UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

RAFAEL GASPAR DE SOUSA

**Trace clustering approach for detection and localization of concept drift in
business processes**

São Paulo

2021

RAFAEL GASPAR DE SOUSA

**Trace clustering approach for detection and localization of concept drift in
business processes**

A dissertation submitted to the School of Arts, Science and Humanities of the University of São Paulo in partial fulfillment of the requirements for the degree of Master of Science from the Graduate Program in Information Systems.

Concentration area: Computing Methodology and Techniques

Corrected version containing the changes request by the reading committee in October 28th, 2021. The original version can be found in reserved collection at EACH-USP's Library and at USP's Digital Library of Thesis and Dissertations (BDTD), according to the Resolution CoPGr 6018, from October 13th, 2011.

Supervisor: Dr. Sarajane Marques Peres

Co-supervisor: Dr. Hajo A. Reijers

São Paulo

2021

Authorize the reproduction and dissemination of total or partial copies of this document, by conventional or electronic media for study or research purpose, since it is referenced.

CATALOGUING IN PUBLICATION

(University of São Paulo. School of Arts, Sciences and Humanities. Library)

CRB 8 -4936

Sousa, Rafael Gaspar de

Trace clustering approach for detection and locatization of
concept drift in business processes / Rafael Gaspar de Sousa ;
supervisor, Sarajane Marques Peres ; co-supervisor, Hajo A. Reijers.
– 2021

134 p. : il.

Dissertation (Master of Science) – Graduate Program in
Information Systems, School of Arts, Sciences and Humanities,
University of São Paulo.

Corrected version.

1. Data mining. 2. Process mining. 3. Trace clustering. I.
Peres, Sarajane Marques, supervisor II. Reijers, Hajo A., co-
supervisor. III. Title

CDD 22.ed.– 006.312

Dissertation authored by Rafael Gaspar de Sousa, entitled “**Trace clustering approach for detection and localization of concept drift in business processes**”, submitted to the School of Arts, Sciences and Humanities of the University of São Paulo, in partial fulfillment of the requirements for the degree of Master of Science from the Graduate Program in Information Systems, in concentration area of Computing Methodology and Techniques, approved on October 28th, 2021 by the reading committee composed of the following members:

Professor Sarajane Marques Peres

School of Arts, Science and Humanities - University of São Paulo

Chair of the committee

Professor André Carlos Ponce de Leon Ferreira de Carvalho

Institute of Mathematics and Computer Sciences - University of São Paulo

Professor Hélio Côrtes Vieira Lopes

Department of Informatics - Pontifical Catholic University of Rio de Janeiro

Acknowledgements

First and foremost, I would like to thank God for providing me the environment, resources, love and opportunities I have encountered during my whole life. I am truly blessed and it is an honor to find myself with such conditions while other people, for multiple reasons, may not have had the same fortune.

I thank my mother, Angélica, for sitting outside my bedroom late at night after work to review my homework when I was little and for showing me the value of studying. I thank my father, Hélio, for being my first teacher on computers, showing me how to format PCs and opening the door to this world that I love. I thank my brother, David, for teaching me confidence and the importance of hard work. I thank my sister, Letícia, for opening the roads for me and showing me that it was possible to enter a great university for free. I am forever grateful to all my family for the love, encouragement and support I received throughout my history.

To my partner in life, Barbara, I am really grateful for all the love, patience, support, thoughtfulness and kindness we have with each other. You have provided me with the fundamental factors of motivation, fun and easiness during the time of this research, specially when needed the most. I would like to thank you for all our coffee, tea, snacks and late nights studying, even though some of them separated, together at heart.

I am sincerely grateful to Professor Sarajane Marques Peres for the inspiring dedication and assistance in every step of this work. For all the conversations about life, cats, academia, research and work. You are a role model of professor, researcher and person that I look up. After every meeting I felt more motivated and less heavy. Thank you for your affection and all the lessons I learned with you that reach far beyond this research.

I would like to offer my special thanks to Professor Hajo Alexander Reijers and Professor Marcelo Fantinato for all their insightful comments and suggestions that contributed a lot to this research. I also thanks all the teachers and professors I have found along the way (specially “*tia* Cris”, Arlen, Alex, Junior and Vanessa). You all believed in me and went beyond the required in your jobs to inspire me.

Finally, I would like to thank all my friends that got me through graduation and all the shared knowledge during this research (specially Thais, Waldyr, Esther, Vitor, Geraldo, Fernando and Matheus). I would not have made it without all of you.

“I would rather have questions that can’t be answered than answers that can’t be questioned.”

(Richard Feynman)

Abstract

SOUSA, Rafael Gaspar de. **Trace clustering approach for detection and localization of concept drift in business processes**. 2021. 133 f. Dissertation (Master of Science) – School of Arts, Science and Humanities, University of São Paulo, São Paulo, Brasil 2021.

Business processes are constantly subject to changes over time due to the need for adaptation and flexibility in the complex environment they operate, such as new clients' demands, competition, or legislation. Process models are one of the fundamental tools when understanding a process behavior, which is key for business success. However, these process models are usually not documented and updated to agree with eventual changes in process behavior over time, leading to misconceptions in the understanding of the actual process. Although process mining aims to provide techniques that discover, analyze, and enhance process automatically based on event logs, most techniques assume that the process is stationary, which is not often the case. Handling the problem of processes changing over time, known as concept drift, leads to the capability of detecting drift as soon as possible and localizing the entities involved in them, providing a much better comprehension of the process behavior that can be a competitive advantage for businesses. Most of the work on dealing with concept drift in the process mining literature focuses on providing a framework that is able to detect drifts, but are generally not adequate to simultaneously localize the change inside the process behavior and exhibit information on the entities involved. Applying clustering techniques to data from event logs, known as trace clustering, supports the identification of patterns in the process behavior that enable simplification and segregation of similar behaviors that produces a model of the process behavior as clusters. However, although common in general process mining, trace clustering has not been widely explored in the context of the concept drift problem. This research presents a method to simultaneously perform concept drift detection and localization based on the same clusters obtained by online trace clustering. The clusters are able to reflect changes in complex process behavior in a simplified manner that serves as a platform for performing effective drift detection and localization online with no additional data structures. Experiments with synthetic and real-world event logs with different types of control-flow changes have shown that, although our method has not outperformed the baseline for drift detection in all cases, our approach was able to correctly detect drifts in most cases according to parameters configuration while also providing information about the entities involved in the drift from the business process perspective

Keywords: concept drift, trace clustering, process mining, data stream, clustering, data mining.

Resumo

SOUSA, Rafael Gaspar de. **Abordagem baseada em agrupamento de traços para detecção e localização de concept drift em processos de negócios**. 2021. 133 p. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, Brasil, 2021.

Processos de negócios estão constantemente em mudança ao longo do tempo devido à necessidade de adaptação e flexibilidade nos ambientes complexos em que eles operam, como novas demandas de clientes, competição ou legislação. Modelos de processos são uma das principais ferramentas utilizadas para se entender o funcionamento de processo. Entretanto, esses modelos costumam não serem documentados ou atualizados frequentemente para se adequarem a eventuais mudanças no comportamento do processo, gerando equívocos no entendimento do processo real. Embora a área de mineração de processos tem por objetivo desenvolver técnicas para descobrir, analisar e melhorar processos automaticamente a partir de *logs* de eventos, a maioria dos métodos assumem que o processo é estacionário, o que frequentemente não é o caso. Lidar com esse problema de processos variando ao longo do tempo, conhecido como *concept drift*, busca prover a capacidade de detectar o quanto antes um *drift* e localizar as entidades envolvidas nele, levando a uma compreensão muito melhor do processo de negócio em questão, o que pode ser uma vantagem competitiva para o negócio. A maioria dos trabalhos que lida com *concept drift* em processos de negócios foca no desenvolvimento de métodos capazes de realizar a detecção de drifts, mas geralmente não são capazes de simultaneamente localizar o drift dentro do comportamento do processo e revelar as entidades envolvidas no *drift*. Aplicando técnicas de clusterização em dados de *logs* de eventos, conhecido como *trace clustering*, é possível identificar padrões dentro do comportamento do processo que possibilitam a simplificação e segregação de comportamentos similares que produzem um modelo que representa o comportamento do processo em forma de *clusters*. Contudo, embora comum em mineração de processos no geral, *trace clustering* ainda não foi amplamente explorado dentro do contexto de *concept drift*. Este trabalho apresenta um método que realiza detecção e localização de *drifts* simultaneamente de forma online baseada nos mesmos *clusters* obtidos por *trace clustering*. Esses *clusters* são capazes de refletir as mudanças que ocorrem em processos de comportamento complexo de forma simplificada, servindo como uma plataforma para a execução de detecção e localização de *drifts* de forma efetiva sem necessidade de estrutura de dados adicional. Experimentos com *logs* de eventos sintéticos e do mundo real com diferentes tipos de mudanças de fluxo-controle demonstram que, embora nosso método não tenha superado o *baseline* em todos os casos na tarefa de detecção, nossa abordagem foi capaz de detectar *drifts* corretamente na maioria dos casos de acordo com a configuração de parâmetros utilizada, enquanto também foi capaz de prover informações sobre as entidades envolvidas no *drift* do ponto de vista do processo de negócio.

Palavras-chave: *concept drift*, agrupamento de traços, mineração de processos, *stream* de dados, agrupamento, mineração de dados.

List of Figures

Figure 1 – Example of <i>concept drift</i> in a process model	20
Figure 2 – An overview of the proposed work, presenting steps and a set of items involved in each one. Highlighted items are part of first experiments described further.	26
Figure 3 – Positioning of the three main types of process mining: discovery, conformance and enhancement.	28
Figure 4 – Example of a process model represented with BPMN	28
Figure 5 – Different structures of concept drift	30
Figure 6 – Dimensions of concept drift in process mining. Highlighted area indicates are covered by this project.	35
Figure 7 – Example of clustering data points. Each color identifies a cluster found.	36
Figure 8 – Overview of time window models	41
Figure 9 – Examples of trace clustering	44
Figure 10 – Examples of control-flow changes in a baseline process.	48
Figure 11 – Overview of the drift detection method proposed	61
Figure 12 – Example of the evolution of clustering: identifying merge, split, disappear and emerge of four clusters over iterations	64
Figure 13 – Example of time independent and time-dependent features extracted from clustering situations	65
Figure 14 – Overview of the drift localization method proposed.	71
Figure 15 – Loan Applications: Baseline process model	76
Figure 16 – An example of synthetic event log generation with concept drift	77
Figure 17 – Loan Applications: Evolution of the <i>average distance between centroids</i> for the 5,000 cases event logs, with activity-binary representation, window of size 100 and <i>k-means</i> with $k = 2$	81
Figure 18 – Loan Applications: Evolution of the <i>average distance between centroids</i> for the 5,000 cases event logs, using activity-binary representation, window of size 100 and <i>k-means</i> with $k = 2$ in not supported change patterns	81

Figure 19 – Loan Applications: Evolution of the <i>average distance between centroids</i> for the 5,000 cases event logs, using transitions-binary representation, window of size 100 and <i>k-means</i> with $k = 2$ in the activity not supported change patterns	82
Figure 20 – Loan Applications: Evolution of different features for the <i>re</i> change pattern with the 5,000 cases event log	82
Figure 21 – Loan Applications: Impact of the <i>window_size</i> parameter reflected on the evolution of <i>average distance between centroids</i> for windows of 50, 100, 150 and 250 traces	83
Figure 22 – Loan Applications: Impact of variation of the k parameter reflected on the evolution of <i>average distance between centroids</i> for values of 2, 3 and 6	83
Figure 23 – Loan Applications: Evolution of the <i>average distance between macro centroids</i> for the 5,000 cases event logs, with activity-binary representation, $h = 150$, $k_macro = 3$, $m = 30$, $t = 2$	84
Figure 24 – Loan Applications: Evolution of the <i>average distance between macro centroids</i> for several values of h in the 5,000 cases event log of change pattern <i>cb</i> , with activity-binary representation, $k_macro = 3$, $m = 30$ and $t = 2$	85
Figure 25 – Loan Applications: Evolution of the <i>average distance between macro centroids</i> for several values of m in the 5,000 cases event log of change pattern <i>cb</i> , with activity-binary representation, $k_macro = 3$, $h = 150$ and $t = 2$	85
Figure 26 – Loan Applications: Evolution of the <i>average distance between macro centroids</i> for several values of t in the 5,000 cases event log of change pattern <i>cb</i> , with activity-binary representation, $k_macro = 3$, $m = 30$ and $h = 150$	86
Figure 27 – Loan Applications: Evolution of the <i>average distance between macro centroids</i> for several values of k_macro in the 5,000 cases event log of change pattern <i>cb</i> , with activity-binary representation, $h = 150$, $m = 30$ and $t = 2$	86
Figure 28 – Loan Applications: Evolution of the <i>squared sum</i> for the 5,000 cases event logs and “no clustering” strategy, with activity-binary representation and <i>window_size</i> = 100	87

Figure 29 – Loan Applications: Process model discovery applied to each cluster before and after the drift for change pattern <i>cb</i> . Batch strategy with $k = 2$, <i>window_size</i> = 100 at the first drift.	88
Figure 30 – Loan Applications: Evolution of the <i>average distance between centroids</i> when using the three different strategies for the change pattern <i>cb</i>	89
Figure 31 – Loan Applications: Process model discovery applied to each cluster before and after the drift for change pattern <i>fr</i> . Batch strategy with $k = 2$, <i>window_size</i> = 100 at the first drift at index 5,000.	89
Figure 32 – Loan Applications: Execution of the drift detection algorithm for the <i>cb</i> with the 10,000 cases event log using $k = 2$, <i>window_size</i> = 100. The detection parameters are $\theta = 2$, <i>mb</i> = 1% and <i>rw</i> = 3	90
Figure 33 – Loan Applications: Execution of the drift detection algorithm for the <i>cb</i> with the 10,000 cases event log using $k = 2$, <i>window_size</i> = 100. The detection parameters are $\theta = 2$ and <i>rw</i> = 3	91
Figure 34 – Loan Applications: Execution of the drift detection algorithm for the <i>cb</i> with the 10,000 cases event log using $k = 2$, <i>window_size</i> = 100. The detection parameters are <i>mb</i> = 1% and <i>rw</i> = 3	92
Figure 35 – Loan Applications: Execution of the drift detection algorithm for the <i>cb</i> with the 10,000 cases event log using $k = 2$, <i>window_size</i> = 100. The detection parameters are <i>mb</i> = 1% and $\theta = 2$	93
Figure 36 – Loan Applications: Process model variation corresponding to change pattern <i>cp</i>	97
Figure 37 – Loan Applications: Drift localization results for the first drift in the 5,000 cases <i>cp</i> event log with activity-frequency representation, <i>k-means</i> with $k = 2$ and <i>window_size</i> = 100.	98
Figure 38 – Loan Applications: MSE over time for three activities in the 5,000 cases <i>cp</i> event log with activity-frequency representation and batch with $k = 2$ and <i>window_size</i> = 100.	99
Figure 39 – Loan Applications: Drift localization results for the 10,000 cases <i>cp</i> event log with activity-frequency representation, <i>k-means</i> with $k = 2$ and <i>window_size</i> = 125	100

