Universidade Estadual de Londrina

GABRIEL MARQUES TAVARES

A FRAMEWORK FOR ONLINE ANOMALY AND CONCEPT DRIFT MONITORING IN EVENT LOG STREAM

Londrina
2019

GABRIEL MARQUES TAVARES

A FRAMEWORK FOR ONLINE ANOMALY AND CONCEPT DRIFT MONITORING IN EVENT LOG STREAM

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Sylvio Barbon Junior

Londrina
2019

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Tavares, Gabriel Marques Tavares.
A Framework for Online Anomaly and Concept Drift Monitoring in Event Log Stream / Gabriel Marques Tavares Tavares. - Londrina, 2019.
72 f. : il.

Orientador: Sylvio Barbon Junior Barbon.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2019.
Inclui bibliografia.

1. Online Process Mining - Tese. 2. Data Stream Mining - Tese. 3. Concept Drift Detection - Tese. I. Barbon, Sylvio Barbon Junior. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

GABRIEL MARQUES TAVARES

A FRAMEWORK FOR ONLINE ANOMALY AND CONCEPT DRIFT MONITORING IN EVENT LOG STREAM

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Sylvio Barbon Junior
Universidade Estadual de Londrina – UEL

Prof. Dr. Evandro Baccarin
Universidade Estadual de Londrina – UEL

Prof. Dr. Rafael Gomes Mantovani
Universidade Tecnológica Federal do Paraná – UTFPR

Londrina, 27 de março de 2019.

ACKNOWLEDGEMENTS

First, I would like to thank everyone who collaborated directly or indirectly with the progress of this research. This journey could not be completed without the aid of so many great people that crossed my path.

Especially, I thank my supervisor, professor Sylvio Barbon Jr., for his academic counselling and for being a friend over several years. I greatly appreciate his contributions to this research and his advice. Moreover, I hope we can continue to collaborate for many works to come.

I would like to thank a more recent academic partner, professor Paolo Ceravolo, for his contributions for advances in the area. Moreover, I also thank my lab friends who always constructively reviewed my works and inspired me to thrive for the best.

I profoundly appreciate the support and love from my beautiful girlfriend, who is always there for me. Further, I thank all my family for supporting me during my life and helping me reach my goals. I thank my friends from *i12*, *404* and *31* who have been by my side for several years and for the great life experiences we had together.

Lastly, I would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for the financial support for this research.

"You cannot carry out fundamental change without a certain amount of madness. In this case, it comes from nonconformity, the courage to turn your back on the old formulas, the courage to invent the future. It took the madmen of yesterday for us to be able to act with extreme clarity today."
(Thomas Sankara)

TAVARES, G. M. **A Framework for Online Anomaly and Concept Drift Monitoring in Event Log Stream.** 2019. 72 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2019.

RESUMO

Organizações geram altos volumes de dados relacionados a processos de negócios e frequentemente enfrentam dificuldades para extrair, organizar e interpretar esses dados. Técnicas tradicionais de mineração de processo apoiam a análise e a recuperação de informações de logs de eventos de negócios, modelando, monitorando e aprimorando processos. No entanto, técnicas tradicionais lidam com logs de eventos em uma abordagem em lote (batch), ou seja, usam um log de um processo já finalizado como ponto de partida para análise. Considerando que processos de negócios são registrados continuamente, surge a necessidade de análise em tempo real. Além disso, ao lidar com fluxos, restrições adicionais aparecem, como a mudança da distribuição de dados ao longo do tempo e instâncias incompletas, o que impossibilita a aplicação de métodos convencionais. Dessa forma, este trabalho propõe um framework capaz de lidar com logs de eventos em um cenário de fluxo de dados, sustentando a descoberta de processos, a verificação de conformidade e o aprimoramento de processos. Experimentos com logs de eventos reais e sintéticos foram realizados. Os resultados mostram que a proposta é capaz de identificar anomalias e adaptar seu modelo a novos conceitos. Finalmente, abordagens similares foram comparadas e discutidas.

Palavras-chave: Mineração de processos online. Mineração de fluxos. Mudança de conceitos.

TAVARES, G. M. **A Framework for Online Anomaly and Concept Drift Monitoring in Event Log Stream.** 2019. 72 p. Dissertation (Master's Deegre in Computer Science) – Universidade Estadual de Londrina, Londrina, 2019.

ABSTRACT

Organisations generate high volumes of data related to business processes and often struggle to extract, organise and interpret this data. Traditional process mining techniques support the analysis and information retrieval from business event logs by modelling, monitoring, and enhancing processes. However, traditional techniques deal with event logs in a batch approach, that is, they consume a log of an already concluded process as a starting point for analysis. Considering that business processes are recorded continuously, a need for real-time analysis is raised. Moreover, when dealing with streams, additional constraints appear, such as the change of data distribution over time and incomplete instances, making conventional methods impractical. Thus, this work proposes a framework capable of handling event logs in a streaming scenario, supporting process discovery, conformance checking, and process enhancement. Experiments with real and synthetic event logs were performed. The results show that the proposal is able to identify anomalies and adapt its model to new concepts. Finally, similar approaches were compared and discussed.

Keywords: Online process mining. Data stream mining. Concept drift detection.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

BPM	Business Process Management
CD	Concept Drift
CDESF	Concept Drift in Event Stream Framework
CP	Check Point
DBSCAN	Density-based Spatial Clustering of Applications with Noise
DM	Data Mining
DSM	Data Stream Mining
GD	Graph Distance
ML	Machine Learning
PM	Process Mining
PMG	Process Model Graph
PN	Petri Net
TH	Time Horizon
THG	Time Horizon Graph

CONTENTS

1 INTRODUCTION

Information and communication technologies are an inherent part of today's society [2]. Led by the growth of automation and availability of devices with higher storage capabilities, data has been produced and stored at surprisingly high rates. Much of the produced data is generated by business processes.

According to Dumas et al. [3], a business process is a collection of inter-related events, activities and decisions that together lead to an outcome that brings value to an organisation's customers. Hence, a process is not necessarily the manufacturing of a product, but any procedural task performed inside an organisation. From the given formulations, it derives that any organisation has to manage business processes to deliver a service or provide a product. Moreover, several entities are involved in a business process, such as stakeholders, actors, resources, objects, among others. Finally, note that an organisation may be a company, a governmental body or a non-profit organisation.

Interested in adding more value and being more competitive, organisations are trying to adapt to new technologies by monitoring and controlling their business processes. Business Process Management (BPM) is the area focused on managing business processes to ensure consistent outcomes and to take advantage of opportunities to improve processes [3]. Thus, BPM offers a set of methods to design, analyse, execute and monitor business processes. However, as a wide management science, BPM techniques do not systematically take advantage of the data (event logs) recorded during the execution of processes [1]. In this sense, BPM presents a lack of intelligent solutions which use event logs as the starting point for process analysis.

In this context, Process Mining (PM) poses as a relevant area to interpret organisational generated data. Its goal is to extract valuable process-related information from event logs to discover, conform and enhance business processes. For that, PM concepts combine BPM, Data Mining (DM) and Machine Learning (ML) to establish a link between actual business processes data and their models [1]. Thus, PM takes advantage of modern information systems logs, which produces high volumes of event data, to analyse processes in a bottom-up approach, giving feedback to organisations based on the real execution of their business processes. The typical tasks addressed by PM are: discovering process models, performing conformance checking and enhancing processes [4].

Considering that organisations continuously generate data, traditional ML methods are not adapted to handle the incoming flux of information. Data Stream Mining (DSM) may offer new opportunities for companies to analyse their data in an online manner, i.e. a continuous data stream. However, dealing with streams may impose new

challenges, such as memory and time limitations and the need for continuous adaptation over time [5, 6].

Data streams are a potentially infinite ordered sequence of data items [5]. Thus, storing large volumes of streaming data is often impractical and infeasible from the memory's perspective. Moreover, in nonstationary environments, the data distribution might change over time, yielding a phenomenon known as *concept drift* (CD) [7].

CD happens in an online environment when the relation between the input data and the target variable changes over time. More formally, for two separate points in time, the underlying relationship between the feature vector $\vec{x}$ and the system response y changes. As a consequence of CD, the learning model decreases its representational capacity and needs to be updated to keep its relevance.

In contrast with DSM, traditional PM techniques expect static data as input, that is, an event log of an already finished business process. This poses a dilemma since organisations are interested in monitoring their processes and acting upon them, preferably while they are still running. For this reason, traditional PM methods are not suitable for online scenarios where (i) the event log is too large (possibly infinite), (ii) processing resources (memory and time) are limited, (iii) immediate response to anomalies is required, or (iv) adaptation to CD is necessary [8].

Given the described scenario, dealing with business processes in an online environment requires knowledge from both PM and DSM. However, more challenges arise regarding a discrepancy at the representation level between both areas. DSM typically works at the event level, where each event represents an instance, and its values are independent of other events. Differently, PM is set at the case level, where multiple recorded events compose a process instance (case), meaning that events are correlated with others. Such constraint requires that online PM techniques deal with unfinished process instances, i.e. incomplete cases.

Recently, a growing interest in extending PM techniques to handle data streams has been increasing [8, 9, 10, 11, 12, 13, 14]. Most approaches rely on an incremental analysis where each PM task is tackled individually every time a settled bucket of events is received [13]. Nevertheless, PM analytic tasks are complementary, and by treating each one separately, the interconnections between them are ignored. This is evident when treating CD. Performing process discovery or conformance checking techniques without taking CD into account might be misleading. At the same time, detecting CD without providing the process insights might not be helpful for stakeholders.

1.1 Hypothesis

Dealing with PM needs in a streaming scenario is still an emerging field in the research community. This leads to several questions which are not entirely investigated and answered, as pointed in the Process Mining Manifesto [15]. However, the intersection between PM and DSM is not trivial as there are several mismatches in the way data is treated. Moreover, PM challenges go further since methods need to be accessible to non-experts stakeholders. This leads to the first hypothesis investigated:

Hypothesis 1. *Evaluate the viability of dealing with PM tasks, namely, process discovery, conformance checking and process enhancement, in a streaming scenario, which requires adaptation to CD. Additionally, this would provide a complete view of the process for stakeholders, acknowledging several PM needs.*

Further, proposed approaches for online PM usually consider only the trace perspective (trace is the sequence of interconnected events from the same process instance[1]). However, process instances contain several attributes, which leads to the second hypothesis:

Hypothesis 2. *Propose a different perspective, namely the time delta between events from the same process instance. Furthermore, when taking into account one more perspective, it would be possible to identify anomalies and new behaviours.*

1.2 Objective

The goal of this work is to investigate the merging of PM and DSM techniques taking into account the continuous generation of event logs for real-time detection of anomalous patterns and adaptation to change in business processes. Thus, offering organisations ways to interpret the currently implied workflow and improve their processes.

In this regard, another objective is to present the Concept Drift in Event Stream Framework (CDESF). The approach compares cases based on multiple perspectives, i.e. trace sequence and time delta between events. This is achieved by the adoption of a graph representation of the process model and by using a density-based clustering algorithm to identify common behaviour over time, highlighting the outliers. Finally, using the latest events, CDESF updates its process model, thus, handling process changes.

1.3 Contributions

This dissertation deals with PM tasks in an online environment. Moreover, it considers several constraints and the interconnection between the several PM tasks. Thus,

[1] The detailed definitions are discussed in Chapter 2

the main contributions are:

- Development of a framework to handle multiple PM tasks in a streaming scenario. Further, presenting a detailed study regarding the relation between hyperparameters and outcomes;
- Proposing time delta between events as a process instance attribute, shining a light into a new perspective for analysis of business processes;
- Reducing the gap between PM and DSM research by adapting a DSM clustering technique to encode business processes correctly;
- Comparing the proposed approach with similar methods [11, 12, 16], and measuring their effectiveness while also considering business stakeholders needs;
- Providing all code for reproducibility[2] and further research while datasets are available according to their respective references.

1.4 Outline

The remainder of this work follows the subsequent order: Chapter 2 presents the theoretical fundamentals necessary to understand the basis of the work. In Chapter 3, we present related research that has similar goals or propose similar techniques. Chapter 4 presents Concept Drift in Event Stream Framework in-depth. Chapter 5 describes the methods and the event logs used, and their characteristics. Chapter 6 presents the results obtained from experiments, its implications and a discussion regarding them. Furthermore, a comparison with similar techniques is performed. Finally, Chapter 7 concludes the work, discusses its limitations and open avenues for future work.

[2] https://github.com/gbrltv/CDESF_v2

2 THEORETICAL FUNDAMENTALS

This work is mostly based on Process Mining and Data Stream Mining techniques. Thus, this chapter presents some essential concepts for this research and necessary for further understanding.

2.1 Process Mining

The intense growth of available technological devices and the capability of recording data produced by such devices give organisations an overabundance of data. The immense quantity of stored data contains valuable information for organisations, and the PM field provides a collection of techniques to extract insightful information so that institutions can improve their processes. In a broad perspective, PM is the area interested in the exploration of process-related data aiming at modelling, monitoring and enhancing processes.

PM comes from the intersection of two traditional areas: Data Mining and Business Process Management. DM is the process of discovering patterns in data by performing automatic (or semiautomatic) methods. The discovery of underlying data relations must lead to an advantage and be meaningful [17]. Furthermore, DM provides a set of techniques that aim at creating models and extract information from datasets. BPM unites process knowledge for the design, configuration and monitoring of business processes to improve the performance of processes inside organisations.

Since process-related data is structured differently from usual DM datasets, traditional DM methods cannot be readily applied to event logs. At the same time, BPM techniques fail to provide intelligent and automated data analysis because BPM analysis starts from an expert's knowledge and not from data. Thus, PM merges knowledge from these areas to provide an insightful analysis based on event logs. The aim is to discover, monitor and enhance business processes by extracting knowledge from event logs [1].

Furthermore, PM contains three main tasks:

- 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 [18], the Inductive Miner [19], the Heuristic Miner [20], among others. Discovering methods need to abstract the connections between process activities to be capable of building a model that represents concurrency and, at the same time, deal with noise data;
- Conformance checking: compares an event log and a model from the same process

to check deviations between the model and the log [21]. When finding deviations, conformance algorithms also may provide metrics to measure such deviations. For instance, finding a process instance that does not conform to the model is an indication of a possible anomalous instance. However, if most of the process instances from the event log do not conform to the model, then the model is probably outdated and does not represent the log;

- Process enhancement: deals with the improvement of the process model using information from the event log. The enhancement may either *repair* the model by correcting it or *extend* the model by updating it with new information extracted from the log.

All three tasks of PM share the same principle, that is, they all use the event log as the starting point. Moreover, the tasks correlate among themselves, e.g. a process discovery algorithm may provide a model which can later be conformed and enhanced.

Figure 1 shows where PM tasks are placed inside a business management environment. The knowledge from organisations, business processes and people is used to create, support and control software systems, which records data in the form of event logs. From there, PM techniques use the event log to generate models, conform executions and enhance processes. Hence, the model interacts directly with the knowledge generation while it can also be used to extract specifications and update the software systems. As seen in the figure, PM methods heavily rely on the use of event logs to be performed.

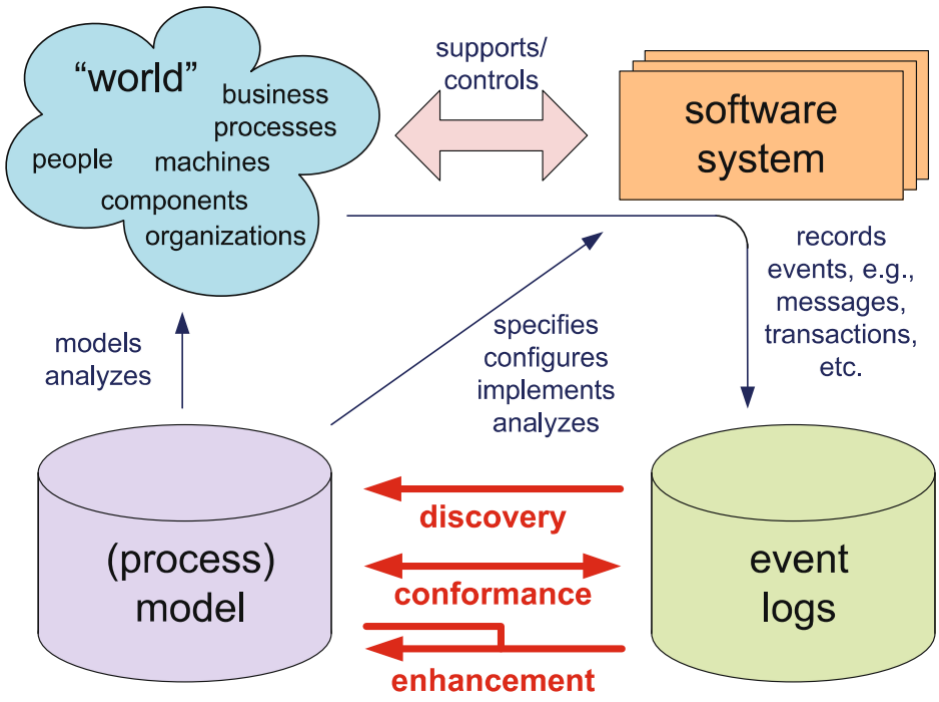


Figure 1 – Application of PM techniques inside a business management environment. PM tasks are performed over a event log (extracted from [1])

2.1.1 Event Log

Analysing an event log is essential to understand better the PMs concepts. The event log represents generated data inside an organisation. It is presumed that an event log is the recording of a single business process. Thus, PM techniques usually expect to consume event log data. Table 1 shows an example of a typical event log used for PM. Each row of the table represents one *event*, which is an execution of an *activity* at a certain *time*. Moreover, each activity corresponds to a single process instance, commonly referred to as *case*. In this example, activities are atomic. However, activities might have start and completion times in case the activity takes time to be executed.

All traditional PM techniques consider a process as a collection of activities which are grouped by their process instances, such that the life-cycle of a case is described. This way, the "Case ID" and the "Activity" columns in Table 1 are necessary elements for any PM application.

Furthermore, a sequence of activities from the same case is known as a *trace* and different cases may present the same trace, i.e., the same sequence of activities. From Table 1, we can infer that Case 4 trace is composed of activities `Process Creation`, `Weight` and `Design Lead`. Also, Case 1 and Case 3 present the trace: `Process Creation`, `CRA` and `Design Checker`. Finally, the group of cases 1, 2, 3 and 4 is a set of recorded events generated from the same business process.

Table 1 – Example of an event log. Each row of the table represents an event, i.e., the execution of an activity at a given time. Moreover, each event is associated with a specific case (process instance)

Case ID	Activity	Complete Time
Case 1	Process Creation	2012/05/07 07:00:00
Case 3	Process Creation	2012/05/07 08:03:58
Case 4	Process Creation	2012/05/07 08:08:43
Case 3	CRA	2012/05/08 14:10:12
Case 1	CRA	2012/05/08 15:15:00
Case 1	Design Checker	2012/05/08 16:00:00
Case 4	Weight	2012/05/08 16:10:35
Case 2	M_P	2012/05/09 12:00:50
Case 2	Stress	2012/05/13 08:27:10
Case 2	ME Assembly Checker	2012/05/13 10:30:00
Case 3	Design Checker	2012/05/15 17:00:20
Case 4	Design Lead	2012/05/15 17:00:56

Events are activities recorded at a specific time. Further, it is expected that event logs have their events timely ordered, this way the event log will represent the real order of execution of the process instances. In Table 1, all necessary attributes of an event log (case identifier, activity and timestamp) are present. However, events may be associated with

more descriptors, such as the execution cost of an activity, the author (who executed), resources exploited during execution, among others.

