Submission date: 18/06/2018
Camera ready submission date: 06/04/2019

1st round notification: 27/03/2019
Available online: 11/04/2019
Publication date: 16/04/2019

Section: extended version
Best papers from the XIV Brazilian Symposium on Information Systems (SBSI 2018)

Leveraging Anomaly Detection in Business Process with Data Stream Mining

Gabriel Marques Tavares[1], Victor G. Turrisi da Costa[1], Vinicius Eiji Martins[1], Paolo Ceravolo[2], Sylvio Barbon Jr.[1]

[1]Computer Science Department – State University of Londrina (UEL)
Londrina, Paraná – Brazil

[2]Dipartimento di Informatica – Università degli Studi di Milano (UNIMI)
Crema, Italy

{gtavares, victorturrisi, barbon}@uel.br, vini9x@gmail.com,
paolo.ceravolo@unimi.it

Abstract. *Identifying fraudulent or anomalous business procedures is today a key challenge for organisations of any dimension. Nonetheless, the continuous nature of business activities conveys to the continuous acquisition of data in support of business process monitoring. In light of this, we propose a method for online anomaly detection in business processes. From a stream of events, our approach extract cases descriptors and applies a density-based clustering technique to detect outliers. We applied our method to a real-life dataset, and we used streaming clustering measures to evaluate performances. By exploring different combinations of parameters, we obtained promising results, showing that the method is capable of finding anomalous process instances in a vast complexity of scenarios. Thus, improving the quality of business processes by providing insights for stakeholders.*

Keywords. *Process Mining, Business Process Modelling, Online, Fraud, Clustering*

1. Introduction

The increasing level of digitisation [Juhaňák et al. 2017], the growth of automation, the availability of devices with higher storage capabilities, and the expanding use of sensors and probes [Kilpeläinen and Tyrväinen 2004] significantly increase data production, leading to a phenomenon known as data explosion [van der Aalst 2011]. However, the availability of a large amount of data represents both an opportunity and a risk. In 2014, Forbes[1] reported that for most organisations, the problem is not the lack of data, but the lack of the right data. Bohmer et al. [Böhmer and Rinderle-Ma 2017] highlight that stored business processes data can be beneficial to both organisations as well as attackers that

[1]http://www.forbes.com/sites/steveculp/2014/06/06/for-banks-better-data-management-means-more-effective-fraud-and-crime-prevention/

Cite as: Tavares, Gabriel Marques., Turrisi da Costa, Victor G., Martins, Vinicius Eiji., Ceravolo, Paolo., & Barbon, Sylvio. 2019. Leveraging Anomaly Detection in Business Process with Data Stream Mining. iSys: Revista Brasileira de Sistemas de Informação (Brazilian Journal of Information Systems), 12(1), 54-75.

may exploit them. Furthermore, anomalies, such as frauds, exceptions, and errors, should be detected as quickly as possible.

PM combines Business Process Management (which uses knowledge from information technology and management sciences to understand operational business processes), Data Mining (DM) and Machine Learning (ML) [van der Aalst 2011]. Furthermore, it aims at extracting valuable information from an event log (stored business processes data) by discovering, monitoring and improving actual processes [Becker and Intoyoad 2017]. The starting point for PM is an event log. The log records the execution of business processes consisting of a sequence of time-ordered events, i.e., performing an activity at a given time. Furthermore, these events other information, such as the actor who executed the activity and the resources exploited during execution. A sequence of activities from the same process instance is known as a trace [van der Aalst 2011].

The traditional assumption of PM is to have access to the whole set of historical data related to a business process. By consequence, most algorithms work in batch mode and rely on an event log composed of complete cases. However, in real-world scenarios, efficient detection is required to be performed in an online fashion. Thus, stakeholders can take action when an anomalous process execution is detected.

Data Stream Mining (DSM) is the area concerned of dealing with streams from running systems [Krawczyk et al. 2017, Gama et al. 2010]. Further, data streams are possibly infinite, creating additional constraints, such as memory and processing time limitation. This way, when treating business processes as a stream of events, it is required to attend DSM demands. Moreover, another challenge arises considering that there is a mismatch at the representation level between PM and DSM. PM organises processes instances as cases composed of several events, meaning that one process instance is the group of events related to one, and only one, case. Contrarily, traditional DSM techniques consider one event as a complete representation of an instance. Utterly, this means that there is a need for encoding business processes before applying traditional DSM methods.

Another underline problem with online PM is the existence of incomplete cases. Traditional PM techniques usually group the cases in order to retrieve their traces. However, in online PM, there is no way to group future events belonging to a running process instance. This way, online PM techniques must deal with incomplete case constraint. A preliminary implementation of online process mining focusing on Cognitive Computing was proposed by Barbon et al. [Barbon et al. 2018]. The authors proposed a strategy to find concept drift in business data streams toward providing a response to evolving environments in near-real time. A concept drift is the change of the distribution of the feature vector in relation to its class over time. The results were promising, but they did not take into account data streaming performance metrics when addressing the anomalous behaviour. Moreover, the proposal was evaluated with only one real-life process. Furthermore, to the best of our knowledge, no state-of-the-art researches are addressing the problem of anomaly detection in online PM. This is due to the intersection between PM and DSM is still new, and little was explored.

In this work, we aim at clarifying and consolidating the validation process of tech-

niques merging PM and DSM. As these techniques have to take into account the continuous generation of data, the cases analysed might be incomplete, i.e. their final activity was not reached. In this regard, the investigated approach has to handle a continuous stream of business events and at the same time, extract knowledge from it by assessing their similarity and organising them in space to differentiate anomalous cases from regular ones. This way, organisations will be able to interpret the currently implied workflow and explore ways of improving their processes.

This work is an extension of that proposed by Tavares et al. [Tavares et al. 2018]. The original work ingested a stream of events to create and maintain a process model representation. For that, a histogram was used. Then, for new events in the stream, their cases were retrieved to extract descriptors. Finally, an online density-based clustering technique was used to cluster cases based on their descriptors, aiding the detection of anomalous behaviour.

To extend the original work, one event log was added. This new example has intricate behaviour, which makes it attractive to analyse. Moreover, we explore the hyperparameters of the proposed algorithms more deeply, going through different combinations. This way the evaluation proposed in this paper can consider more scenarios and offer a better understanding of the capabilities provided by the proposed approach.

The paper is organised as follows. Section 2 gives an overview of PM preliminaries and shows an example of an event log. Section 3 presents our approach and some concepts related to streaming theory. Section 4 presents the data set and its characteristics, our experimental setup and metrics. Section 5 explores the obtained results. Finally, Section 6 concludes the paper.

2. Process Mining Preliminaries

PM is the area interested in extracting knowledge from business processes using the event logs recorded by information systems as input. Table 1 shows an example of a typical event log where each row corresponds to an event. An *event* represents the execution of an *activity* at a certain time. Moreover, an event is associated with a unique process instance, also referred to as a *case*. Then, the group of events from the same case is known as a *trace*. From this example, we can infer that Case 46 is composed of two events and its trace is composed of the activities *Weight* and *Design Checker*. Further, Case 42 has its trace composed of *Process Creation* and *CRA*. Finally, the cases 1, 42, 44 and 46 represent a set of process instances generated from the same business process.

This way, PM strives to discover, monitor and improve real processes based on event logs. It is important to highlight that all PM techniques ingest an event log. Moreover, there are three main types of process mining tasks: process discovery, conformance checking and process enhancement [van der Aalst 2011].

