Anomaly Detection in Business Process based on Data Stream Mining

Gabriel Marques Tavares
State University of Londrina (UEL)
Londrina, Brazil
gtavares@uel.br

Victor G. Turrisi da Costa
State University of Londrina (UEL)
Londrina, Brazil
victorturrisi@uel.br

Vinicius Eiji Martins
State University of Londrina (UEL)
Londrina, Brazil
vini9x@gmail.com

Paolo Ceravolo
Università degli Studi di Milano (UNIMI)
Crema, Italy
paolo.ceravolo@unimi.it

Sylvio Barbon Jr.
State University of Londrina (UEL)
Londrina, Brazil
barbon@uel.br

ABSTRACT

Identifying fraudulent or anomalous business procedures is today a key challenge for organisations of any dimension. Nevertheless, the continuous nature of business 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 for evaluating performances. In particular, we obtained Cluster Mapping Measure of 95.3% and Homogeneity of 98.1% discovering anomalous cases in real-time.

CCS CONCEPTS

• **Applied computing** → **Business process management**; **Business process modeling**; • **Information systems** → *Data stream mining*;

KEYWORDS

Process Mining, Business Process Modelling, Online, Fraud, Clustering

ACM Reference format:
Gabriel Marques Tavares, Victor G. Turrisi da Costa, Vinicius Eiji Martins, Paolo Ceravolo, and Sylvio Barbon Jr.. 2018. Anomaly Detection in Business Process based on Data Stream Mining. In *Proceedings of XIV Brazilian Symposium on Information Systems, Caxias do Sul, Brazil, June 4–8, 2018 (SBSI'18),* 8 pages.
https://doi.org/10.1145/3229345.3229362

SBSI'18, June 4–8, 2018, Caxias do Sul, Brazil

ACM ISBN 978-1-4503-6559-8/18/06.
https://doi.org/10.1145/3229345.3229362

1 INTRODUCTION

The growth of automation and the availability of devices with higher storage capabilities greatly increase data production, leading to a phenomenon known as data explosion [**?**]. The increasing level of digitisation and the expanding use of sensors and probes in support of production and business management [**?**] has established information and communication technologies as an intrinsic part of today's society [**?**]. As a consequence, companies are motivated to improve their internal management by increasing the amount of data that are collected for monitoring and controlling business.

However, we have to remark that the availability of a large amount of data represents both, 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. [**?**] highlight that stored business processes data can be beneficial to both organisations as well as attackers that may exploit them. Furthermore, anomalies, such as frauds, exceptions, and errors, should be detected as quickly as possible.

Discovering, performing conformance checks and enhancing procedures comprehend to the field of Process Mining (PM), which is an important tool to aid an organisation to increase the understanding of their data. PM aims at extracting valuable information from an event log (stored business processes data) by discovering, monitoring and improving actual processes [**?**]. PM combines Business Process Management (which uses both information technology and management sciences knowledge to understand operational business processes), data mining and machine learning (ML) [**?**].

The starting point for PM is the log of events. This log records business cases consisting of a sequence of time-ordered events related to the implementation of a business process. Furthermore, these events contain information related to some recorded activity, such as the actor who executed the activity, the time in which it happened, the associated case, among others. A sequence of activities shared by multiple cases is known as a trace [**?**]. The absence or presence of patterns in the event log may indicate the cause of an anomaly, such as fraudulent insurance claim, security violation or a system malfunction [**?**]. Lastly, the output of a PM

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

application is a model that describes the characteristics of a process and how it behaves in the real world, according to the input log.

Gray and Debreceny [?] explored the application of process mining techniques to fraud detection in the audit of financial statements. Jans et al. [?] addressed PM as an adequate answer to mitigating internal transaction frauds inside the corporate scenario. Once this type of fraud is committed using the information systems of a company, all activities are stored in log servers. For example, financial fraud, including credit card fraud, corporate fraud, and money laundering, has attracted a great deal of concern and attention in recent work [? ?]. Another example is in health-care management systems. Many health insurance systems rely on human experts to manually review insurance claims and identify suspicious ones. Additionally, most computer systems, which are intended to help detect undesirable behaviour, require human experts to identify a set of features to develop the core of detection models [?].