Figure 2 puts into perspective the relations of an event log. Thus, we can infer that a process consists of cases. A case contains several events where each event relates to one and only one case. Events have attributes, such as activity name, time, and resource. Finally, the sequence of activities from the same case is referred to as a trace.

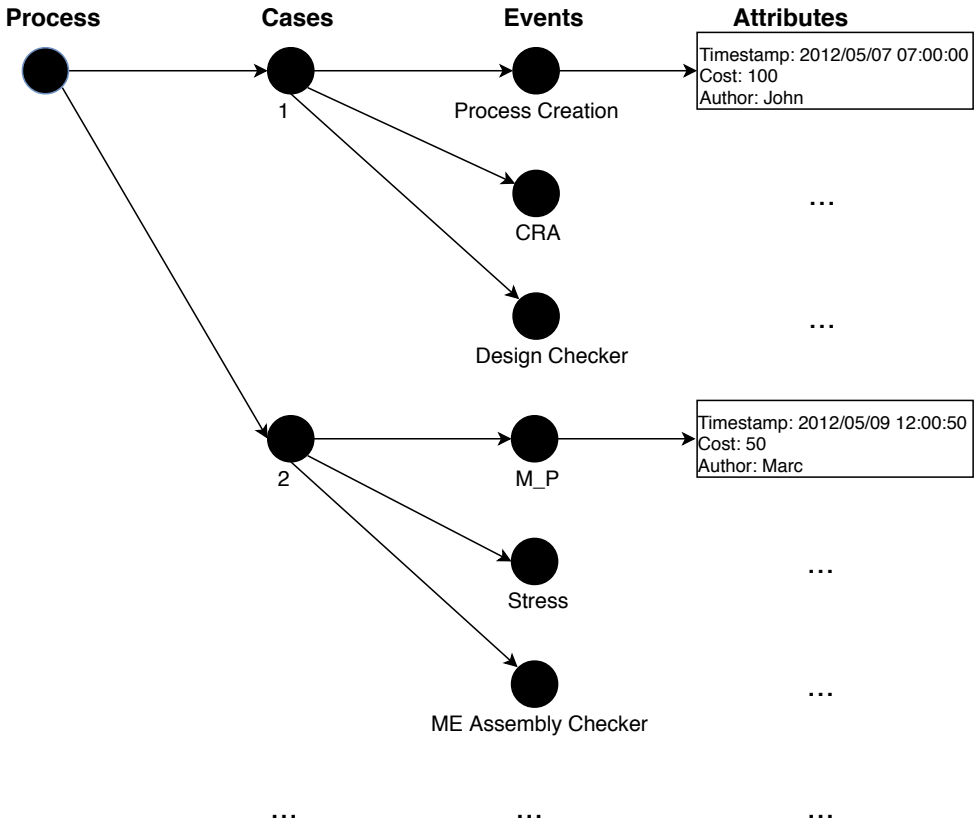


Figure 2 – Relations between attributes, events and cases inside an event log

Definition 1 (*Event*, *attribute* [1]). Let Σ be the *event universe*, i.e., the set of all possible event identifiers. Events may have several *attributes*, such as timestamp, activity, resource, associated cost, among others. Let AN be the set of attribute names. For any event $e \in \Sigma$ and name $n \in AN$, then $\#_n(e)$ is the value of attribute n for event e, if event e has an attribute n, else $\#_n(e)$ is null.

In this work, we assume the following standard attributes of an event:

- $\#_{cid}(e)$ is the *case identifier* of event e.
- $\#_{activity}(e)$ is the *activity* associated to event e.
- $\#_{time}(e)$ is the *timestamp* of event e.

Definition 2 (*Trace*, *case*, *event log* [22, 1]). A *trace* is a non-empty sequence of events $\sigma \in \Sigma^*$ where each event appears only once and time is non-decreasing, i.e., for $1 \leq i < j \leq |\sigma|$: $\sigma(i) \neq \sigma(j)$. Let $\mathcal{C}$ be the *case universe*, that is, the set of all possible case identifiers. An *event log* is a set of cases $L \subseteq \mathcal{C}$ where each event appears at most once in the log, i.e., for any two different cases, the intersection of their set of events is empty.

2.1.2 Model Notation

Even though event logs are essential to process analysis, they may not be readily understandable for non-experts. This way, high-level organisations use process modelling as an important tool to create a human-readable representation of a process. Thus, models can be used to analyse and be acted upon. For that, BPM offers a plethora of modelling notations, such as Petri Nets, Transitions Systems, Workflow Nets, YAWL and BPMN [1].

In this work, we have chosen to use Petri Nets (PN) as a model to represent processes because PNs are a frequently used model for business processes, being one of the oldest and best investigated process modelling languages [1]. Also, PNs notation is intuitive and at the same time several analysis techniques can be used with them [23, 24, 25]. Moreover, PNs are more intuitive than other notations for non-experts.

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 [26]. This way, PNs can represent casual dependencies in sets and systems with different levels of abstraction while also verifying systems properties [27].

Furthermore, since PN models are powerful yet intuitive, one can easily apply an analysis technique. PNs relate to event logs in the way that process discovery algorithms consume event logs to produce a PN that represents the model. It is worth noticing that different discovery methods may produce different PNs, meaning that there is no gold standard PN for an event log since several ways of building relations within the process exist.

Figure 3 shows the PN produced by applying the Inductive Miner [19] algorithm to the event log from Table 1 as an input. Since the event log is small, the produced PN is not complex, but it shows the main characteristics of a PN model. The token is placed at the starting point of the PN. From there, the token is consumed to either follow the arc to `M_P` or `Process Creation`, which means that those are the two initial activities for any case in this process. Hence, both activities never follow or precede one another, either one or the other will start the case. This assumption is true when checking Table 1: three cases start with `Process Creation`, and the remaining one starts with `M_P`. Following the same principle, the token is then consumed until it reaches a final state, i.e., where it can no more be absorbed, thus forming a trace. All traces from Table 1 event log conform

to the model and can be replayed.

Definition 3 (*Petri Net* [1]). A *Petri Net* is a triplet $N = (P, T, F)$ where P is a finite set of *places*, T is a finite set of *transitions* such that $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed *arcs*, called the flow relation.

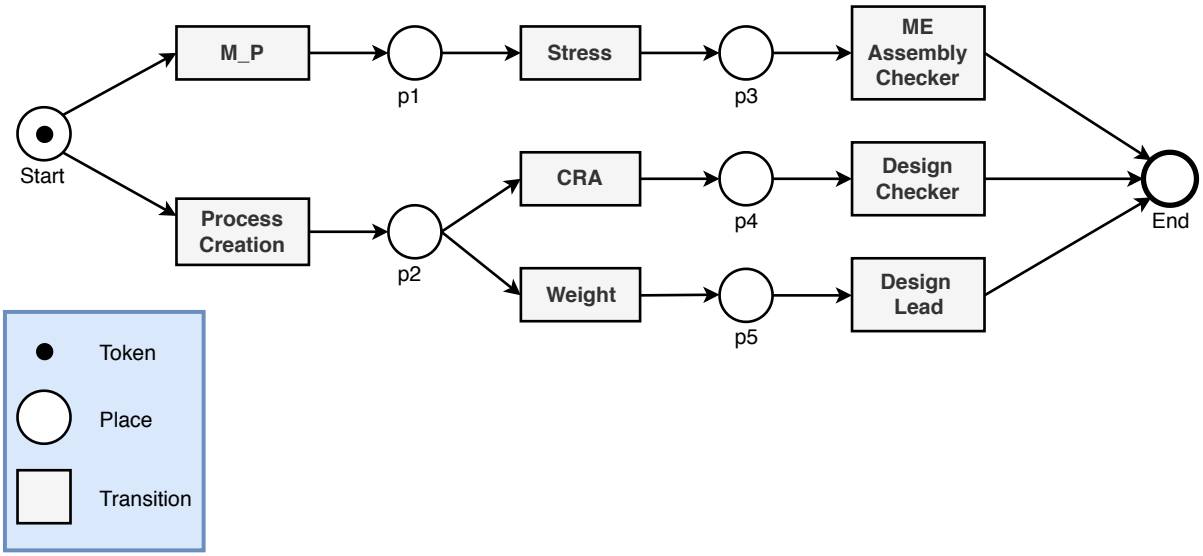


Figure 3 – PN model created using Inductive Miner as the discovery algorithm and the event log from Table 1 as input. The token is consumed and goes through the places. Transitions represent the events executed and are denoted by their activity names

Formally, the Petri Net from Figure 3 can be defined as: $P = \{$*Start*, *p1*, *p2*, *p3*, *p4*, *p5*, *End*$\}$, $T = \{$*M_P*, *Process Creation*, *Stress*, *CRA*, *Weight*, *ME Assembly Checker*, *Design Checker*, *Design Lead*$\}$, and $F = \{$(*Start*, *M_P*), (*Start*, *Process Creation*), (*M_P*, *p1*), (*Process Creation*, *p2*), (*p1*, *Stress*), (*p2*, *CRA*), (*p2*, *Weight*), (*Stress*, *p3*), (*CRA*, *p4*), (*Weight*, *p5*), (*p3*, *ME Assembly Checker*), (*p4*, *Design Checker*), (*p5*, *Design Lead*), (*ME Assembly Checker*, *End*), (*Design Checker*, *End*), (*Design Lead*, *End*)$\}$.

Inspired by the fact that PN is a graph-based representation of a model, the Concept Drift in Event Stream Framework uses graphs as the form of process models. Business processes have been represented as graphs in several works [28, 29]. By using a graph-based process representation, we take advantage of graph distance metrics for feature extraction. In our representation, the graphs' nodes represent activities, while edges represent that an activity is either followed or preceded by another, which we refer to as a *transition*. Each transition contains additional information regarding activities frequency and the computed mean time between two linked activities.

The proposed framework creates a graph model to represent the process, referred to as the *Process Model Graph* (PMG). Figure 4 depicts a simple PMG. Nodes are labelled as activity names while edges are transitions. A tuple is associated with each transition. The first value is a frequency count and the second is the mean time of the transition.

Example 2.1.1. Given the trace $\langle A_1, A_2, A_3 \rangle$ from case c_1, we can infer that c_1 contains two transitions: A_1A_2 and A_2A_3. Moreover, transition A_1A_2 occurred 10 times and its mean time is 5000 (independent of the time unit). Likewise, transition A_2A_3 occurred 9 times and its mean time is 3000.

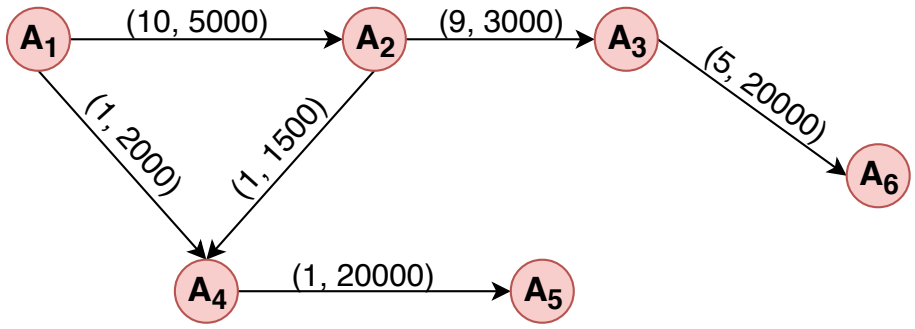


Figure 4 – Process Model Graph (PMG) of a simple business process. Nodes represent activities and edges the transition between two activities. The tuple contains frequency and mean time of a transition

Definition 4 (*Process Model Graph*). A Process Model Graph PMG is a directed graph $PMG(A, T)$ where $a_i \in A$ is an activity $\#_{activity}(e)$ and $t_i \in N \times N$ is a transition between two activities a_i, a_j, denoted a_ia_j. The paths of a PMG describe the traces that can be generated in the PMG, this set is infinite if a path includes loops (at least one edge relates to an activity already included in the path).

2.2 Data Stream Mining

Traditionally, ML algorithms create models using static datasets. However, with higher availability of technological devices, real-time generation of data is a modern phenomenon. For instance, network flow, sensor data, web and e-commerce usage generate a high volume of data and demand for new ML solutions to deal with incoming data [30]. Traditional ML techniques can operate with multiple passes over the data without worrying about processing time and memory usage. On data streams, the assumption that all training data is available to create a model is not correct, i.e., over different time periods, the learning system has to deal with different distributions of data. Moreover, the stream is potentially infinite, which causes restraints in the usage of processing capabilities.

Definition 5 (*Stream* [5]). A *stream* $\mathcal{S}$ is defined in the format $\mathcal{S} = \{i_1, i_2, i_3, ..., i_n, ...\}$, where i corresponds to a pair $P(\vec{x}, y)$ when the class for that instance is known or simply $\vec{x}$ when it is not, with $\vec{x}$ being the feature vector of that instance and y being its label, and n is possibly infinite.

Dealing with sequential data creates an additional challenge, which is the distribution of data changing over time. The phenomenon is known as *concept drift*, which

deteriorates the model performance.

Definition 6 (*Concept drift* [5, 6]). Given the sequence of streams $\langle \mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_i, ... \rangle$ where $\mathcal{S}_i$ is a set of examples generated by some distribution $\mathcal{D}_i$. Given $P^t(\vec{x}, y)$, at each timestamp t, the feature vector $\vec{x}^t$ corresponds to a class y^t. Given two distinct points in time t and $t + \Delta$, given $\mathcal{D}^t \neq \mathcal{D}^{t+\Delta}$, if there is a $\vec{x}$ that satisfies $P^t(\vec{x}, y) \neq P^{t+\Delta}(\vec{x}, y)$, then a *concept drift* has happened.

To counter the drift, the model may update using new data, improving the way it represents the data. In a drift scenario, old observations become obsolete since they represent the nature of past instances. Consequently, the learning agent must forget this information and induce knowledge from newer instances.

Several types of changes through time can take place, i.e. there are different types of drift. Identifying the type of drift can be handy when dealing with its consequences. There has been an ongoing discussion to characterise drifts [31], and taxonomies have been proposed [32].

Figure 5 shows the most common drift types:

- Sudden drift, also known as abrupt, occurs in a specific moment t when a different distribution $\mathcal{D}_{t+1}$ suddenly replaces a distribution $\mathcal{D}_t$;
- Recurring drift is a sequence of changes where prior distributions reappear after some time. This behaviour can be caused by a seasonal occurrence, for example;
- Gradual drift is a state of transition between two distributions, $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$, where the probability of $\mathcal{D}_i$ occurrence gets less frequent while the probability of $\mathcal{D}_{i+1}$ happening increases;
- Incremental drift is characterised by small-scale changes over time. These small changes are represented by the orange colour in Figure 5. During this transition phase, the distribution is slowly mutated until it reaches a final form.

It is important to notice that drift detection is different from anomaly detection. Drift is the result of a change in the distribution between the feature vector and its class while an anomaly is a deviation from the standard behaviour and should be identified by the model without the necessity to update it.

2.2.1 Density based clustering

Though DSM offers several algorithms for online classification, they are not applicable in our scenario because that would require a labelled business process, which is impractical in most cases due to the stream rate. For this reason, we opted to use clustering based techniques.

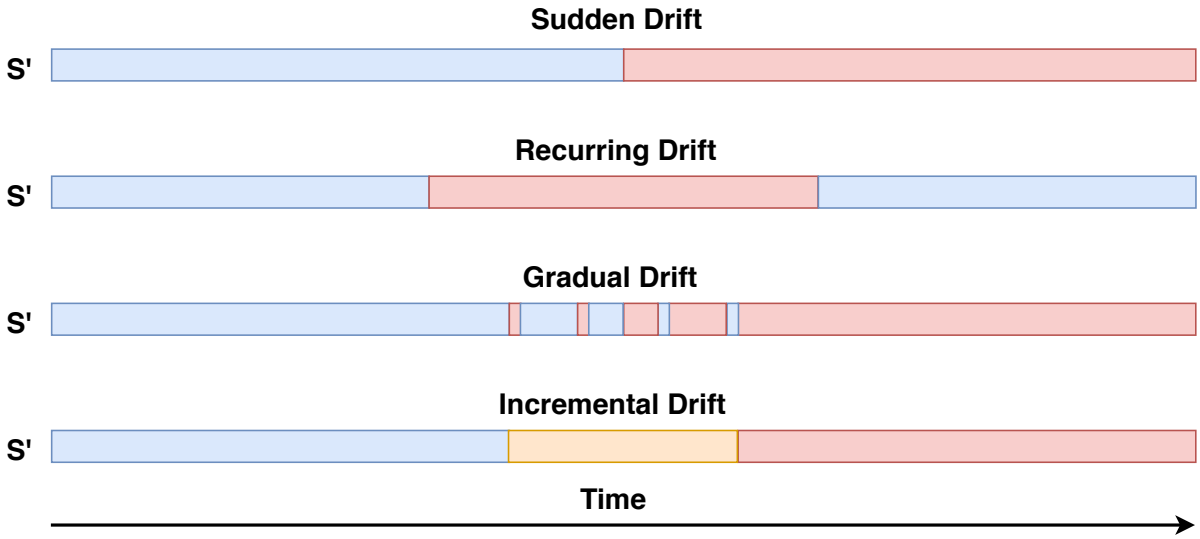


Figure 5 – Several CD types and how they affect the stream over time

Despite several proposed stream clustering algorithms, the DenStream algorithm was chosen 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.

Cao et al. [33] proposed DenStream as an adaptation of a density-based clustering algorithm for data streams which imposes no assumptions over the number of clusters and their shape while it also handles outliers. Furthermore, DenStream dynamically creates, updates and deletes clusters using limited memory by applying the concept of micro-clusters (MC) proposed by Aggarwal [34].

Definition 7 (*micro-cluster*, *centre*, *radius* [34]). A *micro-cluster* is 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. A micro cluster has a *centre* and a *radius* defined as Equations 2.1 and 2.2, respectively.

$$\frac{\overline{CF^1}}{w} \tag{2.1}$$

$$r = \sqrt{\frac{|\overline{CF^2}|}{w} - (\frac{\overline{CF^1}}{w})^2} \tag{2.2}$$

The MC concept is then applied in three ways. The core MC (C-MC) is a dense MC that respects the $w \geq \mu$ rule, where μ is a hyperparameter that controls the minimum density of a MC to be considered a C-MC. The potential C-MC (P-MC) is a less dense MC that follows $w \geq \beta\,\mu$, where β is a hyperparameter that determines the threshold of

an outlier relative to C-MC and assumes $0 < \beta \leq 1$. Lastly, the outlier MC (O-MC) is composed of outlier instances where $w < \beta\,\mu$.

DenStream is then divided into three main parts. In the initialisation step, a modified version of the Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [35] is used to cluster the initial instances. Starting with a random instance, all distances with maximum distance ϵ (hyperparameter) are considered to be part of the same group (the implied distance measure is optional, we have used the traditional Euclidian distance). If the number of instances in the group is higher than $\beta\,\mu$, then the group is initialised as a P-MC. This process is repeated until all instances are either part of a group or cannot form any new MCs.

Then, for each new instance i, DenStream tries to incorporate this instance to the nearest P-MC pmc_n. If, after incorporating i, $r_{pmc_n} \leq \epsilon$ holds true, then the instance is added to that P-MC. Otherwise, the algorithm tries to add i to the nearest O-MC omc_n, employing the same test. If it cannot incorporate i (i.e. because it is too far away), then a new O-MC containing only i is created. Additionally, a procedure is employed to check if the O-MC weight. If $w_{omc_n} > \beta\,\mu$, then omc_n is promoted to a P-MC. The pseudo-code 1 illustrates the described steps.