Process discovery focus on the extraction of a model using only event log information. There are several proposed algorithms for discovering a model, such as the α-algorithm [van der Aalst et al. 2004], the Inductive Miner [Leemans et al. 2014], the Heuristic Miner [Weijters and van der Aalst 2003], among others. Discovering methods need to abstract the connections between process activities to be capable of building a

Case ID	Activity	Complete Timestamp
Case 1	ME Assembly Checker	2011/04/15 10:00:39
Case 42	Process Creation	2011/02/22 17:00:22
Case 42	CRA	2011/02/23 15:00:08
Case 44	Process Creation	2011/03/05 10:00:43
Case 46	Weight	2011/03/09 15:00:23
Case 46	Design Checker	2011/03/09 17:00:07

Table 1. Example of an event log from a business process. Each row represents one event, which is the execution of an activity at a certain time

model that represents concurrency and, at the same time, deal with noise data.

Conformance checking detects deviations between the model and the execution log by comparing an existing process model with the event log of that same process [Rozinat and van der Aalst 2008]. Given a set of prescribed rules, conformance verifies if processes, operations, and practices conform to them. Moreover, conformance may be able to measure the degree of deviation between the model and the event log [Valle et al. 2017]. Thus, a conformance technique supports the detection of anomalies since it identifies deviations between the log and the model.

Finally, the goal of process enhancement is to improve the process model based on information from a log. One of the ways enhancement may act is using the comparison provided by the conformance checking for adapting the reference model. This task is known as model repair. Another enhancement type is the extension, which tries to cross-correlate the model and the log [van der Aalst 2011].

It is important to highlight that traditional PM applications are based on a static event log, i.e. using batch processing. With the rising need of online solutions, several works have been proposed to deal with online process discovery [van Zelst et al. 2018, Leno et al. 2018, Burattin et al. 2014, Burattin et al. 2015], online conformance checking [Burattin and Carmona 2017, Al-Ali et al. 2016] and business concept drift detection [Bose et al. 2014, Bose et al. 2011, Maggi et al. 2013, Barbon et al. 2018]. However, no work has addressed the online detection of process anomalies.

3. Proposed Approach

Figure 1 gives an overview of the proposed method. Cases are evaluated based on two features computed using the Edit Weighted Distance (EWD) and the Time Weighted Distance (TWD) respectively. Those features describe a case in two points of view; the first is trace-based and the latter time-based. The features are extracted by comparing the trace with the model representation. For that, cases are characterised as profiles, where a profile is a set of related items which describe the trace from a specific perspective [Ceravolo et al. 2017a]. We call the process of encoding cases in features the conversion phase. The extracted case features are then fed to an online clusterisation algorithm, which organises cases in a dynamic feature space and continuously updates the case position according to the arrival of newer events.

Since we are proposing an approach to handle event streams, only one event can be

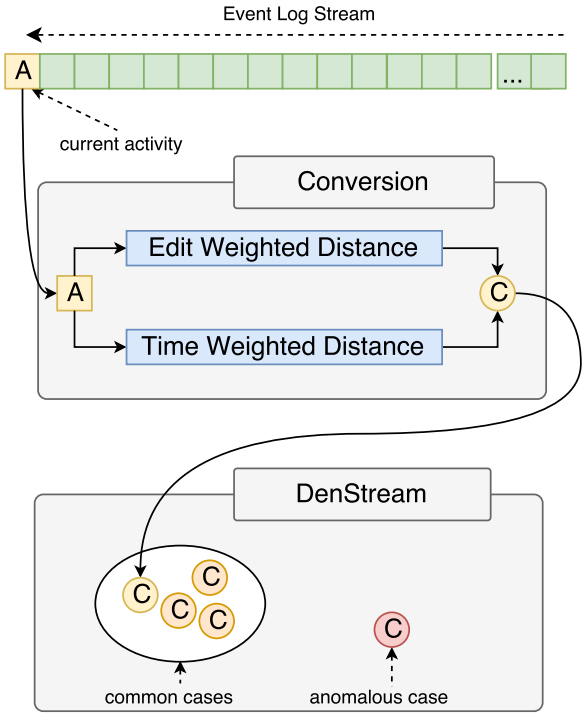

Figure 1. Proposed approach overview. Each event is consumed considering its time of arrival. The conversion phase goal is to extract features from the case. Then, the case is online clustered according to its features

assessed at a time, representing the constant arrival of information. As a consequence, the acquisition of a new event implies in the update of the profile of a case and the consequent update of its position in the feature space. The feature space also preserves the original order of event acquisition by including a feature for tracing the temporal evolution of the system, that can be explored to identify interesting properties such as the convergence or stability.

A density-based clustering algorithm was considered the appropriate solution to constantly update the regions of the feature space, and by consequence, to mark as outliers cases that are dislocated into low-density regions. In Section 3.2 we discuss the DenStream algorithm, that realises the paradigms adopted, and the way it handles the presented dataset.

3.1. Conversion

Traditional PM techniques are applied in batch and can consider complex structures of data. Dealing with data streams requires to focus on profiling as a viable way to reduce space complexity. Our approach addresses this issue through a conversion phase that fits the anomaly detection phase. The goal of this phase is to extract features describing the cases within a process.

Two profiles were used. The first one deals with the trace and the activities related to it, while the second one deals with the time interval at which the activities occurred. By doing that, we can identify both anomalous cases with an odd set of activities and

anomalous cases with time bottlenecks. The profiles adopted in this work encode the behaviour represented by a case in the form of a histogram.

A trace is a sequence of activities. Thus, the order of the activities is a way of telling if the process protocol is being followed or not. Just as important as the order, is the time in which the activities happen. This way, the trace analysis can identify instances where activities happen in an odd sequence or contain irregular time-intervals.

Two hyperparameters control the conversion phase. Since the algorithm starts without a priori data, it has to gather some data to build an initial model. Inspired by the concept of Grace Period (GP) presented in [Domingos and Hulten 2000], data is collected until it reaches the number established by the GP hyperparameter. The GP limit is the number of cases used to create the first model according to the algorithm.

The next hyperparameter introduced is Time Horizon (TH). Its goal is to set a time interval where the algorithm checks the current number of cases and reevaluates if the model needs an update. The optimal number of cases comes from an adaptation of the Nyquist sampling theorem [Lévesque 2014]. Nyquist theory establishes that the sampling frequency should be twice the highest frequency contained in the signal. In our approach, the highest frequency is the number of new cases that appeared during the last TH. Thus, at the end of a TH, the number of new cases (N_c) that had appeared in the last TH is compared with the Nyquist (N_y) previously calculated. If N_y is higher than N_c, older cases are released from memory, and a new Nyquist is calculated.

One of the problems when dealing with data streams is memory limitation. With this in mind, a histogram was chosen to serve as the basis for comparison between events. The event log contains information related to activities and the time of processing. By taking into account both these characteristics, our approach proposes a trace and time analysis.

For the trace analysis, all cases activities are accounted and summed. The result is a histogram summarising the number of occurrences of each activity. At the arrival of new events, the corresponding case is retrieved and its trace updated and then compared with the histogram to update its feature values. Inspired by the widely used Edit Distance algorithm [Wagner and Fischer 1974], which compares two strings, we used the Edit Weighted Distance (EWD). This metric identifies different events in both strings and sums the normalised weighted distance based on the histogram occurrence.