In traditional PM techniques, one has access to all available event logs, producing feedback related to processes that already happened. Moreover, most algorithms rely on an event log composed of complete cases, working in batch mode. Actually, an efficient detection has to be performed in real or run time. For this reason, data stream mining (DSM), focusing on prediction models for data streams, is today a relevant research topic. An alternative implementation of online process mining focusing on Cognitive Computing was proposed by Barbon et al. [?]. In this work, the authors proposed a strategy to find concept-drift in business data streams toward providing a response to evolving environments in near-real time. The results were promising, but they did not take into account data streaming performance metrics when addressed the anomalous behaviour. Moreover, the proposal was evaluated with only one real-life process.

In this work, we propose merging PM and DSM techniques taking into account the continuous generation of data and real-time detection of anomalous patterns, even though the data might be incomplete (e.g. cases that not went through its final activity). In this regard, our approach can handle a continuous stream of business event log while, at the same time, extract knowledge from it. Our main goal is to analyse processes and identify their similarity, separating anomalous cases from regular ones. This way, organisations will be able to interpret the currently implied workflow and explore ways of improving their processes of detecting anomalies.

The rest of this paper is organised as follows: Section 2 presents our approach and some concepts related to streaming theory; Section 3 presents the data set and its characteristics, our experimental setup and metrics; Section 4 explores the obtained results; Finally, Section 5 concludes our proposal.

2 PROPOSED APPROACH

Figure 1 gives an overview of our 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 last time-based. Both descriptors encode the behaviour represented by a case in the form of a histogram. This encoding process is known as the conversion phase, and it is discussed in depth in Section 2.1.

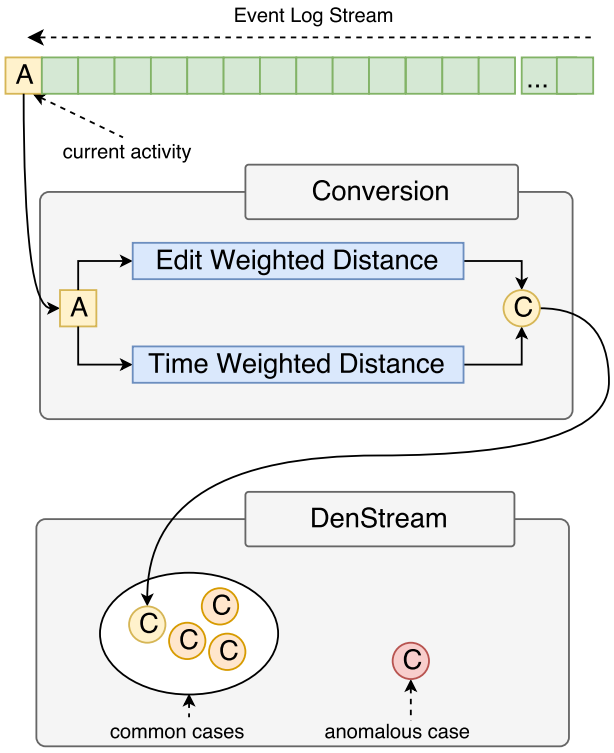

Figure 1: Proposed approach overview

Since we are proposing a streaming approach, the event log allows the access of one event at a time, simulating the constant arrival of information. In other words, the acquisition of a new event implies the consequent update of the encoding of a case.

The conversion phase transforms the event log into a temporary dataset of cases and its associated features, preserving the original order of event acquisition. Using a density-based clustering algorithm, we can identify those cases that belong to the same region, and by consequence, we can mark as outliers instances that are dislocated into low-density regions. The examination of the DenStream algorithm, the paradigms adopted and the way it handles the dataset is presented in Section 2.2.

2.1 Conversion

Traditional PM techniques are applied in batch, considering complex representations of complete cases. Dealing with data streams requires to focusing on data encoding schema as a viable way to reduce space complexity. Our approach address this issue by 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 descriptors were used. The first one deals with the trace and the activities related to it, while the second one deals with the time in 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.

A trace is a sequence of activities, e.g., when arriving at a hospital, a patient is submitted to a protocol. The patient is registered, then

submitted to primary care, then medical advice, then exams are performed, and so on. The order of the activities is a way of telling if the protocol is being followed or not. A patient going through several exams before even being registered is a sign of anomaly. This way, the trace analysis can identify instances where activities happen in an odd order, i.e., possible anomalies. Just as important as the order, is the time in which the activities happen. Assuming the mean time between registration and primary care is thirty minutes, a new case presents a four hours interval. This new case is a possible anomaly too. Thus, a time analysis is also needed to understand processes.