Algorithm 1 Merging (i)

```
Try to add i into its nearest P-MC pmc_n
if r_{pmc_n} ≤ ε then
    Merge i into pmc_n
else
    Try to add i into its nearest O-MC omc_n
    if r_{omc_n} ≤ ε then
        Merge i into omc_n
        if w_{omc_n} > β μ then
            Promote omc_n to a P-MC
        end if
    else
        Create a new O-MC with i
    end if
end if
```

This way, P-MCs and O-MCs are updated online as new instances arrive and are assigned to them. After v instances (hyperparameter), the P-MCs and O-MCs which were not updated by any new instances have an exponential decay applied to them, defined as $2^{-\lambda}$ (λ is a hyperparameter). Such behaviour implies that P-MCs and O-MCs situated in inactive regions of the feature space suffer the decay factor until an eventual fade away. In contrast, P-MCs and O-MCs that receive new instances increase their weight, becoming denser.

The last step applies the DBSCAN algorithm to create the final clusters, i.e. C-

MCs, for user visualisation. For that, an arbitrary P-MC p_a is selected, and the algorithm scans for other P-MCs that satisfy $distance(p_a, p) <= r_{p_a} + r_p$, where r_{p_a} and r_p are the radii of selected P-MCs. The P-MCs that meet the condition are then added to the same cluster. This same scanning process is repeated for each new P-MC in the cluster until no new P-MCs are added into the cluster. DenStream then checks the weight of the formed cluster (which is the summed weight of the P-MCs located inside), if the weight is higher than μ, the cluster is a C-MC. The last step ends either when all P-MCs are part of a final cluster or cannot be incorporated into any.

Finally, Table 2 presents the relation between DenStream hyperparameters and what they are associated with during the execution of the algorithm. Their relation with the output of the algorithm will be explored in the further chapters.

Table 2 – List of DenStream hyperparameters and their roles in the algorithm

Hyperparameter	Description
ϵ	Represents the maximum value a MC radius can assume
μ	Controls the minimum density of a MC to be considered a C-MC
β	Determines the threshold of an outlier relative to P-MC. It is used along with μ
λ	Decay factor applied into existing MCs. Sets the importance of historical data
v	Controls the application of the decay factor. That is, the decay factor is applied after v instances

3 RELATED WORK

In recent years, several works addressed the PM topic in streaming environments, with many focusing on process discovery [8, 36, 37, 38], conformance checking [39, 40] and concept drift detection [11, 41, 42]. These techniques vary in several ways, such as model representation, updating strategy, addressed attributes, among others. We focus our review on works that detect CD in business processes. Lastly, assume that an event log in a streaming scenario is also referred to as an *event stream*, i.e., a stream of recorded business processes events.

3.1 Definitions of Concept Drift for Business Processes

As in other scenarios, a business process may present drifts during its execution, especially considering that a process may run for several years. Thus, being susceptible to changes either from inside or outside the organisation. These changes are also referred to as hidden contexts. However, the study of the intersection between PM and DSM areas is still new. Therefore, there are few works that propose definitions for drifts in business processes [10, 11, 43]. Usually, they inherit definitions from data stream literature.

In DSM research, the traditional drift definition considers the change of the data distribution ($\vec{x}$) over time in relation to its label (y) [5]. However, traditional PM event logs do not contain label information, which results in a mismatch between DSM and PM descriptions of drift. PM definitions overlooked the fact that drifts occur in scenarios where each instance has an associated label. Finally, labelling is time-consuming and frequently impractical in real event logs.

Given this context, Burattin et al. [44] create an analogy with traditional data streams to state that an event stream can either be *stationary* or *evolving*. The authors define that a stationary stream is the result of a business process that does not change over time. Contrarily, an evolving stream is generated by a process that changes during time. Further, there are three types of evolving streams: (i) drift of the process model, (ii) shift of the process model, and (iii) cases distribution change. Process model drift is a gradual change of the process model while a model shift in an abrupt change between two process models. The change in cases means the distribution of the features of the process cases changes with time, i.e., distribution of cases is not stationary.

Seelinger et al. [43] define a *process drift* as a significant behavioural change of the process execution over time. Additionally, almost all traces that arrive after a process change follow the drift behaviour. Moreover, the authors state that process drifts can be *planned* or *unexpected*. The first type refers to changes in guidelines or regularities. Fur-

thermore, organisations can check if the desired change is executed correctly. The second type refers to unexpected business scenarios, such as new employees or changes in the resources capacities. Further, the early detection of unexpected changes helps organisations prevent financial consequences.

More quantitatively, Maaradji et al. [10] formulate business process drift detection as a point in time where there is a statistical difference between the process behaviour before and after the said point.

In [11], Bose et al. state that drifts are situations where the process changes while being analysed, and the authors go further to distinguish two types of drift: *offline* and *online* drift. The offline drift refers to scenarios where drift is detected after a process has ended, which is better employed in situations that do not require fast responses. Oppositely, online drift refers to real-time identification of changes.

Carmona et al. [45] raise three main problems of CD in the context of PM:

1. Detect when a process changes;
2. Characterise the nature of the change and identify the regions of change in a process;
3. Discover the evolution of a process change and how it affects the model.

Despite some efforts to define CD in business processes, there is still a need for more formal definitions. Previous attempts leave gaps to different interpretations. Therefore, it is essential to provide requirements for PM techniques over data streams. This way, different approaches can be compared more fairly.

3.2 Detecting Concept Drift in Business Processes

Bose et al. [11, 41] proposed solutions for dealing with concept drifts in PM using an offline analysis, consuming the event log in a batch strategy. The authors proposed a framework composed of five steps: (i) feature selection and extraction from the event log, (ii) definition of sample populations to explore trace changes, (iii) comparison between previously generated populations, (iv) interactive results visualisation and (v) drift plotting to an expert decision. This technique, however, is not suitable for streaming scenarios since it needs a complete event log. This way, organisations cannot react to changes as fast as possible. Hence, affecting business performance. Moreover, an offline approach handles only complete cases. In an event stream, incomplete cases are typical, and the proposed feature extraction is compromised. Finally, the approach only extracts trace features while other critical perspectives, such as time, were neglected in case description.

Seeliger et al. [43] proposed the use of a statistical significance test (G-test) to find drifts in event logs. Two adaptive non-overlapping sliding windows go through the

log, each one covering subsequential parts of the log. Those sub-logs are inputted to a process discovery technique grounded by the Heuristic Miner [20], which outputs the process models. Both process models are compared, and if they are significantly different, a drift is warned. The authors take advantage of graph metrics, such as the number of nodes/edges, network degree, among others, to quantify the difference between the models. Limited by the use of the Heuristic Miner algorithm, which cannot correctly identify loops, the gradual drift detection (a common drift behaviour in real-world scenarios) is compromised. Moreover, they do not consider time delta between activities, which is an important descriptor of a case. Finally, the incomplete trace problem is not discussed.

Maaradji et al. [10] consider partially ordered runs to detect the occurrence of sudden drifts. A run represents a set of traces equivalent to each other which models concurrency relations between activities. The technique performs statistical hypothesis tests over two consecutive sliding windows. Furthermore, the windows can adapt their sizes. This way, creating a trade-off between accuracy and drift detection delay. The drawback is to consider only complete traces, ignoring that traces may take days or weeks to be completed in real-world processes, which compromises their method in a streaming perspective.

Further, Maaradji et al. [46] extended the previous work to detect gradual drifts. The authors assume two consecutive sudden drifts delimit a gradual drift. The location detection is performed in a post-processing analysis where a separate statistical test is applied. This way, they identify if there is a gradual drift between two sudden drifts.

Ostovar et al. [12] proposed a sequel to the previous work. This approach performs over a stream of events, i.e. an online approach. Instead of using runs to characterise process behaviour, the authors proposed the use of $\alpha+$ relations. An $\alpha+$ relation is a set of rules to represent the relation between two activities, e.g., if one activity follows or precedes another [47]. Thus, the technique is suitable to perform over an online setting. However, only the sudden drift type is detected, and no other perspectives were applied.

Accorsi et al. [48] proposed another offline approach. The authors introduced the activity distance measure that considers the distance between a pair of activities in the event log. Then, a distance matrix is calculated for all possible pairs of activities. Finally, traces are clustered according to their activities distances. However, the approach (i) relies on a complete event log (offline), (ii) do not support loops, and (iii) ignore other case descriptors.

Zheng et al. [16] introduced a three-stage approach to detect concept drifts from event logs. First, the log is converted into a relation matrix by extracting direct succession and weak order relations from traces. Then, each relation is checked for variation trends to obtain candidate change points. Lastly, the candidate change points are clustered (using the DBSCAN algorithm) and combined into real change points. Like previous works,

Zheng et al. [16] technique is offline and also depends on a complete event log.

Liu et al. [49] proposed a framework for process drift detection divided into five steps. Initially, the framework creates a model which stands as a benchmark model. Then, noise reduction is applied to reduce infrequent behaviour in the model. After, to detect the drift, relationships between activities are mapped into an adjacency matrix. Further, if the framework identifies a drift, it creates a new model with the newer traces. Finally, it classifies the drift type by comparing the latest two process models. Though the authors state the framework identifies the four main types of drift, their experiments do not explore this affirmation. Moreover, the framework relies on a stream of traces, characterising as an offline approach.

An earlier version of Concept Drift in Event Stream Framework was proposed by Barbon et al. [50]. There, a histogram posed as the process model representation and DBSCAN was the used clustering technique. Though the approach reached promising results, there was space for progress. Firstly, a histogram is not a suitable representation of a process since it limits the transition aspect, i.e. the relation between activities that follow (or precede) others. Moreover, DBSCAN is intended for batch analysis. That is, the algorithm expects to deal with chunks of collected data, and is not prepared to handle a stream. Hence, the drift detection mechanism was incomplete since it depended on manual analysis of the generated clusters.

Table 3 compares the main characteristics of the presented works. Most approaches follow an offline strategy using trace streams as a form of data consumption. Moreover, sudden drift is the most commonly identified. Since the majority of techniques rely on statistical analysis, they do not provide models. Hence, model adaptation is not included in those works. Finally, the table lists the available techniques.

3.2.1 Limitations of Current Techniques

Despite CD being a widely researched aspect of DSM with a plethora of techniques and algorithms, they do not translate well to the PM field. This happens because traditional stream drift detection techniques treat each instance as a complete representation of a phenomenon. However, in event logs, a case is composed of several interconnected events, which do not occur atomically.

Reviewed works present several approaches to solve drift in PM scenarios, although most solutions are usually for a specific scenario. Hence, there are some gaps in previous techniques. The offline approaches deal only with complete event logs, which is impractical in streaming environments. Furthermore, some approaches only handle a stream of traces, i.e. complete cases. This is also infeasible in online scenarios since cases are not atomic. Finally, all reviewed works use trace related features. Thus, other critical attributes, as time, were neglected in case description.

Table 3 – Summary of related works regarding CD detection in business processes. A hyphen is placed where there is no available information

References	Analysis	Data Ingestion	Identified Drift Types	Model Adaptation	Open Source
[11, 41]	Offline	Trace Stream	Sudden Gradual	No	Yes
[43]	—	—	Sudden	No	No
[10]	Offline	Trace Stream	Sudden	No	Yes
[46]	Offline	Trace Stream	Sudden Gradual	No	Yes
[12]	Online	Event Stream	Sudden	No	Yes
[48]	Offline	Trace Stream	—	No	No
[16]	Offline	Trace Stream	Sudden	No	Yes
[49]	Offline	Trace Stream	Sudden	Yes	No
[50]	Online	Event Stream	—	Yes	Yes

Our proposal tackles those challenges by representing the process as a graph and applying a density-based clustering technique grounded in DSM. Hence, we can take advantage of graph-based metrics for feature extraction. Furthermore, our approach is unique in considering time delta between activities as a case descriptor. Thus, both anomaly and concept drift monitoring are possible.

4 CONCEPT DRIFT IN EVENT STREAM FRAMEWORK

The original Concept Drift in Event Stream Framework introduced in [50] modelled processes using a histogram of activities and implemented case clustering using the histogram to profile the attributes of traces. The DBSCAN algorithm was used to cluster cases in the feature space, detecting denser regions (common behaviour) and less dense regions (anomalous behaviour). As presented in [50], the original CDESF is capable of identifying anomalies both in trace composition and in time delta between activities.

However, its ability to handle drifts and adapt through time is imprecise due to two reasons: (i) the histogram representation and (ii) the use of DBSCAN as the clustering algorithm. Regarding process representation, a histogram is limited since it only counts the occurrence of activities, while it loses the activities order and their relation, narrowing the representational role. Concerning DBSCAN, the difficulty is that the algorithm is not suitable for online scenarios since it was proposed for batch learning.

Our modification addresses these limitations by revising the representation profile of cases using Process Model Graphs and by adopting a clustering algorithm for data streams (DenStream). Figure 6 presents CDESF workflow, which is divided into four main steps: Transformation, Distance Computation, Stream Processing and Check Point. Finally, for reproducibility purposes, CDESF was implemented in Python and is publicly available[1].

4.1 Transformation

The goal of this step is to model a case into a graph-based representation. Following a streaming paradigm, each event from the event stream arrives at a time. This implies that the framework cannot consume chunks of data. In turn, each event is consumed at the time of arrival. Thus, each time an event is presented to the framework, a transformation step is performed.

If the new event belongs to a new case (i.e., a case that has never been seen before), a graph containing this single event is created. However, if the event is from an already existing case, its corresponding trace is updated, and a new node added to its graph. The graphs' nodes represent activities, while edges represent that an activity is followed or preceded by another, which we introduced as *transitions*.

Example 4.1.1. Given a new event c in the stream from case c_1. Case c_1 trace is retrieved and updated: $\langle a, b\rangle \rightarrow \langle a, b, c\rangle$. Hence, we can infer that case c_1 contains two transitions:

[1] https://github.com/gbrltv/CDESF_v2

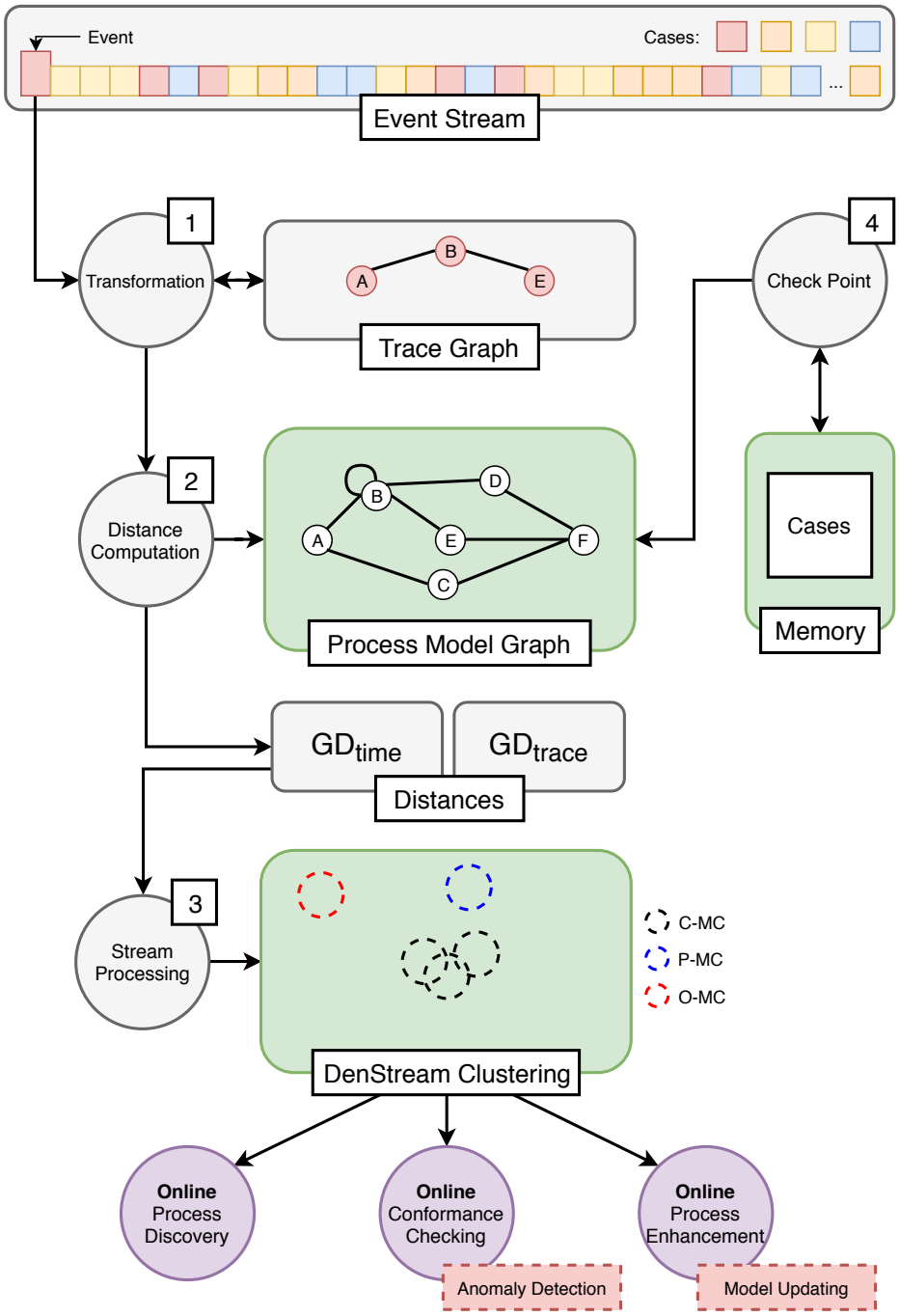


Figure 6 – Overview of main CDESF aspects. Given a new event, CDESF transforms its case into a graph. Then, distances are calculated by comparing the Trace Graph with the Process Model Graph. After that, the case is clustered by DenStream. Finally, the Check Point guarantees model adaptation and memory control

ab and *bc*.

4.2 Distance Computation

The second step aims to extract features from the case. In this context, the case is compared with the PMG, which represents the whole process and is built gradually over

time. The comparison insists on two perspectives: *trace* and *time* and is operated on a normalised version of the graph. From a trace perspective, the normalisation procedure consists of dividing the occurrence of a specific transition by the value of the most occurred transition. The result of this operation is referred to as *weight*. From a time perspective, edge normalisation computes the mean time of a transition between activities.

First, consider two functions, $PMG_{weight}(transition)$ and $PMG_{time}(transition)$. Given a transition, these functions retrieve the weight and mean time span of the corresponding transition in the PMG. Given a trace tr. T_{tr} is the total amount of transitions in tr. $tr[i]$ corresponds to the ith transition in tr. $tr[i]_{time}$ is the time delta in $tr[i]$ and $tr[i]_{weight}$ is the weight of $tr[i]$. Trace distance GD_{trace} and time distance GD_{time} are computed according to Equation 4.1 and 4.2, respectively.

$$GD_{trace}(tr) = \frac{\sum_{i=1}^{T_{tr}} 1 - PMG_{weight}(tr[i])}{T_{tr}} \tag{4.1}$$

$$GD_{time}(tr) = \log_{10}\left(\frac{\sum_{i=1}^{T_{tr}} |PMG_{time}(tr[i]) - tr[i]_{time}|}{\sum_{j=1}^{T_{tr}} PMG_{time}(tr[j])}\right) \tag{4.2}$$

Example 4.2.1. Given a trace $t_1 = \langle a, b, c \rangle$, given transitions ab and bc with time deltas of 4700 and 3000 (in seconds), respectively. Given a normalised PMG where transitions ab and bc time deltas are 5000 and 3000 (in seconds) and weights are 1 and 0.9, respectively. We can infer that $GD_{trace}(t_1) = [(1-1) + (1-0.9)]/2 = 0.05$ and $GD_{time}(t_1) = \log_{10}[((5000-4700) + (3000-3000))/(5000+3000)] = -1.42$

4.3 Stream Processing

The graph distances previously calculated (GD_{trace} and GD_{time}) are fed to the DenStream algorithm, which clusters the cases. DenStream provides an aggregation of similar process behaviour, where denser clusters represent common patterns and sparser clusters represent anomalous patterns. Moreover, it can detect cases which do not belong to any cluster or are not dense enough to form one. These points are also considered anomalous since they are isolated in the feature space.