For example, given a set of traces $L = \{\langle a, b, c\rangle, \langle b, c, c, d\rangle, \langle c, c, d, d\rangle, \langle d, e\rangle\}$, and a set of activities $A = \{a, b, c, d, e\}$, we construct the histogram $H = \{1, 2, 5, 4, 1\}$ following the order of activities in A and recording the frequencies observed in L for each activity. This histogram represents the number of occurrences of each activity, in order. When a new trace $N = \langle c, d, e\rangle$ arrives, the histogram is normalised (according to Equation 1) between 0 and 1, and the EWD is computed. The normalised histogram is $Hnorm = \{0, 0.25, 1, 0.75, 0\}$ and the different activities between both strings (*abcde* and *cde*) is *ab*. Finally, as the sum of the normalised weighted distances, EWD = 0.25. In case of an activity that is in the trace but not in H, its weighted value is 0.5.

$$Xnorm = \frac{X - Xmin}{Xmax - Xmin} \quad (1)$$

We computed the Time Weighted Distance (TWD) based on the same principle of the weighted distance applied to time analysis. Here, the histogram construction takes the following steps: for each case, the time difference (in seconds) between each activity is computed; after that, the list of time differences are given as inputs for a quartile calculation. Quartiles are cutpoints that divide the input data into four equally distributed groups, so each group has a quarter of the data [Frigge et al. 1989]; the time differences are binned into the quartiles, and the sum of all cases quartiles is the histogram of timestamps. Given a new event, its cases timestamps go through this binning process, and the normalised weighted distance is the difference between the binned case and the time histogram.

3.2. DenStream

DenStream is an online clustering algorithm developed for evolving data streams proposed by Cao et al. [Cao et al. 2006]. It was selected for our solution for a set of reasons: (i) it is well-established in DSM literature, (ii) deals with outlier detection, (iii) identifies arbitrarily shaped clusters, (iv) does not require previous knowledge about the optimal number of clusters, (v) performs well memory and time wise.

Firstly, it uses the concept of micro-clusters proposed by Aggarwal et al. [Aggarwal et al. 2003] to create, delete and update clusters dynamically using limited memory. The micro-clusters used by DenStream can be defined as $\{\overline{CF^1}, \overline{CF^2}, w\}$, where $\overline{CF^1}$ corresponds to the weighted linear sum of the instances in that cluster, $\overline{CF^2}$ is the weighted squared sum of the instances and w is the weight of that micro-cluster. The centre of a micro-cluster is defined as $\frac{\overline{CF^1}}{w}$ and its radius is $r = \sqrt{\frac{|\overline{CF^2}|}{w} - (\frac{\overline{CF^1}}{w})^2}$.

Cao et al. [Cao et al. 2006] uses the micro-cluster concept in three different ways.

1. Core micro-clusters (c-micro-clusters), which can be viewed as a dense micro-cluster with $w \geq \mu$, where μ is a hyperparameter to control the minimum density of a micro-cluster to be considered a c-micro-cluster;
2. Potential c-micro-clusters (p-micro-clusters) which can be viewed as a potential c-micro-cluster with $w >= \beta\,\mu$, where β is a hyperparameter;
3. Outlier micro-clusters (o-micro-clusters), which is composed of outlier instances with $w < \beta\,\mu$.

When new instances arrive, p-micro-clusters and o-micro-clusters weights are updated by an exponential decay, defined as $2^{-\lambda}$, with λ being a hyperparameter. Such behaviour implies that p-micro-clusters and o-micro-clusters situated in inactive regions of the feature space suffer the decay factor until an eventual fade away. In contrast, p-micro-clusters and o-micro-clusters that receive new instances increase their weight, becoming denser.

The DenStream algorithm can be divided into three main parts:

1. First, there is the initialisation step, where a modified version of the DBSCAN algorithm [Ester et al. 1996] is applied to the first *n_min* instances. Starting at a random instance, all instances within a maximum distance of ϵ are considered to be part of the same group. If the resulting group weight is greater than $\beta\,\mu$, then this group is initialised as a p-micro-cluster. The process is repeated until all instances are part of a p-micro-cluster or cannot form any new micro-clusters;
2. After that, p-micro-cluster and o-micro-clusters are updated whenever the algorithm receives a new instance i. At first, DenStream will try to incorporate this instance to the nearest p-micro-cluster. If the radius of that p-micro-cluster would stay less than ϵ after incorporating that instance (intuitively because it is close to that p-micro-cluster), then the instance is added to it. Otherwise, it will try to add this instance to the nearest o-micro-cluster, employing the same test. If it cannot incorporate that instance (intuitively because it is too far away from all existing micro-clusters), then a new o-micro-cluster is created containing only that instance. Additionally, it checks if any o-micro-cluster has enough weight to be promoted to a p-micro-cluster;
3. The last part consists of creating the final clusters (composed by the c-micro-cluster) for the user by applying the DBSCAN algorithm to the p-micro-clusters. First, starting a cluster with an arbitrary p-micro-cluster p_a, it scans other p-micro-clusters, where $distance(p_a, p) <= r_{p_a} + r_p$, where r_{p_a} is the radius of the p-micro-cluster p_a and r_p if the radius of the other p-micro-cluster, and adds them to the same cluster. Then, for each new p-micro-cluster in that cluster, the same scanning process is performed. This is performed until no new p-micro-cluster is added to that cluster. After that, if the total summed weight of all p-micro-clusters in that cluster is greater than μ, then it becomes a final cluster. This process is repeated until all p-micro-clusters are part of a final cluster or cannot be incorporated into any.

4. Materials and Methods

4.1. Event Log Data Stream

The business processes chosen for the experiments are from a manufacturing company in Italy, which was first studied and explored in [Ceravolo et al. 2017b]. The event logs include different business processes related to product management.

The data contains five processes that were selected considering several behavioural characteristics, such as the number of events and cases, the time span of cases and the variety of traces. The selected processes were:
1. *Assembly_Frozen-Final_Rel*; 2. *Assembly_IW-Frozen*; 3. *Detail_Frozen-Final_Rel*; 4. *Detail_IW-Frozen*; 5. *Detail_Supplier_IW-Frozen*. These processes will be referred to as P_1 through P_5, respectively.

Additionally, one more dataset was chosen. The *Hospital_Billing* event log was first presented in [Mannhardt et al. 2017], and it was obtained from the ERP system of a hospital. The event log relates to the billing of medical services, its events are anonymised and the time within activities has not been modified. A full version of *Hospital_Billing*

dataset can be found in the PM repository[2]. From now on, this event log will be referred to as P_6.

The use of P_6 comes from the fact that the log is complex and comes from a different source, thus presenting different behaviour. Moreover, its record ranges more data, containing more than 450,000 events and 100,000 traces. Such extensive data presents a variety of anomalous cases, both from time and trace perspective.

4.2. Business Processes Statistics

Table 2 presents several metrics that describe the chosen processes, with all of them having a good amount of cases and events. The mean value of cases and events per day range significantly in the selected processes. The processes that present a higher number of cases also have a higher number of events. An interesting way of understanding the behaviour of a process on a daily basis is to look at the mean cases per day. On the other hand, the maximum cases per day show an unusual day, which may be an indication of erratic behaviour.

Regarding trace size, P_1 and P_3 both have the longest trace with only two activities. P_1 has a mean trace size of 1.99, which can imply that traces with only one activity are outliers. Though all processes have the smallest trace size of one activity, P_2, P_4 and P_5 have longer traces, reaching 9, 10 and 12 activities, respectively. The mean trace size of those processes corroborates with the idea of a good distribution of activities per case. Since P_1 and P_3 usually have fewer activities, their case duration is also smaller compared to other processes, with a mean duration of no more than a couple of minutes. P_1 has a maximum case duration of almost ten days, a high value compared to its mean and median, which is strong evidence of outlier behaviour. Contrarily, P_2, P_4 and P_5 have longer case duration. That aligns with the fact that these processes have more activities and consequently take more time to complete.

Since P_6 is roughly 18 times larger (in the number of events) than the previously largest process (P_4), we have chosen to analyse the first 30,000 events of P_6. These events represent around five months of the event log, which is a fair representation of the process. This way, the statistics presented in Table 2 only consider the first 30,000 events of P_6.