Two hyperparameters control the conversion phase. Since the algorithm starts without a priori data, one has to gather some data to build a model. Inspired by the concept of Grace Period (GP) presented in [**?**], 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 will check the current number of cases and reevaluate if the model needs an update. The optimal number of cases comes from an adaptation of the Nyquist sampling theorem [**?**]. 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, so the Nyquist number is two times that value. 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. Additionally, in this situation, a new histogram is built to represent the standard behaviour better.

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 of comparison between events. The event log contains information related to activities and the time of processing, and by taking into account both these characteristics, our approach proposes a trace analysis, which handles the activities, and a time analysis, which handles the time wherein the activities took place. 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. The trace string, i.e., the sequence of activities, is then compared with the histogram. The same applies to a new case, but the trace string would be composed of only one activity. Inspired by the widely used Edit Distance algorithm [**?**], 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 activities a, b, c, d, and e, we construct the histogram $H = \{1, 2, 5, 4, 1\}$ according to L. 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.

$$Xnorm = \frac{X - Xmin}{Xmax - Xmin} \tag{1}$$

Based on the same principle of the weighted distance but applied to time analysis, we computed the Time Weighted Distance (TWD). 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 [**?**]; 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 histogram.

2.2 DenStream

DenStream is a clustering algorithm developed for evolving data streams proposed by Cao et al. [**?**]. It works around the principles that there are no assumptions about the number of clusters and the shapes of each cluster in the stream. Additionally, it also handles outliers. DenStream uses the concept of core-micro-clusters (c-micro-clusters) to create, delete and modify clusters dynamically. They consist of an approximation of the clustering results of a group of instances and, given that memory is limited, this approximation is necessary. Additionally, c-micro-clusters have weights associated with them which takes into consideration the timestamps of the instances in that micro cluster. Since outliers can often become part of clusters and instances in clusters may become outliers after some time, c-micro-clusters are divided into two basic structures:

(1) Potential c-micro-cluster (p-micro-cluster) which corresponds to the resulting clusters from the clustering process. Since clusters may change along the stream, they can be dismissed and new ones created, hence the name potential;
(2) Outlier micro-cluster (o-micro-cluster) which corresponds to a cluster of outliers. Intuitively, if an o-micro-cluster receives too many instances, then it is promoted to a p-micro-cluster.

The algorithm is divided into two parts: the online micro-cluster maintenance and offline creation of the final clusters. In the first part, both the micro-clusters are updated whenever the algorithm receives a new instance *i*. At first, DenStream will try to add this instance to the nearest p-micro-cluster (according to EWD and TWD, in our case). If it does not succeeds, it will try to add this instance to the nearest o-micro-cluster, categorising that instance as an outlier. If the nearest o-micro-cluster can not incorporate that instance (because it is too far away), then this instance becomes an o-micro-cluster by itself. Additionally, if an o-micro-cluster receives too many instances, it will be promoted to a p-micro-cluster. The second part consists of creating the final clusters for the user by applying the DBSCAN algorithm to the p-micro-clusters. DBSCAN is a clustering algorithm that starts with an arbitrary instance and expands its cluster according to density-based metrics. Given two hyperparameters, *n_min* and ϵ, which corresponds to the minimum

number of instances to build a cluster and the maximum density between two clusters to merge them, the algorithm expands regions until all instances are contained in a cluster or considered outliers and thus cannot be part of any cluster [**?**]. For more information related to DenStream and DBSCAN, please refer to [**?**] and [**?**], respectively.

The DenStream clustering algorithm was selected for this work since it can deal with outlier detection. Additionally, it is also aligned with our time and memory constraints imposed by the data stream. Please note that we interpret outliers as anomalous cases since these cases differ from the normal behaviour of the cases in each process.

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

Table 1: Involved processes statistics

3 METHODS

3.1 Event Log Data Stream

The dataset chosen for the experiments is a business event log from a manufacturing company in Italy, which was first studied and explored in [**?**]. The event log includes 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.

Table 1 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 an outlier. On the other hand, 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.

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 (PNs). PNs are a popular way of representing a process in PM, being the oldest process modelling language [**?**]. A PN is a bipartite graph of *places* and *transitions* connected by arcs [**?**], which is capable of describing concurrent, asynchronous, distributed, parallel, non-deterministic, and stochastic systems [**?**]. 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 [**?**].

In PM, process discovery techniques intent to find a best fitting model that can represent the event log [**?**]. 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 [**?**], we created the PNs representations of our processes. Figures 2, 4, 3, 5 and 6 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.