Figure 40 – Loan Applications: Drift localization for all index and the three different strategies. Using 5,000 cases <i>cb</i> event log with activity-binary representation and <i>window_size</i> = 125, <i>k</i> = 2, <i>h</i> = 200, <i>k_macro</i> = 3, <i>m</i> = 50 and <i>t</i> = 2	101
Figure 41 – High trace variability process: Baseline model	105
Figure 42 – High trace variability process: Evolution of the <i>centroids sum of squares</i> with activity-binary trace representation for the supported change patterns, using the batch strategy with <i>window_size</i> = 150 and <i>k</i> = 3.	108
Figure 43 – High trace variability process: Execution of the drift detection algorithm for the log ConditionalMove-1 with different levels of noise. a) Using <i>window_size</i> = 100, <i>k</i> = 3 for the batch strategy with <i>rw</i> = 3, <i>std_tol</i> = 2.5 and <i>min_tol</i> = 0.01 for drift detection; b) Using <i>h</i> = 200, <i>k_macro</i> = 3, <i>m</i> = 50 and <i>t</i> = 2 for the <i>CluStream</i> strategy with <i>rw</i> = 200, <i>std_tol</i> = 1.5 and <i>min_tol</i> = 0.02 for drift detection.	109
Figure 44 – High trace variability process: Execution of the drift detection algorithm for the log ConditionalMove-1 with different levels of noise with “no clustering” <i>window_size</i> = 100 and <i>rw</i> = 3, <i>std_tol</i> = 2.5 and <i>min_tol</i> = 0.01 for drift detection.	110
Figure 45 – High trace variability process: localization for all index and the three different strategies in the log ConditionalMove-1. Using <i>window_size</i> = 150, <i>k</i> = 3, <i>h</i> = 200, <i>m</i> = 50, <i>t</i> = 2 and <i>k_macro</i> = 3	114
Figure 46 – High trace variability process: Process model discovery before and after the drift for each cluster.	115
Figure 47 – BPIC 2015: Most frequent behavior for high level abstraction in M1	115
Figure 48 – BPIC 2015: Execution of our approach for all five municipalities with the different clustering strategies.	116
Figure 49 – BPIC 2015: Results of drift localization for the first drift found in all event logs with the batch strategy. Showing the activities that had an MSE above 0 in at least one event log.	117
Figure 50 – BPIC 2015: Occurrence of the top 10 most frequent activities over time for each municipality. Vertical lines indicate found drifts. Activities that presented the highest MSE in each municipality in localization are highlighted.	118

Figure 51 – BPIC 2020: Described business process model BPI 2020	119
Figure 52 – BPIC 2020: Execution of our approach for all five municipalities with different clustering strategies	120
Figure 53 – BPIC 2020: Results of drift localization for the first drift found in all event logs with the batch strategy	122
Figure 54 – BPIC 2020: occurrence of the top twelve most frequent activities over time for each municipality. Vertical lines indicate drifts found and highlighted activities are the ones with highest MSE in the localization task.	123

List of Tables

Table 1 – Event log and a list of possible traces from the model in figure 4	29
Table 2 – Example of event log with information of activity, resource and timestamp.	46
Table 3 – Example of vector representations for profiles based on occurrence, with the counting model frequency	47
Table 4 – Example of vector representations for performance profile (in minutes). .	47
Table 5 – Comparison between the trace representations activity-binary and transitions- binary for the baseline and “a)” process models	48
Table 6 – Comparison between the trace representations activity-binary and activity- order for the baseline and “a)” process models	48
Table 7 – Comparison between the trace representations activity-binary and activity- frequency for the baseline and “b)” process models	49
Table 8 – An overview of related work in concept drift detection from the detection perspective	57
Table 9 – An overview of related work in concept drift detection from the repre- sentation perspective	58
Table 10 – An overview of related work on drift localization from the output perspective	59
Table 11 – List of time-independent features extracted for cluster tracking	67
Table 12 – List of time-dependent features extracted for cluster tracking	68
Table 13 – Loan Applications: Control-flow change patterns present in the dataset .	77
Table 14 – Loan Applications: Support of trace representation for each control-flow change patterns	78
Table 15 – Loan Applications: Parameters in the experiment for the Loan Applica- tions set of event logs.	80
Table 16 – Loan Applications: F1 score results for the drift detection task for all change patterns. The best results for each case are highlighted in bold. .	93
Table 17 – Loan Applications: Parameters settings for the best average results for each trace representations.	94
Table 18 – Loan Applications: Parameters for the best result for change pattern <i>fr</i>	95

Table 19 – Loan Applications: Comparison between the baseline (MAARADJI <i>et al.</i> , 2015b) and our concept drift detection method (F-Score results) when considering the best parameters setting per change pattern.	96
Table 20 – Loan Applications: Area under the ROC curve results for the drift localization task per change patterns	102
Table 21 – Loan Applications: Parameters for the best average results for each trace representations for the drift localization task	102
Table 22 – Loan Applications: Area under the ROC curve results for the drift localization task per change patterns for the clustering parameters from the drift detection task using the batch strategy. Highlighted items indicate differences from the results according table 20	104
Table 23 – High trace variability process: Change patterns present in event log and support by trace representation profile of our approach	106
Table 24 – High trace variability process: Parameters in the experiment for the Loan Applications set of event logs.	107
Table 25 – High trace variability process: Best results achieved for drift detection, considering the best parameter combinations for each change pattern individually	111
Table 26 – High trace variability process: Parameters for the best average results for each trace representations	112
Table 27 – High trace variability process: Best results achieved for drift detection, considering the best parameter combinations for each change pattern individually	113

List of abbreviations and acronyms

XOR	<i>Exclusive Or</i>
ROC	<i>Receiver Operating Characteristic curve</i>
AUC	<i>Area Under the Curve</i>
ERP	<i>Enterprise Resource Planning</i>
CRM	<i>Customer Relationship Management</i>
BPMN	<i>Business Process Model Notation</i>
TPR	<i>True Positive Rate</i>
FPR	<i>False Positive Rate</i>
MDT	<i>Multidisciplinary Team Review</i>
DBSCAN	<i>Density Based Spatial Clustering of Application with Noise</i>
BIRCH	<i>Balanced Iterative Reducing and Clustering using Hierarchies</i>
ADWIN	<i>Adaptative Windowing</i>
MSE	<i>Mean Squared Error</i>
BPIC	<i>Business Process Intelligence Challenge</i>

List of symbols

$\mathcal{L}$	Event log
$\mathcal{C}$	Case set
$\mathcal{E}$	Event set
$\in$	Is member of
$\mathcal{A}$	Attributes set
σ	Trace
$\subseteq$	Is subset of
$\cap$	Set intersection
$\emptyset$	Empty set
$\mathcal{S}$	Trace stream
$\mathcal{P}$	Business process
$\mathbb{R}$	Real number space
$\mathcal{C}$	Cluster
ζ	Clustering (set of clusters)

Contents

1	Introduction	19
1.1	<i>Problem definition</i>	21
1.2	<i>Hypothesis</i>	23
1.3	<i>Justification</i>	23
1.4	<i>Objectives</i>	24
1.5	<i>Methods</i>	25
1.6	<i>Document organization</i>	26
2	Background	27
2.1	<i>Process mining</i>	27
2.2	<i>Concept drift</i>	29
2.3	<i>Concept drift in process mining</i>	31
2.4	<i>Clustering</i>	35
2.4.1	The <i>k-means</i> algorithm	38
2.4.2	Streaming clustering	39
2.4.3	<i>CluStream</i> algorithm	41
2.4.4	Clustering in concept drift	42
2.5	<i>Trace clustering</i>	43
2.5.1	Trace clustering categories	44
2.5.2	Trace representation in vector spaces	45
3	Drift detection and localization	50
3.1	<i>Related work</i>	51
3.1.1	Drift detection	51
3.1.2	Drift localization	54
3.1.3	A comparative analysis	56
3.2	<i>Proposed method</i>	60
3.2.1	Vector representation	62
3.2.2	Clustering	63
3.2.3	Tracking clustering evolution	64
3.2.4	Drift detection	68

3.2.5	Drift localization	70
3.3	<i>Summary</i>	73
4	Experiments	74
4.1	<i>Synthetic event logs</i>	75
4.1.1	Loan Applications	76
4.1.2	High trace variability process	104
4.2	<i>Real-world event logs</i>	112
4.2.1	BPIC 2015	113
4.2.2	BPIC 2020	119
4.3	<i>Summary</i>	124
5	Final considerations	125
5.1	<i>Limits</i>	125
5.2	<i>Contributions</i>	125
5.3	<i>Future work</i>	126
	BIBLIOGRAPHY	127

1 Introduction

Traditional methods of data analysis are prepared for application in stationary environments, in which underlying concepts do not change over time (JUNIOR. *et al.*, 2014). Flexibility and adaptability have been studied in the business process management field, since organizations are frequently embedded in contexts that require a growing need to adapt to uncommon scenarios, such as recent clients' demands, competition, economy, or legislation (BOSE *et al.*, 2011). Being able to notice and understand changes as soon as possible can be a competitive advantage for organizations, specially unwanted divergences that demand correction or new improvements to be spread for all personnel involved.

Such scenarios influence the dynamics of operations within organizations and change the processes underlying them. This problem is known as concept drift and it consists of changes in the underlying distribution of data alter, which the target concept in a learning situation, and it is common in data stream scenarios (ŽLIOBAITÉ; PECHENIZKIY; GAMA, 2016). The data originating from the event logs of the processes within organizations is subject to concept drift. Therefore, process mining tools that are based on the analysis event logs need to deal with changes related to its dynamic environments since they are subject to the concept drift problem and most existing techniques in the field assume business processes to be steady (SEELIGER; NOLLE; MÜHLHÄUSER, 2017).

The starting point of process mining is the concepts of events, logs and traces. According to Aalst (2014), an event log is a collection of several events pertaining to instances of a single process. Each instance is named a *case* and describe the course of a particular trace in terms of executed activities, which are steps of the given process. Events register the execution of a particular activity, correspond to a single case, and may carry additional information such as time, resource, or cost. These attributes are useful when analyzing the process from unique perspectives.

Dealing with concept drift provides better conditions for analysis in all types of process mining. Contributing to the understanding of the process evolution in an online setting enables the organization to react quickly to correct, adapt, or improve a process when needed. The enhancement for process analysis come from the ability of splitting the event log according to the actual concepts and process behavior they hold. For example, in

the process discovery task (the primary type of process mining), not segregating concepts would occasionally lead to inaccurate interpretations and *spaghetti* process models (AALST, 2011).

The key challenges in the context of handling concept drift in process mining are: (i) detection, to detect at what moment the concept drift has occurred; (ii) localization and characterization, to identify the elements involved in the change of the process; and (iii) explanation, to explain the process evolution and the reason of the drifts (BOSE *et al.*, 2011). Figure 1 shows two process models to help understand these challenges. The model had an XOR gateway indicating that only activity B or C is executed in a process instance and, after instant t_c , all activities occur sequentially. The drift detection (i) should be capable of noticing it occurred at instant t_c ; the localization and characterization (ii) should point out that the drift is related to the activities B and C in a sequentialization of activities, i.e., a change in the control-flow perspective; a hypothetical explanation (iii) for this process evolution could be a reduction in the need for decision making.

Figure 1 – Example of *concept drift* in a process model

Source: Rafael Gaspar de Sousa, 2021

Several approaches can deal with concept drift in online settings, but they are primarily focused on the detection task, which, alone, may not provide enough information to clarify or support the organization's act of correcting, adapting or improving the process (CARMONA; GAVALDÁ, 2012; MAARADJI *et al.*, 2015b; ZHENG; WEN; WANG, 2017). Drift localization and characterization methods usually require an additional graph-based structure of the process, other than the representation used in the detection task (ARMAS-CERVANTES *et al.*, 2016; OSTOVAR, 2019).

Since most existing methods for online drift analysis rely on statistical tests over trace distributions, it is reported they are not able to perform well in processes with high variability, i.e., a process that exhibits a large proportion of distinct traces over the total number of cases in an event log (OSTOVAR, 2019). In this scenario, there might not be

enough data samples of specific behavior, compromising the reliability of statistical tests (OSTOVAR, 2019).

As an alternative to overcome these limitations, we propose an approach for dealing with concept drift based on trace clustering. Clustering is the main machine learning technique applied in the process mining context (MAITA *et al.*, 2017). In the usage of this technique, traces are encoded into a feature vector representation based on a specific profile of the process (e.g. activities, resources, transitions, etc.). Trace clustering is commonly used as a preprocessing tool to find patterns and split the event logs into groups of similar process behavior (BOSE; AALST, 2009). This is specially helpful when dealing with more complex and less structured processes, as the ones encountered in logs with high trace variability.

The problem pointed out in this project is how to perform concept drift detection and localization tasks effectively in an online setting, while dealing with the high trace variability problem and providing an accurate and informative description of the drift based on the same process behavior representation.

1.1 Problem definition

Formally, consider an event log $\mathcal{L}$ composed by a set of n cases $\mathcal{L} = \{c_1, c_2, \dots, c_n\}$ and a set of all cases $\mathcal{C}$, such that $c \in \mathcal{C}$. Also, consider a set of all events $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$. A case c is composed of a sequence of events e such that each one is related to specifically one case. Each event e and case c are associated with a set of attributes $\mathcal{A}$. For each attribute $a \in \mathcal{A}$, the function $\#_a(e)$ denotes the value of attribute a for the event e . Each case c has a special mandatory attribute *trace* (AALST, 2016). In the context of this project, the following attributes are considered $\mathcal{A} = \{id, trace, timestamp, activity\}$, where $\#_{id}(c)$ identifies the case, $\#_{trace}(c)$ contains the sequence of events for each case, $\#_{timestamp}(e)$ the current time when the event occurred and $\#_{activity}(e)$ denotes the activity associated with event.

Adapting the notations from Bose (2012), each trace σ corresponds to a ordered set of events e that belongs to a specific case $\sigma = \#_{trace}(c)$. Let $\mathcal{E}_\sigma$ be the sequence of events e that are part of trace σ such that $\mathcal{E}_\sigma \subseteq \mathcal{E}$. A trace σ represents a non-empty finite sequence of events such that for $1 \leq i < j \leq |\sigma|$, the inequality $\sigma(i) \neq \sigma(j)$ is

satisfied, indicating that no two events inside a trace are the same. Events inside a trace should respect the ascending temporal order, such that for $1 \leq i < j \leq |\sigma|$, the inequality $\#_{timestamp}(\sigma(i)) \leq \#_{timestamp}(\sigma(j))$ is satisfied. Each event should occur only once inside an event log $\mathcal{L}$ such that for any two traces σ_1, σ_2 where $\sigma_1 \neq \sigma_2$, the equation $\mathcal{E}_{\sigma_1} \cap \mathcal{E}_{\sigma_2} = \emptyset$ is satisfied.