P_6 is a process that differs significantly from the others. It has 142.37 mean cases per day, while also presenting a more chaotic behaviour. None of the previous processes has maximum cases per day as higher as the mean of P_6, which has a maximum of 271 cases in only one day. This is a reflection of the mean number of events per day (211.26). Even though P_6 has the mean trace size with no more than 2.5 activities, it has some larger traces, with the most expressive one being composed of 213 activities. However, 43.8% of P_6's cases have only one activity. It is also observable how P_6 presents a unique case duration. The mean case duration is more than 15 days, which is comprehensible whereas the log comes from a healthcare background and treatments are time-dependent. One case lasted for almost 130 days, but the median case duration is low due to a large portion of cases being time-atomic, i.e. having only one activity. Although the other processes are real event logs, P_6 has a more complex behaviour, where the log depends on a plethora of

[2]https://data.4tu.nl/repository/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741

variables, such as the size of the hospital, dependencies within activities, among others. Using such an event log will further explore the capabilities of our approach to handling more complex situations.

	P_1	P_2	P_3	P_4	P_5	P_6
# cases	1185	4199	3817	11549	9488	12631
mean cases per day	5.46	7.96	14.4	18.41	25.1	142.37
max cases per day	33	48	96	101	99	271
# events	2355	9324	6722	25131	24952	29999
mean events per day	10.85	17.69	25.36	40.08	66.01	211.26
max events per day	66	202	187	483	347	521
mean trace size	1.99	4.57	1.76	4.63	5.43	2.37
longest trace size	2	9	2	10	12	213
smallest trace size	1	1	1	1	1	1
mean duration (days)	0.03	2.51	0.001	2.5	2.14	15.5
max duration (days)	9.91	50.87	0.12	77.83	47.7	129.77
median duration (hours)	0.0002	23	0.0002	22.01	19.99	0.004

Table 2. Processes statistics considering case and event frequencies, trace sizes and time duration

4.3. Model Representations

The structure of this work comes from a PM point of view and so, it is important to further understand the processes behaviour by analysing their Petri Nets. Petri Net (PN) is a popular way of representing a process in PM, being the oldest process modelling language [van der Aalst 2011]. A PN is a bipartite graph of *places* and *transitions* connected by arcs [Murata 1989], which is capable of describing concurrent, asynchronous, distributed, parallel, non-deterministic, and stochastic systems [Murata 1989]. Major points of PNs consist of: representing casual dependencies and in sets and systems with different levels of abstraction; the possibility of verifying systems properties; and the easiness of applying an analysis technique [Reisig 1985].

In PM, process discovery techniques intent on finding the best fitting model that can represent the event log [van der Aalst 2011]. Consequently, PN modelling uses an event log, describing its behaviour. Creating a model from a dataset is not a trivial task. Thus, there are several algorithms for PN creation, and there is no consensus on a better one. Using the Inductive Miner-infrequent (IMi) discover algorithm proposed in [Leemans et al. 2014], we created the PNs representations of our processes. Figures 2, 3, 4, 5, 6 and 7 show the resulting PNs. However, since we are dealing with a stream, the events are processed at the time of their arrival. Therefore, cases are incomplete and the PNs of such cases compromised, making traditional approaches not viable.

P_6's PN shows the complexity of the process, with several arcs and dependencies among activities. However, several activities are not so typical in the event log. For instance, A_{14} and A_{16} occur only 4 and 5 times, respectively, in the whole event log. This sort of rare activity usually makes the representation of a PN more complex, as seen in

Figure 7, and though the activities are infrequent, they are not necessarily anomalous [Mannhardt et al. 2017]. Furthermore, different process discovery algorithms may implicate in different PNs. Some discovery algorithms focus on a more precise representation of the event log behaviour, while others may sacrifice some of the infrequent behaviours to present a simpler PN.

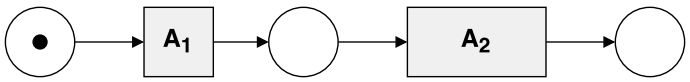

Figure 2. P_1 **Petri Net representation extracted with Inductive Miner-infrequent**

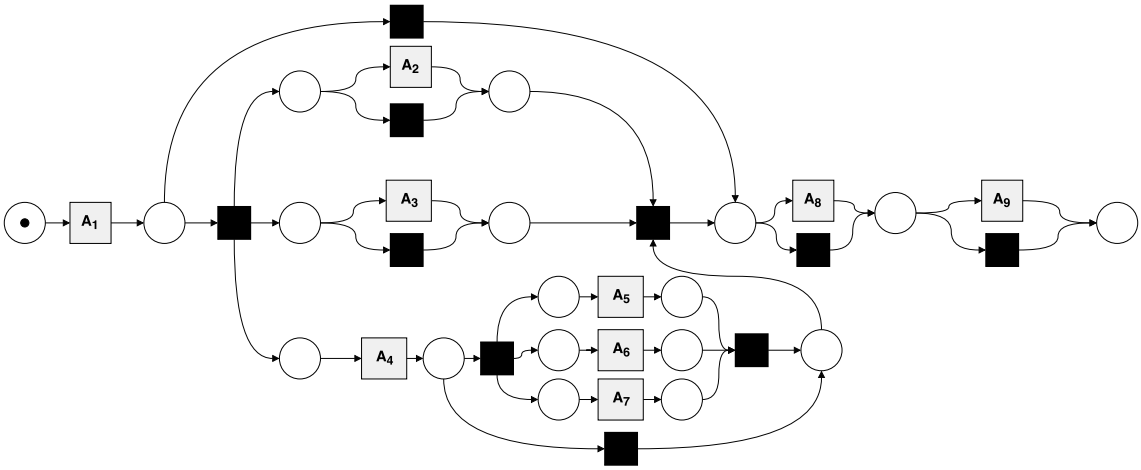

Figure 3. P_2 **Petri Net representation extracted with Inductive Miner-infrequent**

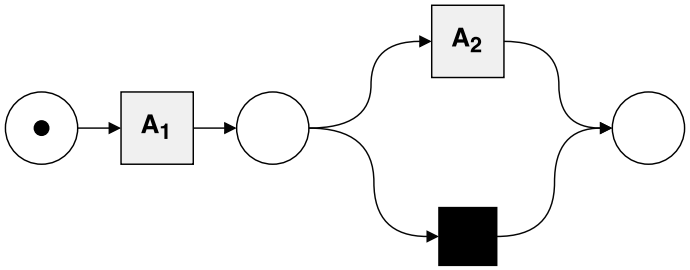

Figure 4. P_3 **Petri Net representation extracted with Inductive Miner-infrequent**

4.4. Experimental Setup

To evaluate our approach, we used the business event logs described earlier. Additionally, we used the DenStream implementation provided by the Massive Online Analysis (MOA) tool. MOA is a software environment that allows the implementation of algorithms and execution of experiments on evolving data streams with the inclusion of an extensive collection of methods and tools including classification, regression, clustering, and concept drifts tools [Bifet et al. 2010].

