - \frac{|\{P_k | P_k \in \mathbf{P} \text{ and } P_k \prec S^i_{t,l}\}|}{|\mathbf{P}|}.$$

The authors only consider closed itemsets over a single dimension, but we can extend FPOF to compute this score for any pattern set, such as sequential patterns, and for multiple pattern sets mined over multiple dimensions of $\mathbf{S}$. Given two patterns sets, $\mathbf{P}_1$ and $\mathbf{P}_2$, the anomaly score is computed as:

$$a(S^i_{t,l}, \mathbf{P}_1 \cup \mathbf{P}_2) = 1.0 - \frac{|\{P_k | P_k \in \mathbf{P}_1 \cup \mathbf{P}_2 \text{ and } P_k \prec S^i_{t,l}\}|}{|\mathbf{P}_1 \cup \mathbf{P}_2|}.$$

It is trivial to extend this formula to d dimensions. The only requirement is that for computing a match from dimension d, we need to check if the pattern mined from dimension d matches the segment of the corresponding dimension. Multiple pattern sets can also be mined over the same dimension using a different algorithm or settings. For example, we can mine both itemsets and sequential patterns in a single dimension S^i.

Generic isolation forest of distance-weighted occurrences. PBAD [7] computes anomaly scores with the help of the isolation forest algorithm applied to an embedding of both maximal itemsets and sequential patterns for each continuous and discrete dimension [7]. For continuous time series, the authors use a distance-weighted match to match both itemsets and sequential patterns with each original, non-discretised, segment. For example, the distance between itemset $P_k = \{'0.5', '0.6'\}$ and segment $S_1 = (\underline{0.50}, \underline{0.61}, 0.11, 0.10)$ will be smaller than the distance to segment $S_2 = (\underline{0.31}, \underline{0.42}, 0.12, 0.04)$. We generalise PBAD by decoupling the pattern mining from the anomaly detection phase. Concretely, the distance-weighted embedding and isolation forest can be used on any pattern set and any number of dimensions. Assume we have two pattern sets $\mathbf{P}_1$ and $\mathbf{P}_2$. First, we compute the distance-weighted match between each pattern and each window for continuous time series, and the exact match for discrete (or multi-dimensional) time series. We now have two matrices of dimensions $|\mathbf{S}| \times |\mathbf{P}_1|$ and $|\mathbf{S}| \times |\mathbf{P}_2|$, and can represent each segment $S^i_{t,l}$ using a feature vector (or embedding) of length $|\mathbf{P}_1| + |\mathbf{P}_2|$. Finally, we feed this embedding to an isolation forest to compute anomaly scores.

Concept Drift. For pattern-based anomaly detection, we assume a stable distribution such that the mined patterns are good descriptors of the new data that enters the system and deviations are anomalies. However, this might not be true in practice, especially over a long period of time where the observed system might change. In such a setting we can use the pattern-based anomaly detection as part of an online adaptive learning procedure [9] and extend our framework to detect *concept drift*. The anomaly score is, in this case, the target variable that is being predicted from the new instances, and the loss function is the deviation from an average anomaly score closer to 0, representing normal behaviour. When the aggregated loss grows too large or some other change point detection algorithms crosses a threshold, the framework signals concept drift. Depending on the application, various strategies can be used to relearn. From maintaining a database of previous data to gradual forgetting old patterns and introducing new mined patterns to the pattern set(s). We refer to Gama et al. for an extensive overview.

3.4 Implementation of the framework

We implemented our framework in Java as an open-source web-based application called TIPM[4].

[4] Source and datasets available at `https://bitbucket.org/len_feremans/tipm_pub`