4.4 Check Point

The PMG needs to be updated at some point to adapt to new concepts that emerge in the stream. However, model updating at the arrival of every single event is time and memory consuming. Also, it is not necessary since one event does not have a significant impact on the model.

Inspired by the DSM literature, we introduce the Time Horizon (TH) window which is a hyperparameter in a time unit. The point of division between two THs is the Check Point (CP). Figure 7 shows the relation between TH and CP. The figure depicts a stream of events where events are represented as squares. The square colours are for case identification. As we can see, a TH is a time-based window. Several events arrive in a respective TH. Between two THs, there is a CP. Note that TH is not a window based on event count. Contrarily, TH is measured in a time unit. Hence, the number of events that arrive in a TH varies. A CP has two main functions: PMG updating and memory control.

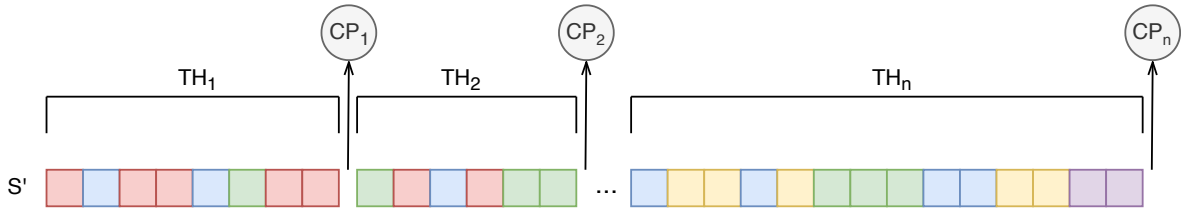


Figure 7 – TH is a time-based window while CP is the point of division between two THs. The number of events in a TH varies according to the business process and its environment

First, regarding PMG updating, all new cases from the last TH are used to build a graph, which we refer as TH graph (THG). The THG is a representation of the latest behaviour from the process since it is built with the newest events that arrived through the stream. A decay factor is applied to the PMG before merging the THG with it. This way, new and historical data are balanced. The decay diminishes the transitions weight by 5%, making the importance of transitions decrease as time passes. In a practical example, if a transition does not appear in a long time, its weight is decayed until the transition has no importance. This step supports process model enhancement by using new information to maintain the PMG faithful to real data.

Algorithm 2 shows how the merging process between both graphs works. The transitions' weight and time deltas are simply added. Figure 8 shows an example of a merging in a given CP. First, the decay factor is applied to the initial PMG. Then, the transitions values are summed. Transitions that are in the original PMG but not in the THG are only affected by the decay. The merging result is the updated PMG.

Due to memory and processing time limitations, events ingested from the event stream are accessed only once and released from memory later. This way, older cases are dispensed at each CP. The number of cases to be forgotten is computed using the Nyquist sampling theorem [51]. The theorem states that a minimum threshold to the sampling rate of data ingestion should be twice the highest frequency included in the signal. Equation

Algorithm 2 Merging Graphs (PMG, THG)

for each transition $tranP$ in PMG **do**
    Apply 5% decay factor to $tranP_{weight}$
end for
for each transition $tranT$ in THG **do**
    if $tranT$ in PMG **then**
        Update corresponding PMG transition ($tranP$) weight and time delta
        $tranP_{weight} = tranP_{weight} + tranT_{weight}$
        $tranP_{time} = tranP_{time} + tranT_{time}$
    else
        Add $tranT$ to PMG
    end if
end for

Figure 8 – PMG and THG merging example. The first graph is the PMG before the merge and the second graph is the THG. The third graph is the merging result, which is the new PMG. The process consists of applying a decay factor in the initial PMG and then summing the tuple values, which are the weight and time delta of a transition

4.3 shows how Nyquist is obtained.

$$Nyquist = 2 * highest\ frequency \tag{4.3}$$

In CDESF, the highest frequency is the number of new cases that have arrived in the last TH. During CP stage, the number of cases in memory (N_c) is compared to the Nyquist (N_y) value obtained in the previous CP. From this, there are two possible outcomes: (i) if $N_y < N_c$, there are more cases than the sampling theorem establishes; therefore the framework releases the excess. The number of cases to be released is given by $N_c - N_y$. Thus only the necessary amount of cases is maintained. Also, the cases are ordered, so the released ones are the oldest. Finally, a new N_y value is calculated. (ii) if $N_y \geq N_c$, there are fewer cases than the sampling theorem establishes, meaning that there is no need for case deletion. Thus, the N_y value stays the same.

Since the Nyquist is an automated solution, its usage facilitates the specialist role,

taking away the need for manual setting. Also, such forgetting mechanism permits the adaptation of the framework to the arrival rate of the stream. Finally, the automatic updating nature of the Nyquist aids the releasing of memory, an important factor in DSM.

CDESF's workflow marries various characteristics of both PM and DSM by tackling the three tasks of PM (process discovery, conformance checking, and process enhancement) in a streaming environment. Online process discovery creates new graphs based on recent data. The kernel process representation follows a graph structure to support distance computations and similar case clustering. Online conformance checking compares the current case with the PMG and extracts its metrics (equations 4.1 and 4.2), which aids the identification of anomalous cases in the clustering (low-density regions). Lastly, online process enhancement updates the PMG with newer cases. Hence, the PMG is always up-to-date with the current process behaviour.

4.5 Process Model Graph Creation

Process Model Graph is a fundamental element of the framework. Both distance computation and CP stages heavily depend on the PMG. Previously, we demonstrated how the PMG updates to keep up with new event data. However, the PMG must be built to support these operations.

Algorithm 3 shows the PMG creation. The PMG is built in the first CP. For this, it uses all events that arrived in the first TH. Given a set of traces, the transitions within them are mapped and grouped as a graph. The same algorithm is used for THG creation.

Algorithm 3 PMG Creation (traces)

```
1: Loop through traces
2: for each trace T in traces do
3:     Loop through transitions of a given trace
4:     for each transition tran in T do
5:         if tran not in Graph then
6:             Create new edge and nodes to host the transition
7:         end if
8:         Add one to the weight of Graph_tran
9:         Add tran_time to time delta of Graph_tran
10:    end for
11: end for
```

Example 4.5.1. Given a set of traces $L = \{\langle a, b, c\rangle, \langle a, b\rangle, \langle a, b, c, d\rangle\}$. Given 10 as the time delta between any two activities. The resulting PMG contains three transitions: ab, bc and cd. The transitions weights are 3, 2 and 1. And the transitions time deltas are 30, 20 and 10. Figure 9 illustrates the resulting PMG given L.

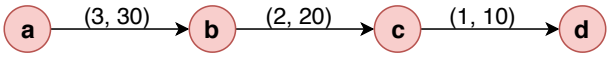


Figure 9 – The resulting PMG given the set of traces L from the Example 4.5.1

4.6 Performing an Example in CDESF

Figure 10 provides an example that shows how the processing occurs given a new event. First, the transformation step is performed. Then, the PMG is normalised, and graph distances are extracted. Finally, the case is clustered using the DenStream algorithm. The figure demonstrates the main aspects of CDESF and how they are executed.

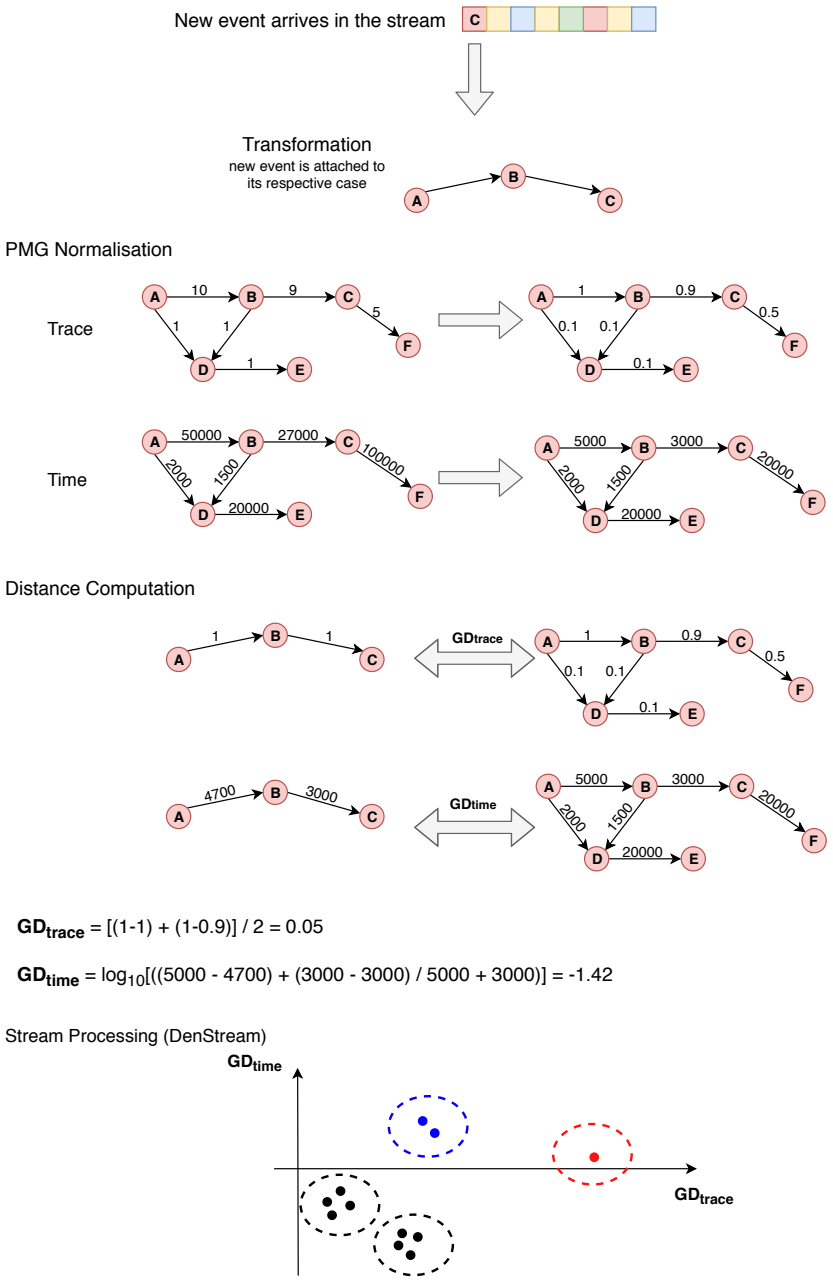


Figure 10 – Example of a new event in a stream and the main CDESF steps to process the event

5 MATERIALS AND METHODS

This chapter presents an in-depth discussion of the event logs that served as the basis for our experiments and analysis. Also, the experimental setup and used technologies are explained.

5.1 Datasets

Three groups of event logs were chosen to support the framework evaluation. Their data have different distributions, i.e., different characteristics such as the number of events, the time span of the process, among others. Two groups contain real-life event logs, and one group contains synthetic event logs:

- The *Healthcare* event log is massive log of a real-life healthcare scenario[1]. The event log contains the activities: `NEW`, `RELEASE`, `MANUAL`, `CODE NOK`, `SET STATUS`, `JOIN-PAT`, `BILLED`, `REOPEN`, `CODE ERROR`, `STORNO`, `CHANGE END`, `REJECT`, `DELETE`, `CODE OK`, `EMPTY`, `FIN`, `ZDBC_BEHAN` and `CHANGE DIAGN`. *Healthcare* was obtained from the ERP system of a hospital. Each trace records the activities executed to bill a package of medical services that were bundled together [52]. Its events are anonymised and the time within activities has not been modified;

- The second group contains several event logs from a manufacturing company in Italy, which was first studied and explored in [53]. The log includes forty different business processes related to product management. Since many of them are too small for streaming simulation, we have chosen the four biggest ones: *Assembly_IW-Frozen*, *Detail_Frozen-Final_Rel*, *Detail_IW-Frozen* and *Detail_Supplier_IW-Frozen*;

- The last group of logs is a synthetic one introduced by Maaradji et al. [10]. The authors created the logs focusing on drift detection. The base model used a textbook example of a business process for assessing loan applications [3]. It contains 15 activities, one start event and three end events. Moreover, the model explores several characteristics, such as loops, parallel and alternative branches. The authors modified the base model to simulate different behaviours. For that, simple changes patterns were used. The change patterns involve: adding, removing or looping a model fragment; swapping two fragments or parallelising two sequential fragments. Then, event logs were created interchanging traces produced by two different models. This way, simulating a drift behaviour. Note that this simulation created only

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

sudden drifts because one behaviour substituted another abruptly in the log. Furthermore, the authors produced logs with 2500, 5000, 7500 and 10000 traces and each log contains nine drifts located at multiples of 10% of the log size [10]. We have chosen only logs with 5000 traces since they are large enough to explore and detect drifts.

5.2 Features from Event Logs

An interesting way of comprehending the behaviour of a process is to look at its statistics, mainly the number of events, cases and their duration. Table 4 summarises extracted statistics from real-life and synthetic event logs.

Regarding the logs from the manufacturing company, the largest trace from *Detail_Frozen-Final_Rel* has only two activities while the mean trace size is 1.76. Since *Detail_Frozen-Final_Rel* has fewer activities, its case duration is also smaller when compared to other processes. *Assembly_IW-Frozen*, *Detail_IW-Frozen* and *Detail_Supplier_IW-Frozen* have longer traces, reaching 9, 10 and 12 activities, respectively. The mean trace size of those processes corroborates with the idea of a proper distribution of activities per case. Moreover, they have longer case duration with the mean being over two days, which aligns with the fact that these processes have more activities and consequently take more time to complete.

The statistics from the *Healthcare* dataset show a more complex behaviour, which complies with it being more than 18 times larger (in the number of events) than the second largest log (*Detail_IW-Frozen*). As seen in Table 4, *Healthcare* has around 328 mean cases per day and a maximum of 592 cases in only one day. None of the previous processes has maximum cases per day as high as the mean of *Healthcare*. Moreover, its biggest trace is composed of 217 activities, a possible outlier.

Although *Healthcare* contains larger traces, 22.4% of its traces only have one activity, and 8% only have two. Hence, the mean trace size is 4.5 activities, a relatable value compared to the other real-life logs.

It is also observable that *Healthcare* presents a unique case duration. The mean case duration is more than 127 days, which is comprehensible whereas the log comes from a healthcare background and treatments are time-dependent. For instance, one case lasted for more than one thousand days. However, the median case duration is lower due to a significant portion of cases having only one activity.

Healthcare has very complex behaviour. This is due to the log depending on a plethora of variables, such as the type of treatment, the size of the hospital, dependencies within medical care, among others. Using such an event log will further explore the capabilities of our approach to handling more complex situations.

Table 4 – Exploring real-life and synthetic business processes by extracting statistical metrics. The time unit used is days

process	total cases	mean cases per day	max cases per day	total events	mean events per day	max events per day	mean trace size	max trace size	min trace size	mean time duration	max time duration	median time duration
Healthcare	99999	327.99	592	451358	398.37	687	4.51	217	1	127.36	1035.42	102.38
Assembly IW-Frozen	2037	7.98	48	9324	17.72	202	4.57	9	1	2.51	50.87	0.95
Detail Frozen Final Rel	3817	14.45	96	6722	25.46	187	1.76	2	1	0.001	0.12	0.00001
Detail IW-Frozen	5418	18.44	101	25131	40.14	483	4.63	10	1	2.50	77.83	0.91
Detail Supplier IW-Frozen	4587	25.16	99	24952	66.18	347	5.43	12	1	2.14	47.70	0.83
cb5k	5000	22.56	28	50418	156.09	196	10.08	21	8	0.64	3.83	0.21
cd5k	5000	23	28	51851	159.54	205	10.37	21	8	0.69	3.83	0.26
cf5k	5000	23.60	29	52989	165.07	209	10.59	21	8	0.70	3.83	0.73
cm5k	5000	22.91	28	50441	155.20	196	10.08	23	8	0.68	3.83	0.24
cp5k	5000	23.52	30	54029	166.24	214	10.80	23	8	0.73	3.86	0.74
fr5k	5000	22.91	28	51699	158.58	203	10.33	21	8	0.68	3.83	0.26
IOR5k	5000	22.92	28	52323	160.00	196	10.46	21	8	0.68	3.83	0.28
IRO5k	5000	23.63	29	53647	166.60	219	10.72	29	8	0.71	3.96	0.73
lp5k	10000	23.97	29	108302	169.48	215	10.83	28	8	0.73	3.88	0.75
OIR5k	5000	23.33	28	55334	170.25	233	11.06	24	8	0.70	3.83	0.74
ORI5k	5000	23.74	29	54618	169.62	222	10.92	32	8	0.72	3.95	0.74
pl5k	5000	23.74	28	51936	161.29	201	10.38	21	8	0.71	3.83	0.75
pm5k	5000	22.58	28	51268	157.74	196	10.25	21	8	0.64	3.83	0.22
re5k	10000	23.22	28	98679	155.4	198	9.86	27	7	0.64	3.80	0.21
RIO5k	5000	23.54	28	53426	165.40	211	10.68	21	8	0.71	3.83	0.73
ROI5k	5000	23.84	29	51995	161.97	202	10.39	21	8	0.73	3.83	0.75
rp5k	5000	23.45	28	51782	161.81	198	10.35	21	8	0.68	3.83	0.29
sw5k	5000	23.44	28	51824	161.95	196	10.36	21	8	0.68	3.85	0.28

Regarding synthetic event logs, it is clear that different processes share many adjacent properties, meaning that they have similar behaviour. This happens because the synthetic event logs heir the aspects from the same base model and change some specific point to characterise a drift. Hence, case-related metrics are similar among the processes. The mean cases per day is around 23 and the maximum cases per day varies from 28 to 30. The number of events per day is high compared to the real-life logs, which implies in a stream with a faster rate. Trace wise, the mean size levitates around 10 activities. All processes have the maximum trace length of more than 21 activities, while the smallest trace usually contains 8 activities. When analysing time metrics, the synthetic logs have a mean duration of less than one day and a maximum duration of less than four days.

In the real-life logs, the mean value of cases and events per day vary significantly in the selected processes, and processes that present a higher number of cases also have a higher number of events. This happens because real processes are prone to be affected by environmental variables. Contrarily, the synthetic event logs are more constant and

present fewer outliers.

5.3 Further Analysis Using Petri Nets

Another way of analysing a business process is to visualise its model. For that, we used the Inductive Miner proposed by Leemans et al. [19] to extract the Petri Nets representation of some processes. The complete event log was used as input for the Inductive Miner. However, when dealing with event streams, the events are processed at the time of their arrival. Therefore, the complete event logs are not available, meaning that the PNs of such processes are compromised, making traditional approaches not viable.

Several PNs contain black squares, known as invisible transitions. An invisible transition is the result of generalisation where the path can be taken without a specific rule attached to it. Usually, the discovery algorithms use such transitions to abstract noisy traces. This way, outlier behaviour is not modelled in the final representation.

Figures 11, 12, 13, 14 and 15 show the resulting PNs for processes *Assembly_IW-Frozen*, *Detail_Frozen-Final_Rel*, *Detail_IW-Frozen*, *Detail_Supplier_IW-Frozen* and *Healthcare*, respectively.