The event log $\mathcal{L}$ is constantly updated during the execution of a business process. Since this project is set to an online environment, the data is processed through a stream of traces corresponding to set of completed cases, that already have their last event registered in $\mathcal{L}$. Based on [Silva et al. \(2014\)](#), a stream $\mathcal{S}$ is a potentially infinite sequence of traces $\{\sigma_1, \sigma_2, \dots, \sigma_n\}$ such that $\mathcal{S} = \{\sigma_i\}_{i=1}^N$ where $N \rightarrow \infty$.

Adapting the definition from [Gama et al. \(2014\)](#), a concept drift occurs in a business process between instant t_i and t_{i+1} when

$$\exists \sigma : p_{t_i}(\sigma, \mathcal{P}) \neq p_{t_{i+1}}(\sigma, \mathcal{P}), \quad (1)$$

where $p_{t_i}(\sigma, \mathcal{P})$ denotes the joint distribution at instant t_i between the trace σ and the business process $\mathcal{P}$. The underlying assumption is that the business process $\mathcal{P}$ behavior reflected in the distribution that guides the stream $\mathcal{S}$ is non-stationary. When there is a change in $\mathcal{P}$, posterior elements in the stream $\mathcal{S}$ are drawn from a new distribution, thus manifesting the concept drift through changes in the joint distribution function between σ and $\mathcal{P}$.

Given the definitions established for the concept drift concept, on the drift detection task, consider:

- The goal of drift detection is to identify the moment in time they occur. Given an event log $\mathcal{L}$, An automated method must determine all instants t_d such that the inequality in 1 is satisfied.
- In the online setting, such detections are performed on the fly, as new traces are fed to the trace stream $\mathcal{S}$ when the last event of a case is logged in $\mathcal{L}$

On the drift localization task, consider:

- With the instant of a drift detection t_d and the set of possible traces completed before and after t_d , the localization goal is to identify the entities involved in the

changes in the business process before and after the current detected drift for each of them.

- Drift localizations are performed on the fly, after the detection task, as the instants t_d are detected.

1.2 Hypothesis

The solution proposed on this work relies on the usage of trace clustering to address the challenges of dealing with concept drift in complex and highly variable situations. Our main hypothesis is that the ability to group and abstract the information on an event log provided by trace clustering, capable of isolating similar behaviors presented in the process model, is useful and insightful when dealing with concept drift. We also stand that such ability also enables the performance of both the drift detection and localization tasks using the same data structure.

Since traces with similar behaviors are grouped together by trace clustering, concept drift analysis can be done specifically for each cluster of traces. In the context of complex problems, clustering makes it possible to identify drifts which would likely be ignored otherwise due to process complexity. Our hypothesis is that trace clustering performs well under high trace variability circumstances in complex business processes.

1.3 Justification

Most process mining techniques are suited for applications that assume business processes are non-stationary, i.e., when there is no change in the process and no concept drift. However, businesses processes are frequently subject to changes that come from the environment and business context they are inserted, hence the need to adequately deal with unexpected changes in the business process in real-time. For instance, when discovering process models from event logs containing concept drift, the process model might not correspond correctly to the current reality of the business process, since it may capture behaviors that are no longer part of the process.

Clustering is one of the data mining tools most commonly applied in process mining (MAITA *et al.*, 2017; GARCIA *et al.*, 2019a). It has been successfully used to

enhance process discovery (GRECO *et al.*, 2006; SONG; GÜNTHER; AALST, 2008; BOSE; AALST, 2009; WEERDT *et al.*, 2013; SEELIGER; NOLLE; MÜHLHÄUSER, 2018). However, it has not been thoroughly explored in the context of dealing with concept drift. Thus, considering the benefits that could facilitate detection and localization of concept drift; consequently assisting the three main types of process mining - discovery, conformance, and enhancement.

The main challenge in trace clustering is related to the definition of a representative feature space that can efficiently map traces into vectors ready for general clustering algorithms. The trace representation schemes can be limited and not provide the complete information to characterize certain types of process behavior. Also, each one of them focuses on a particular perspective of the trace, such as the activities, time, or resources involved.

In the field of general data mining, most concept drift work has been focused on supervised learning settings, in which there is immediate availability of labels, leading to approaches based on tracking the performance of a supervised model (IWASHITA; PAPA, 2019). For the unsupervised scenario, evolutionary clustering approaches have been successfully applied to learn adaptively and represent the current concept in vector spaces where drifts in incoming data can be detected and models can be adapted based on changes in the clusters (SETHI; KANTARDZIC, 2017; GARCIA *et al.*, 2019b).

Specifically, work in concept drift in the process mining based on trace clustering strategies are presented in Luengo and Sepúlveda (2012), Accorsi and Stocker (2012) and Hompes *et al.* (2015), but are focused on offline applications with business context-related data, instead of pragmatic information about process control flow. In an online scenario, Junior *et al.* (2018) have shown interesting results on using a streaming trace clustering approach to support dealing with concept drift. Another approaches, such as Zheng, Wen and Wang (2017), Tavares *et al.* (2019) and Yeshchenko *et al.* (2021) also employ clustering techniques to their methods, but in an indirect way not focused on trace clustering.

1.4 Objectives

The main goal of this project is to implement an online method to perform concept drift detection and localization by using trace clustering. The approach must be suited to

several trace representations, focused on control-flow changes in the process, and able to perform both the detection and localization tasks in an integrated way, without requiring additional data structures or process models. To validate our approach, we measured the accuracy in each task, in synthetic and real-world event logs with high traces variability and with a variety set of process changes.

1.5 Methods

This research project has an application nature, focusing on developing solutions to improve problems in process mining by implementing methods to deal with concept drift. As the objective of this project is to propose a new approach and compare it with related work through tests accepted by the literature, from the methodological viewpoint, this project can be classified as empirical research, as defined in [Wazlawick \(2017\)](#).

The main procedure of the study in this project is the computational experiment. Also, procedures of exploratory review of the literature are part of the development of this project. An exploratory review has been done to explore the fundamental ideas on dealing with concept drift, process mining, trace clustering, and online clustering.

In terms of experiments, this study aims to develop, discuss, and analyze results in:

1. Perform and evaluate different online trace clustering strategies (batch with offline algorithm and streaming clustering algorithms) in event logs considering multiple vector space representations for traces;
2. Track and extract features from the evolution of trace clustering through time;
3. Detection of concept drift in synthetic event logs based on the products of item [2](#);
4. Localization of concept drift in synthetic event logs based on the products of item [2](#) and [3](#);
5. Evaluate the detection and localization results compared to the results of our framework when not performing trace clustering.

The proposed work can be understood as a pipeline containing a set of steps. Figure [2](#) shows an overview of the proposed work as a pipeline, detailing the scope of the techniques utilized in each step of this research. The windowing step is only required when using an offline clustering algorithm, as online clustering algorithms are fed with new data and automatically deal with them discarding the need of windowing.

Figure 2 – An overview of the proposed work, presenting steps and a set of items involved in each one. Highlighted items are part of first experiments described further.

Source: Rafael Gaspar de Sousa, 2021

For the concept drift detection task, the results were evaluated as a classification problem, through F_β score measures, to capture the trade-off between precision and recall. Drift localization results were ranked and then evaluated through the Area Under the ROC Curve (AUC) to measure how well our method can localize entities related to the process drift. For the real-world event logs without ground truth available, the results were analyzed according to the process model context.

Finally, the products of this research project are properly published in terms of scientific dissemination; and source codes, documentation, and intermediate results are available¹ to promote research reproducibility.

1.6 Document organization

Chapter 2 presents concepts and definitions related to concept drift, process mining and trace clustering. Next, chapter 3 presents an overview of the related work and describe the details of our drift detection and localization approach. Then, chapter 4 presents the setup, results and discussion of the experiments carried out in this study. Finally, chapter 5 presents the limits and contributions of this research project.

¹ Source code: https://github.com/rafaelgasp/trace-clustering_process-drift-detection

2 Background

This chapter provides background information on the main topics related to this research, presenting basic definitions, examples of applications, and their relationship to the approach presented in this work. Section 2.1 provides information on process mining and, in section 2.2 we present the problem of concept drift. In the intersection of both, section 2.3 details the concept drift problem in the process mining context. Then, section 2.4 introduces the task of clustering and some of the main algorithms in the literature. Finally, section 2.5 shows clustering approaches in the process mining context.

2.1 Process mining

Organization's processes are the core part of their activities but are frequently not well documented and, when they are, they might not behave in conformance with what is expected. A process is a collection of events, activities, and decisions that, when coordinately executed, reach a business goal (DUMAS *et al.*, 2013). Information systems have become the support platform for the organization's operational processes, logging every step and activity that occurred in the process with details, such as ERP¹ and CRM². Real life processes are subject to human management, decisions, and acting, making it very important that they are completely understandable for the organization's success. In this context, the process mining research area emerges, aiming to enhance the comprehension of process models through extracting value from recorded event logs and turn it into insights and actions (AALST, 2011).

A process model is a description of a process that happens in the real world, which involves people, businesses, resources, and organizations themselves. These processes are supported by software systems that record the events and steps of the process in the form of event logs. According to Aalst (2016) there are three main types of process mining: process discovery, that aims to obtain a process model that adequately matches what is recorded in the event logs that reflects the true process behavior; process conformance, that can, given a process model, check how the event logs reflect the process described in the model; and process enhancement that focuses on improving the process by modifying

¹ Enterprise resource planning

² Customer relationship management

or extending it based on the process models and event logs by analyzing both the process models and event logs. Figure 3 shows an overview of process mining and describes the relationship between the main entities related to it.

Figure 3 – Positioning of the three main types of process mining: discovery, conformance and enhancement.

Source: Aalst (2016)

In process mining, a common notation is to represent process models with Business Process Model Notation (BPMN) (DUMAS *et al.*, 2013). Figure 4 shows an example of process model with that notation. The process starts with activity A and ends at G. The B and C activities are part of an XOR gateway, represented by the symbol X , indicating that only one of them is executed for each case. The E and F activities are part of a AND gateway, represented by $+$, meaning that both activities occur in parallel.

Figure 4 – Example of a process model represented with BPMN

Source: Rafael Gaspar de Sousa, 2021

According to Aalst (2016), a process consists of cases, which are a sequence of ordered events. An event can have several attributes often describing the conditions of execution of a specific activity. Every case is associated with a trace, which is a non-empty finite sequence of unique events. The event logs are composed of a list of activities executions and the identification of the trace they belong to. Table 1a has an example of event log from the process model in figure 4, with the Trace ID, Activity and Timestamp

columns. Table 1b presents the list of possible traces and their respective activities for the example process model.

Table 1 – Event log and a list of possible traces from the model in figure 4

(a) Example of event log

Trace ID	Activity	Timestamp
T_1	A	2020-01-01 00:00
T_2	A	2020-01-01 00:05
T_1	B	2020-01-01 01:12
T_1	D	2020-01-01 01:35
T_2	C	2020-01-01 02:40
T_2	D	2020-01-01 03:45
T_3	A	2020-01-01 12:30

(b) List of possible traces

Trace ID	Activities
T_1	A, B, D, E, F, G
T_2	A, C, D, E, F, G
T_3	A, B, D, F, E, G
T_4	A, C, D, F, E, G

Source: Rafael Gaspar de Sousa, 2021

2.2 Concept drift

Traditional techniques in the data mining field generally target on learning concepts in static environments. In dynamic real world environments, those concepts are subject to changes over time, leading the problem known as concept drift (WANG; SCHLOBACH; KLEIN, 2010). As a result, historical data utilized for learning may suggest a relationship for input and target concept that has changed by some hidden context not explicit in data (WIDMER; KUBAT, 1996).

Dealing with concept drift has become more important along with the growth in the use of data in a streaming format rather than only static datasets (ŽLIOBAITĖ; PECHENIZKIY; GAMA, 2016). For example, cases when the problem consists of learning the behavior of an adversarial, such as fraud detection, they are also aware that they are being studied and keep constantly trying to change their behavior. Another example would be to learn users' preferences for recommendation systems, which is subject to change and are generally due to hidden context, i.e. not observable in the data available.

Concept drifts can manifest itself in data through different structures, as shown in figure 5 with a one dimension data example. a) sudden drifts consists of an abrupt change from one concept to another; b) incremental drift is when the drift goes through many intermediate concepts in between; c) gradual drifts occur when two concepts keep alternating back and forth gradually until stability of the new one; d) recurring drifts are structured as repeated sudden drifts alternating between concepts; e) presents an important distinction to not mix a concept drift structure with an outlier or random noise in the data.

Figure 5 – Different structures of concept drift

Source: Adapted from [Gama et al. \(2014\)](#)

In the literature, dealing with concept drift focus on providing methods of adaptative learning that can learn from data in dynamic environments. According to the survey in [Iwashita and Papa \(2019\)](#), most approaches are meant for supervised learning scenario and rely on instance selection method that only considers more recent data, inside a time window, for learning, disconsidering older data ([TSYMBAL, 2004](#)). This kind of approach does not detect or characterize the drift itself but are primarily focused on adaptative learning and avoid loss of performance of supervised models over time.

Another line of work is focused on developing approaches to detect the occurrence of concept drift in data streams. That detection can be applied in various forms, for example inside other adaptative learning models structures, investigation on what caused the drift, characterizing the changes, or for data analysis in general. The survey on [Junior. et al. \(2014\)](#) presents a study on the comparison of concept drift detectors. Most detectors are based on window strategies, analyzing the incoming data mean for more recent windows or the performance in a supervised learning situation.

On detailing and analyzing characteristics of the drifts, the work in [Webb et al. \(2016\)](#) provides a framework through some quantitative proprieties that can describe and compare drifts. One of them is the drift magnitude which is a measurement of how different is the new active concept than the previous one. Also, the type of drift as shown in figure

5 and the characterization of the drift subject, which depends on the context. Tests on generated examples of concept drift data with different characteristics have shown that the performance of the detection methods varies depending on those characteristics.

2.3 Concept drift in process mining

Most process mining techniques, as in the general data mining field, assume that processes are stationary and stay the same during all event log. [Bose et al. \(2014\)](#) have analyzed processes in more than 100 organizations and state that processes are commonly altered over time, hence the rise of studies on concept drift in the process mining context. In process mining, concept drift is defined based on the change of behavior of the process itself, meaning that there is a point in time where a statistically significant difference can be found in the process behavior before and after it ([MAARADJI et al., 2015b](#)).

In business processes, the main perspectives subject to concept drift: control-flow, data, resources, and performance ([BOSE et al., 2014](#); [APPICE](#); [MALERBA, 2016](#)).