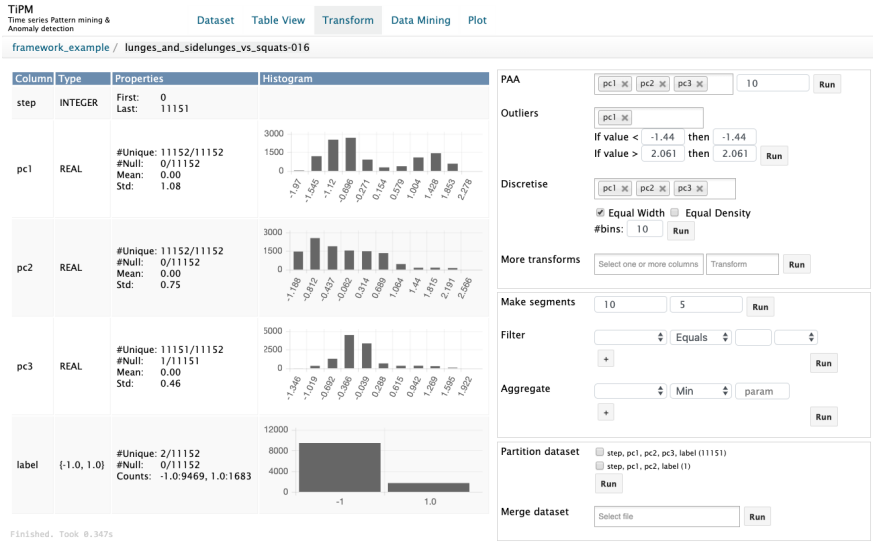

Fig. 7: TiPM: Time series representation options. In the first use case, we apply PAA with $l_{\text{PAA}} = 10$, cap outlier values, discretise all time series using 16 equal width bins and create overlapping segments with a $l_{segm} = 10$ and $i_{segm} = 5$.

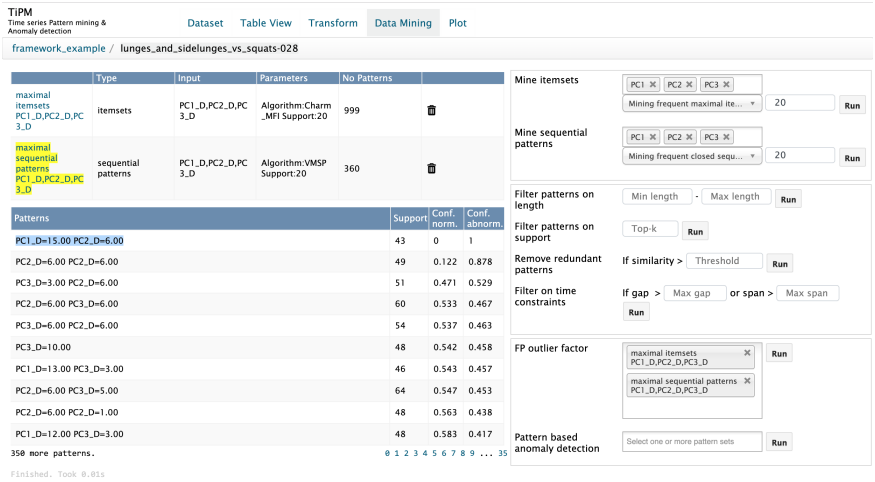

Fig. 8: TiPM: Using an early integration strategy, we mined maximal itemsets and maximal sequential patterns with minimal *support* = 20 and compute the generalised FPOF anomaly score. An example of an anomalous pattern is highlighted.

Interactive workflow. TiPM takes in any dataset that contains at least a timestamp and one or more value columns. TiPM visualises the histogram and

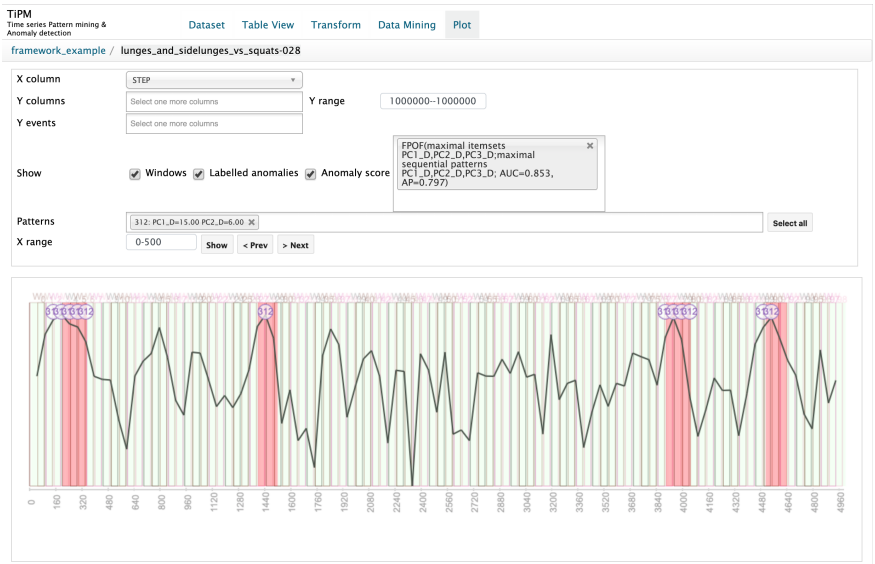

Fig. 9: TIPM: Visualisation of anomaly scores and patterns. We show occurrences of the interpretable anomalous pattern $pc1 = 15, pc2 = 6$.