We executed the DenStream algorithm for each process while varying the TH value. To generate the ground truth for each case (labeled as common or anomalous),

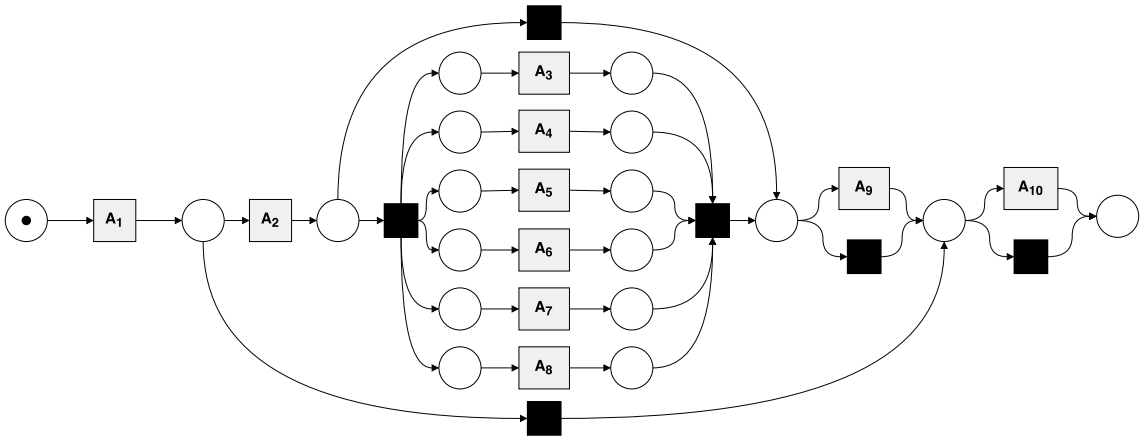

Figure 5. P_4 **Petri Net representation extracted with Inductive Miner-infrequent**

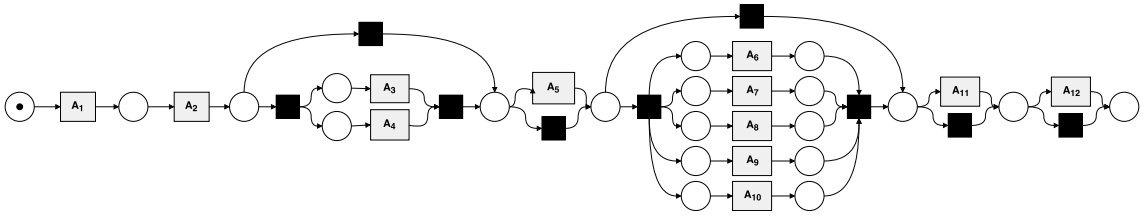

Figure 6. P_5 **Petri Net representation extracted with Inductive Miner-infrequent**

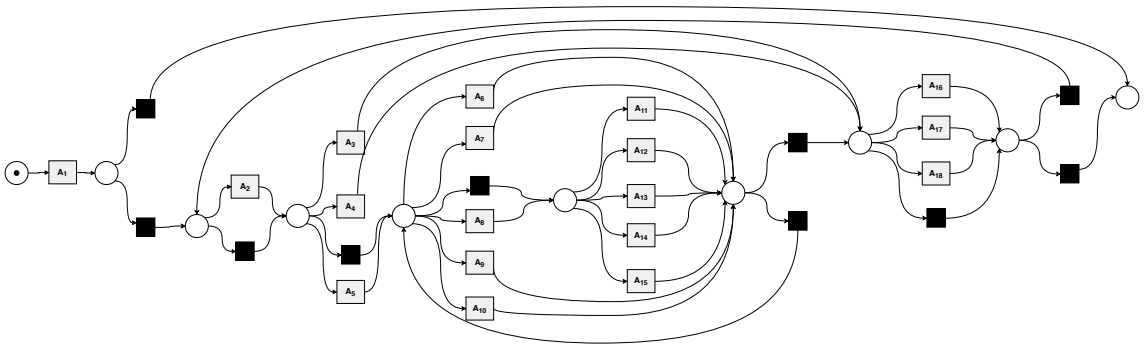

Figure 7. P_6 **Petri Net representation extracted with Inductive Miner-infrequent**

we used the Principal Component Analysis (PCA) [Witten et al. 2016]. PCA was applied in PM for process interpretation in some works [Carmona and Cortadella 2010, Wang et al. 2012]. They highlighted the main advantage of PCA usage in PM as its unsupervised modelling since the availability of anomalous cases identification by a human expert is very poor in real-life datasets. Also, PCA exposes an overview of a process in a more comprehensible space domain built by the most representative principal components. However, the PCA technique requires a batch processing of the log event. In this way, we computed the behaviour of each case using the whole event log considering the outliers cases as anomalous, as suggested by [Bose and van der Aalst 2010].

Figure 8 was obtained by applying the PCA to P_5 and shows the first and second principal components, PC1 and PC2 respectively. PC1 was responsible for explaining

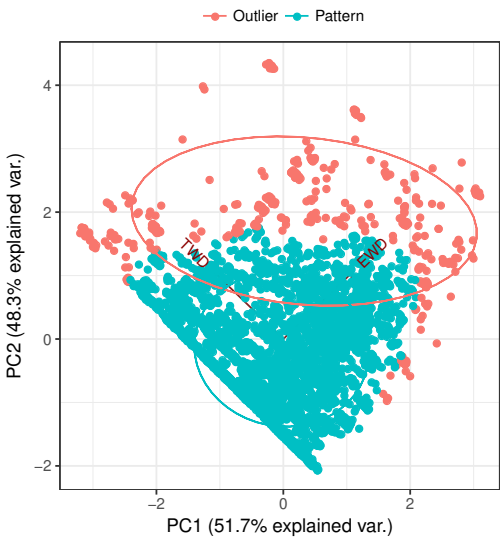

Figure 8. PCA space of *Detail_Supplier_IW-Frozen* (P_5) for anomaly detection (outlier), reddish points are outliers and greenish are common pattern

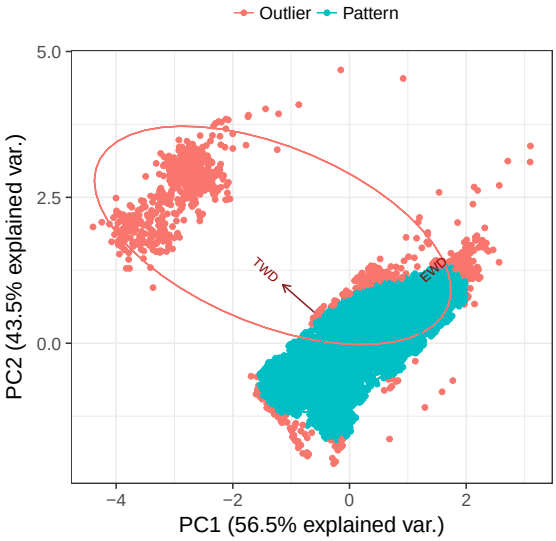

Figure 9. PCA space of *Hospital_Billing* (P_6) for anomaly detection (outlier), reddish points are outliers and greenish are common pattern

51.7% of the variance while PC2 48.3%. Figure 9 shows the PCA result of P_6 with 24 hours as TH. Reddish points indicate outliers and greenish ones represent common behaviour. Common cases are concentrated while outliers are sparser and localised in extreme regions of the feature space. There is a group of anomalous cases in the left upper corner of the image. These cases are identified as outliers due to their TWD value, showing that an abnormal time variance within activities is indicative of anomalous behaviour. From this experiment, 9% of cases were classified as outliers while 91% were common.

The DenStream has the following hyperparameters:

1. *Horizon*: comprehends how many instances on the stream are considered;
2. Epsilon (ϵ): represents the maximum value a micro-cluster radius can assume;
3. Mu (μ): controls the minimum density of a micro-cluster to be considered a c-micro-cluster;
4. Outlier Threshold (β): determines the threshold of an outlier relative to p-micro-cluster. It is used along with μ;
5. Decay Factor (λ): applied into existing micro-clusters weight. Sets the importance of historical data;
6. Stream Speed (v): controls the application of the decay factor. That is, the decay factor is applied after v instances;
7. *n_min*: the number of points used for initialisation of the DenStream algorithm.