As seen in Table 4, *Detail_Frozen-Final_Rel* only has two activities. Thus its PN is the most simplistic one. The other manufacturing company processes offer a more robust example of multiple activities and how they correlate. The *Healthcare* PN stands out as the most complex of the real-life event logs, featuring 18 activities and several dependencies among them. The PN showcases the extensive event log, which covers a plethora of available procedures inside a healthcare facility.

Figure 16 shows the baseline model used in the synthetic event logs. This model was modified to create different behaviour. The drift comes from interchanging traces from the baseline model and a modified one. Figures 17 and 18 show the PNs of two modified models. Each synthetic log is a variation of a change pattern which creates different types of drift.

In the *cb5k* process, *cb* stands for "make fragment skippable/non-skippable" [10]. As we can see from the images, *cb5k* has a skippable fragment between activities A_9 and A_{10} that does not exist in the base model, implying in different trace behaviour.

A more drastic change is performed in the *OIR5k* process, as seen in Figure 18. *O*, *R* and *I* stands for optionalization, resequentialization and insertion, respectively. In this example, *O* makes an activity optional (e.g. skippable), *R* makes an activity parallel to others and *I* inserts a new activity to the process (which is depicted by the `Added Activity` label in Figure 18). Further, we can see that the resulting *OIR5k* PN has more significant changes when compared to *cb5k* PN.

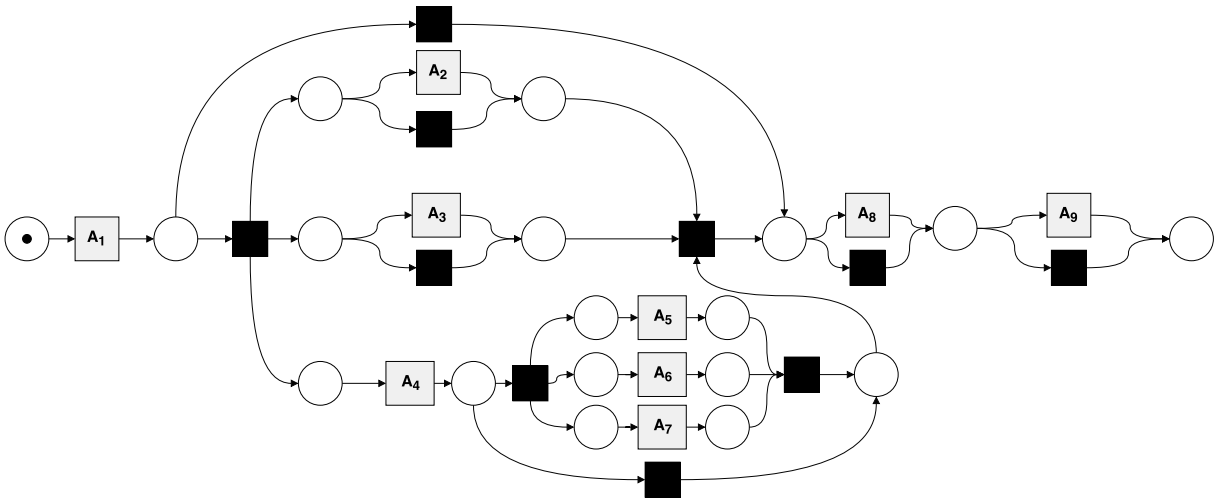


Figure 11 – *Assembly_IW-Frozen* Petri Net by applying the Inductive Miner algorithm

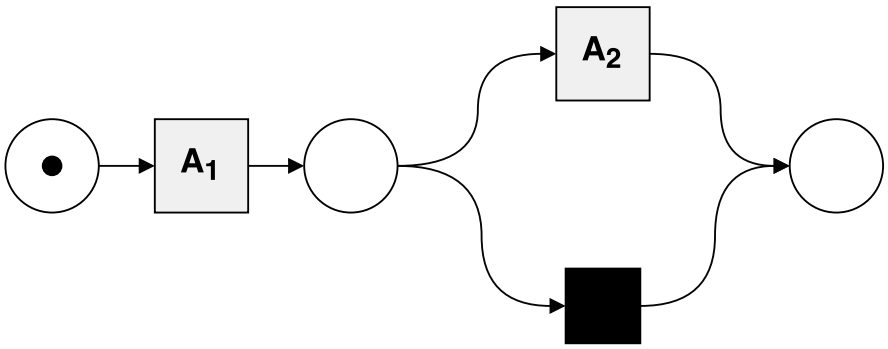


Figure 12 – *Detail_Frozen-Final_Rel* Petri Net by applying the Inductive Miner algorithm

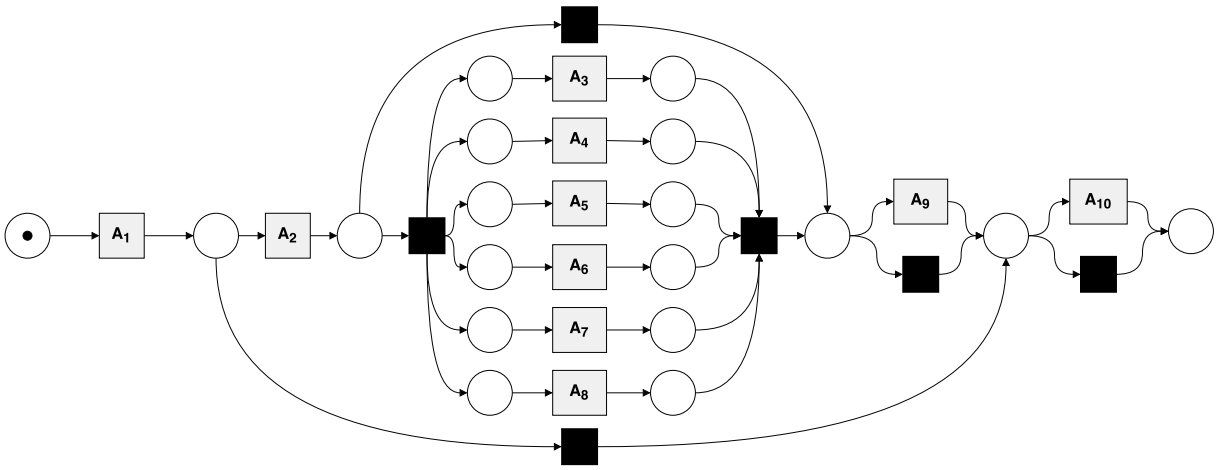


Figure 13 – *Detail_IW-Frozen* Petri Net by applying the Inductive Miner algorithm

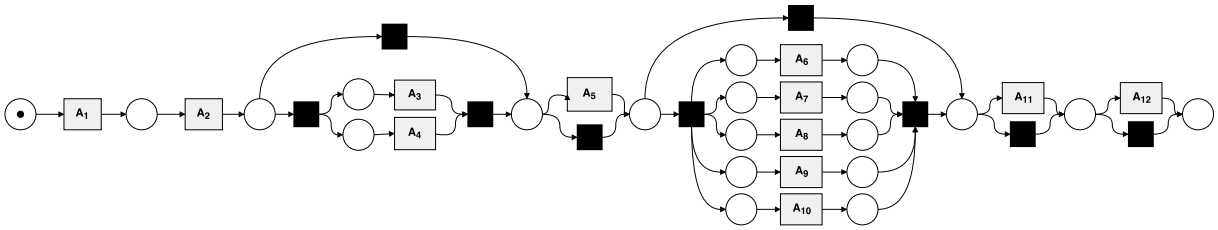


Figure 14 – *Detail_Supplier_IW-Frozen* Petri Net by applying the Inductive Miner algorithm

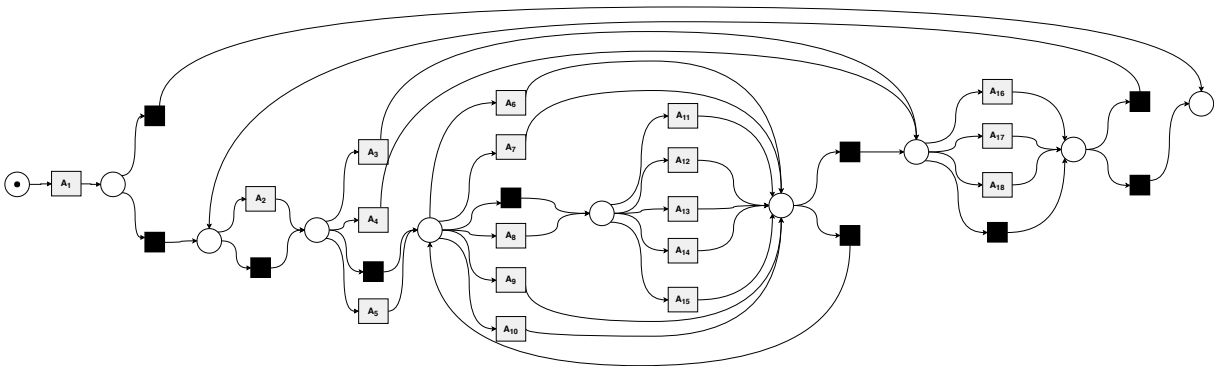


Figure 15 – *Healthcare* Petri Net by applying the Inductive Miner algorithm

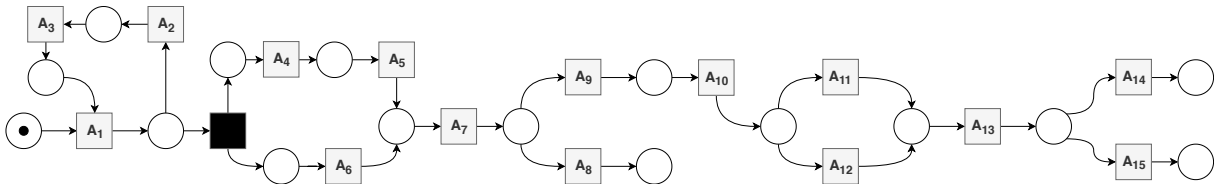


Figure 16 – Synthetic baseline Petri Net by applying the Inductive Miner algorithm

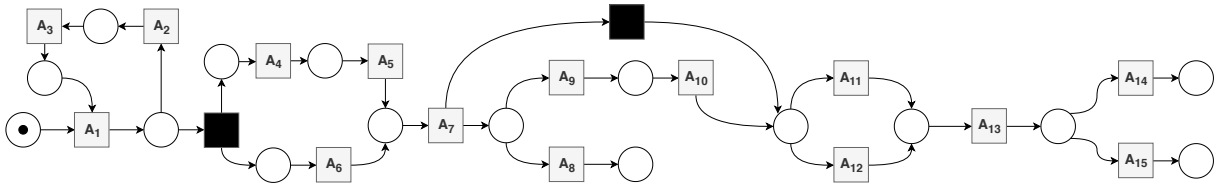


Figure 17 – *cb5k* Petri Net by applying the Inductive Miner algorithm

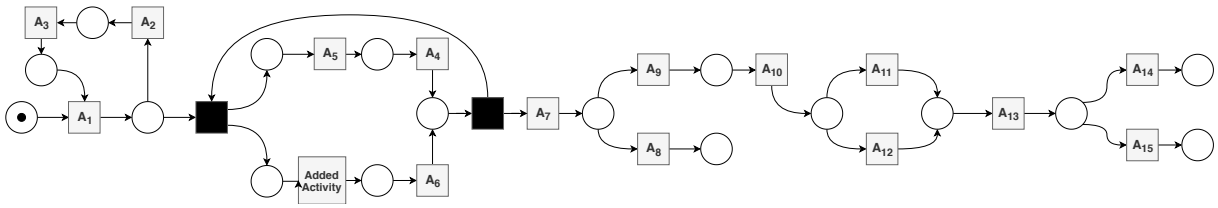


Figure 18 – *OIR5k* Petri Net by applying the Inductive Miner algorithm

5.4 Experimental Setup

To evaluate our approach, we used the business event logs presented in Section 5.1. CDESF is implemented in Python (version 3.6). The embedded DenStream algorithm strictly followed the proposal by Cao et al. [33].

Since there are several event logs, the TH value ranges significantly in the experiments. A good starting point for a TH is the mean case duration because it indicates the average behaviour of a business process. From there, we explored both ends of the spectrum, using THs either lower or higher than the mean case duration. For the three groups of business event logs, the experimented THs were:

- *Healthcare*: by far the largest event log, the *Healthcare* provides sufficient material for extensive analysis. The mean case duration has more than four months, so applying a compatible TH would be inappropriate in a streaming environment. That is, a four months TH would imply in very few model updates, making it less adaptable to drift. Also, storing 120 days worth of data is not feasible in a streaming approach. Thus, we explored the THs of 1, 2 and 3 days, which implies that CDESF will have to handle several unfinished cases and at the same time maintain the PMG;
- Manufacturing company: these logs come from a common source and their time characteristics are not sparse (see Table 4). Thus, considering the mean case duration of 2.5 days, the used THs were 1, 2, 3 and 4 days. This way, we have further support for the analysis of these processes encompassing very small THs to larger ones;
- Synthetic: all synthetic logs are very similar time-wise since the logs were created for trace drifts, resulting in the mean case duration fluctuating around 0.7 days. Thus, the THs used were 0.25, 0.5, 1, and 3 days.

Apart from TH, CDESF heirs the DenStream hyperparameters, which control the clusterisation phase of the framework. Table 5 summarises the configurations used for all event logs. As explained, TH values vary for different event logs. DenStream hyperparameters mainly follow recommendations from the original paper. Nevertheless, several combinations between the hyperparameters were explored based on a grid design. The goal of these configurations is to investigate the influence of the different hyperparameters in CDESF, mainly regarding PM tasks.

Previous configurations support the analysis of several CDESF aspects. However, a comparison regarding CD detection is still needed. This way, we selected all available solutions from Table 3 that can detect at least one CD type. Thus, three groups of approaches were selected: [11, 41], [10, 46, 12] and [16]. We selected the latest release of each approach and ran with a standard configuration. Furthermore, the synthetic event logs were used for CD detection, once they were created solely for this purpose.

Table 5 – CDESF hyperparameters configurations for experiments

Hyperparameters	Values
TH	0.25, 0.5, 1, 2, 3, 4
ϵ	0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
β	0.1, 0.2, 0.3, 0.4
μ	1, 2, 3, 4, 10, 50, 100, 500, 1000
λ	0.05, 0.1, 0.15, 0.2
v	1, 10, 50, 100, 500, 1000, 2000, 3000, 4000, 5000

6 RESULTS

This chapter delves into implications of the Concept Drift on Event Stream Framework results and raises a discussion based on PM interpretations in a streaming scenario. We explore anomaly detection and drift adaptation. Moreover, a comparison with traditional drift detection techniques is provided.

6.1 Experiments in an Online Environment

This section evaluates how CDESF handles traditional PM tasks in an online environment.

6.1.1 Online Process Discovery

Conventionally, process discovery techniques create a model that describes a business process solely based on the event log as the input. Usually, the first step of a process discovery method is to group all cases into traces, which implies that a case went through the process and reached an end up activity. When dealing with an event log in a stream fashion, the complete case is not available, so dealing with incomplete cases when discovering a process model is a requirement.

CDESF addresses this restriction by representing the model as a graph with connections among activities, independently of the state of the case (complete or incomplete). The graph needs to build up itself upon new arriving events and update its weights to maintain a close representation of the process.

We ran the first 26000 events of real-life *Healthcare* event log to test the PMG adaptability through time during online process discovery. The hyperparemeters were set to: TH = 3 days; $\epsilon = 0.1$; $\lambda = 0.1$; $\beta = 0.3$; $\mu = 4$; $v = 100$. Figure 19 shows the obtained PMG at CPs 1, 7, 11, and 43, respectively. Due to the small amounts of events at that time, only a simple graph exists in CP 1 (Figure 19a). Arriving events aid the evolution of the graph, seen at CPs 7 and 11 (Figures 19b and 19c). The enhanced PMG is built around the initial one seen in CP 1. Finally, at CP 43 (Figure 19d) the PMG presents more complex connections, representing all traces that went through the stream. PMG at CP 43 can be seen as a result of a batch analysis of the first 26000 events.

Complementarily, we compared the PMGs of each CP to the last PMG generated in this experiment. This way, we can observe the evolution of the PMG through time. Equations 4.1 and 4.2 were used to calculate the distances between the PMGs. Figure 20 shows the similarity between each CP graph and the batch process graph. Early on the stream, an accurate representation of the process is already at hand. Not only the

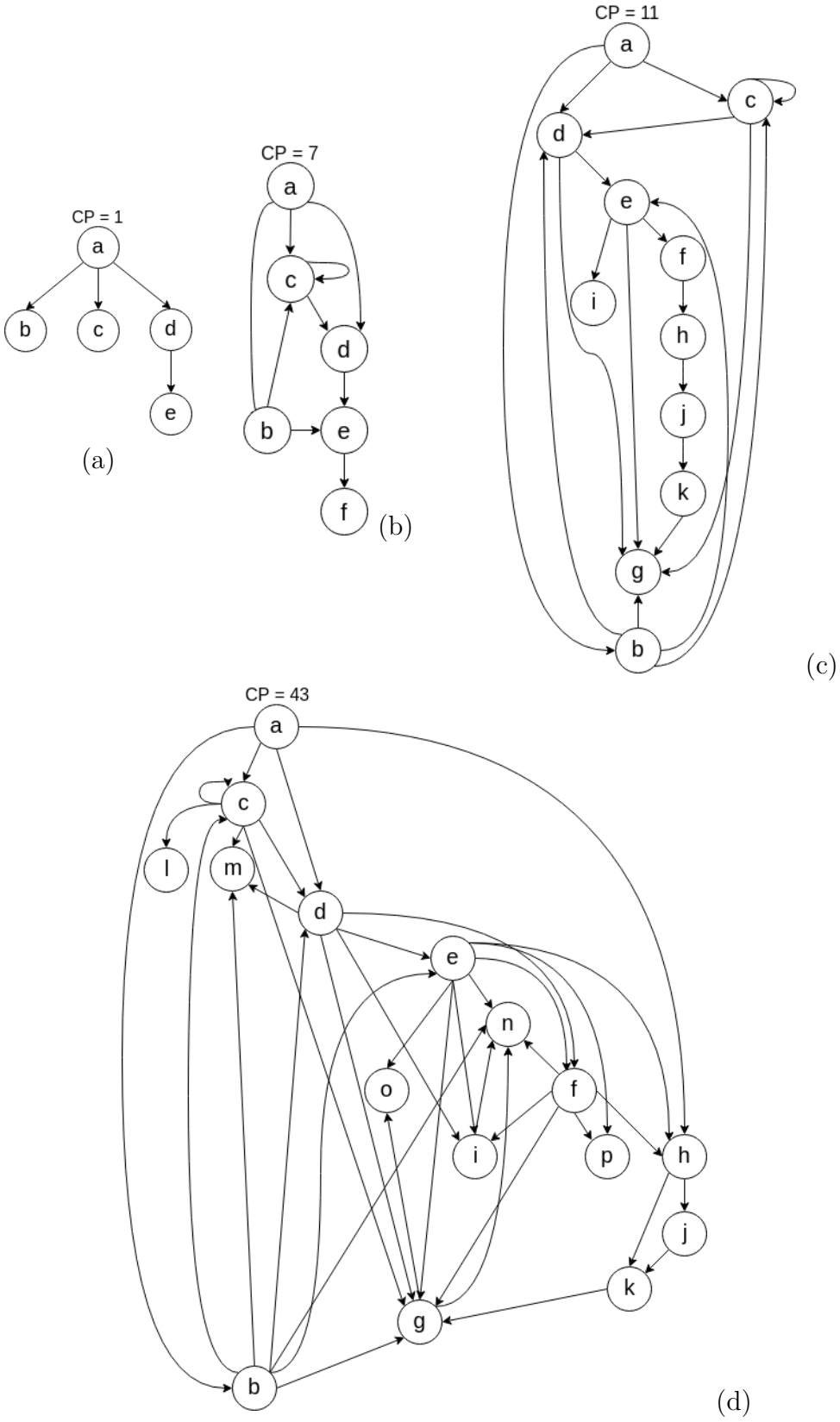


Figure 19 – PMG state in CPs 1 (a), 7 (b), 11 (c) and 43 (d) for the *Healthcare* event log

trace part of the graph converges but the time within activities also does, meaning that we can rely on our trace and time approach. The same behaviour is observed in different

processes, agreeing with our hypothesis that the model can adapt and learn in a small amount of time.