summary statistics for each column and allows transforming continuous time series using our framework, as shown in Figure 7. Pattern mining is done using the algorithms implemented in SPMF. Multi-dimensional mining transformations, external constraints and anomaly detection algorithms are implemented in our framework as shown in Figure 8. TIPM can plot continuous time series values, transformed values, discrete event logs, labels, and segmentation, on different levels of granularity in time (raw, hourly, daily, yearly, etc.). For validation purposes, the tool can render pattern occurrences and anomaly scores as shown in Figure 9. We remark that TIPM saves intermediate files after each operation allowing end-users to undo any operation.

Representing mixed-type real-world datasets. Many real-world datasets, such as supervisory control and data acquisition datasets for wind turbines, contain missing values, non-continuous periods, and timestamped values stored together with event log data in a single file. In our framework, we stay close to this *tabular format* as this is most convenient for collaborating with domain experts who prefer to look at the raw data for validation. In addition, we provide two explicit temporal join operations: *partition* and *merge*. Partition takes a subgroup of columns having non-zero values and saves them in a separate table. This is useful for extracting event log data from continuous time series data. Merge is the opposite operation and takes the union of two tables and sorts them on time. If two column names match in both tables, merge takes the column value of the first table. For example, merge can be used to join time series datasets from multiple devices.

Scaling to large datasets. In our implementation, we use *streaming* operations as much as possible. Each procedure implemented in this way, processes one row at a time, instead of loading all data into main memory. Alternative, we load data in a *paginated* way, that is, we only load data required in the user interface, i.e. only the current period of the time series. By processing data using streaming operations and loading data paginated, the interface and pre- and postprocessing transformations can handle large time series with millions of samples instantaneously. For pattern mining, we can manage resources by setting support to a relatively high value, and reducing time series as discussed before. A possible extension would be to include streaming pattern mining algorithms.

4 Use cases

In this section, we will illustrate the usefulness of our framework, implemented in TIPM, using two use cases.

Anomaly detection on multivariate times series. For the first use case, we detect anomalies in a multivariate time series dataset that was obtained by using a Kinect sensor to track the body movements during indoor physical exercises [2]. The goal is to assist people in performing the exercises correctly. We focus on detecting incorrectly executed exercises during a continuous workout session consisting of 60 lunges and 10 squats. The ground-truth values are known. The original dataset consists of 75 time series and we reduced this to 3 time series using principal component analysis [7].

First, we upload the time series in tabular file format. TIPM shows statistics and histograms for each time series (*pc1*, *pc2* and *pc3*) as well as the label as shown in Figure 7. We can now select options to preprocess each time series. First, we cap outlier values based on the 1% and 99% quantiles. Next, we compute and store the average value every 10 time steps, that is $l_{\text{PAA}} = 10$, to reduce the 3 continuous dimensions using PAA. We then apply equal-width discretisation with $b = 16$ bins. For multi-dimensional mining, our input dataset thus consists of 16×3 discrete items. We create sliding windows with a duration of $l_{segm} = 10$ (3 seconds in absolute time) and $i_{segm} = 5$, resulting in 223 windows of length 10 that overlap for 50%. With all continuous data time series represented as discretised segments, we start mining patterns. We opt for early integration and select all three dimensions as input. We select an algorithm for mining maximal itemsets (CHARM_MFI) with a minimum *support* of 20%. For reducing patterns, we remove itemsets that co-occur in at least 90% of windows, resulting in 999 itemsets. We compare this set of patterns by mining maximal sequential patterns (VMSP) with the same settings, resulting in 360 patterns. Finally, we run the generic outlier factor and compute an anomaly score for both types of pattern sets individually. The screens in TIPM are shown in Figure 8 for mining and Figure 9 for visualisation. We show occurrences of the (anomalous) pattern $pc1 = 15, pc2 = 6$, found by sorting all maximal sequential patterns with minimal support of 20 on decreasing confidence towards labelled anomalies. Figure 10 shows the first minute of the Kinect dataset. TIPM shows the transformed time