- **Control-flow perspective** holds changes in the structure of the process through changing activities or conditional/concurrent branches. There are special cases in this category when the process structure remains intact, but the frequency aspect of the process behavioral changes, for example when a rare behavior becomes frequent. A list of common patterns of control-flow changes is presented in [Weber, Reichert and Rinderle-Ma \(2008\)](#).
- **Resource perspective** is related to changes in resources, their roles, and the structure of the organization that may affect the process. For example, change in personnel or roles inside an organization. Since each resource tends to work in a specific manner, drifts can occur when changing resources because it reflects on changes in the process behavior.
- **Data perspective** refers to changes in the requirement and generation of data during the process. For example to not require a specific document as a step in a process.
- **Performance perspective** addresses changes reflected on timestamps of traces, altering the performance in terms of duration of the process, for example when an activity takes longer to complete because of changes in its execution.

The main tasks in the study of concept drift in process mining are: detection, localization and characterization, and change process discovery.

Detection

Concept drift detection has been the main task studied in the process mining context, and it has been approached both in an online scenario, by processing a stream of traces as they come, and in an offline situation, by analyzing a fixed-size event log. Our approach is focused on the online scenario.

Detecting a process drift consists of identifying a moment in time that separates two statistically significant different behaviors in the process. For this task, it is required that we have ways to measure and describe the behavior of the process model to monitor its variations over time. Then, we can set up the identification of changes and the decision of whether they are significant enough.

In this setting, some challenges are the subject of research in this field. How to effectively compare two sets of traces? How to decide whether this difference is related to some outlier, noise, or an actual concept drift? Each of those questions brings up different trade-off situations and are discussed in the literature.

To address the first challenge, there are approaches based on the similarity that use feature extraction to represent traces in a vector space and make them comparable through similarity metrics ([BOSE; AALST, 2009](#); [ACCORSI; STOCKER, 2012](#); [HOMPES *et al.*, 2015](#)). This kind of approach is a powerful tool to support analysis and provide insights, however, it is not, alone, an autonomous technique to detect drifts. To provide that ability and address the second challenge, there are good results in using statistical tests, along with windowing methods, that compare subsequent windows of traces and can automatically detect process drifts ([CARMONA; GAVALDÁ, 2012](#); [KUMAR; THOMAS; BASAVA, 2015](#); [MAARADJI *et al.*, 2015b](#); [OSTOVAR, 2019](#)). Further discussion on this topic are presented in chapter 3.

This problem can be understood as a binary classification problem from supervised learning as, at every iteration, whether a concept drift has occurred or not. Therefore, for evaluation of the performance of the detected methods, we can consider the usage of such

metrics for classification scenarios as presented in [Tharwat \(2018\)](#). For this project, the validation of detection relies on the F_β score composed of the precision and recall.

Precision indicates the ratio between the number of true concept drifts detected over the total number of drifts detected. Higher Precision implies that lower false positive cases will be detected by the method. The Recall measures the ratio of concept drifts correctly detected over the total number of existing drifts. Higher values for Recall indicates that most of the existing drifts are detected. The F_β score is a weighted mean between both Precision and Recall where the weight attached to recall is β times higher the weight of Precision. In this work, we used this metric with $\beta = 1$ for weighing both terms equally. This metric is also used to evaluate drift detection performance in related work ([MAARADJI *et al.*, 2015b](#)).

Localization and Characterization

Once a drift has been detected, the next step in dealing with concept drift consists of being able to identify and describe what has changed. This task comprehends the identification of perspective (control-flow, data, resource or performance), the structure of the drift (sudden, gradual, incremental, etc.) and the change in the process itself (for example the removal of a specific activity or insertion of a conditional branch between two activities) ([BOSE *et al.*, 2014](#)).

Approaches that deal with this challenge are usually based on a comparison of sub-logs extracted from before and after the detected drift, identifying the differences. Though not focused on dealing with concept drift, there are frameworks in the literature to compare two distinct processes based on abstractions of event logs ([BEEST *et al.*, 2015](#); [ARMAS-CERVANTES *et al.*, 2016](#); [BOLT](#); [LEONI](#); [AALST, 2018](#)). There is also work focused directly on concept drift characterization based on the same idea ([OSTOVAR *et al.*, 2017](#); [SEELIGER](#); [NOLLE](#); [MÜHLHÄUSER, 2017](#); [OSTOVAR](#); [LEEMANS](#); [ROSA, 2020](#)).

The approach proposed in this project fits the localization and characterization task defined in [Bose *et al.* \(2014\)](#), though we specifically focus on identifying the activities involved in the process drift, as further discussed in chapter 3. While there is not a

systematic definition of the exact information to be localized and characterized after the process drift, some similar approaches might have different targets on this task.

To evaluate the results for the localization task, we want to analyze whether our approach can identify correctly the entities involved and the ones not involved in the drift. Based on a binary classification approach, we are going to consider the area under the ROC (Receiver Operator Characteristic) curve as an evaluation metric for this task. The ROC curve can be obtained by plotting the True Positive Rate (TPR) against the False Positive Rate (FRP), for different thresholds. For our proposes as a numerical index, the Area Under the Curve (AUC) is used to describe the ROC curve behavior ([CALI; LONGOBARDI, 2015](#)). The AUC score ranges from 0 to 1, where 1 indicates the best result and 0 the opposite.

Process evolution

After detecting, locating, and characterizing the concept drift, this task aims to put all this information in perspective according to the business process context ([BOSE *et al.*, 2014](#)). It includes the discovery of the new process over time and its characteristics, such as monitoring statistics and performance levels related to business context.

[Li *et al.* \(2017\)](#) presented a work that combines several techniques from process mining analysis in order to unravel the process evolution. A great example of an application for such an approach is presented in [Kurniati *et al.* \(2020\)](#), applied to the specific context of healthcare event logs. Through analyzing several metrics (such as frequency and duration of activities, model precision, and fitness) over time, combined with concept drift detection and process discovery techniques, they were able to provide a useful understanding of the process over time. One concern raised when handling process drifts was that not all patients were being submitted to a Multidisciplinary Team Review (MDT Review), which is a required activity in their process. Later, domain experts have confirmed that the absence of the given activity in the logs was related to the fact that it was usually done automatically during a medical appointment and not logged properly. Other discovered changes were associated with modifications and updates in the software employed by the organization.

The dimensions of study of concept drift have been summarized in [Bose et al. \(2014\)](#) and are shown in figure 6. Highlighted areas represent the coverage of this work.

Figure 6 – Dimensions of concept drift in process mining. Highlighted area indicates are covered by this project.

Source: Adapted from [Bose et al. \(2014\)](#)

2.4 Clustering

Clustering is an unsupervised learning task in data mining that consists of grouping data points according to their similarity. The goal is to obtain groups composed of points similar to each other but, different from points in other groups. In this context, the similarity is defined through distance metrics. Each group is named cluster and a set of clusters obtained after the analysis is named clustering ([HAN; KAMBER; PEI, 2012](#)).

This task has been applied in various domain fields such as biology, business intelligence, image recognition, etc. For example, it is useful in business to be able to better know your clients, a possible approach is to apply clustering techniques to obtain groups of clients that share similar preferences and be able to satisfy their needs more specifically. Clustering is also applied for outlier detection, where outliers lie distant from any cluster since it is able to understand patterns and synthesize data into clusters that represent the overall distribution of data points.

Formally, the clustering algorithms take as input a data matrix $X \in \mathbb{R}^{n \times m}$, with n lines and m columns able to represent data from certain domain. This matrix is composed from a set of vectors $V = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$. Each vector is one of the n data points in a vector space of m dimensions. The main goal of clustering is to find k parts of V , denoted by ordered subsets $\mathcal{C}_i \subseteq V$, where $i \in 1, \dots, k$. Then, the obtained set of clusters

$\zeta = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k\}$ is composed by the obtained groups of vectors from V resulting from the algorithm.

As a result, the clustering algorithm intends to obtain clusters made of points more similar to each other inside their cluster than they are to data points belonging to different other clusters (JAIN; MURTY; FLYNN, 1999). Figure 7 presents an illustrative example of clustering in a two dimensions dataset. In a) the data points are displayed in a scatter plot and in b) represents a result of clustering by assigning a cluster for each data point. Each of the six clusters $\mathcal{C}_c$ is identified by a different color.

Figure 7 – Example of clustering data points. Each color identifies a cluster found.

Source: Adapted from Jain, Murty and Flynn (1999)

The main fundamental clustering algorithms were divided in Han, Kamber and Pei (2012) in the following categories: partitioning, hierarchical or density-based methods.

- **Partitioning** methods are the most intuitive and common approach, that aims to partition data into k clusters. Also, each element is assigned to one cluster only. Most partitioning methods are based on a distance metric, usually Euclidean distance. The clusters in this kind of approach are represented through the centroid, which is defined as the mean of all data points belonging to that cluster. The *k-means* algorithm is an example from this category, and it works by, given k , creating a random initial partitioning and iteratively relocate elements between groups in order to improve the quality of partitioning based on Euclidean distance.
- **Hierarchical** methods create a hierarchic structure to the data points. This hierarchy can be obtained in an agglomerative way that starts with each data point being a cluster and then merging similar points to each other until there is only one group remaining. There is also the divisive approach that works in the opposite way. The similarity for algorithms in this group is also usually distance-based. The hierarchies can be visualized through a tree structure diagram (dendrogram).

- **Density-based** methods are based on the idea that clusters are formed by dense regions of data points separated by low-density regions. This approach can detect clusters of any arbitrary shape. DBSCAN is the main example of an algorithm in this group. It works by growing clusters based on data points neighborhood. Two points are neighbors whether they are within a given radius of each other. When growing, there must be a minimum number of points within a neighborhood, otherwise, a dense region is not formed and it considers that point as an outlier.

As shown in figure 7, clusters may have different shapes, densities, or sizes. As the clustering algorithms comprehend an array of distinct implicit assumptions about cluster shapes, similarity measures, or grouping criteria, there is not a clustering technique universally applicable in a mixture of structures present in multidimensional data sets (JAIN; MURTY; FLYNN, 1999). A comparison of the performances of the main algorithms is presented in Rodriguez *et al.* (2019). Additionally, the fundamental clustering algorithms have been extended in numerous ways in order to achieve better performance in different scenarios, such as dealing with fuzzy clustering, a large amount of data, high dimensionality, or streaming situations. Further information on these topics are covered in Han, Kamber and Pei (2012), Xu and Tian (2015).

When evaluating the validity of the clustering algorithm, there are internal indicators that intend to measure how good are the obtained clusters in terms of having similar points inside a cluster that is different from points in other groups. Additionally, there are external indicators based on external data and their analysis between the different clusters, for example when there are labels available the clustering groups can be evaluated with metrics for supervised learning (XU; TIAN, 2015). Since this project is based on an unsupervised approach, the following internal validating index were applied: Silhouette (ROUSSEEUW, 1987) and Davies-Bouldin (DAVIES; BOULDIN, 1979).

The Davies-Bouldin index is defined as the average similarity of each cluster with its most similar cluster and can be calculated through equation . This similarity is defined by the ratio of average intra-cluster distances to inter-cluster distances. The minimum score is zero, as lower values indicate better clustering. The intention is that clusters more

compact and distant to others and more results in a lower score (DAVIES; BOULDIN, 1979).

$$\begin{aligned} \mathcal{C}_j\text{-avg_intra} &= \frac{1}{n} \left(\sum_{i=1}^n d(\vec{x}_i, c_j) \right) \forall \vec{x} \in \mathcal{C}_j \\ \text{Davies-Bouldin} &= \frac{1}{k} \left(\sum_{i=1}^k \frac{\mathcal{C}_i\text{-avg_intra} + \mathcal{C}_j\text{-avg_intra}}{d(c_i, c_j)} \right) \end{aligned} \quad (2)$$

The Silhouette index can be calculated using equation . The score is calculated to each data point $x_i \in \mathcal{C}_j$ and it is represented as $s(\vec{x}_i)$. It is based on the average intra-cluster distance (a) and the average distance to the closest cluster (b). Then, it is possible to obtain the average Silhouette for each cluster ($s_avg(\mathcal{C}_j)$) and to calculate a general coefficient based on the maximum value between all clusters. The coefficient ranges between -1 and 1, where higher values indicate better clustering. Values near 0 indicate overlapping clusters. Negative values usually indicate that a point has been assigned to a wrong cluster, as a result a different cluster is more similar than the assigned one (ROUSSEUW, 1987).

$$\begin{aligned} a(\vec{x}_i) &= \frac{1}{|\mathcal{C}_j| - 1} \sum_{l \in \mathcal{C}_j, i \neq l} d(\vec{x}_i, \vec{x}_l) \\ b(\vec{x}_i) &= \min_{k \neq j} \frac{1}{|\mathcal{C}_k|} \sum_{l \in \mathcal{C}_k} d(\vec{x}_i, \vec{x}_l) \\ s(\vec{x}_i) &= \frac{b(\vec{x}_i) - a(\vec{x}_i)}{\max\{a(\vec{x}_i), b(\vec{x}_i)\}} \\ s_avg(\mathcal{C}_j) &= \frac{1}{n} \left(\sum_{i=1}^n s(\vec{x}_i) \right) \\ \text{Silhouette} &= \max_{i=1}^k s_avg(\mathcal{C}_i) \end{aligned} \quad (3)$$

2.4.1 The *k-means* algorithm

The *k-means* algorithm is one of the most simple and widely used clustering algorithm, with its nomenclature being first introduced in Macqueen (1967). As it will be seen later in chapter 3, our approach requires a descriptive representation of the clusters in the form of a vector that allow us to analyze clusters changes over time. An approach such as the *k-means* algorithm suits our necessities by providing centroids that represent

each cluster. Its goal is to partition a dataset composed of vectors with N dimensions into k subsets with efficiency in terms of within subsets variance, by minimizing the average squared distance between points in the same cluster. The key idea of the algorithm is: given an initial (but not necessarily optimal) set of k points as primary centroids, assign each point in the dataset to its centroids (this is done by using a distance metric between the two points, usually Euclidean), then update the centroids to become the mean of all points assigned to that cluster (hence the name “k-means”). This process is repeated several times until a convergence criteria (such as specific number of iterations) is satisfied.

Several variants were proposed on this basic algorithm focusing on improve its performance (e.g. different initialization methods) or usage in specific contexts (e.g. interactive or streaming clustering). In this study, we worked with the *k-means++* algorithm, proposed in [Arthur and Vassilvitskii \(2007\)](#), that provides an randomized seeding technique to generate better initial centroids. This works by selecting the first centroid randomly, then the following ones are selected in a way to increase the distance between the centroids from each other, resulting in improved clustering and with faster convergence.

2.4.2 Streaming clustering

The recent evolution of computational and storage capacities in information systems has allowed massive and continuous data acquisition. When dealing with an uninterrupted flow of data in applications such as monitoring data from sensors, video surveillance, stock market analysis, or interactions on social networks, it is of summary importance that the techniques employed can learn with data continuously. This requires the algorithm to deal with memory and time constraints, usually processing one data point at a time ([GAMA, 2010](#)). Formally, data streams $\mathcal{S}$ are unbounded sequences of data objects constantly generated at rapid rates that can be formally defined as a potentially infinite sequence of data points $\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$ such that $\mathcal{S} = \{\vec{x}_i\}_{i=1}^N$ where $N \rightarrow \infty$ ([GAMA et al., 2014](#)).