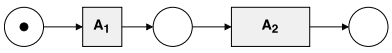

Figure 2: P_1 Petri Net representation

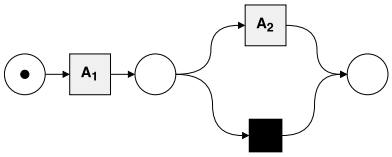

Figure 3: P_3 Petri Net representation

3.2 Experimental Setup

To evaluate our approach, we used the processes on dataset described in Section 3.1. 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 a large collection of methods and tools including classification, regression, clustering and concept drifts tools [**?**].

We executed the DenStream algorithm for each process while varying the TH value. To generate the ground truth for each case

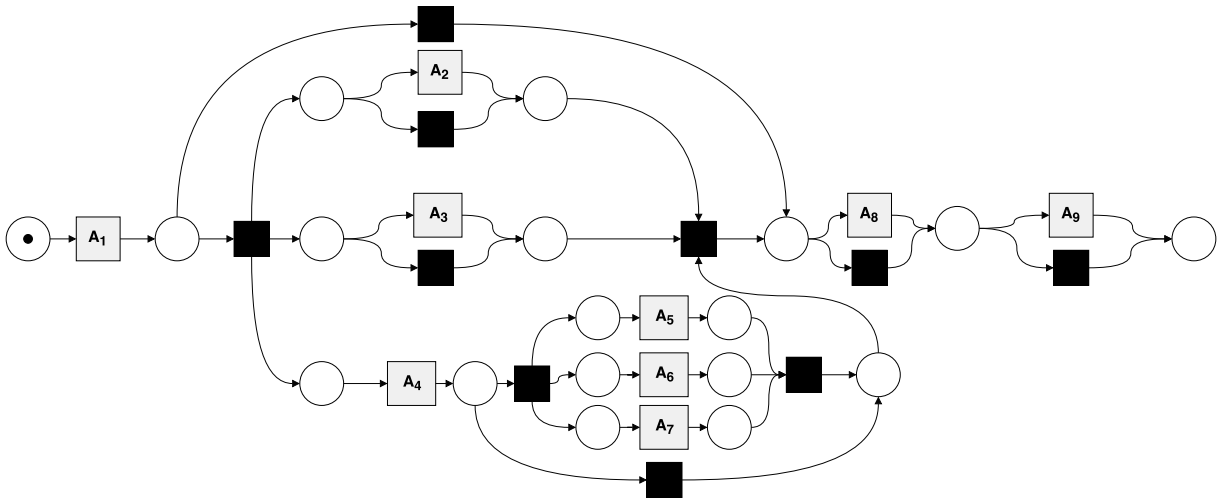

Figure 4: P_2 Petri Net representation

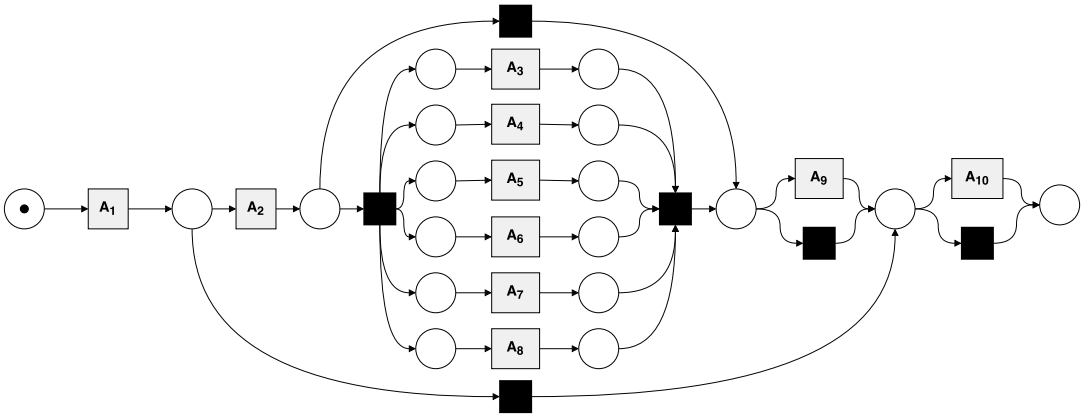

Figure 5: P_4 Petri Net representation

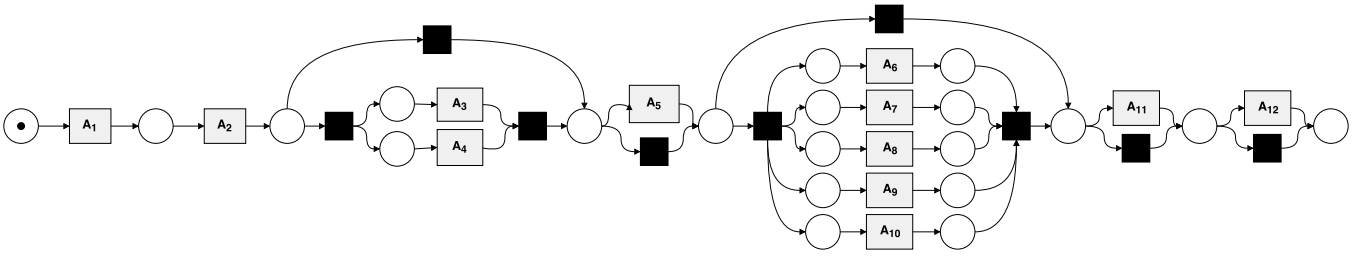

Figure 6: P_5 Petri Net representation

(labelled as common or anomalous), we used the Principal Component Analysis (PCA) [**?**]. PCA was applied in PM for process interpretation in some works [**? ?**]. 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 [**?**]. Figure 7 was obtained from first and second principal components, PC1 and PC2 respectively. PC1 was responsible to explain 51.7% of variance, on the other hand PC2 48.3%.

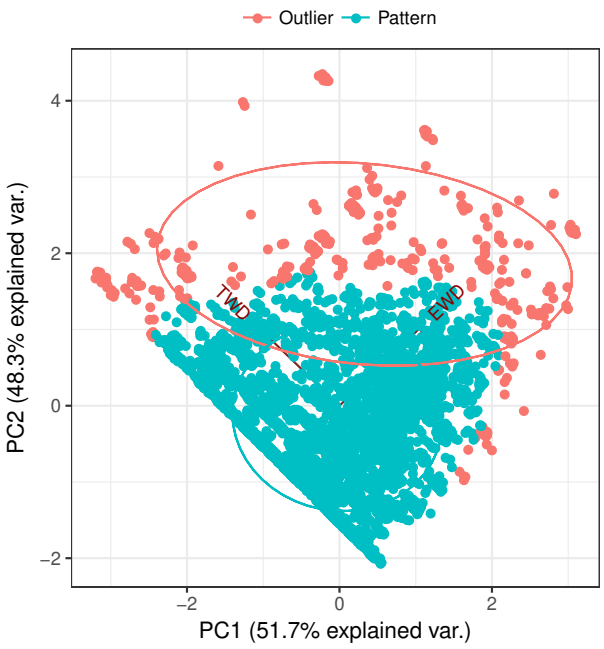

Figure 7: PCA space of *Detail_Supplier_IW-Frozen* (P_5) for anomaly detection (outlier), reddish points are outliers and greenish are common pattern

The DenStream has the following hyperparameters:

(1) Horizon: comprehends to how many instances on the stream are considered - (in our experiments, 1100);
(2) Epsilon (ϵ): this value defines the maximum distance between an instance and a cluster for that instance to be considered inside the cluster - (in our experiments, 0.02);