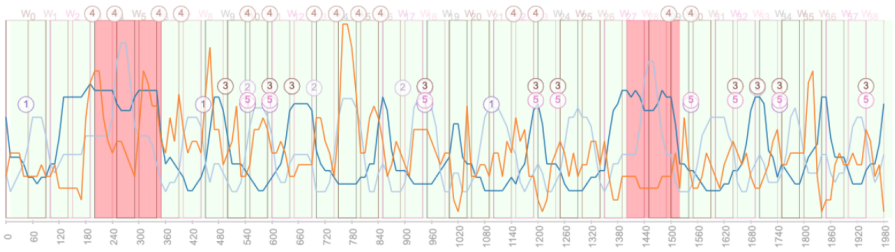

Fig. 10: Visualisation in TIPM of the first minute of data. The blue line is *pc1*, the light blue line *pc2*, and the orange line *pc3*. We show the top-5 most frequent maximal itemsets, mined over all three dimensions.

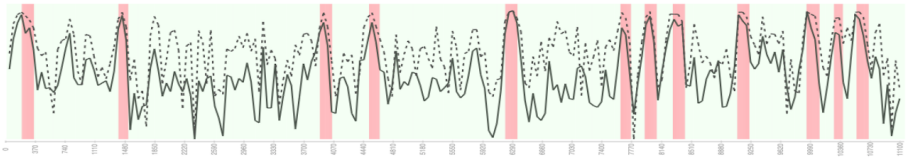

Fig. 11: Visualisation in TIPM of anomaly detection results. Segments with a red background are labelled anomalies, and the black line is the anomaly score predicted (unsupervised) using generic outlier factor using maximal sequential patterns. The dotted line is the results using maximal itemset patterns.

series and overlapping segments. We selected the top-5 most frequent maximal itemsets for visualisation. The first itemset is $\{pc1 = 1, 2, 4 \wedge pc2 = 1 \wedge pc3 = 4, 5\}$ which has a support of 56 (or relative support of 0.25). This means that 25% of segments contain both (discretised) values of 1, 2 and 4 in time series *pc1*, 1 in *pc2*, and 4 and 5 in *pc3*. Notice that the first frequent pattern, as well as the 2^{nd}, 3^{th} and 5^{th}, but not 4^{th}, almost never occur in any anomalous segment highlighted in red. Consequently, the patterns are examples of frequent interpretable patterns that occur during normal behaviour. Deviations from these patterns are marked as anomalies. As mentioned before, TIPM allows sorting patterns on confidence towards normal or abnormal segments, thereby assuming labels. We find that sequential patterns containing high values of *pc1* are the most predictive towards abnormal behaviour. Figure 11 shows the anomaly scores over the entire 6 minute time series. Using the generic outlier factor anomaly detection method we can report an AUROC of 0.839 and average precision of 0.767 for maximal itemsets, and an AUROC of 0.884 and average precision of 0.833 for maximal sequential patterns.

Exploratory analysis of real-world heterogeneous time series. In the second use case, we perform an exploratory analysis of a supervisory control and data acquisition dataset collected from a wind turbine farm [5]. This dataset is challenging because: (i) the data was collected over different years, (ii) there are multiple continuous time series, (iii) there is an event log containing more than hundreds of different types of events, (iv) behaviour of a wind turbine is strongly