An approach to cluster data streams with traditional clustering algorithms and learn with new data points is to run clustering periodically while adding more recent data. As pointed in [Carnein and Trautmann \(2019\)](#), this is expensive in computational and storage terms. Therefore, the goal of streaming clustering algorithms is to update existing clusters and account new data points to the existing model by identifying changes

in clustering structure, such as rise, merge, or split of clusters. Additionally, processing data streams should be aware of eventual concept drift, since the data stream may hold multiple concepts over time (NGUYEN; WOON; NG, 2015).

According to Carnein and Trautmann (2019), many streaming clustering algorithms rely on a two-phase approach that has an online component that evaluates and summarizes the data points in real-time and stores a statistical synopsis of the stream. This results in a group of micro-clusters as a first draft of the final clusters. The final clustering results are obtained in an offline phase that clusters the micro-clusters into a final set of macro-clusters. A general summary of the stream is represented as micro-clusters, then providing more flexibility to clustering in the offline phase. The most common algorithms are based on this method, such as BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), CluStream (based on hierarchical clustering), and DenStream (based on DBSCAN). A minority of streaming clustering algorithms rely on incremental approaches, where the final macro-clusters are generated incrementally in real-time. The algorithms STREAM (based on *k-means*) and SWEM (Sliding Window Expectation-Maximization) are examples in this category.

To better adapt to the eventual concept drifts and enhance the clustering performance, the algorithms can employ time window models (NGUYEN; WOON; NG, 2015). They allow processing only a portion of the entire data stream and focus the algorithm's interest in a more recent portion of data. The different types of window models are shown in figure 8. The (a) *damped time window* model (also referred as *fading*) assigns a weight for each data point that decays over time. The (b) *sliding* window model considers, at each iteration, only a specific amount of the most recent data points, and works as a queue by removing the oldest point from the window when a new data point arrives. The (c) *landmark* window model focus on splitting the stream into disjunct slices based on a landmark (either a specific number of points or time has passed). Finally, the (d) *pyramidal* window model (also referred to as *titled time*) uses different granularity levels based on the time recency of data, considering more recent data in a finer scale than long-term data from past data points.

Extensive reviews gathered information on the different streaming clustering approaches available and comparative analysis are available in Nguyen, Woon and Ng (2015) and Carnein and Trautmann (2019) for further reference. In this project, we intend to

Figure 8 – Overview of time window models

Source: (CARNEIN; TRAUTMANN, 2019)

evaluate the usage of the CluStream algorithm, as well as the traditional offline clustering approach combined with a time window model to deal with the data streams.

2.4.3 *CluStream* algorithm

In this work, we chose the *CluStream* to experiment with as it is considered one of the reference algorithms when dealing with data streams due to its ability to consider different time horizons when clustering (PEREIRA; MELLO, 2011). It was first introduced in Aggarwal *et al.* (2003) and it aims to divide the clustering task in two phases: online that produces microclusters, and the offline phase to produce the macroclusters. In the offline step, only data obtained in the online stage is used. This method borrows many ideas from the BIRCH hierarchical clustering algorithm in Zhang, Ramakrishnan and Livny (1996), extending it to streaming capabilities.

In the streaming scenario, each data point can be processed once and, as the data is not stored, can not be reiterated on later by the algorithm. Hence the purpose of the online phase is to produce a summary of the data that it has seen so far. These microclusters are stored as tuple that contains the number of elements, linear and squared sums of the elements in the microcluster, as well as temporal information. The number of microcluster is user defined and larger than the number of k clusters desired at the end of the offline phase.

Whenever a new data point is processed after the microclusters have been initialized, either it is absorbed by an existing microcluster if it lies inside the microcluster boundary (which can be controlled by a user defined parameter); or it will require a new microcluster

to be created. When the number of existing microclusters exceeds the parameter, the algorithm attempts to delete the oldest microcluster if there is any considered to be old enough (according to a user defined parameter); or it merges the two closest microclusters into one.

Microclusters are periodically stored in what is called snapshots, which intend to capture information on the data in that specific moment. The snapshots capture follows a pyramidal time frame, that adopt a logarithmic scale, storing more information about recent data points than older ones. This snapshots are used as input to the offline phase, which clusters the microclusters into macroclusters, considering a particular time horizon and number of clusters k .

The limitations on this algorithm are similar to the ones in *k-means*: dependence on distance metrics that do not necessarily work well under high dimensionality data; and the inability of finding clusters of non-spherical shape (PEREIRA; MELLO, 2011). However, it is a reference algorithm for dealing with data streams, specially since the snapshots store the microclusters situation at several timestamps, which makes it possible to obtain clusterings in different time resolutions.

2.4.4 Clustering in concept drift

Unsupervised approaches for detecting concept drift in data streams have been proposed in the literature, although not in the business process domain. One category of such approaches includes the usage of clustering-based strategies, as presented in Sethi and Kantardzic (2017). The main idea is to analyze variations on distributions, patterns, or density information and detect novel concepts. For that, clustering and outlier detection are commonly used since the data stream is summarized into clusters and new data points can be compared in terms of similarity to what has been previously seen and incrementally update the model. The work is focused on classification problems that include supervised data.

On the clustering-based drift detection strategies, drifts can be detected when there is a need to create new clusters representing new data arriving in the stream, or when new data changes the preexisting clusters features, such as positioning or density (VALLIM *et al.*, 2014). Such methods combine the incremental learning representation of data from

streaming clustering algorithms with a novelty or outlier detection mechanism. Relying on clustering stability properties, these methods assume robustness to outliers, since an exceptional data point does not necessarily reflect a real behavior change and clustering would be altered significantly only when an effective behavior change occurs.

For example, the work in [Faria, Gama and Carvalho \(2013\)](#) applies the CluStream algorithm to cluster data points that are unknown to the supervised model. Then, samples are queued and new clusters are either incorporated into an existing class or assigned to a new class (novelty). A similar approach is also present in [Garcia *et al.* \(2019b\)](#) that compares new data points with the nearest micro-clusters to detect novelty and concept drift, and adapt the learning model. In [Sethi, Kantardzic and Hu \(2016\)](#) a grid-based clustering approach is applied and the variations on the density are analyzed to detect concept drifts. These approaches are focused on improving the performance of classification models over time.

2.5 Trace clustering

Clustering is the main machine learning technique applied in the process mining context ([MAITA *et al.*, 2017](#)). Commonly known as “trace clustering”, it aims to analyze the similarities of traces in a process, identifying patterns that can decompose different profiles of a process behavior ([BOSE; AALST, 2009](#)). This task is extremely helpful when working on a less structured process, which is common in healthcare service, for example, since they deal with specific conditions in each case and there are possible deflections from the procedures in order to deal with different circumstances. These kinds of processes are classified as spaghetti: there is a comprehension of the process flow and their activities are usually oriented by experience, intuition, heuristic rules, or vague information ([AALST, 2016](#)).

When decomposing the event log according to their similar profiles, the process is subdivided into understandable parts and process mining techniques can be applied more easily. Figure 9 shows an illustrative example of how trace clustering works by splitting a spaghetti process model through trace clustering and the obtained results after applying process discovery techniques into the complete event and to the three clusters separately.

Figure 9 – Examples of trace clustering

Source: Adapted from [Bose \(2012\)](#)

2.5.1 Trace clustering categories

The review in [Lu \(2018\)](#) states that trace clustering approaches can be divided into three categories: based on trace sequence similarity, model similarity, and feature vector similarity.

- *Trace sequence similarity* approaches are based on the trace syntax and order of activities, as a sequence. A trace can be edited in order to become equal to other traces, by adding or removing events. For similarity, distance metrics for sequences such as Levenshtein are applied in order to measure how many edit operations are needed to make two traces equal. With a matrix of similarity between the traces in an event log, general clustering algorithms are applied. [Bose and Aalst \(2009\)](#) is an example of this approach with Levenshtein distance and hierarchical clustering.
- *Model similarity* approaches are based not directly on trace, but on the quality of the process model discovered from the traces. A trace is more similar to a group of traces if a more fit, precise, and simple model can be obtained with this group ([LU,](#)

- 2018). [Hompeš *et al.* \(2015\)](#) and [Koninck, Weerdt and Broucke \(2016\)](#) are examples of work that employ this type of approach and apply it to find variant cases in traces.
- *Feature vector similarity* approaches are similar to general data mining clustering techniques, and it is the strategy chosen in this work. The traces are mapped to a vector space by extracting features from a specific profile of the process. For example, counting the presence or frequency of occurrence of activities in the trace. With this vector representation, several similarity measures can be used along with distance-based clustering algorithms. Example of approaches in this type are: [Song, Günther and Aalst \(2008\)](#) represents traces through bag of activities and *k-means* as clustering algorithm; [Appice and Malerba \(2016\)](#) uses co-training to combine different profile representation using control-flow, resource and timestamp information; and [Koninck, Broucke and Weerdt \(2018\)](#) proposes a trace embeddings representation based on the co-occurrence of activities.

Since the strategy based on feature vectors is the one adopted in this work, the different forms of representation are going to be discussed further at [2.5.2](#).

2.5.2 Trace representation in vector spaces

Event logs contain information about the traces from different points-of-view such as the name of activities or resources involved. In trace clustering, we need to establish what makes two traces similar, and that can be done in different ways. As defined in [Song, Günther and Aalst \(2008\)](#), a *profile* is a set of related metrics, that assigns to each trace a specific value, that is able to define a trace from a specific perspective. This process results in a vector with the metrics from a profile in a specific order and that can be used to measure trace similarity and feed them to clustering algorithms.

The main vector representations profiles are based on (i) the occurrence of activities, (ii) resources, (iii) transitions of activities, or (iv) statistics from trace performance ([APPICE; MALERBA, 2016](#)). Information on these perspectives is detailed for each event in the log, as an event log provides information for each event, identifying the activity, timestamp and the resource (in this case, the person that executed the activity) is provided in [table 2](#) as an example.

Table 2 – Example of event log with information of activity, resource and timestamp.

Trace ID	Activity	Timestamp	Resource
T_1	A	2020-01-01 00:00	Ana
T_1	B	2020-01-01 00:10	João
T_2	A	2020-01-01 02:00	Ana
T_1	D	2020-01-01 03:40	Caio
T_2	C	2020-01-01 03:59	João
T_2	D	2020-01-01 05:50	Ana

Source: Rafael Gaspar de Sousa, 2021

Vector representations for profiles based on occurrence are obtained by performing semantic models based on frequency: Binary, frequency or *tf-idf* (term frequency-inverse document frequency). In the binary counting model, only the presence or absence of activities/transitions in the trace is considered. The frequency counting model considers the frequency of activities/transitions and the *tf-idf* weights the term frequencies with the inverse frequency of each term in other traces of the event log. These techniques are common in the text mining and information retrieval research fields and can be explored in process mining. Another representation is the one based on “order”, that consists in, for each term in the trace, count the position that it first appeared in the trace, then normalize the values according to the length of the trace. These counting models can be applied to all profiles. In this work, we will explore the activity and transitions profiles.

Table 3 shows the representation of traces T_1 and T_2 from the event log at previous table for the activity, resource and transitions profiles. The activity profile (i) characterizes a trace solely by the activities that have occurred in them. Each activity present in the event log becomes a dimension in the vector representation, in this case, A , B , C , and D . Similarly, for the resource profile (ii), each resource is mapped to a dimension in the vector, in this case, Ana, João, and Caio. The transitions profile (iii) is based on the control-flow perspective and each dimension is defined as a pair of activities that occur subsequently inside each trace. For example, dimension $(A : B)$ represents activity A being followed by activity B .

The statistics from trace performance profile (iv) are based on the timestamp information from the events. The example on table 4 presents the following set of features: size (number of events in the trace); duration (difference in time from the last to the

Table 3 – Example of vector representations for profiles based on occurrence, with the counting model frequency

Activity					Resource			Transitions			
Trace ID	A	B	C	D	Ana	João	Caio	A:B	A:C	B:D	C:D
T_1	1	1	0	1	1	1	1	1	0	1	0
T_2	1	0	1	1	2	1	0	0	1	0	1

Source: Rafael Gaspar de Sousa, 2021

first event); the minimum, maximum, average and standard deviation of time difference calculated between consecutive events in the trace.

Table 4 – Example of vector representations for performance profile (in minutes).

Trace ID	Size	Duration	Minimum	Average	Maximum	Std. Deviation
T_1	3	220	10	110	210	141,42
T_2	3	230	111	115	119	5,65

Source: Rafael Gaspar de Sousa, 2021

A different approach, based on learned representation, is presented in [Koninck, Broucke and Weerd \(2018\)](#). The strategy, also referred to as embeddings, relies on learning highly informative vectors of low dimensionality through neural networks application. This representation has presented promising results in trace clustering.

The main issue of feature vector approaches for trace clustering is in their definition ([APPICE; MALERBA, 2016](#)). In this project, trace clustering based on feature similarity was performed online to deal with concept drift in the control-flow aspect. This requires that vector representation incorporates features that can distinguish the process behavior that has changed. Each profile and counting model has its limitations and that can be especially problematic when dealing with concept drift.

Figure 10 shows examples of control-flow changes to a baseline process model. The model in a) exchanges the order of activities *A* and *D*; model b) inserted a loop around activity *A*.

The activity profile does not consider information on the order of events when representing a trace since it only considers their frequency on the trace. Table 5 shows the representation in both activity and transitions profile for possible traces from the baseline and the altered model. As model a) is an example of process change in the order of activities, the activity profile is not able to distinguish between the *A, B, D*, and *D, B, A*

Figure 10 – Examples of control-flow changes in a baseline process.

Source: Rafael Gaspar de Sousa, 2021

traces. At the cost of requiring more dimensions, the Transitions representation is able to easily discriminate both behaviors.

Table 5 – Comparison between the trace representations activity-binary and transitions-binary for the baseline and “a)” process models

Process model	Trace	Activity-binary				Transitions-binary						
		A	B	C	D	A:B	A:C	B:D	C:D	D:B	D:C	B:A
Baseline	<A, B, D>	1	1	0	1	1	0	1	0	0	0	0
a)	<D, B, A>	1	1	0	1	0	0	0	0	1	0	1

Source: Rafael Gaspar de Sousa, 2021

Considering the same example, table 6 shows the comparison of the vectors using activity-binary and activity-order representations. Similar to the Transitions case, the change in order can be distinguished in the vector generated by the Order representation, as the value for features *A* and *D* changes as they switch positions in the trace, while the other activities remains the same.