(3) Outlier Threshold (β): this value defines the threshold for a micro-cluster's weight for it to be considered an either an o-micro-cluster or an p-micro-cluster - (in our experiments, 0.2);
(4) Mu (μ): This value is the minimum weight an overall neighbourhood needs to be considered a core object - (in our experiments, 1);
(5) initN: the amount of points used for initialisation of the DenStream algorithm;
(6) Decay Factor (λ): sets the importance of old data for their current clusters - (in our experiments, 1000);
(7) Stream Speed (v): the amount of instances that arrive at each time - (in our experiments, 100).

3.3 Performance Evaluation

To evaluate our results, we selected CMM (Cluster Mapping Measure) [**?**] and Homogeneity [**?**], since CMM was proven effective when evaluating clustering on data streams [**?**] and Homogeneity is a criteria introduced as a score for clustering solutions and highly considered in the V-measure metric [**?**].

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 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 [**?**].

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 [**?**].

4 RESULTS

The plots in Figures 8, 9, 10, 11 and 12 show the values for CMM and Homogeneity for each of the five processes. All five processes were submitted to a set of 8 TH variations that range from 6 to 96 hours. Since each process describes a precise protocol with specific characteristics, no single TH will present the best metrics across all processes.

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 were aligned.

The graphs show that Homogeneity score was even higher than CMM, in all cases. This means that, for most of the time, our approach was able to differentiate common and anomalous cases, rarely putting them together in the same cluster. This level of abstraction is decisive in an anomaly detection approach.