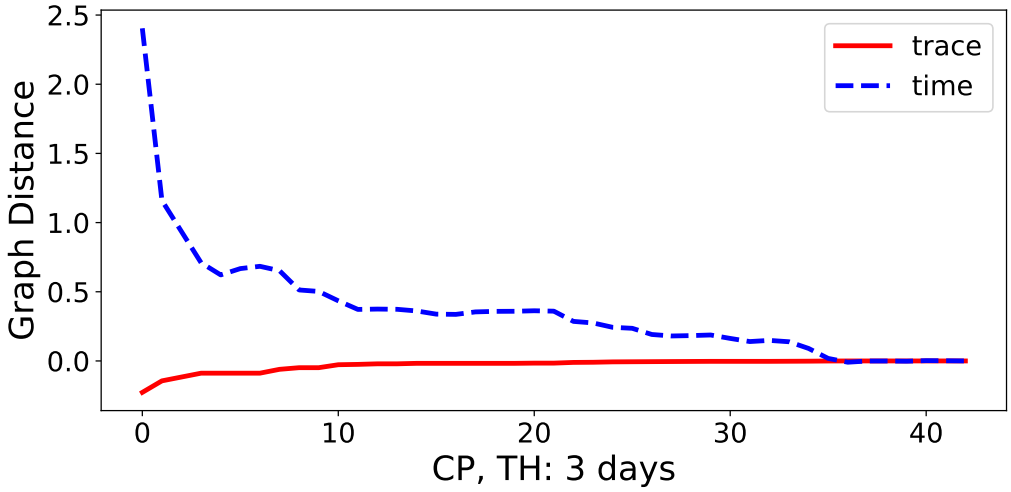


Figure 20 – PMG adaptation in *Healthcare* event log. The graph shows the difference between each CP PMG and the last PMG

Likewise, we performed the same experiment to event logs from the other groups. Figures 21 and 22 show the results of PMG similarity for *Assembly_IW-Frozen* and *cb5k*, respectively. Though the business processes are from different contexts, the same PMG adaptability is observed. This way, the process discovery task in online environments is satisfied by the PMG. Therefore, demonstrating CDESF capability to represent a model and adapt it through time.

Figure 21 shows a sudden trace adaptation in CP 21, meaning that a new transition appeared in the previous TH. Thus, when the PMG was updated, it became equal (trace wise) to the last PMG. Moreover, in Figure 22, there is a slow time adaptation over time starting in CP 200. That may be the result of activities being performed faster in the business process.

6.1.2 Online Conformance Checking

Conformance checking may detect, locate, explain deviations, and measure their severities [1]. Regarding deviations, conformance checking offers a way of finding anomalous cases. Aalst et al. [54] state that an anomaly can be formally described as a rare or infrequent event that deviates from a standard rule, meaning that an anomalous trace is a trace that does not fit an appropriate model.

CDESF uses the clustering phase to early detect anomalous traces, i.e. it performs online conformance checking analysis. At the arrival of an event, the framework retrieves its case and compares to the PMG, extracting both GD_{trace} and GD_{time}, which in turn are inputted into DenStream. In the DenStream domain, the O-MCs detected are clusters

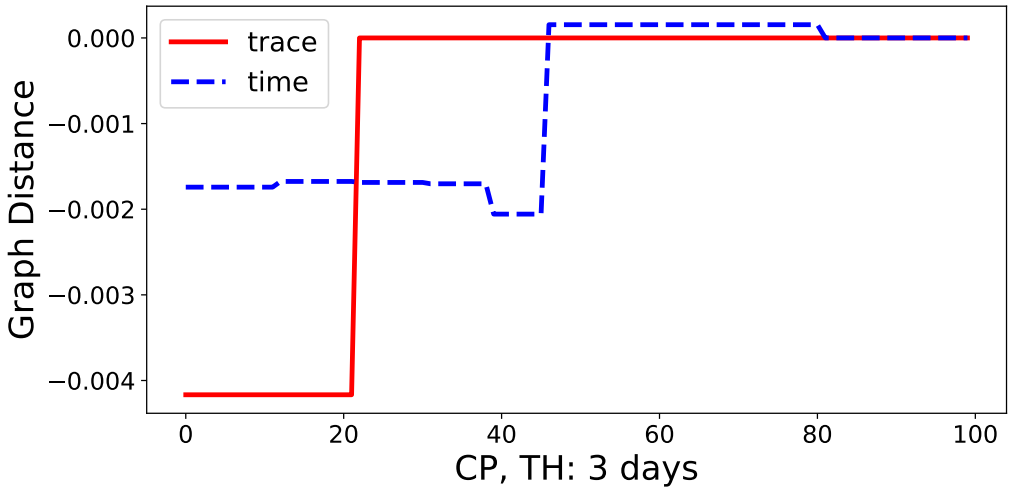


Figure 21 – PMG adaptation in *Assembly_IW-Frozen* event log. The graph shows the difference between each CP PMG and the last PMG

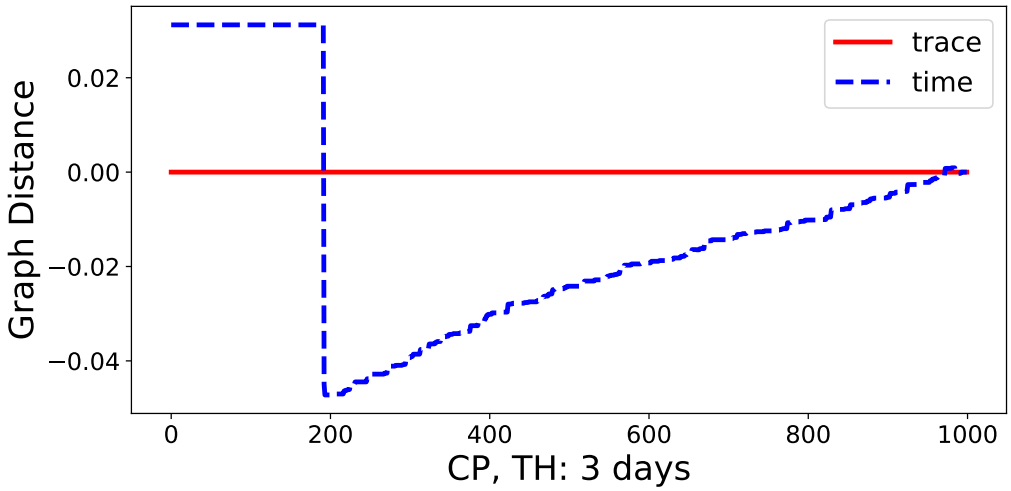


Figure 22 – PMG adaptation in *cb5k* event log. The graph shows the difference between each CP PMG and the last PMG

with low density. Hence, being an O-MC is related to anomalous behaviour in the feature space. Therefore, the notion of O-MC is crucial to support the activation of real-time alerts for monitoring a process.

We explored the creation of new O-MCs with the dataset *Detail_Supplier_IW-Frozen* using the following set of parameters: TH = 3 days; $\epsilon = 0.2$; $\lambda = 0.1$; $\beta = 0.3$; $\mu = 4$; $v = 100$. We could observe that several O-MCs permute the event log. Though some of them may be the first example of new behaviour (concept drift), most are an indication of outlier behaviour, whether from trace or time perspective.

We selected case 16280 from this event log to showcase the anomalous behaviour that results in an O-MC in the clusterisation phase. This case has a regular set of activities, which is the same one seen in the majority of the cases. However, it presents an

abnormal GD_{time} value. Case 16280 starts with activity `Process Creation` completed in "2011/03/29 14:00:44", which is followed by activity `EngSupplierDesign Lead` completed in "2011/04/05 17:00:16". The time span between those two activities is more than 7 days. In the PMG, the mean time between those same two activities was around 4 hours. There is an apparent gap between normal time span (4 hours) represented in the PMG and case 16280 time span (7 days). Therefore, the GD_{time} for this case was high, meaning that a considerable distance between the case and the model exist, and when clustered, the case was grouped into an O-MC.

Moreover, statistics from Table 4 show that the mean case duration of *Detail_Supplier_IW-Frozen* is around 2.5 days, corroborating with the idea that case 16280 is an outlier. Traditional approaches would fail to identify this case as an outlier since they do not take the inter-activity time feature as a case descriptor. In a real-life scenario, however, abnormal time spans are an indication of anomalous behaviour.

When applying the *Healthcare* event log to CDESF using TH = 3 days; $\epsilon = 0.1$; $\lambda = 0.1$; $\beta = 0.3$; $\mu = 4$; $v = 100$, we found an outlier trace sequence from case 291, which was also discovered as an outlier by Barbon et al. in [50]. Its trace is composed by activities `NEW`, `FIN`, `RELEASE` and `DELETE` activities. By looking at the PMG, we see that the transitions `NEW` to `FIN` and `FIN` to `RELEASE` happen only twice in all other cases, and the transition from `RELEASE` to `DELETE` happens only once. This unusual behaviour resulted in an abnormal GD_{trace}. Moreover, case 291 also presents an anomalous GD_{time}, because its time inter-activities is not common compared to the PMG. Thus, when clustered by DenStream, case 291 was considered an outlier.

Furthermore, we also found a direct correlation between the hyperparameter ϵ and the number of O-MCs detected. ϵ comprehends to the maximum actuation range of a MC. In this way, the higher the ϵ, the fewer O-MCs will be formed because clusters would have a higher actuation range. Figure 23 presents the number of O-MCs while varying ϵ for three event logs, two real-life (*Detail_IW-Frozen* and *Healthcare*) and one synthetic (*ROI5k*).

It is visible that the synthetic log presents fewer O-MCs. This happens because synthetic event logs are less prone to have outlier behaviour. Moreover, this event log was created to simulate a drift and presents very few outliers. Contrarily, real-life logs have more outliers due to real processes being more flexible and inconstant.

6.1.3 Online Process Enhancement

Process enhancement improves an existing process model using the event log as input [1]. In batch approaches, an already existing process model (either created by a process discovery technique or by an expert) is updated with enhancement techniques based on the event log. In our approach, CDESF uses new data to update the PMG alike

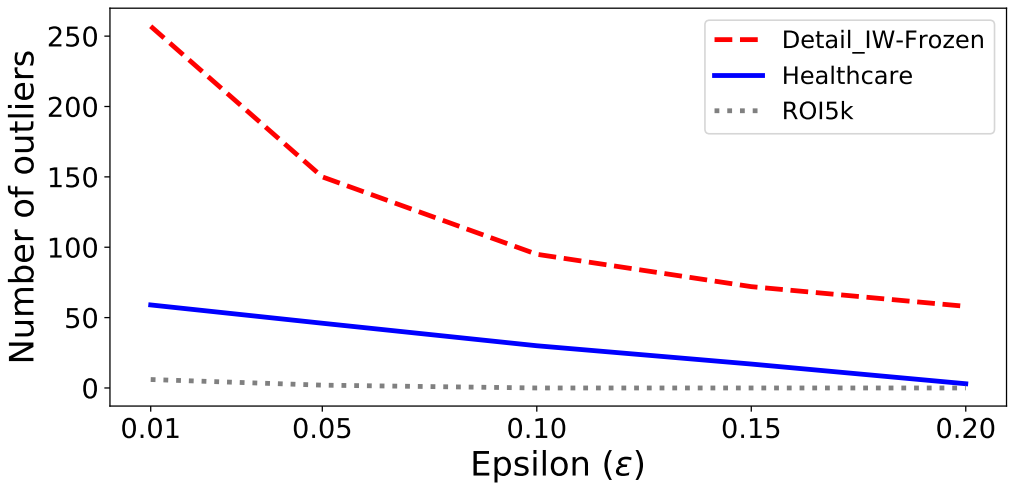


Figure 23 – Number of O-MCs detected in real-life (*Detail_IW-Frozen* and *Healthcare*) and synthetic (*ROI5k*) process versus ϵ value. A higher ϵ yields less O-MCs

an enhancement technique can repair the model by modifying it and cross-correlating the model with the new log.

In an online environment, each new event is part of a new log from the model point of view. However, CDESF does not update the model after each new event because model updating is resource consuming and only one event has little to no effect to the PMG. Thus, the framework updates an existing PMG with new events at every CP, where a group of the latest events exist, smoothly adapting during the event log processing. In the case of an incremental drift in the stream, where the data distribution changes slowly with each event, the PMG updates gradually, using the CP graph as the basis of representation of the slightly different behaviour. In a sudden drift scenario, the PMG will suffer more influence of the CP graph since the latter is entirely composed of a new concept. The consequence is that the PMG will adapt as fast as possible, using all new examples of the new concept.

6.1.3.1 CDESF Online Cluster Adaptation

Figure 24 shows the adaptation of a specific C-MC over the *Assembly_IW-Frozen* event log with TH = 3 days; $\epsilon = 0.2$; $\lambda = 0.05$; $\beta = 0.2$; $\mu = 4$; $v = 100$. For this analysis, the densest C-MC's centroid position variance was tracked after every new event in the stream. The black line represents the trace variance while the red one represents time variance.

It is possible to observe a converging behaviour because new events corroborate with the current position of the cluster, i.e. new events are placed inside the C-MC. In some cases, trace and time variance occur together, meaning that an event that impacts in trace configuration also often changes the time perspective. It is important to highlight that

the variations presented at the beginning of the log are related to an adaptation period. Furthermore, as the C-MC gets denser, new events have less impact on the cluster's position. Moreover, the *Assembly_IW-Frozen* event log has no drifts in the evaluated chunk of events.

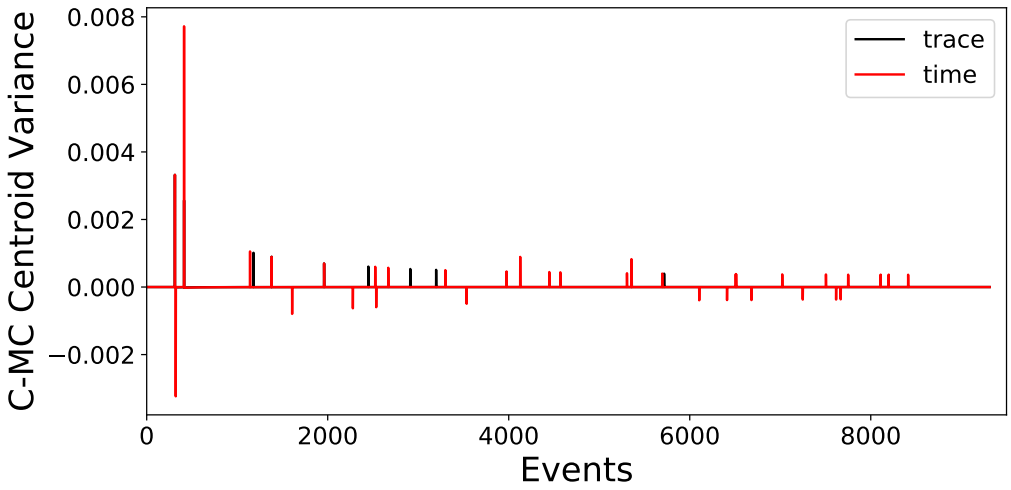


Figure 24 – Tracked changes in the C-MC centroid position in the *Assembly_IW-Frozen* event log. Black and red lines represent trace and time variance

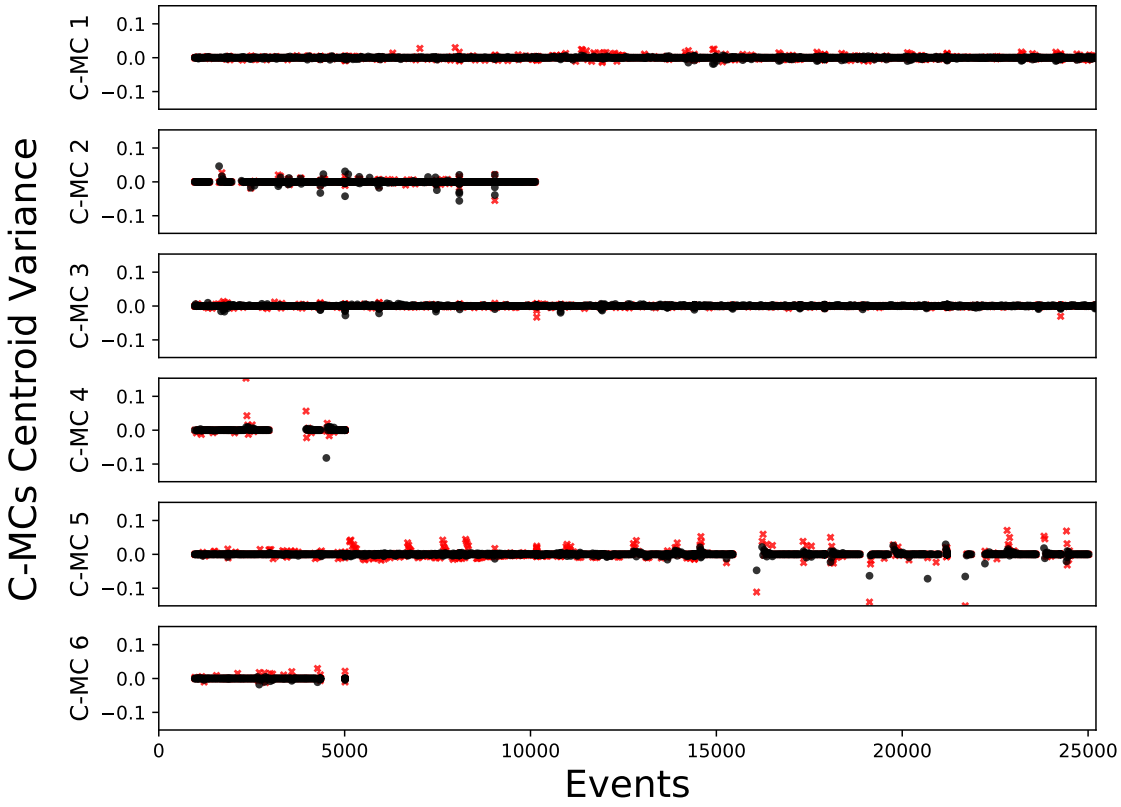
6.1.3.2 CDESF Drift Detection

CDESF has two triggers to identify drifts in the stream. This evaluation is grounded by the DenStream notion of common behaviour, represented by the C-MC. Thus, either the detection of a new C-MC or the fading of a current C-MC is interpreted as a drift.