Table 6 – Comparison between the trace representations activity-binary and activity-order for the baseline and “a)” process models

Process model	Trace	Activity-binary				Activity-order			
		A	B	C	D	A	B	C	D
Baseline	<A, B, D>	1	1	0	1	0.33	0.66	0.0	1.0
a)	<D, B, A>	1	1	0	1	1.0	0.66	0.0	0.33

Source: Rafael Gaspar de Sousa, 2021

Additionally, the binary counting model is not suited for dealing with loops, since it only considers whether the event has or not occurred. Table 7 shows the representation from the activity profile in both Binary and Frequency counting model for the baseline and the

altered model b). As the altered model suggests that activity A is repeatable, the activity-binary representation is not able to distinguish the trace A, B, D from A, A, A, B, D . In these cases, the Frequency counting model is able to effectively discriminate against both behaviors. Also, both the Transitions and Order representations can differentiate the vectors when loops occur.

Table 7 – Comparison between the trace representations activity-binary and activity-frequency for the baseline and “b)” process models

Process model	Trace	Activity-binary				Activity-frequency			
		A	B	C	D	A	B	C	D
Baseline	$\langle A, B, D \rangle$	1	1	0	1	1	1	0	1
b)	$\langle A, A, A, B, D \rangle$	1	1	0	1	3	1	0	1

Source: Rafael Gaspar de Sousa, 2021

3 Drift detection and localization

When dealing with concept drift, the first task is drift detection, which must be performed efficiently to enable further analysis. Once a drift is detected, we are able to localize, characterize or explain the drift (BOSE *et al.*, 2014). Early and accurate drift detection can largely contribute to process improvement and to the organization’s success, enabling fast correction measures and avoidance of negative and unexpected behavior of their business process. Hence the importance of having a reliable online method able to accurately detect drifts as early as possible and to understand what it impacts and what may have caused it.

While the drift detection task is broadly understood as the problem of finding the instant in time when a drift has occurred, the drift localization, however, does not have a unique definition. In general data mining, the problem of describing concept drift characteristics has been structured in Webb *et al.* (2016) through a framework able to define several quantitative features, such as the drift duration or magnitude, and categorical taxonomy describing the types of drifts. In process mining, the concept drift localization task is interested in detailing the drift subject, i.e., identifying the aspects of control-flow changes that explain the process drift. Most of the work on dealing with concept drift in process mining focus only on the detection task, not providing an approach able to perform localization in a systematic manner (OSTOVAR, 2019).

In this chapter, we propose an online method able to perform detection and localization of concept drift supported by trace clustering. The method can be divided into three steps: (i) to perform trace clustering in a streaming manner and extract descriptive measures about the clustering variation over time, (ii) to analyze clustering variations along the windows to detect the drift, and (iii) compare the clustering before and after the detected drift to identify the entities showing higher involvement in the drift. We have carried out experiments and evaluated the results in different scenarios of changes in the process models and considering different feature representations for the trace clustering step.

This chapter is organized as follows. Section 3.1 discuss some related work on dealing with concept drift in process mining. Section 3.2 presents a detailed description

of the proposed method. Finally, section 3.3 contains a summary of the main topics in discussion for this chapter.

3.1 Related work

Numerous approaches have been proposed for concept drift detection and localization in process mining. The detection methods are usually based on comparing traces in consecutive windows to decide whether differences found are significant enough to be considered a drift. Localization methods are generally based on comparisons between the process behavior before and after the drift to report differences. Most approaches focuses solely on only one of these tasks. The studies gathered in this section were a product of exploratory research.

3.1.1 Drift detection

The detection methods are going to be divided into two groups: the ones based on a similarity measure related to some feature extraction, and the ones adopting statistical tests to detect the drift. This section covers studies in both groups. Such studies gathered this section were a product of exploratory research.

Studies exploring statistical tests based approaches

One of the first studies to propose a method based on statistical tests is presented in Bose *et al.* (2011). First, the traces are represented as feature vectors that describe the relationship of activities by looking at their occurrences in the bag of traces. When comparing two consecutive windows of traces in a sliding manner, a significance probability is computed through a statistical test to evaluate whether both windows come from the same distribution. In Bose *et al.* (2014) and Martjushev, Bose and Aalst (2015) the method was extended to deal with gradual drifts, use an adaptive size window, and was more extensively evaluated.

Similar approaches are presented in Kumar, Thomas and Basava (2015) and Maaradji *et al.* (2015b). Both apply statistical tests over a sliding window of traces,

varying only on how to prepare and represent the trace as vectors. [Kumar, Thomas and Basava \(2015\)](#) uses a co-occurrence matrix, which limits its usage to an offline setting. However, [Maaradji *et al.* \(2015b\)](#) rely on the frequency of traces in the windows, thus being suitable for an online application. The work was also extended in [Maaradji *et al.* \(2017\)](#) in order to deal with gradual drifts.

This kind of approach has performed well in logs with simpler process models with low trace variability, since it relies on the frequency of unique traces in the event log. When in a complex process model with high trace variability, [Ostovar *et al.* \(2016\)](#) addressed the limitations in [Maaradji *et al.* \(2015b\)](#) by considering a different representation schema using features that describe the relationships between activities in the log, and then applying statistical test over trace windows. This approach has presented improved accuracy and it is able to detect a more vast set of changes in a complex business process that produced event logs with high trace variability.

More recently, [Lu, Chen and Poon \(2021\)](#) proposed an window based detection method with a similar representation approach. The idea is to deal with different behavior observed in the directly-follows relationship between activities as candidate drift points. Those points are then subject to an statistical G-test to support their significance to be considered as drift.

Studies exploring similarity-based approaches

One example of an approach based on similarity is presented in [Accorsi and Stocker \(2012\)](#). First, the traces are represented in a feature vector space as the average distance between every pair of activities presented in the log. Then, by clustering in a moving window all the pairs that have presented different distances, the method counts how many activities pairs have shown variations in each window. One downside of this method is its inability to autonomously decide whether a drift has been found, instead, it provides a chart of the variations along time to be used by an analyst. Also, the representation is not suited to deal with loops. [Hompes *et al.* \(2015\)](#) perform a Markov Clustering algorithm on a graph-based trace representation and compute a similarity matrix considering additionally some context-related data. The detection is performed by measuring the variation of the similarity matrix through trace windows. Similar to [Accorsi and Stocker \(2012\)](#), this work

does not introduce a method to autonomously decide whether a large variation is to be considered a drift or not.

In order to address the problem of an autonomous detection of concept drift, [Carmona and Gavalda \(2012\)](#) present a method based on ADWIN (Adaptative Windowing) generally used in concept drift detection. The windows of traces are represented as a convex polyhedron based on the frequency of each trace. Then, they are compared in subsequent windows and the algorithm can detect a drift if a significant number of samples lie outside the reference polyhedron.

The work in [Zheng, Wen and Wang \(2017\)](#) represents traces as a relationship matrix to indicate the occurrence of each pair of activities for every trace over time. The matrix is analyzed in fixed-size windows to identify points when there is a difference in the behavior between windows. Those points are then clustered using the DBSCAN algorithm, small groups are considered as outliers and larger ones as drifts.

[Liu, Huang and Cui \(2018\)](#) shows an online concept drift detection method that relies on using process discovery and conformance techniques to compare a trace window to a reference process model, by calculating the precision of the model. If it is lower than a pre-defined threshold, it indicates that the current behavior does not match the reference process model. This kind of approach depends heavily on the process discovery algorithms used, which does not guarantee perfect accuracy.

Building on a graph-based trace representation, [Tavares *et al.* \(2019\)](#) propose a online framework that extracts features by calculating graph distances considering both the control flow perspective of the trace, as well as the time between occurrence of activities. Then, the streaming clustering algorithm DenStream is used to capture the process behavior and, when a new clusters arise, a drift is detected. This work proposes a similar framework to ours, although it relies on a graph-based trace representation method and provide no option for using the models from trace clustering as base information to localize the drifts.

Another graph-based approach is presented in [Impedovo *et al.* \(2020\)](#) with a framework based in representing the process behavior in dynamic networks and obtaining the differences between them over time. In this work, the dynamic networks are subdivided and analyzed as sub-graphs. Drifts are detected when a relevant difference in frequency between a set of usual connected sub-graphs is found. It also provides a quantified magnitude of the changes found and the change patterns involved.

[Brockhoff, Uysal and Aalst \(2020\)](#) propose a sliding window drift detection approach that represent the process behavior in the window with stochastic languages and uses the Earth Mover's Distance (suitable when comparing probability distributions) as a similarity measure to compare the two subsequent windows with each other in order to detect drifts.

In [Yeshchenko *et al.* \(2021\)](#) the authors introduce a drift detection and visualization approach that represents the process behavior using Declare constraints ([AALST; PESIC; SCHONENBERG, 2009](#)). Those constraints specify the relationship between activities in terms of the observed events in the log (e.g. precedence, succession, response). Then, they are analyzed over time using time series clustering techniques to group for similar constraints. Each cluster is analyzed for concept drifts, and the final result of the framework produces several charts and visual artifacts to support the process analysts when analysing concept drifts.

3.1.2 Drift localization

An overall idea of performing concept drift localization is initially mentioned in [Bose *et al.* \(2011\)](#) and [Carmona and Gavalda \(2012\)](#) by proposing a comparison between the process behavior before and after the drift. Such comparison is suggested to be based on features that describe the relationship between activities before and after the detected drift. However, they do not provide an actual method and present no evaluation of their suggested approach. Also, they do not establish a scheme for framing the results of the localization task.

A method to compare differences between two process variants is proposed in [Bolt, Leoni and Aalst \(2018\)](#). It performs the comparison of the process variants before and after the drift through calculating some process metrics, such as activities or transitions frequencies, and performing statistical tests to detect significant differences. Statistically significant changes indicate the characteristic involved in the change before and after the drift. Since it relies on statistical tests, this approach may have problems in event logs with high trace variability, as common fluctuations in process behavior differences might be identified even though they are not necessarily occasioned by the drift itself. The output result of the localization analysis in this work is a visual transitions systems highlighting the intensity of changes.

Similarly, [Beest *et al.* \(2015\)](#) and [Nguyen *et al.* \(2018\)](#) presented approaches based on the construction of graphs that describe relationships (e.g. directly-follows, co-occurrence, causality, conflict, etc.) between entities in the event log (e.g. activity, resource, location). These approaches can deal with drifts from different perspectives other than the control-flow but have limitations in terms of high execution time since they deal with large graph structures. The method in [Beest *et al.* \(2015\)](#) is able to output natural language statements describing the differences, while [Nguyen *et al.* \(2018\)](#) presents the results in the form of visual matrix heatmaps.

A different approach to event log comparison is to first discover the process models on the sub-logs before and after the drift and then apply a comparison directly on the model level, as presented in [Armas-Cervantes *et al.* \(2016\)](#), with the usage of reduced graph-based event structure. One downside of this approach is that no automatic process discovery technique can guarantee perfect discovery accuracy, which might be associated with the localization of spurious differences discovered in the model but are not part of the actual process behavior.

Based on process model references, [Liu, Huang and Cui \(2018\)](#) provides a framework to compare the current process model in the stream to candidate process models learned from data over time. This work aims to classify the type of the drift (e.g. sudden, gradual, recurring or incremental) in order to adapt their process model references accordingly. In [Impedovo *et al.* \(2020\)](#), the changes in the process are identified by comparing the dynamic networks generated in subsequent windows, the same used when detecting the drift. They select the most representative traces and compare the differences in support between them in the two windows. This output provides a information on what aspect of the process behavior has changed with greater intensity.

Focusing on providing visual assets as support for understanding the concept drifts, [Pauwels and Calders \(2018\)](#) and [Yeshchenko *et al.* \(2021\)](#) propose several plots, charts and process model visualizations that require interpretation of process analysts. In terms of technique, [Pauwels and Calders \(2018\)](#) relies on the usage of Dynamic Bayesian Networks to capture conditional dependencies between activities that represent the process behavior learned. New traces are subject to an anomaly score based on the model learned from the reference window and then are displayed as attribute-density plots, where differences in the activity level can be visually spotted. While in [Yeshchenko *et al.* \(2021\)](#), segmented heatmaps resulting from the time series clustering of the DECLARE constraints obtained

in the detection step are presented, as well as the visualization of graphs highlighting more intense changes in the process model.

In [Ostovar *et al.* \(2017\)](#) a set of alpha relations is extracted from the event log that describes the observed relationship between activities. Then, statistical tests are performed to measure the association between each relationship with the actual drift. The relations are ranked and matched to predefined change templates that can translate the changes into natural language statements. As mentioned by the authors, this method needs improvement when dealing with more complex changes that involve multiple activities and overlapping or nested changes at the same time, given the level of representation obtained by the extracted activities relationships. The result of localization is presented as electing the change template most closely associated to the drift analyzed.

To address these previous problems, the work in [Ostovar, Leemans and Rosa \(2020\)](#) is based on the idea of discovering two block-structured process models from the sub-logs before and after the drift named *process trees*. Then, it finds the minimum cost sequence of edit operations that transform the process tree before the drift to the process tree after it, detailing changes in the process by exploring the sequence of edit operations obtained. This also outputs information regarding the relationship between activities, denoting whether a sequentialization of parallelization has occurred. Though this method works well for nested and overlapping changes with multiple activities, it does not have the same performance when localizing simpler drifts that involve individual activities.

3.1.3 A comparative analysis

Considering the drift detection task, [table 8](#) summarizes the work presented in this section in comparison with the approach proposed in this project. This table indicates the category for each work, type of drift that it aims to detect, whether it is suited for an online scenario and if it is capable of performing autonomous detection. Also, we structured the technique in which the approach is based and the windowing method applied when adequate.

The majority of work is focused on detecting sudden drifts, as it is the most common type of drift. Also, most studies are able to autonomously detect a drift, which is also the goal in this project. The techniques vary, being the similarity ones the most common, as

Table 8 – An overview of related work in concept drift detection from the detection perspective

Work	Category	Drift Type	Online	Autonomous	Technique	Windowing
Yeshchenko et al. (2021)	Similarity	Sudden/Gradual	No	Yes	Constraints time series	Sliding
Lu, Chen and Poon (2021)	Statistical	Sudden	No	Yes	G-test	Sliding
Impedovo et al. (2020)	Similarity	Sudden/Gradual	No	Yes	Dynamic networks	Sliding
Brockhoff, Uysal and Aalst (2020)	Similarity	Sudden	No	Yes	Earth Mover's Distance	Sliding
Tavares et al. (2019)	Similarity	Sudden/Gradual	Yes	Yes	Distance in graphs	-
Liu, Huang and Cui (2018)	Similarity	Sudden	Yes	Yes	Conformance checking	Sliding
Maaradji et al. (2017)	Statistical	Gradual	Yes	Yes	Chi-squared test	ADWIN
Zheng, Wen and Wang (2017)	Similarity	Sudden	No	Yes	Relationship change points	fixed-size
Ostovar et al. (2016)	Statistical	Sudden	Yes	Yes	G-test	ADWIN
Hompes et al. (2015)	Similarity	Sudden	No	No	Markov clustering	-
Kumar, Thomas and Basava (2015)	Statistical	Sudden	No	Yes	Mann-Whitney U test	ADWIN
Maaradji et al. (2015b)	Statistical	Sudden	Yes	Yes	Chi-squared test	ADWIN
Martjushev, Bose and Aalst (2015)	Statistical	Gradual	Yes	Yes	Kolmogorov-Smirnov test	ADWIN
Bose et al. (2014)	Statistical	Sudden	No	Yes	Kolmogorov-Smirnov test	ADWIN
Accorsi and Stocker (2012)	Similarity	Sudden	No	No	Cluster cuts	fixed-size
Carmona and Gavalda (2012)	Similarity	Sudden	Yes	Yes	Polyhedron overlapping	ADWIN
Bose et al. (2011)	Statistical	Sudden	No	Yes	Kolmogorov-Smirnov test	Sliding
This work	Similarity	Sudden	Yes	Yes	Trace clustering evolution	Sliding

Source: Rafael Gaspar de Sousa, 2021

the usage of adaptative windowing. Another important feature our work is that it focus on dealing with concept drift in an online form, processing traces as they arrive, while the most common approaches are meant for offline analysis.