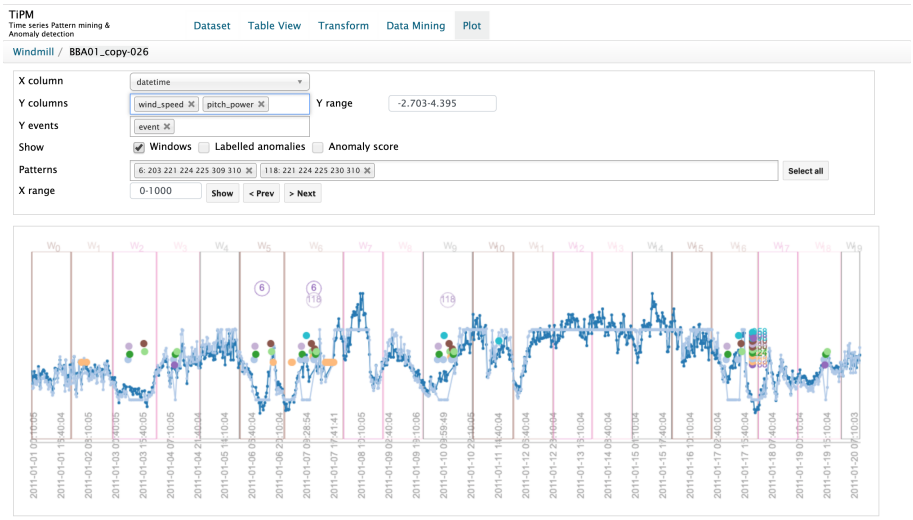

Fig. 12: TIPM: Use case for exploratory analysis of heterogeneous wind turbine data. We show two interesting patterns that are characteristic of operational behaviour, mined from the event log.

dependent on current weather conditions. Figure 12 shows the preprocessed data and two interesting patterns for a selected period of 1 month in TIPM. For the wind turbine, we have selected two continuous variables representing the wind speed and power output as well as an event log containing warning, error, and operational codes. First, we normalised both continuous time series. We verify that wind speed and power output are highly correlated. The main difference is that power output is capped to a maximal value. The different occurring events are shown as colour-coded dots. We mine maximal itemsets with a minimal support of 1 and found about 140 patterns. Next, we filter patterns with a minimum size of 2. Next, we remove redundant patterns by setting a Jaccard similarity threshold of 0.9, thereby removing patterns that co-occur in 90% of windows. From the remaining patterns, we show two maximal itemsets of size 5 and 6 that occur in the selected period. Both patterns correspond to a specific series of operator actions for remotely stopping and restarting the turbine. If these patterns occur, the power output drops to 0, regardless of the current wind speed. From this second use case, we conclude that our framework can be used to explore complex multi-dimensional datasets, using patterns extracted from the event log, to capture meaningful operational behaviour.

5 Related work

Most general data mining libraries, such as WEKA or KNIME, are incomplete concerning pattern mining. TIPM is complementary to SPMF [8] by implementing temporal constraints, multi-dimensional pattern mining, and pattern-based

anomaly detection algorithms. In contrast to SPMF, and other libraries that implement time series transformations on consecutive numeric vectors, we support timestamped tabular data with multiple dimensions, and mixed-type attributes. Other tools for anomaly detection in time series uses either shapelets or motifs (or discords) in single-dimensional continuous time series [18]. Interactive pattern mining tools, such as MIME [10] or SNIPER [15], do not support continuous time series.

There exist algorithms for directly mining patterns with temporal constraints [16]. However, by providing temporal constraints as an external post-processing filter, we can apply them to any pattern mining algorithm. This is of interest for many efficient algorithms for mining closed, maximal or interesting patterns that do not support temporal constraints. Many more transformations for reducing the length of the time series exist [3]. We prefer PAA for two reasons. First, different authors have confirmed that more advanced techniques are not necessarily more effective [3,14]. Second, many other representation techniques, i.e., transformation to spectral space, single value decomposition, or clustering, make interpretation much harder while patterns of binned values are easy to interpret. Remark that other transformations, such as differencing or smoothing the raw time series are not problematic regarding interpretation.

Two popular techniques for classification and anomaly detection in time series are the *matrix profile* [20], that computes an outlier score relative to the euclidean or dynamic time warping (DTW) distance to its nearest neighbour, and time series *shapelets*, which are subsequences from a continuous time series and are used in combination with the DTW distance to classify time series segments [19]. A key difference is that frequent patterns naturally handle both continuous time series and event logs. If we compare sequential patterns to shapelets, we argue that on the one hand, sequential patterns generalise shapelets, because we use non-continuous subsequences with gaps. On the other hand, sequential patterns are more specific, because they consist of discretised values instead of continuous values. The latter argument against sequential patterns, however, can be relaxed by using a weighted distance. Itemsets, however, are radically different from shapelets and of value for predicting anomalies. In future work, an *ensemble* of representations could have value. That is, we can compute itemset and sequential pattern distances, exact pattern matches, shapelet distances, motif distances, and combine those in one feature vector, as input for existing classification or anomaly detection algorithms.