4.5. Performance Evaluation

To evaluate our results, we selected CMM (Cluster Mapping Measure) [Kremer et al. 2011] and Homogeneity [Rosenberg and Hirschberg 2007]. CMM

was proven effective when evaluating clustering on data streams [Kremer et al. 2011] and Homogeneity is a criterion introduced as a score for clustering solutions and highly considered in the V-measure metric [Rosenberg and Hirschberg 2007].

CMM consists of the normalised sum of penalties for errors occurring in the clustering process, a score of 1 (or 100%) means that no errors were found in the clustering, while 0 (or 0%) indicates the maximum error. It takes into consideration the age of the data, the points that were missed by the moving clusters, the points that were misplaced in the clusters and the noise found in each cluster to identify three fault cases: missed points, misplaced points, and noise inclusion. The penalties that are used to calculate CMM are generated from these errors taking into account their seriousness, age, and clustering model [Kremer et al. 2011].

On the other hand, Homogeneity is defined by the number of clusters that contain only data points from the same class, meaning the smallest amount of entropy. Along with completeness, it was defined as the two criteria used to measure V-Measure [Rosenberg and Hirschberg 2007].

5. Results and Discussion

The plots in Figures 10, 11, 12, 13, 14 and 15 show the values for CMM and Homogeneity for each of the five processes. All six processes were submitted to a set of 8 TH variations that range from 6 to 96 hours. The DenStream parameters were: *horizon* $= 1100$; $\epsilon = 0.02$; $\beta = 0.2$; $\mu = 1$, *n_min* $= 1000$; $\lambda = 0.25$; $v = 100$. Since each process describes a precise protocol with specific characteristics, no single TH presents the best metrics across all processes.

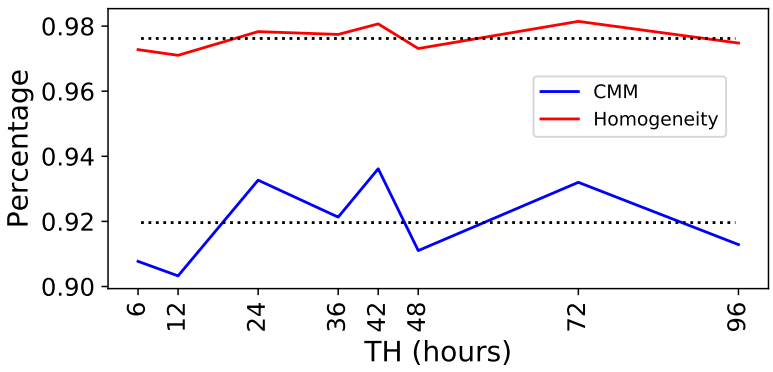

Figure 10. P_1 metrics for different THs. Mean CMM = 0.919; Mean Homogeneity = 0.976

There is a definite correlation between CMM and Homogeneity, which is perceived due to their similar performance behaviour according to the TH. On several occasions the metrics behaviour mimics each other, meaning that they are aligned.

The graphs show that the Homogeneity score was even higher than the CMM, in all cases. This means that, for most of the time, our approach was able to differentiate

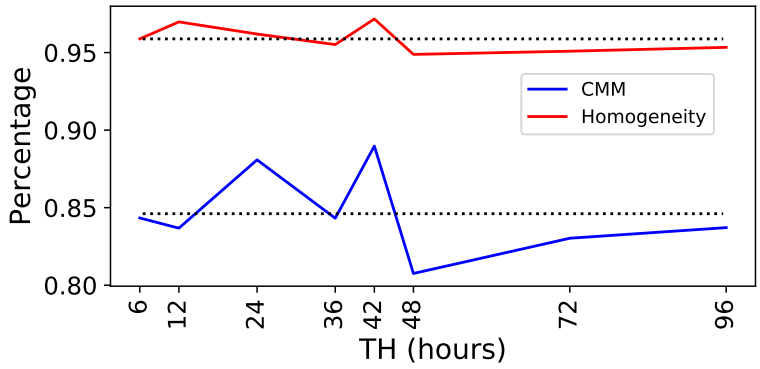

Figure 11. P_2 **metrics for different THs. Mean CMM = 0.846; Mean Homogeneity = 0.958**

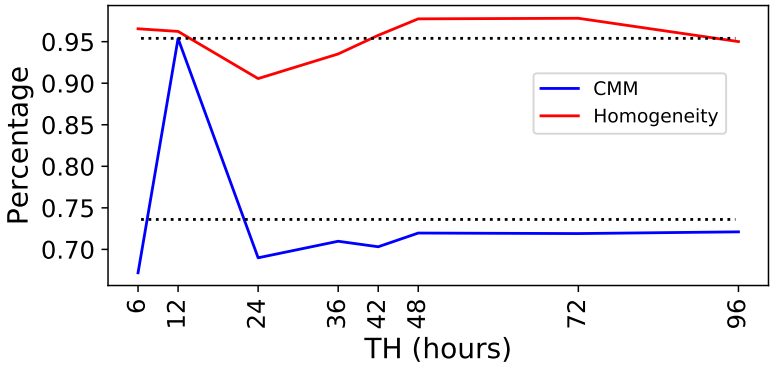

Figure 12. P_3 **metrics for different THs. Mean CMM = 0.736; Mean Homogeneity = 0.954**

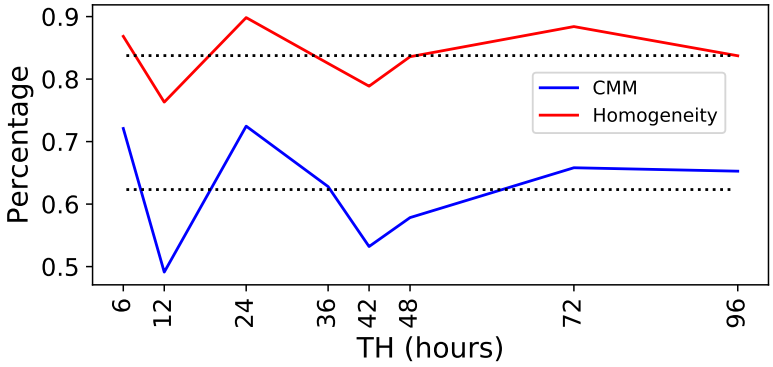

Figure 13. P_4 **metrics for different THs. Mean CMM = 0.623; Mean Homogeneity = 0.837**

common and anomalous cases, rarely putting them together in the same cluster. This level of abstraction is decisive in an anomaly detection approach.

Table 3 presents the performance metrics of the DenStream algorithm in this con-

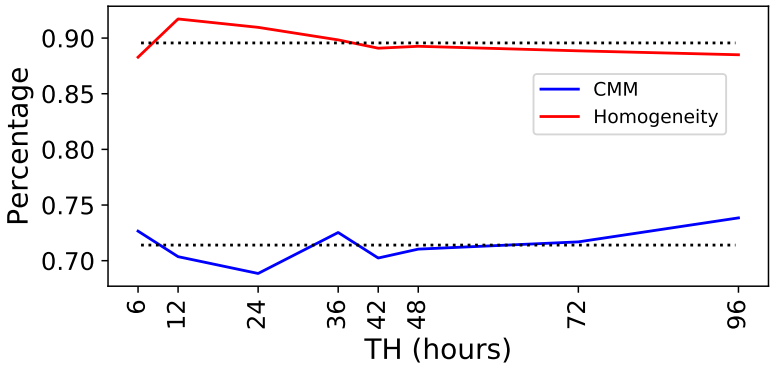

Figure 14. P_5 metrics for different THs. Mean CMM = 0.714; Mean Homogeneity = 0.895