In business processes of real life, there is not a general clear threshold in how much difference found represents a drift. A more subtle definition is required, depending on the process behavior, domain and people involved. For that reason, our drift detection algorithm provides parameters to adjust the detection sensibility adequately. While methods from the statistical category rely on the results of statistical tests with a specified threshold to detect drifts.

The detection technique performance for drift detection is directly related to the representation of the data in the event log that it is based on. Table 9 presents an overview of these representations in the related work and in this project. The representations can describe each trace individually or gather features from process behavior and describe the log completely. From a control-flow perspective and having no prior information on the drift to be detected, the representation should be adequate when dealing with changes in process relatives to loops, parallelism, and order. For these particular aspects, we indicate whether the representation distinguishes it fully (✓), indirectly (∼) or not (−). The work in [Hompes et al. \(2015\)](#) focuses on drifts in the data perspective and it is not subject to this analysis.

Table 9 – An overview of related work in concept drift detection from the representation perspective

Work	Representation	Subject	Loops	Parallelism	Order
Yeshchenko <i>et al.</i> (2021)	DECLARE constraints	Log	✓	✓	✓
Lu, Chen and Poon (2021)	Direct follows relations	Log	✓	✓	✓
Impedovo <i>et al.</i> (2020)	Dynamic networks	Log	✓	✓	✓
Brockhoff, Uysal and Aalst (2020)	Stochastic language	Log	✓	~	✓
Tavares <i>et al.</i> (2019)	Process model graph	Trace	✓	✓	✓
Liu, Huang and Cui (2018)	Process model discovery	Log	✓	✓	✓
Maaradji <i>et al.</i> (2017)	Trace frequency	Trace	~	✓	~
Zheng, Wen and Wang (2017)	Activities relationship matrix	Log	~	-	✓
Ostovar <i>et al.</i> (2016)	Alpha relationships	Log	✓	✓	✓
Kumar, Thomas and Basava (2015)	Co-occurrence matrix	Log	~	~	~
Maaradji <i>et al.</i> (2015b)	Trace frequency	Trace	~	✓	~
Martjushev, Bose and Aalst (2015)	Activity follows/precedes	Log	~	-	✓
Bose <i>et al.</i> (2014)	Activity follows/precedes	Log	~	-	✓
Accorsi and Stocker (2012)	Distance between pair of activities	Trace	~	-	✓
Carmona and Gavalda (2012)	Convex polyhedron based on frequency	Trace	✓	-	-
Bose <i>et al.</i> (2011)	Activity follows/precedes	Log	~	-	✓
This work	Activity and transitions profile	Trace	✓	✓	✓

Source: Rafael Gaspar de Sousa, 2021

Approaches that have the event log as subjects are mainly focused on capturing the order aspect of activities, usually by extracting a matrix that describes the relationship between all activities in terms of precedence, causality, conflict, and concurrence. When the subject of representation is the individual traces, usually one trace representation is employed that focuses on a specific perspective of control-flow behavior, leading to uncovered change aspects. For example, a representation based on activity frequency such as Carmona and Gavalda (2012) covers loops but are not able to distinguish changes in activities order. Representations that consider trace frequency, such as Maaradji *et al.* (2015b) and Maaradji *et al.* (2017) indirectly cover all aspects but are limited when having high trace variability.

To address this issue, our approach is able to use trace representations that cover completely the three aspects. More than that, our approach is flexible in terms of representation, as any vector trace representation can be employed in trace clustering. This aspect provides more adaptability to the dynamics of each process being studied.

Additionally, our detection approach is able to generate a model, consisting of trace clusters, that represents the process behavior present in traces processed from a stream. This model is applied to the drift localization task and also is able to support further analysis of process changes. Few approaches are able to perform both the detection and localization tasks within the same process representation. Table 10 shows an overview of the work on drift localization focusing on their different representations and outputs.

Table 10 – An overview of related work on drift localization from the output perspective

Work	Category	Representation	Output
Yeshchenko <i>et al.</i> (2021)	Similarity	Constraints time series	Visual system
Impedovo <i>et al.</i> (2020)	Graph	Dynamic networks	Weighted list of traces
Ostovar, Leemans and Rosa (2020)	Graph	Process trees	Natural language statements
Liu, Huang and Cui (2018)	Similarity	Process model reference	Type of drift
Bolt, Leoni and Aalst (2018)	Statistical	Transitions frequency	Visual transitions systems
Nguyen <i>et al.</i> (2018)	Graph	Perspective graph	Visual matrix
Pauwels and Calders (2018)	Graph	Dynamic Bayesian Network	Attribute density plot
Ostovar <i>et al.</i> (2017)	Statistical	Activities relationships	Change templates
Armas-Cervantes <i>et al.</i> (2016)	Graph	Process model discovery	Visual and textual description
Beest <i>et al.</i> (2015)	Graph	Direct acyclic graph	Natural language statements
This work	Similarity	Trace clustering	Weighted list of entities involved

Source: Rafael Gaspar de Sousa, 2021

Representations based on graph structures are the most common in the localization task, as their are one of the most frequent forms to represent process behaviors as models. Similarity and statistical studies aim to capture the relationship between activities and their frequencies, which is more alike to the trace representations applied in our work.

In terms of output, there is also a considerable variety. Some approaches focus on providing visual resources (e.g. plots, graph visualizations, etc.) to be interpreted by analysts, while others intend to describe the changes in more textual forms, such as natural language statements, lists or matrices. Our work focus on the latter form, but our results can additionally be inspected visually to when different perspectives are of interest.

The method proposed in our work is also based on comparing the sub-logs before and after the drift. In our case, we rely on the usage of trace clustering as an abstraction to compare differences in process behavior. The comparison is quantified through Mean Squared Error (MSE), which gives advantage information of not only pointing the differences found but also supplying the analysis with a sense of variation's severity that can be also utilized as a visual asset when analyzed over time. We selected MSE as it is indifferent to the direction of errors and, as it squares the errors, large differences are emphasized. Our approach has minimum computational costs, given that the trace clustering is a by-product of our detection method and can deal with smaller and natural fluctuations on processes behavior not related to drift, as clusters capture the essence of the process and are not affected by small variations. Although most works do not provide a systematic evaluation of the localization task performance, our results in experiments show that our method is able to accurately present an weight of the activities involved in the drift.

3.2 Proposed method

In order to detect drifts using a stream of traces, our approach first clusters the traces as they arrive in the stream. Then, it tracks the clustering evolution by extracting features that measure some properties of the clustered data. Next, our drift detection method iterates through the feature over time, comparing every value to a adaptative estimation of lower and upper boundaries that indicate whether the value at the current moment represents a significant difference to what it had analyzed before. Finally, the centroids resulting of trace clustering are compared by calculating the MSE to perform the localization task.

To illustrate the proposed method, figure 11 shows an overview of the steps that will be discussed in the following subsections. Initially, each trace σ_i from a stream is represented into a feature vector space. In this example, it is a 2D vector space for better visualization. Then, to cluster the traces, two different strategies can be applied: (a) based on offline clustering algorithms, use a windowing schema to buffer recent traces and cluster windows of traces independently; and (b) based on streaming clustering algorithms cluster traces in an evolving manner. In this example, the (a) strategy applies a landmark windowing model with a fixed-size of 3, while the (b) strategy is based on an aging strategy that specifies a weight for each trace depending on how recent it is according to the streaming clustering algorithm employed. For a trace arriving at i , this weight is halved at $i + 3$ and is set to zero after $i + 4$. The next step is to analyze the clustering evolution and it consists of, for each new trace processed, measure characteristics of the clusters that can represent the process behavior in a stable way and sensitive to changes caused by concept drift. In the example, the average distance between the centroids is extracted. Our drift detection method is based on analyzing the behavior of that given feature and check its variation over the iterations. When a significant change in the process behavior occurs at iteration 6, the centroids positioning are impacted and the feature value lies outside the expected threshold, causing the detection algorithm to report the detection of a drift.

Figure 11 – Overview of the drift detection method proposed

Source: Rafael Gaspar de Sousa, 2021

3.2.1 Vector representation

As previously mentioned in section 2.5.2, the vector representation is key to the performance of the trace clustering algorithm. This drives our work to analyze different representations since each one can represent a specific behavior of the process and can capture specific types of concept drifts.

There are two important conceptual aspects to discuss regarding the feature vector representation in our approach:

- Conceptually, each trace representation is based on a specific view and have inherent limitations, being able to represent only specific kinds of behaviors in the process. For example, a binary trace representation based on the occurrence of activities can only identify whether the activity is present or not in the trace, hence it does not hold information about loops and repeated occurrence of activities. This makes the activity binary trace representation not suited to detect drifts in which the change in the process behavior involves loops. Such changes, however, would be detectable in an activity frequency-based trace representation that counts the occurrence of activities in the trace.
- To create such vector spaces able to represent traces, one must identify the entities in the given perspective. For example, when considering the activities profile, the vector representation maps one dimension for each activity in the trace. As there are several different traces in a event log, that would lead us to potentially one different vector space per trace in a event log. In order to enable comparable results when clustering sub-logs, the vector space should be the same over all traces processed by our approach. To standardize the vector space over the event log, we have considered prior knowledge of all possible entities in each perspective to enable our vector representations.

In this study, the traces were mapped to a feature vector space based on the control flow perspective, considering the views of activities and transitions, and representation models based on occurrence, frequency and order. The main tasks performed in this step were:

- Implementation of functions to map traces into a vector space for each representation.

- Evaluation of limitations and advantages of what types of concept drift can each representation support for the event logs considered in the experiments.
- Experiments to evaluate the quality of trace clustering for each possible representation.

3.2.2 Clustering

Once the traces are mapped into the feature vector space, they are ready to be clustered. There are several clustering algorithms able to perform this step, as elaborated in section 2.4. The key challenge in this phase is the ability to deal with traces in an online setting. This is an inherent problem in streaming clustering, as studied in the data mining field, usually done using a window model to weight more recent data and fade out older data (CARNEIN; TRAUTMANN, 2019). Though streaming clustering approaches deal with this problem naturally, offline clustering algorithms can be adapted to be applied in this context, by using a window model and clustering every window of recent data independently.

In this project, we evaluated the impact of different clustering strategies on the quality of clustering and the influence on the concept drift detection. The main activities performed in this step were:

- Evaluation of the results of trace clustering using the *k-means* offline clustering algorithm. The algorithm was chosen because it is very common in process mining and in general data mining. Since it is not adapted to streaming environments, it was applied together with a window model, processing only the window of traces at each time and producing independent clustering results for each window.
- Evaluation of results when applying the CluStream streaming clustering algorithm. This algorithm is an extensions of traditional clustering algorithms adapted to work in the streaming environment. We investigated the dynamics they can provide on the concept drift detection, and compare it with the use of a windowed offline approach.
- Compare drift detection and localization results obtained with clustering to results obtained without using clustering.

3.2.3 Tracking clustering evolution

Through trace clustering, we have an abstract representation and clustered information that represents the process behavior. Since concept drift is a problem related to evolution and changes of concepts, we need to keep track of the transformation of that clustering. Work done in [Spiliopoulou *et al.* \(2006\)](#) and [Oliveira and Gama \(2010\)](#) have presented frameworks capable of identifying external transitions such as merging, splitting, disappearing and emerging of clusters; and internal transitions such as expansion, shrinkage and density variations.

Figure 12 shows an example of monitoring the evolution of 4 clusters over 10 iterations of an evolutionary clustering approach. Each colored line represents the presence of a given cluster in an iteration. In this example, there is a merge (denoted as a yellow circle) between clusters 0 and 1 at iteration 4 indicating that both clusters have become one. Cluster 2 emerges (yellow arrow) at iteration 5, possibly by the presence of new data points distant from what has processed before, while cluster 1 disappears (red “x”) at iteration 6 meaning that older concepts might not as relevant in the data stream as previously. Then, cluster 2 is split (yellow star) into two different clusters at iteration 7.

Figure 12 – Example of the evolution of clustering: identifying merge, split, disappear and emerge of four clusters over iterations

Source: Adapted from [Gong, Zhang and Yu \(2017\)](#)

Different types of process changes can impact the clustering in various forms, such as changing the position of the existing centroids, merging two clusters, or emerging a new one. To detail specific properties presented in the clustering evolution, we propose to extract features that characterize the clustering situation. A clustering situation $\zeta_i = \{\mathcal{C}_1^i, \mathcal{C}_2^i, \dots, \mathcal{C}_k^i\}$ is obtained at the end of each i iteration of the clustering algorithm and is a set of k clusters $\mathcal{C}$. Figure 13 shows the strategies for two types of features extracted in this project.

time-independent features (a) only consider properties of the current clustering situation, for example, the intra-cluster and inter-clusters distances; and time-dependent features (b) are related to the comparison of the two subsequent clustering situations (ζ_{i-1} is the previous situation; ζ_i is the current situation), such as the Mean Squared Error (MSE) and distance between clustering positioning.

Figure 13 – Example of time independent and time-dependent features extracted from clustering situations

Source: Rafael Gaspar de Sousa, 2021

Time-dependent features have the additional requirement of matching clusters from situation ζ_{i-1} to their corresponding ones in ζ_i . Situations when a cluster in ζ_{i-1} matches none or more than one cluster in ζ_i or there is a cluster in ζ_i that does not have a match in ζ_{i-1} , indicate that a merge, split, disappear or emerge have occurred. In [Spiliopoulou et al. \(2006\)](#) and [Oliveira and Gama \(2010\)](#) this is done through calculating the proportion of data points that overlap in each cluster, considering the aging of points, and defining a threshold to consider whether the cluster $\mathcal{C}_j^{i-1}$ matches $\mathcal{C}_j^i$ for $j \leq k$. This method is adequate when having a great number of data points in both windows that generate the clusterings, for example when using a sliding window model in a buffer stream of data. Other techniques rely on matching clusters by similarity. The problem can be understood as an assignment problem to find the pairs of clusters with a minimum possible distance. In this experiment, this problem is solved by the Hungarian combinatorial optimization algorithm introduced in [Kuhn \(1955\)](#). With a similar approach, the work in [Tanca et al. \(2019\)](#) has also applied a similarity-based matching, through minimizing Jaccard distance between all clusters in ζ_{i-1} and ζ_i and defining the matches by choosing clusters pairs with the smaller distance between them. This type of approach is useful in streaming situations

where processed data points are not stored and the only reference of previous data is in the form of information on clusters ζ_{i-1} .