6 Conclusion

Existing pattern-based anomaly detection algorithms focus on a particular combination of time series representation, pattern mining, and computation of the anomaly score. In PBAD, the authors remarked that this method is a promising general framework for time series anomaly detection, where certain variations might be more effective in different applications [7]. In this paper, we implement such a framework and discuss a wealth of general building blocks, that can be

composed to create new variations. This allows data scientists to create novel unsupervised anomaly detection models. We also present TIPM, an interactive, easy-to-use, and open-source tool that implements our framework. TIPM is unique since we have a rich set of options for interactively preprocessing and mining patterns from mixed-type time series, supported by visualisation of (raw and transformed) time series, event logs, segments, patterns and anomaly scores. With our framework, we show how to discover interesting interpretable patterns and detect anomalies in multi-dimensional time series in two different use cases.

Our framework and corresponding tool are designed to support real-world applications. For applications such as condition monitoring of devices, it is important to support devices that log both sensor values and events. We focus on contextual anomalies, i.e. we only consider outlier values as anomalous if they are abnormal given the current operational conditions, by capturing normal behaviour using patterns and predicting anomalies as deviations from normal behaviour. We also discussed the integration of concept drift within our framework as an important next step.

Acknowledgements

The authors would like to thank the VLAIO SBO HYMOP project for funding this research.

References

1. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 15 (2009)
2. Decroos, T., Schütte, K., De Beéck, T.O., Vanwanseele, B., Davis, J.: AMIE: Automatic monitoring of indoor exercises. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 424–439. Springer (2018)
3. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proceedings of the VLDB Endowment **1**(2), 1542–1552 (2008)
4. Esponda, F., Forrest, S., Helman, P.: A formal framework for positive and negative detection schemes. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **34**(1), 357–373 (2004)
5. Feremans, L., Cule, B., Devriendt, C., Goethals, B., Helsen, J.: Pattern mining for learning typical turbine response during dynamic wind turbine events. In: ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. pp. V001T02A018–V001T02A018. American Society of Mechanical Engineers (2017)
6. Feremans, L., Cule, B., Goethals, B.: Mining top-k quantile-based cohesive sequential patterns. In: Proceedings of the 2018 SIAM International Conference on Data Mining. pp. 90–98. SIAM (2018)
7. Feremans, L., Vercruyssen, V., Cule, B., Meert, W., Goethals, B.: Pattern-based anomaly detection in mixed-type time series. Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2019)

8. Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The spmf open-source data mining library version 2. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 36–40. Springer (2016)
9. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM computing surveys (CSUR) **46**(4), 44 (2014)
10. Goethals, B., Moens, S., Vreeken, J.: Mime: a framework for interactive visual pattern mining. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 757–760. ACM (2011)
11. He, Z., Xu, X., Huang, Z.J., Deng, S.: FP-outlier: Frequent pattern based outlier detection. Computer Science and Information Systems **2**(1), 103–118 (2005)
12. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and information Systems **3**(3), 263–286 (2001)
13. Lam, H.T., Mörchen, F., Fradkin, D., Calders, T.: Mining compressing sequential patterns. Statistical Analysis and Data Mining: The ASA Data Science Journal **7**(1), 34–52 (2014)
14. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. pp. 2–11. ACM (2003)
15. Moens, S., Jeunen, O., Goethals, B.: Interactive evaluation of recommender systems with sniper - an episode mining approach. In: Proceedings of Thirteenth ACM Conference on Recommender Systems. RecSys '19 (September 2019)
16. Pei, J., Han, J., Wang, W.: Constraint-based sequential pattern mining: the pattern-growth methods. Journal of Intelligent Information Systems **28**(2), 133–160 (2007)
17. Petitjean, F., Li, T., Tatti, N., Webb, G.I.: Skopus: Mining top-k sequential patterns under leverage. Data Mining and Knowledge Discovery **30**(5), 1086–1111 (2016)
18. Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A.P., Chen, C., Frankenstein, S., Lerner, M.: Grammarviz 2.0: a tool for grammar-based pattern discovery in time series. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 468–472. Springer (2014)
19. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 947–956. ACM (2009)
20. Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A., Keogh, E.: Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: 2016 IEEE 16th international conference on data mining (ICDM). pp. 1317–1322. IEEE (2016)
21. Zaki, M.J., Meira, W.: Data mining and analysis: fundamental concepts and algorithms. Cambridge University Press (2014)
22. Zimmermann, A.: Understanding episode mining techniques: Benchmarking on diverse, realistic, artificial data. Intelligent Data Analysis **18**(5), 761–791 (2014)