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

TH value each process

* optimal

Table 2: Performance metrics (CMM and Homogeneity) for each process.

Table 2 presents the performance metrics of the DenStream algorithm. 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 it better, 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. 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.

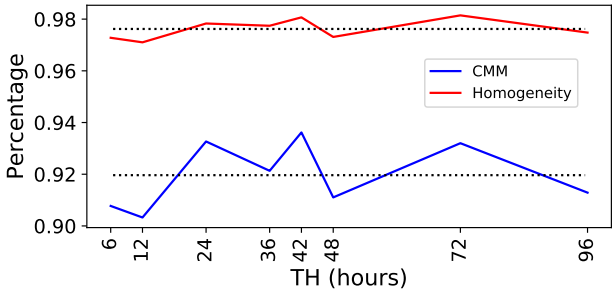

Figure 8: P_1 metrics for different THs. Mean CMM= 0.919; Mean Homogeneity= 0.976

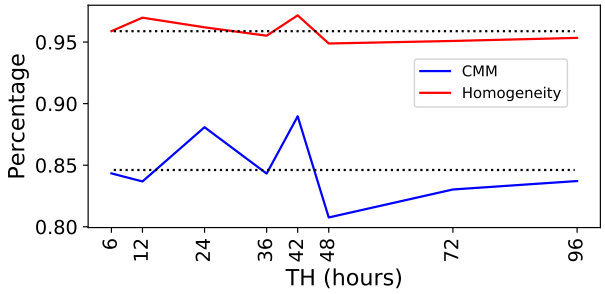

Figure 9: P_2 metrics for different THs. Mean CMM= 0.846; Mean Homogeneity= 0.958

5 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 log at runtime. Traditional approaches fail to adapt themselves to situations of a continuous flow of data. Differently, 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 the incomplete ones.

The evaluation metrics used were CMM and Homogeneity. We obtained promising results with a CMM score of 95.3% and a Homogeneity score of 98.1%, meaning that our proposed technique was able to identify similar cases and differ the common from the anomalous ones in several scenarios.

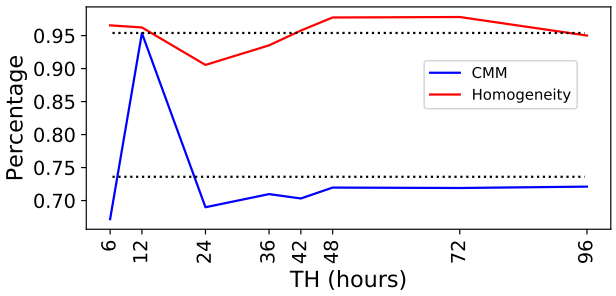

Figure 10: P_3 metrics for different THs. Mean CMM= 0.736; Mean Homogeneity= 0.954

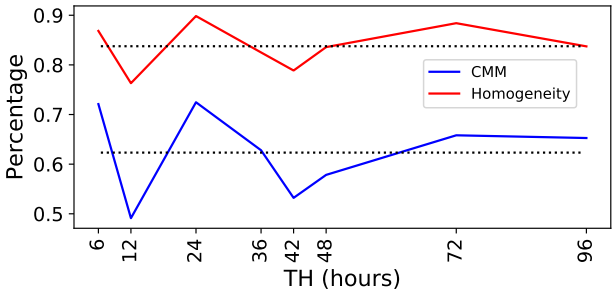

Figure 11: P_4 metrics for different THs. Mean CMM= 0.623; Mean Homogeneity= 0.837

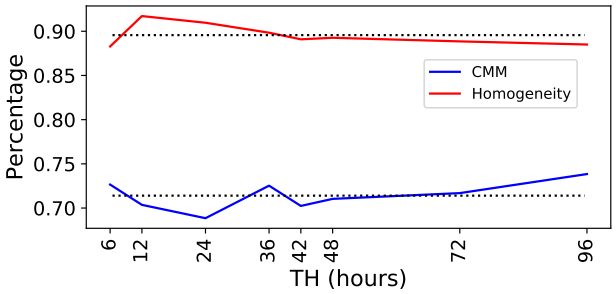

Figure 12: P_5 metrics for different THs. Mean CMM= 0.714; Mean Homogeneity= 0.895

For future work, we aim at dealing with concept-drift, which is the change of standard behaviour through time.

6 ACKNOWLEDGEMENTS

We would like to thanks CAPES for providing scholarships.