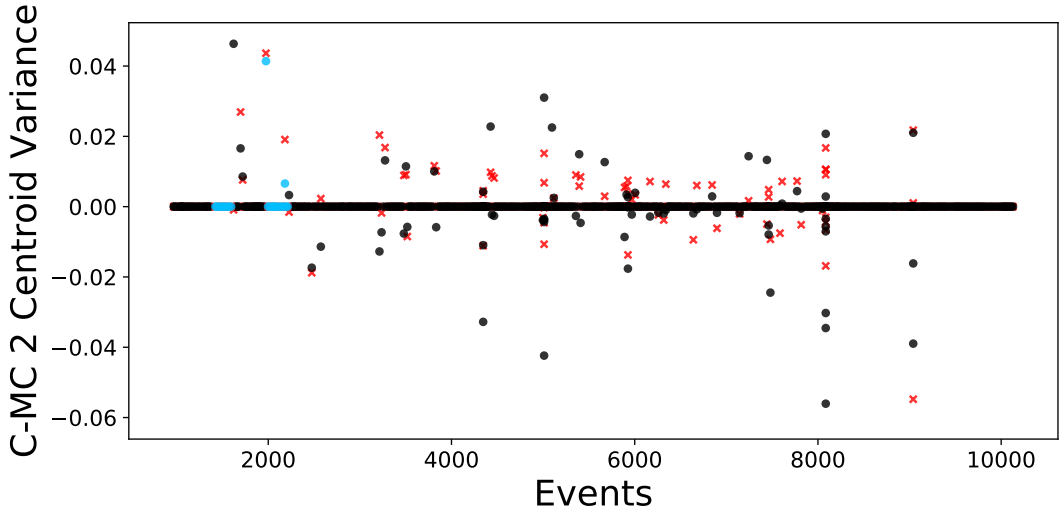
To explore CDESF drift detection, we used *ORI5k* synthetic event log using TH = 3 days; $\epsilon = 0.1$; $\lambda = 0.05$; $\beta = 0.3$; $\mu = 4$; $v = 100$. Figure 25a shows the five C-MCs detected from the first 25000 events in the process. The MCs centroid variation was tracked, black dots represent a trace variation and red crosses represent time variation.

The authors in [10] created this dataset by interspacing two different concepts every 10000 events. This way, representing drift behaviour. By analysing Figure 25a, we can see that C-MC 2 faded a short time after 10000 events, meaning that our framework was able to identify the stated drift at that point. At that point in the stream, new events presented new characteristics, which imply in different GD_{trace} and GD_{time}, thus instances ceased to fall inside that MC, and its weight decayed over time.

According to the authors [10], the *ORI5k* event log also has a drift occurring around 20000 events. However, the concept that appears is not novel since it had already appeared in the stream. Therefore, Figure 25a does not show a C-MC fading or emerging at that point. This means that the PMG was able to model the reappearing concept. Thus, when the drift occurred after 20000 events, the PMG was prepared to handle the



(a) *ORI5k* C-MCs centroid position variance. Black dots represent trace variance and red crosses relate to time variance



(b) Investigation of C-MC 2 behaviour. Black and blue dots represent C-MC and P-MC, respectively. After 10000 events the cluster fades, indicating a concept drift

Figure 25 – *ORI5k* C-MCs centroid position variance

recurring concept. Hence, the existing clusters will shelter the new cases, and no alarming is needed since the nature of these cases has been modelled as common.

We selected C-MC 2 to evaluate its behaviour. Figure 25b shows C-MC 2 centroid variation. The colour of the dots denote the MC state, blue represents a P-MC and black represents a C-MC. Red crosses represent time variance. At the beginning of the log, after 2000 events, C-MC 2 lost weight and became a P-MC (denoted by blue dots). However, it comes back to being a C-MC and sustains that position for some time. Around 9000 events, the cluster maintains its centroid position stable after one last variation in both trace and time, seen as the last vertical variation in Figure 25b. Indeed, a cluster without centroid position variance is likely to be a cluster that is not receiving new points, so it never changes its position. Consequently, this means that the cluster is probably dying out, which is the case not so long after ten thousand events.

6.1.3.3 Tracking Micro-Cluster Density: An Example

This experiment aims at investigate the density variation in a given MC. Thus, demonstrating that in different points of the stream, the MC behaviour might evolve. For that, we applied the *Assembly_IW-Frozen* to CDESF with TH = 3 days; $\epsilon = 0.1$; $\lambda = 0.05$; $\beta = 0.3$; $\mu = 4$; $v = 100$.

Figure 26 shows the behaviour of MC 60 during its existence in the stream. Around 6000 events, MC 60 starts as an O-MC (denoted by red dots), then alternates between a P-MC (denoted by blue dots) and a C-MC (denoted by black dots) until it stabilises. At around 7500 events, it becomes a P-MC. Then, its weight decays until it disappears.

This is a recurring behaviour in real-life event logs where some concept starts as an outlier, becomes common and then stops existing. MC 60 starts as an outlier because its behaviour is new and very few examples are available at event 6041. However, with the arrival of more similar behaviour, the MC's weight rises. Hence, becoming a high-density region. Thus, it evolves into a C-MC in event 6189. It stays as a C-MC for some period in the stream. However, with a possible change in the organisation, its behaviour becomes more and more rare, which implies weight decay. Finally, MC 60 becomes a P-MC and fades away shortly after. Both the appearance and fading of MC 60 as a C-MC are considered drifts.

In order to clarify the phenomenon seen in MC 60, we explored the feature space at events 6041, 6189 and 7806. First, we refer to Figure 27a that shows all clusters after the event 6041. There are 1400 different cases (TC), and the current CP is 123. Here, we also use the same colour scheme as before, C-MCs are black, P-MCs blue and O-MCs red. The green number attached to the cluster is the MC ID. Yellow crosses represent cases placed outside all clusters' boundaries, which are anomalous behaviour, with differences in trace and time.

MC 60's placement compared to MC 1, which is the most common behaviour throughout the whole event stream, is shifted up to the right in the feature space. Varia-

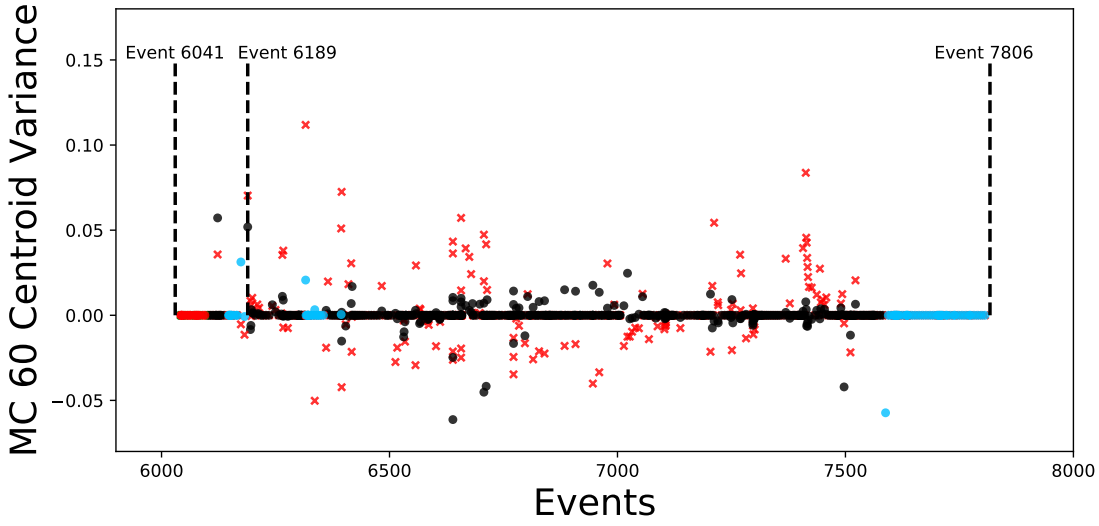


Figure 26 – MC 60 behaviour from *Assembly_IW-Frozen*

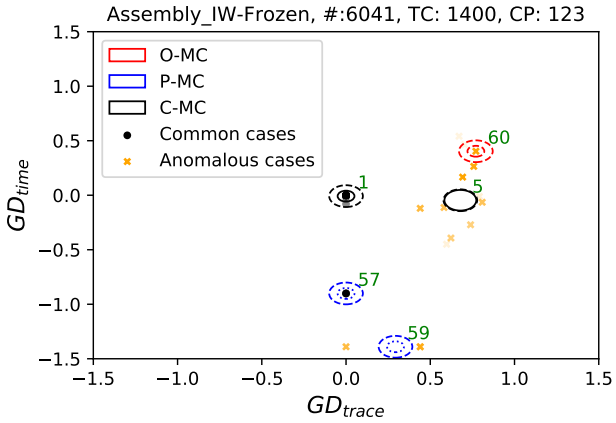
tions in the horizontal mean a difference in trace composition (GD_{trace}), whereas variations in the vertical mean differences in the time inter-activities (GD_{time}). At this point, MC 60 is an O-MC (red colour). Figure 27b shows the evolution of MC 60 after 6189 events (148 events after Figure 27a). Now, MC 60 is a C-MC, representing common behaviour similar to C-MC 10 concerning GD_{trace}, but different in the time inter-activities (GD_{time}). Figure 27c shows the last depiction of MC 60 (at event 7806) when it is a P-MC. There are no recent cases placed inside the MC at this moment. At the same time, the DenStream decay factor (λ) is affecting MC 60 weight. Thus, making MC 60 fade away in the next few events, as shown in Figure 26.

6.2 The Impact of Hyperparameters Configurations

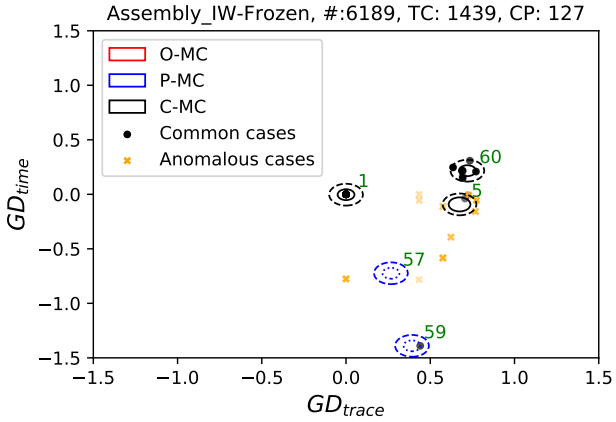
To make its use simpler, CDESF abstracts the use of hyperparameters. TH is the only hyperparameter non-related to clustering. Its impact is mostly related to the adaptation of the PMG.

Smaller THs result in cases being processed mostly in an incomplete form, since they imply more frequent CPs, generating graphs with less information and capturing an excessive number of incomplete cases. Hence, THs lower than the mean case time produce a higher rate of O-MCs since there is less time for learning. Moreover, C-MCs are harder to be maintained, making the identification of standard behaviour less practical.

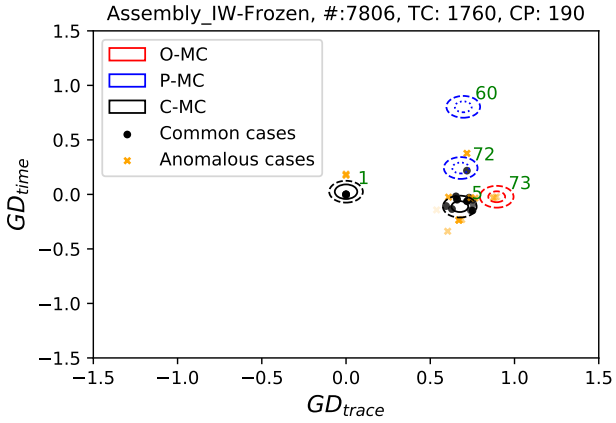
Contrarily, a large TH implies fewer CPs, which are crucial for the characterisation of outliers and common behaviour. The larger the TH, the framework gets more similar to a batch approach. However, with larger THs, the PMG takes a longer time to adapt to drifts, since CPs are not as frequent as with smaller THs. Hence, a TH slightly higher



(a) Snapshot in event 6041. MC 60 is an O-MC at this moment



(b) Snapshot in event 6189. MC 60 is an C-MC at this moment



(c) Snapshot in event 7806. MC 60 is an P-MC at this moment

Figure 27 – *Assembly_IW-Frozen* MCs snapshots in different points in the stream. MC 60 first appears as an O-MC. Then, it evolves into a C-MC. Finally, it becomes a P-MC before fading away

than the mean case time yield the optimal solution capable of obtaining good response times, identifying core behaviours and generating fewer outliers. Since a specialist knows the mean time of a process instance, choosing this hyperparameter is usually trivial.

Though the TH hyperparameter affects the PMG adaptation, CDESF controls its influence with the Nyquist sampling theorem. This way, CDESF balances adaptation by using Nyquist to control the number of cases necessary for updating.

Regarding the DenStream hyperparameters, we performed experiments to evaluate them separately. For that, we varied one hyperparameter and maintained the others. The standard hyperparameters were TH = 3 days; $\epsilon = 0.1$; $\lambda = 0.1$; $\beta = 0.2$; $\mu = 4$; $v = 100$. The number of O-MCs and CDs detected were counted and summed for all event logs.

There is a direct correlation between ϵ and the number of O-MCs detected since it comprehends to the maximum actuation range of an MC. The higher the ϵ, the fewer O-MCs are formed. Moreover, we found that higher ϵ causes the framework to detect fewer CDs, as shown in Figure 28a. Thus, we can observe that a higher ϵ makes the framework less sensitive. Therefore, when choosing the right ϵ, a specialist should consider a trade-off between sensibility (which may generate more false positives but would likely detect all anomalous cases) and stability (which may generate less false positives at the cost of maybe missing some anomalous events).

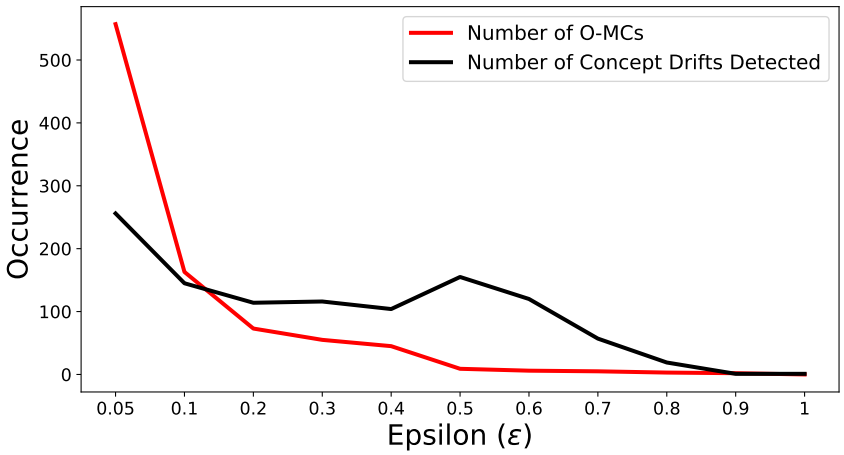
The same pattern happens with the v hyperparameter, as Figure 28b shows — higher v yields fewer CDs and O-MCs. However, for v from 1000 to 5000, we observed that the number of O-MCs were maintained. This phenomenon happened because of higher v impact less in the MCs weights. Thus, the O-MCs were able to stay active for a longer period.

Figure 28c shows the result for the μ test. Regarding CD detection, a higher μ usually finds more drifts. This happens because C-MCs are harder to be maintained in such larger μ. Thus, the MC probably switches between the P-MC and C-MC states often, raising the alerts for drifts. Regarding O-MC detection, the peak is at μ 500, then it decays slightly.

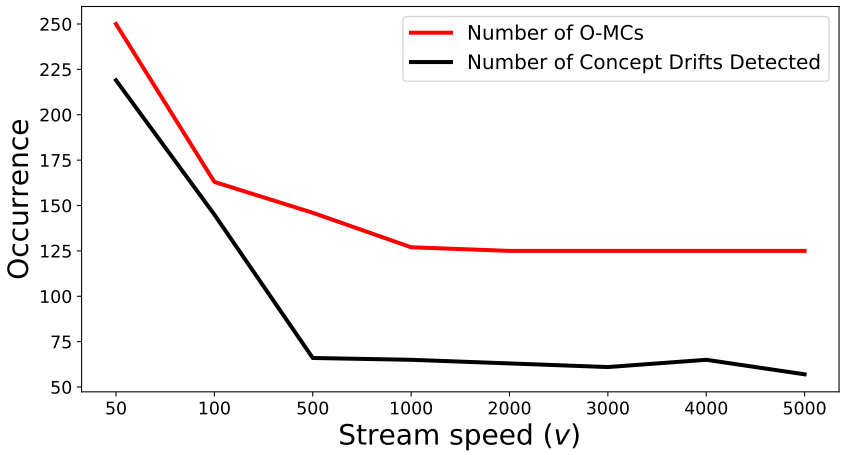
The β hyperparameter is tied with μ. Thus, the same behaviour is expected. A small β allows many clusters to be a core cluster, and a high β makes it harder for clusters to climb to a core position. Finally, as stated by Cao et al. [33], a good β value for clustering quality ranges between 0.2 and 0.6.

Usually, either a relatively high or low λ value decreases the clustering quality. Hence, the higher the λ value, the lower importance has historical data compared to more recent data. The contrary happens for lower λ, making the clustering more resistant to newer knowledge, i.e. harder to update when a drift occurs.

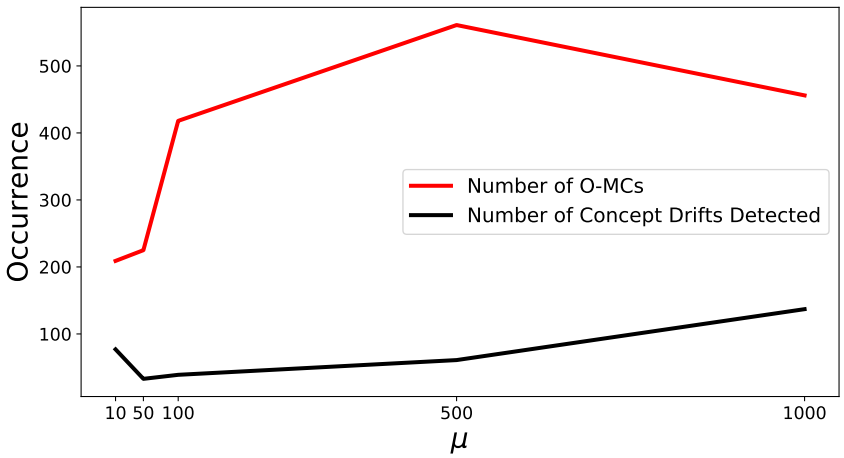
In this context, the sensitivity of DenStream hyperparameters permits a wide range



(a) Evaluation of ϵ hyperparameter. With higher ϵ, less O-MCs and CDs are detected



(b) Evaluation of v hyperparameter. With higher v, less O-MCs and CDs are detected



(c) Evaluation of μ hyperparameter. O-MC and CD detection behave differently with higher μ

Figure 28 – Evaluation of isolated hyperparameters. The number of O-MCs and CDs detected were counted

of configurations, which give users room to work towards their goal and adapt according to the business process characteristics.

6.3 Comparing Concept Drift Detection Techniques

This experiment aims at comparing the available CD detection techniques listed in Table 3. For that, we used the 18 synthetic event logs proposed in [10]. Each log contains 9 concept drifts. All tools were performed with default hyperparameters. For CDESF, we used TH = 3 days; $\epsilon = 0.1$; $\lambda = 0.1$; $\beta = 0.2$; $\mu = 4$; $v = 100$. Note that we used the latest version of each technique. The results are listed in Table 6.