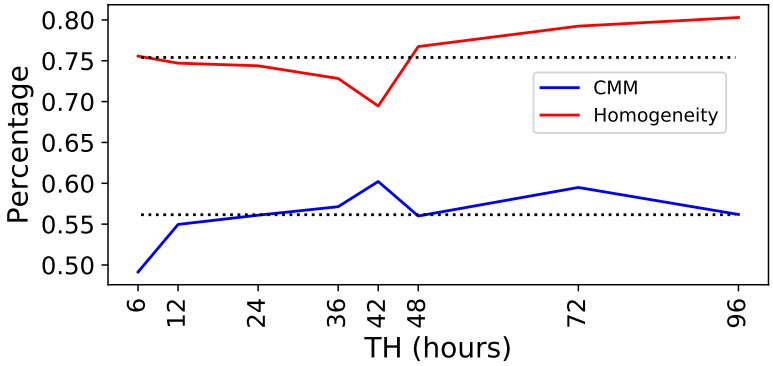

Figure 15. P_6 metrics for different THs. Mean CMM = 0.562; Mean Homogeneity = 0.754

Statistics	P_1	P_2	P_3	P_4	P_5	P_6
$\overline{CMM}$	91.9%	84.6%	73.6%	62.3%	71.4%	56.2%
CMM(TH*)	93.6%	88.9%	95.3%	72.4%	73.8%	60.2%
$\overline{Homo}$	97.6%	95.8%	95.4%	83.7%	89.5%	75.4%
Homo(TH*)	98.1%	97.1%	95.4%	89.8%	91.7%	80.3%

* optimal TH value for each process

Table 3. Performance metrics (CMM and Homogeneity) for each process

figuration. We calculated the mean for both metrics for each process, while additionally, presenting the best performance values according to the optimal TH value for each metric in each process. Since TH is a hyperparameter, it can be easily adapted to different processes.

Process P_1 obtained 91.9% and 97.6% for mean CMM and Homogeneity, respectively. With an optimal TH value, the CMM and Homogeneity raise to 93.6% and 98.1%. This behaviour indicates the simplicity of the process, which is corroborated by its PN. Thus, making it easier to find a possible fraudulent case. On the other hand, P_3 presents

lower mean CMM (73.6%), but when selecting the optimal TH value, the CMM increases to 95.3%. This shows a lot about the process behaviour, in the sense that the PN for P_3 seems relatively simple, but only one TH value was able to describe more precisely, which implicates that the process usually takes the same time to be completed. The behaviour of process P_3 reinforces the necessity of taking time into account when finding anomalous cases. By considering only the trace, it would not be as easy to find irregular behaviour.

The PNs of processes P_4 and P_5 show how they are more complex than the others. However, even with higher complexity, we obtained more than seventy percent in CMM with optimal THs for both processes. Moreover, their Homogeneities are 89.8% and 91.7%, respectively, which shows that our approach hardly clusters different elements together.

The mean CMM and Homogeneity for process P_6 were 56.2% and 75.4%, respectively. Both results are lower than previously discussed processes due to the complexity of process P_6, as seen in Figure 7. Figure 15 shows the metrics while varying the TH. Using 42 hours as TH, the optimal CMM value was reached (60.2%). However, the same TH had the lowest Homogeneity (69.4%). The optimal Homogeneity (80.3%) was obtained with 96 hours as TH, showing that with this TH most clusters contained only one case class. Several TH configurations (from 6 hours to 96 hours) were explored, however, none was able to understand the process behaviour completely; consequently, the method had difficulty pinning anomalous cases.

Although varying the TH is interesting to optimise performance metrics, it is not enough in some situations. Complementary, DenStream parameters should not be neglected and should also be explored for better results, once they cause an impact in the online clusterisation of the event stream. Using TH $= 42$ hours; *horizon* $= 1100$; $\epsilon = 0.15$; $\beta = 0.1$; $\mu = 2$; *n_min* $= 1000$; $\lambda = 0.05$; $v = 100$, we obtained a CMM of 99.87% and Homogeneity of 100% for process P_6, despite its complexity. Lowering ϵ implicates in clusters with a smaller range of action, thus suffering less effect from points in the feature space. Also, once cases are situated outside all clusters, they tend to form their clusters, either it being an outlier or not. Moreover, smaller λ implicates that historical data is less affects, hence common behaviour is not forgotten to early. Both parameters impact highly on the performance metrics results, which implies that finding optimal parameters configuration is essential for better performance metrics.

Figure 16 shows the results of CMM and Homogeneity for different THs using *horizon* $= 1100$; $\epsilon = 0.1$; $\beta = 0.1$; $\mu = 2$; *n_min* $= 1000$; $\lambda = 0.05$; $v = 100$. Performance metrics are higher compared to Figure 15, which is a result of the deeper exploration of DenStream parameters, showing, therefore, that even a complex process can have its behaviour modelled. The mean CMM and Homogeneity were 98.6% and 96.9%, respectively.

Finally, Table 4 shows the best metrics obtained in the experiments and the configuration that enabled such performance for each process. For this, the parameters *horizon* $= 1100$; $\beta = 0.1$; $\mu = 2$; *n_min* $= 1000$; $v = 100$ were fixed while ϵ, λ and TH varied. The varying parameters were chosen because they showed a higher impact on the results. The highest CMM was 99.9% (P_3) and the highest Homogeneity was 100% (P_6). When

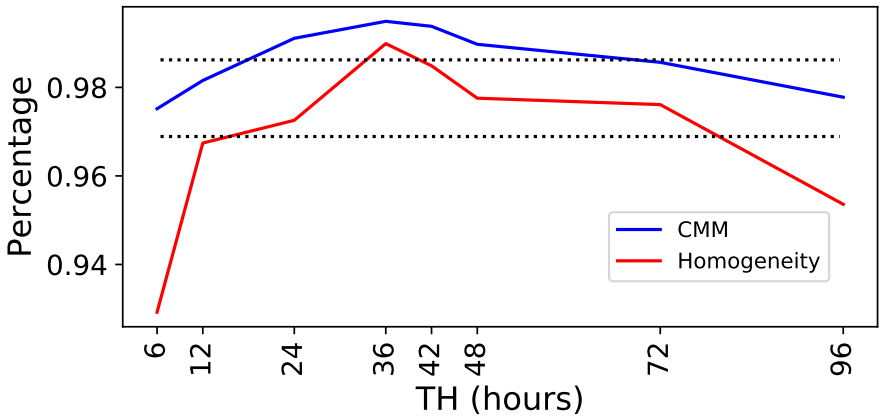

Figure 16. P_6 **metrics for different THs with** $horizon = 1100$**;** $\epsilon = 0.1$**;** $\beta = 0.1$**;** $\mu = 2$**;** $n\_min = 1000$**;** $\lambda = 0.05$**;** $v = 100$**. Mean CMM = 0.986; Mean Homogeneity = 0.969**

adapting the parameters to processes' needs, the complexity of them has less effect on the performance of the method, as seen in the results of processes P_4, P_5 and P_6, which are among the most complicated to model.

Different processes will adapt better to different THs. Thus, process particularities and expected interval of observation must be taken into consideration when choosing a TH value. Also, it is important to consider that this value is related to the desired "range of time" to discover anomalies and deviation in the common behaviour. However, even without the best TH hyperparameters, our approach was able to achieve high metric values.

Process	TH	ϵ	λ	CMM	Homogeneity
P_1	172800	0.2	0.15	98.4%	98.6%
P_2	43200	0.2	0.15	99.3%	97.8%
P_3	21600	0.1	0.15	99.9%	98.9%
P_4	21600	0.15	0.05	99.6%	98.4%
P_5	172800	0.15	0.15	98.9%	95.8%
P_6	151200	0.15	0.05	99.8%	100%

Table 4. Parameters configuration for best performance metrics (CMM and Homogeneity) for each process

6. Conclusions

Organisations currently produce and archive a high volume of data nowadays. This data may contain important information about the organisation's internal processes and may be a helpful tool for improvement.