Table 6 – Number of drifts found by each technique

Event Logs	# Drifts	[16]	[12]	CDESF
cb5k	9	11	1	8
cd5k	9	11	9	8
cf5k	9	11	9	8
cm5k	9	11	9	8
cp5k	9	11	9	8
fr5k	9	11	0	8
IOR5k	9	11	9	8
IRO5k	9	16	9	8
lp5k	9	11	9	0
OIR5k	9	12	9	8
ORI5k	9	13	9	8
pl5k	9	2	9	8
pm5k	9	11	9	8
re5k	9	1	9	0
RIO5k	9	11	9	8
ROI5k	9	11	9	8
rp5k	9	11	9	8
sw5k	9	11	9	8

Bose et al. [11] perform an offline technique and was not able to find any drift in the synthetic logs. Among the four tested methods, [11] is the one with the highest number of hyperparameters, providing a wide range of configurations at the cost of making the tool rather complex to set up. Thus, these results may come from lack of hyperparameters tuning.

Ostovar et al. [12], which was proposed by the same authors that provided the log datasets, had a better performance identifying all drifts from 16 out of 18 processes, as reported in Table 6. The window size is one of the most influential hyperparameters of this technique, and its standard value complies with the characteristics of the synthetic logs. Moreover, this tool provides hyperparameters for filtering out the noise. Though the

noise is filtered, this information is not outputted. However, noisy cases in a business event log are anomalous and may be of interest for stakeholders.

Zheng et al. [16] detects 11 drifts in 14 out of 18 event logs. This technique is also based on statistic tests over windows. The results are acceptable considering the datasets contained 9 drifts and that no tuning was performed.

In CDESF, either the detection or fading of a C-MC is considered a drift, with that, the tool was able to find 8 out of 9 drifts in 16 event logs. Since the PMG learns several behaviours over trace and time, when a group of traces is repeated in the stream, that does not necessarily yield a drift warning because the framework recognises the recurring behaviour.

CDESF performance is close to techniques specialised in drift detection. Moreover, CDESF offers a complete view of the business process by providing the current model, detecting anomalous cases and identifying drifts. Contrarily, the other detection techniques fail to provide insights of the business process.

In this context, a comparison among different tools is not an easy task considering there is no standard definition of drift in PM and how to measure it. Furthermore, the synthetic event logs available in the literature are not a completely general representation of drifts in processes since they only contain trace related sudden drifts, ignoring both inter-activity time and other types of drifts.

7 CONCLUSION

Organisations run several business processes to produce a service or a product. Process Mining is the field that gives organisations insights about their processes. Moreover, PM analysis always use the event log (recorded business processes events) as the starting point for analysis. Traditional PM techniques need a complete event log, and for that, the business process must have run for a considerable time. However, organisations need a fast response to understand their processes. Considering that data is being produced at a fast rate, online solutions are needed.

This work proposes Concept Drift in Event Stream Framework, a framework for clustering process instances in the form of event streams. The framework uses graphs as a model representation of the process, supporting the extraction of graph distances between the model and the case. The DenStream algorithm was used for clustering, sustaining anomalous case identification and adaptation through time. Regarding concept drift challenges, CDESF presented good results by using new data as the basis of adaptation to update the model representation. Furthermore, CDESF deals with incomplete cases in a stream of events. This is supported by the graph distance approach, which does not require complete cases when computing distance.

7.1 Main Contributions

In the experiments, we applied a varied set of both real and synthetic event data to analyse CDESF behaviour. As a result, the framework showed it could detect anomalous cases as much as adapt to different concepts through time.

Hypothesis 1 proposes the viability of dealing with PM tasks in an online scenarios. CDESF handles PM needs (process discovery, conformance checking and process enhancement) in an online environment, which presents several constraints and challenges, such as drift detection and memory limitations. Thus, Hypothesis 1 is satisfied since the framework provides stakeholders with a complete view of the process. Lastly, CDESF yields information such as the process model, anomaly detection and concept drift monitoring.

Hypothesis 2 proposes the use of the time delta between events as a case feature. This way, CDESF also meets Hypothesis 2 since it successfully used time delta as a case descriptor. This enabled the clusterisation phase to find anomalous behaviour from a time perspective.

We compared CDESF concept drift detection with other state-of-the-art techniques. CDESF had a good performance by identifying almost all drifts from the synthetic event logs. Moreover, CDESF provides additional information that other techniques do

not.

7.2 Limitations

Some limitations are intrinsic to the area of this work. First, there are few available event streams for testing. Additionally, there is no standard way of measuring and detecting drifts.

Regarding CDESF, an initial limitation is that the framework is more academic. Thus, it is not so close to organisations day-to-day reality. An effort to implant CDESF in real life scenarios would be interesting to overcome this gap.

Furthermore, this work main focus is to propose a framework to detect CD in online event streams and to provide insights into business processes. For that, we have compared the approach with other drift detection techniques. However, no comparison with other process discovery techniques was performed. Though this path was not explored mostly because there is no standard way of comparing discovery tools in online PM. Moreover, there are no conformance checking metrics for online PM.

7.3 Future Work

Though CDESF covers several aspects of online PM, there are several avenues for future work:

- Test CDESF's capability of finding specific drift types (sudden, incremental, gradual, recurring);
- Expand CDESF to act on different case descriptors, such as the author of an action, resources used and cost required;
- Explore other graph distances in the PMG;
- Given that there is only one group of synthetic event logs with limited drift variations, we aim at providing benchmark event logs to explore all types of drifts. Furthermore, drifts from other perspectives (e.g. time drift) should be simulated;
- Currently, there is no consensus around what consists a process drift. Moreover, basic PM tasks are not well formalised for online scenarios. Thus, it would be interesting to propose requirements for Process Mining on Data Streams.

BIBLIOGRAPHY

[1] AALST, W. M. P. van der. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2011. ISBN 3642193447, 9783642193446.

[2] JUHAňáK, L.; ZOUNEK, J.; ROHLíKOVá, L. Using process mining to analyze students' quiz-taking behavior patterns in a learning management system. *Computers in Human Behavior*, 2017. ISSN 0747-5632. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0747563217306957>.

[3] DUMAS, M. et al. *Fundamentals of Business Process Management*. Germany: Springer, 2013. ISBN 978-3-642-33142-8.

[4] AALST, W. M. P. van der. Business process management demystified: A tutorial on models, systems and standards for workflow management. In: _____. *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 1–65.

[5] KRAWCZYK, B. et al. Ensemble learning for data stream analysis: A survey. *Information Fusion*, v. 37, p. 132 – 156, 2017. ISSN 1566-2535. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1566253516302329>.

[6] GAMA, J. *Knowledge Discovery from Data Streams*. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2010. ISBN 1439826110, 9781439826119.

[7] GAMA, J. a. et al. A survey on concept drift adaptation. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 46, n. 4, p. 44:1–44:37, mar. 2014. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2523813>.

[8] ZELST, S. J. van; DONGEN, B. F. van; AALST, W. M. van der. Event stream-based process discovery using abstract representations. *Knowledge and Information Systems*, Springer, v. 54, n. 2, p. 407–435, 2018.

[9] Jagadeesh Chandra Bose, R. *Process mining in the large : preprocessing, discovery, and diagnostics*. Tese (Doutorado) — Department of Mathematics and Computer Science, 2012. Proefschrift.

[10] MAARADJI, A. et al. Fast and accurate business process drift detection. In: *Proceedings of the 13th International Conference on Business Process Management - Volume 9253*. Berlin, Heidelberg: Springer-Verlag, 2015. p. 406–422. ISBN 978-3-319-23062-7. Disponível em: <https://doi.org/10.1007/978-3-319-23063-4_27>.

[11] BOSE, R. P. J. C. et al. Dealing with concept drifts in process mining. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 1, p. 154–171, Jan 2014. ISSN 2162-237X.

[12] OSTOVAR, A. et al. Detecting drift from event streams of unpredictable business processes. In: COMYN-WATTIAU, I. et al. (Ed.). *Conceptual Modeling*. Cham: Springer International Publishing, 2016. p. 330–346. ISBN 978-3-319-46397-1.

[13] BURATTIN, A. Streaming process discovery and conformance checking. In: *Encyclopedia of Big Data Technologies*. [S.l.]: Springer, 2018.

[14] Batyuk, A.; Voityshyn, V. Streaming process discovery for lambda architecture-based process monitoring platform. In: *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*. [S.l.: s.n.], 2018. v. 1, p. 298–301.

[15] AALST, W. van der et al. Process mining manifesto. In: DANIEL, F.; BARKAOUI, K.; DUSTDAR, S. (Ed.). *Business Process Management Workshops*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 169–194. ISBN 978-3-642-28108-2.

[16] ZHENG, C.; WEN, L.; WANG, J. Detecting process concept drifts from event logs. In: PANETTO, H. et al. (Ed.). *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*. Cham: Springer International Publishing, 2017. p. 524–542. ISBN 978-3-319-69462-7.

[17] WITTEN, I. H.; FRANK, E.; HALL, M. A. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123748569, 9780123748560.

[18] AALST, W. van der; WEIJTERS, T.; MARUSTER, L. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, v. 16, n. 9, p. 1128–1142, Sept 2004. ISSN 1041-4347.

[19] LEEMANS, S. J. J.; FAHLAND, D.; AALST, W. M. P. van der. Discovering block-structured process models from event logs containing infrequent behaviour. In: LOHMANN, N.; SONG, M.; WOHED, P. (Ed.). *Business Process Management Workshops*. Cham: Springer International Publishing, 2014. p. 66–78. ISBN 978-3-319-06257-0.

[20] WEIJTERS, A. J. M. M.; AALST, W. M. P. van der. Rediscovering workflow models from event-based data using little thumb. *Integr. Comput.-Aided Eng.*, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 10, n. 2, p. 151–162, abr. 2003. ISSN 1069-2509. Disponível em: <http://dl.acm.org/citation.cfm?id=1273320.1273325>.

[21] ROZINAT, A.; AALST, W. van der. Conformance checking of processes based on monitoring real behavior. *Information Systems*, v. 33, n. 1, p. 64 – 95, 2008. ISSN 0306-4379. Disponível em: <http://www.sciencedirect.com/science/article/pii/S030643790700049X>.

[22] AALST, W. van der; SCHONENBERG, M.; SONG, M. Time prediction based on process mining. *Information Systems*, v. 36, n. 2, p. 450 – 475, 2011. ISSN 0306-4379. Special Issue: Semantic Integration of Data, Multimedia, and Services. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0306437910000864>.

[23] JENSEN, K.; KRISTENSEN, L. M. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 3642002838, 9783642002830.

[24] REISIG, W.; ROZENBERG, G. (Ed.). *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the Volumes Are Based on the Advanced Course on Petri Nets*. London, UK, UK: Springer-Verlag, 1998. ISBN 3-540-65306-6.

[25] AALST, W. van der; STAHL, C. *Modeling Business Processes: A Petri Net-Oriented Approach*. [S.l.]: The MIT Press, 2011. ISBN 0262015382, 9780262015387.

[26] MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, Apr 1989. ISSN 0018-9219.

[27] REISIG, W. *Petri Nets: An Introduction*. Springer, 1985. v. 4. (EATCS Monographs on Theoretical Computer Science, v. 4). ISBN 3-540-13723-8. Disponível em: <https://doi.org/10.1007/978-3-642-69968-9>.

[28] HUANG, Y. et al. Efficient business process consolidation: combining topic features with structure matching. *Soft Computing*, v. 22, n. 2, p. 645–657, Jan 2018. ISSN 1433-7479. Disponível em: <https://doi.org/10.1007/s00500-016-2364-y>.

[29] DIJKMAN, R. et al. Similarity of business process models: Metrics and evaluation. *Information Systems*, v. 36, n. 2, p. 498 – 516, 2011. ISSN 0306-4379. Special Issue: Semantic Integration of Data, Multimedia, and Services. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0306437910001006>.

[30] COSTA, V. G. T. da; CARVALHO, A. C. P. de Leon Ferreira de; JUNIOR, S. B. Strict very fast decision tree: a memory conservative algorithm for data stream mining. *CoRR*, abs/1805.06368, 2018. Disponível em: <http://arxiv.org/abs/1805.06368>.

[31] TSYMBAL, A. *The Problem of Concept Drift: Definitions and Related Work*. [S.l.], 2004.

[32] WEBB, G. I. et al. Characterizing concept drift. *Data Min. Knowl. Discov.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 30, n. 4, p. 964–994, jul. 2016. ISSN 1384-5810. Disponível em: <http://dx.doi.org/10.1007/s10618-015-0448-4>.

[33] CAO, F. et al. Density-based clustering over an evolving data stream with noise. In: SIAM. *Proceedings of the 2006 SIAM international conference on data mining*. [S.l.], 2006. p. 328–339.

[34] AGGARWAL, C. C. et al. A Framework for Clustering Evolving Data Streams. *Proc. of the 29th int. conf. on Very large data bases*, p. 81–92, 2003. ISSN 10844627. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8650>.

[35] ESTER, M. et al. 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*. AAAI Press, 1996. (KDD'96), p. 226–231. Disponível em: <http://dl.acm.org/citation.cfm?id=3001460.3001507>.

[36] LENO, V. et al. Discovering process maps from event streams. *arXiv preprint arXiv:1804.02704*, 2018.

[37] BURATTIN, A.; SPERDUTI, A.; AALST, W. M. van der. Control-flow discovery from event streams. In: IEEE. *Evolutionary Computation (CEC), 2014 IEEE Congress on*. [S.l.], 2014. p. 2420–2427.

[38] BURATTIN, A. et al. Online discovery of declarative process models from event streams. *IEEE Transactions on services computing*, IEEE, v. 8, n. 6, p. 833–846, 2015.

[39] BURATTIN, A.; CARMONA, J. A framework for online conformance checking. In: SPRINGER. *International Conference on Business Process Management*. [S.l.], 2017. p. 165–177.

[40] AL-ALI, H. et al. Translating bpmn to business rules. In: SPRINGER INTERNATIONAL PUBLISHING. *6th International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA)*. [S.l.], 2016. p. 22–36.

[41] BOSE, R. P. J. C. et al. Handling concept drift in process mining. In: MOURATIDIS, H.; ROLLAND, C. (Ed.). *Advanced Information Systems Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 391–405. ISBN 978-3-642-21640-4.

[42] MAGGI, F. M. et al. Online process discovery to detect concept drifts in ltl-based declarative process models. In: SPRINGER. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. [S.l.], 2013. p. 94–111.

[43] SEELIGER, A.; NOLLE, T.; MüHLHäUSER, M. Detecting concept drift in processes using graph metrics on process graphs. In: *Proceedings of the 9th Conference on Subject-oriented Business Process Management*. New York, NY, USA: ACM, 2017. (S-BPM ONE '17), p. 6:1–6:10. ISBN 978-1-4503-4862-1. Disponível em: <http://doi.acm.org/10.1145/3040565.3040566>.

[44] BURATTIN, A.; SPERDUTI, A.; AALST, W. M. P. van der. Heuristics miners for streaming event data. *CoRR*, abs/1212.6383, 2012. Disponível em: <http://arxiv.org/abs/1212.6383>.

[45] CARMONA, J.; GAVALDÀ, R. Online techniques for dealing with concept drift in process mining. In: HOLLMÉN, J.; KLAWONN, F.; TUCKER, A. (Ed.). *Advances in Intelligent Data Analysis XI*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 90–102. ISBN 978-3-642-34156-4.

[46] Maaradji, A. et al. Detecting sudden and gradual drifts in business processes from execution traces. *IEEE Transactions on Knowledge and Data Engineering*, v. 29, n. 10, p. 2140–2154, Oct 2017. ISSN 1041-4347.

[47] Alves De Medeiros, A. et al. *Process mining: extending the alpha-algorithm to mine short loops*. [S.l.]: Technische Universiteit Eindhoven, 2004. (BETA publicatie : working papers). ISBN 90-386-1978-2.

[48] ACCORSI, R.; STOCKER, T. Discovering workflow changes with time-based trace clustering. In: ABERER, K.; DAMIANI, E.; DILLON, T. (Ed.). *Data-Driven Process Discovery and Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 154–168. ISBN 978-3-642-34044-4.

[49] Liu, N.; Huang, J.; Cui, L. A framework for online process concept drift detection from event streams. In: *2018 IEEE International Conference on Services Computing (SCC)*. [S.l.: s.n.], 2018. p. 105–112. ISSN 2474-2473.

[50] JUNIOR, S. B. et al. A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. In: *Companion Proceedings of the The Web Conference 2018*. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018. (WWW '18), p. 319–326. ISBN 978-1-4503-5640-4. Disponível em: <https://doi.org/10.1145/3184558.3186343>.

[51] LÉVESQUE, L. Nyquist sampling theorem: understanding the illusion of a spinning wheel captured with a video camera. *Physics Education*, v. 49, n. 6, p. 697–705, 2014. ISSN 0031-9120.

[52] MANNHARDT, F. et al. Data-driven process discovery-revealing conditional infrequent behavior from event logs. In: SPRINGER. *International Conference on Advanced Information Systems Engineering*. [S.l.], 2017. p. 545–560.

[53] CERAVOLO, P. et al. Toward a new generation of log pre-processing methods for process mining. In: _____. *Business Process Management Forum: BPM Forum 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*. Cham: Springer International Publishing, 2017. p. 55–70. ISBN 978-3-319-65015-9. Disponível em: <https://doi.org/10.1007/978-3-319-65015-9_4>.

[54] BEZERRA, F.; WAINER, J.; AALST, W. M. P. van der. Anomaly detection using process mining. In: HALPIN, T. et al. (Ed.). *Enterprise, Business-Process and Information Systems Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 149–161. ISBN 978-3-642-01862-6.

WORKS PUBLISHED BY THE AUTHOR

Works published by the author during the program.

1. S. Barbon, G. M. Tavares, V. G. T. da Costa, P. Ceravolo, E. Damiani, **A Framework for Human-in-the-loop Monitoring of Concept-drift Detection in Event Log Stream**, The 2018 Web Conference Companion (WWW), Lyon, France, 2018, (Qualis CC 2018, A1)

2. G. M. Tavares, V. G. T. da Costa, V. E. Martins, P. Ceravolo, S. Barbon, **Anomaly detection in business process based on data stream mining**, Brazilian Symposium on Information Systems (SBSI), Caxias do Sul, Brazil, 2018, (Qualis CC 2018, B2) — Best Paper Award

3. G. M. Tavares, V. G. T. da Costa, V. E. Martins, P. Ceravolo, S. Barbon, **Leveraging Anomaly Detection in Business Process with Data Stream Mining**, Brazilian Journal of Information Systems (iSys), 2019, (Qualis CC 2018, B3)

4. N. J. Omori, G. M. Tavares, P. Ceravolo, S. Barbon, **Comparing Concept Drift Detection with Process Mining Tools**, Brazilian Symposium on Information Systems (SBSI), Aracaju, Brazil, 2019, (Qualis CC 2018, B2)

5. J. I. Morais; H. Q. Abonizio; G. M. Tavares; A. A. Fonseca; S. Barbon, **Deciding among Fake, Satirical, Objective and Legitimate news: A multi-label classification system**, Brazilian Symposium on Information Systems (SBSI), Aracaju, Brazil, 2019, (Qualis CC 2018, B2)

6. G. M. Tavares, S. M. Mastelini, S. Barbon, **User classification on online social networks by post frequency**, Brazilian Symposium on Information Systems (SBSI), Lavras, Brazil, 2018, (Qualis CC 2018, B2)

7. S. Barbon, G. M. Tavares, G. S. Kido, **Artificial and Natural Topic Detection in Online Social Networks**, Revista Brasileira de Sistemas de Informação (iSys), March, 2017, (Qualis CC 2018, B3)

8. S. Barbon, G. F. C. Campos, G. M. Tavares, R. G. Igawa, M. L. Proença, R. C. Guido, **Detection of Human, Legitimate Bot, and Malicious Bot in Online Social Networks Based on Wavelets**, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), March, 2018, (Qualis CC 2018, B1)