This paper addressed the problem of finding anomalous cases from an event stream. For that, trace and time features are extracted for each case. Then, the features are feed to an online density-based clustering technique. Since event streams are record-

ings of a running business process, traditional approaches cannot be applied in similar scenarios since they need a complete event log.

Our approach deals with a stream of events and can identify anomalous cases in near real-time by clustering similar cases. On top of that, we have eliminated the need for full-storage and use of traditional batch analysis. In other words, our novel approach allows on the fly detection of anomalous cases, even in front of incomplete executions.

The evaluation metrics used were CMM and Homogeneity. We obtained promising results in both stream performance metrics while varying the hyperparameters, meaning that our proposed technique was able to identify similar cases and differ the common from the anomalous ones in several scenarios.

One of the event logs analysed presented a more complex behaviour than the others, which resulted initially in lower CMM and Homogeneity scores. This motivated the evaluation of different hyperparameters of the DenStream algorithm, which then yield satisfactory results. In this sense, further works should focus on exploring configurations of the DenStream hyperparameters to find for optimal values, which may be unique to each event log.

7. Acknowledgements

We would like to thanks CAPES for providing scholarships.

References

Aggarwal, C. C., Watson, T. J., Ctr, R., Han, J., Wang, J., and Yu, P. S. (2003). A Framework for Clustering Evolving Data Streams. *Proc. of the 29th int. conf. on Very large data bases*, pages 81–92.

Al-Ali, H., Damiani, E., Al-Qutayri, M., Abu-Matar, M., and Mizouni, R. (2016). Translating bpmn to business rules. In *6th International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA)*, pages 22–36. Springer International Publishing.

Barbon, S., Tavares, G. M., da Costa, V. G. T., Ceravolo, P., and Damiani, E. (2018). A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France*. ACM.

Becker, T. and Intoyoad, W. (2017). Context aware process mining in logistics. *Procedia CIRP*, 63:557 – 562. Manufacturing Systems 4.0 – Proceedings of the 50th CIRP Conference on Manufacturing Systems.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604.

Böhmer, K. and Rinderle-Ma, S. (2017). Anomaly detection in business process runtime behavior–challenges and limitations. *arXiv preprint arXiv:1705.06659*.

Bose, R. J. C. and van der Aalst, W. M. (2010). Trace alignment in process mining: Opportunities for process diagnostics. In *BPM*, volume 6336, pages 227–242. Springer.

Bose, R. P. J. C., van der Aalst, W. M. P., Žliobaitė, I., and Pechenizkiy, M. (2011). Handling concept drift in process mining. In Mouratidis, H. and Rolland, C., editors, *Advanced Information Systems Engineering*, pages 391–405, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bose, R. P. J. C., van der Aalst, W. M. P., Žliobaitė, I., and Pechenizkiy, M. (2014). Dealing with concept drifts in process mining. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):154–171.

Burattin, A. and Carmona, J. (2017). A framework for online conformance checking. In *International Conference on Business Process Management*, pages 165–177. Springer.

Burattin, A., Cimitile, M., Maggi, F. M., and Sperduti, A. (2015). Online discovery of declarative process models from event streams. *IEEE Transactions on services computing*, 8(6):833–846.

Burattin, A., Sperduti, A., and van der Aalst, W. M. (2014). Control-flow discovery from event streams. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2420–2427. IEEE.

Cao, F., Estert, M., Qian, W., and Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM.

Carmona, J. and Cortadella, J. (2010). Process mining meets abstract interpretation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 184–199. Springer.

Ceravolo, P., Damiani, E., Torabi, M., and Barbon, S. (2017a). Toward a new generation of log pre-processing methods for process mining. In *International Conference on Business Process Management*, pages 55–70. Springer.

Ceravolo, P., Damiani, E., Torabi, M., and Barbon, S. (2017b). *Toward a New Generation of Log Pre-processing Methods for Process Mining*, pages 55–70. Springer International Publishing, Cham.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press.

Frigge, M., Hoaglin, D. C., and Iglewicz, B. (1989). Some implementations of the boxplot. *The American Statistician*, 43(1):50–54.

Gama, J., Rodrigues, P. P., Spinosa, E., and Carvalho, A. (2010). Knowledge Discovery from Data Streams. *Web Intelligence and Security - Advances in Data and Text Mining Techniques for Detecting and Preventing Terrorist Activities on the Web*, pages 125–138.

Juhaňák, L., Zounek, J., and Rohlíková, L. (2017). Using process mining to analyze students' quiz-taking behavior patterns in a learning management system. *Computers in Human Behavior*.

Kilpeläinen, T. and Tyrväinen, P. (2004). The degree of digitalization of the information over-flow: A case study. In *ICEIS 2004, Proceedings of the 6th International Conference on Enterprise Information Systems, Porto, Portugal, April 14-17, 2004*, pages 367–374.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., and Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132 – 156.

Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., and Pfahringer, B. (2011). An effective evaluation measure for clustering on evolving data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 868–876. ACM.

Leemans, S. J. J., Fahland, D., and van der Aalst, W. M. P. (2014). Discovering block-structured process models from event logs containing infrequent behaviour. In Lohmann, N., Song, M., and Wohed, P., editors, *Business Process Management Workshops*, pages 66–78, Cham. Springer International Publishing.

Leno, V., Armas-Cervantes, A., Dumas, M., La Rosa, M., and Maggi, F. M. (2018). Discovering process maps from event streams. *arXiv preprint arXiv:1804.02704*.

Lévesque, L. (2014). Nyquist sampling theorem: understanding the illusion of a spinning wheel captured with a video camera. *Physics Education*, 49(6):697–705.

Maggi, F. M., Burattin, A., Cimitile, M., and Sperduti, A. (2013). Online process discovery to detect concept drifts in ltl-based declarative process models. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 94–111. Springer.

Mannhardt, F., de Leoni, M., Reijers, H. A., and van der Aalst, W. M. P. (2017). Data-driven process discovery - revealing conditional infrequent behavior from event logs. In Dubois, E. and Pohl, K., editors, *Advanced Information Systems Engineering*, pages 545–560, Cham. Springer International Publishing.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

Reisig, W. (1985). *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer.

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420.

Rozinat, A. and van der Aalst, W. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64 – 95.

Tavares, G. M., da Costa, V. G. T., Martins, V. E., Ceravolo, P., and Barbon, S. (2018). Anomaly detection in business process based on data stream mining. In *SBSI'18: XIV*

Brazilian Symposium on Information Systems, June 4–8, 2018, Caxias do Sul, Brazil. ACM.

Valle, A. M., Santos, E. A., and Loures, E. R. (2017). Applying process mining techniques in software process appraisals. *Information and Software Technology*, 87:19 – 31.

van der Aalst, W., Weijters, T., and Maruster, L. (2004). Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142.

van der Aalst, W. M. P. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Publishing Company, Incorporated, 1st edition.

van Zelst, S. J., van Dongen, B. F., and van der Aalst, W. M. (2018). Event stream-based process discovery using abstract representations. *Knowledge and Information Systems*, 54(2):407–435.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *J. ACM*, 21(1):168–173.

Wang, J., Wong, R. K., Ding, J., Guo, Q., and Wen, L. (2012). On recommendation of process mining algorithms. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 311–318. IEEE.

Weijters, A. J. M. M. and van der Aalst, W. M. P. (2003). Rediscovering workflow models from event-based data using little thumb. *Integr. Comput.-Aided Eng.*, 10(2):151–162.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.