The experiments rely on centroid-based clustering algorithms, that represent a cluster $\mathcal{C}$ by a centroid. According to [Han, Kamber and Pei \(2012\)](#), a centroid is, conceptually, the center point of a cluster and it is defined in our experiment by the average of all data points that compose a cluster, as shown in equation 4 how to obtain the centroid for a cluster $\mathcal{C}_j$ considering $\vec{x} \in \mathcal{C}_j$. The time-independent features extraction is based on calculating individual features for each cluster based on its centroid, such as its standard deviation of a cluster (equation 5) or skewness (equation 6). For the inter-cluster distances, we apply Euclidean distance between all k centroids in ζ . For the intra-clusters distances, we consider the average pairwise distance for all data points in the cluster (equation 7). Each feature is then summarized by aggregating the same individual feature from all clusters $\mathcal{C}$ into one value that represents that feature for the ζ situation of clusters. Table 11 presents the list of time-independent features explored in this experiment.

$$c_j = \frac{1}{n} \left(\sum_{i=1}^n \vec{x}_i \right) \forall \vec{x} \in \mathcal{C}_j \quad (4)$$

$$\mathcal{C}_{j_std} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (\vec{x}_i - c_j)^2} \quad (5)$$

$$\mathcal{C}_{j_skewness} = \frac{\frac{1}{n} \sum_{i=1}^n (\vec{x}_i - c_j)^3}{\sqrt{\frac{1}{n-1} \sum_{i=1}^n (\vec{x}_i - c_j)^2}} \quad (6)$$

$$\mathcal{C}_{j_intra} = \frac{2}{n(n-1)} \left(\sum_{i=1}^n \sum_{j=i}^n d(\vec{x}_i, \vec{x}_j) \right) \quad (7)$$

The time-dependent features are based on the comparison of a cluster $\mathcal{C}_j^{i-1}$ with its respective match $\mathcal{C}_j^i$ that corresponds to the next iteration of the algorithm. Considering d the number of dimensions in the centroids vector space, we calculate the squared error between centroids by measuring the variation in each specific dimension using equation 8 and summarize it, as in time-independent features, with the sum, average and counting the number of dimensions where MSE is different from 0. Additionally, we also consider the euclidean distance between $\mathcal{C}_j^{i-1}$ and $\mathcal{C}_j^i$ to measure how much variation in spatial

Table 11 – List of time-independent features extracted for cluster tracking

Feature	Description	Equation
<i>avg_cluster_std</i>	Average of all clusters' standard deviation	$\frac{1}{k} \left(\sum_{j=1}^k \mathcal{C}_j\text{-std} \right)$
<i>avg_dist_intra_cluster</i>	Average of all clusters' intra cluster distance	$\frac{1}{k} \left(\sum_{j=1}^k \mathcal{C}_j\text{-intra} \right)$
<i>avg_skewness</i>	Average skewness of all clusters	$\frac{1}{k} \left(\sum_{j=1}^k \mathcal{C}_j\text{-skewness} \right)$
<i>avg_dist_between_centroids</i>	Average of euclidean distances between all centroids	$\frac{2}{k(k-1)} \left(\sum_{i=1}^k \sum_{j=i}^k d(c_i, c_j) \right)$
<i>max_dist_between_centroids</i>	Maximum euclidean distance between all centroids	$\max_{\forall i \in k, \forall j \in k} d(c_i, c_j)$
<i>min_dist_between_centroids</i>	Minimum euclidean distance between all centroids	$\min_{\forall i \in k, \forall j \in k} d(c_i, c_j)$
<i>sum_dist_between_centroids</i>	Sum of euclidean distances between all centroids	$\sum_{i=1}^k \sum_{j=i}^k d(c_i, c_j)$
<i>std_cluster_std</i>	Standard deviation of all clusters' standard deviation	$\sqrt{\frac{1}{k-1} \sum_{j=1}^k (\mathcal{C}_j\text{-std} - \bar{\mathcal{C}}\text{-std})^2}$
<i>std_dist_intra_cluster</i>	Standard deviation of all clusters' intra cluster distance	$\sqrt{\frac{1}{k-1} \sum_{j=1}^k (\mathcal{C}_j\text{-intra} - \bar{\mathcal{C}}\text{-intra})^2}$
<i>std_skewness</i>	Standard deviation of all clusters' skewness	$\sqrt{\frac{1}{k-1} \sum_{j=1}^k (\mathcal{C}_j\text{-skewness} - \bar{\mathcal{C}}\text{-skewness})^2}$
<i>ssq_centroids</i>	Squared sum of the clusters' centroids	$\sum_{i=1}^k c_i^2$
<i>Silhouette</i>	Silhouette coefficient	See section 2.4
<i>DBi</i>	Davies-Bouldin cluster validation index	See section 2.4

Source: Rafael Gaspar de Sousa, 2021

positioning has happened. Table 12 presents the list of time-dependent features explored in this experiment.

$$\text{MSE}(\mathcal{C}_j) = \frac{1}{d} \left(\sum_{l=1}^d c_{j,l}^{i-1} - c_{j,l}^i \right)^2 \quad (8)$$

Additionally, these features were adapted to be extracted in a “no clustering” scenario for the experiments presented in chapter 4. In this case, we consider all data points in the windows to extract the time-dependent features and the time-independent features. For features that perform internal measurements on clusters, such as *avg_dist_intra_cluster*, we consider that all data points as belonging to the same cluster. The objective with this

Table 12 – List of time-dependent features extracted for cluster tracking

Feature	Description	Equation
<i>avg_MSE</i>	Average of all clusters' MSE	$\frac{1}{k} \left(\sum_{j=1}^k \text{MSE}(\mathcal{C}_j) \right)$
<i>count_non_zero_MSE</i>	Count the number of dimensions with nonzero MSE	$\sum_{j=1}^k \begin{cases} 1, & \text{if } \text{MSE}(\mathcal{C}_j) \neq 0 \\ 0, & \text{otherwise} \end{cases}$
<i>total_MSE</i>	Sum of all cluster's MSE	$\sum_{j=1}^k \text{MSE}(\mathcal{C}_j)$
<i>avg_diff_centroids</i>	Average euclidean distance between matched clusters at i-1 and i	$\frac{1}{k} \left(\sum_{j=1}^k d(c_j^{i-1}, c_j^i) \right)$

Source: Rafael Gaspar de Sousa, 2021

adaptation is to obtain a baseline reference of the performance of our drift detection and localization in this framework without clustering.

3.2.4 Drift detection

By tracking the clustering evolution we obtain a series of values for each one of the cluster tracking features at each new trace (or new window of traces, if using an offline clustering algorithm with Batch strategy). The drift detection process consists of checking if the current value has suffered a significant variation. To define what is significant in that scenario, we estimate a lower and upper tolerance based on the average and standard deviation of previous values presented in that feature over the previous traces. Since each value in the series being analyzed is already a representation of a significant group of traces, we assume that the clustering features are going to remain stable according to the process behavior in the traces. To detect drifts, we check if the current value is out of the boundary, which indicates that the current window of traces displays a significantly different behavior than what it has previously seen.

A detailed description of the proposed method for detecting concept drift is presented in algorithm 1. It has four parameters: v series of values corresponding to the chosen feature extracted from clustering over the traces windows; rw the size of a rolling window used for analyzing the behavior underlying v ; θ as a threshold regarding how many standard deviations to tolerate in considering a normal behavior in v ; and mb that sets a minimum tolerated variation to deal with noise and oscillation in v . The output is a list of the indexes where it found drifts.

Algorithm 1 Drift detection algorithm

```

1: Input
2:    $v$    Feature vector
3:    $rw$   Rolling window size
4:    $\theta$  Standard deviation tolerance
5:    $mb$  Minimum percentage for lower and upper boundaries
6: Output
7:    $d$    List of index of found drifts

8: procedure DRIFT_DETECTION( $v, rw, \theta, mb$ )
9:    $d = \emptyset$  ▷ *List of drifts found
10:   $buffer = \emptyset$  ▷ *Initialize window buffer queue
11:   $rolling\_avg = none$  ▷ *Store rolling average
12:   $rolling\_std = none$  ▷ *Store rolling standard deviation
13:   $index = 0$  ▷ *Initialize index for  $v$ 
14:  for each  $value \in v$  do
15:    if  $size(buffer) = rw$  then
16:      ▷ *Calculates upper and lower tolerance boundaries based on rolling mean and
      standard deviation
17:       $low\_tol = \min(rolling\_avg - (rolling\_std * \theta), (1 - mb) * rolling\_avg)$ 
18:       $up\_tol = \max(rolling\_avg + (rolling\_std * \theta), (1 + mb) * rolling\_avg)$ 
19:      if  $value < low\_tol$  or  $value > up\_tol$  then
20:        ▷ *Drift found!
21:         $d.append(index)$ 
22:         $buffer = \emptyset$ 
23:         $buffer.pop(0)$ 
24:         $buffer.append(value)$ 
25:      ▷ *Updates rolling average and standard deviation
26:       $rolling\_avg = average(buffer)$ 
27:       $rolling\_std = std(buffer)$ 
28:       $index += 1$ 

```

The algorithm iterates over all the values in v and appends values to the *buffer* until it has reached the rw size (line 15). Once the *buffer* is ready, it calculates the upper and lower boundaries based on the current *rolling_avg*, the product of the *std_tolerance* parameter, and the current *rolling_std* (lines 17-18). The next step tests whether it can consider the current index as a drift if the current value does not lie between the upper and lower boundaries (line 19). If a drift is found, we empty the *buffer* to discard values that represent old behavior and make it ready for the new concept (lines 20-21). The *buffer* is updated by removing the earliest value and append the current value (lines 23-24). Finally, it updates the current *rolling_avg* and *rolling_std* based on the updated *buffer* to be ready for the next iteration (lines 26-28) and increases the index indicating that it has

successfully iterated over the current element of v . There is also a variant version of this algorithm that checks if the *rolling_avg* is inside the boundaries, instead of the current value (line 19).

Through the parameter θ , we can be less or more flexible in how much variation is tolerated in the estimation of the boundaries. The idea behind this strategy is that the rolling average is capable of representing the process model average behavior during some time window and the standard deviation holds how much variation is present in the process model behavior seen inside the rolling window. Also, the parameter rw holds the size of the rolling window considered in the calculation of mean and standard deviation. This parameter impacts how much to smooth the values of the series, to make it more or less sensitive to anomalies and small changes. A drift is detected when the current value is outside of the upper or lower boundaries calculated in the previous iteration. When the boundary is too large, no drifts are detected and when the boundary is too small, there are too many false positives. The parameter mb sets a minimum boundary size based on a fraction of the current rolling average.

3.2.5 Drift localization

The main goal of our localization method is to compare the clustering situations before and after the detected drift to identify the activities involved in the control-flow change. An overview of the proposed approach is presented in figure 14.

A trace stream is processed and a drift is detected at a specific point in time. From that detection, two trace windows are subject to our method for localizing changes: the windows immediately before W_{before} and after W_{after} the drift. Our detection approach generates clusterings that describe the trace windows, which are then compared to localize the drift (denoted by ζ_{before} and ζ_{after} in the figure). The comparison is performed through the calculation of the Mean Squared Error (MSE) between the centroids from ζ_{before} and their corresponding ones in ζ_{after} , for each dimension. In the given example, no change in behavior is found in dimension A, while dimension B has an MSE of 0.16. In activity-based trace representation, the localization method indicates that the drift is associated more intensively with activity B.

Figure 14 – Overview of the drift localization method proposed.

Source: Rafael Gaspar de Sousa, 2021

The drift detection method is executed in real-time, enabling the localization strategy to profit from the availability of clusters from the current window. When one drift is detected, the clustering information on the current iteration is stored and, when the next iteration is available, the localization method takes place, taking into account the clustering situation in the interactions immediately before and after the detected drift. In the streaming clustering algorithms scenario, however, it is wise to wait for newer clusters to arrive to get a better representation of the new process behavior after the drift.

The comparison, relating the clustering situations before and after the drift relies on evaluating the variation on individual centroids positioning, as a control-flow drift might affect only a specific part of the process. In this step, centroids matching is necessary to identify the correspondents centroids from ζ_{before} and ζ_{after} . This issue was also addressed when dealing with time-dependent features for tracking cluster evolution in the detection task, as mentioned in subsection 3.2.3. In this case, we adopt the same strategy of applying the Hungarian combinatorial optimization algorithm introduced in [Kuhn \(1955\)](#) to match centroids by minimizing the distance between them.

Our proposed localization step can be performed independently of the clustering strategy being applied, as it relies solely on the centroids which are the core product of clustering. However, when using an streaming algorithm such as CluStream, the immediately new clustering situation does not fully captures the process behavior after the drift, thus requiring us to adapt the reference window of comparison accordingly.

The metric applied in this comparison is the mean squared error between the position of the centroid before and after the drift. Then, we average it over all clusters to obtain a mean variation for each dimension. Equation 9 shows the calculation of the MSE for a dimension d .

$$\text{MSE}(d) = \frac{1}{k} \sum_{j=1}^k (c_{j,d}^{\text{before}} - c_{j,d}^{\text{after}})^2 \quad (9)$$

Our goal in the localization is not to output a list of dimensions involved in the drift, but to provide the ordered list of dimensions with their corresponding MSE. With this information, it is straightforward to identify what dimensions are related to the drift and to what degree. Also, this value can be calculated at any instant and can also support visual monitoring or analysis of changes in the process as the data arrives in the stream, independently of drifts being detected.

3.3 *Summary*

In this section, we have presented a drift detection and localization approach based on tracking the evolution of trace clustering in the event log. For trace clustering, we propose the usage of a multiple trace representations in order to cover a broader range of changes in the process behavior. Various clustering strategies can be applied as to generate models that represent the process behavior. To track the evolution of clustering, we extract features based on the centroids and analyze them over time. When a significant variation on clustering is found, a concept drift is detected. Once a drift is detected, we localize the drift by comparing the clustering situation before and after the drift, obtaining a rank and intensity of the entities involved in the process drift.

4 Experiments

In this chapter, we present the results of several experiments carried out to assess the performance of different configurations of our approach over synthetic and real-world event logs. We analyzed the performance of clustering with batch windowing *k-means++* and compared it to *CluStream*. Also, we have experimented with an “no clustering” approach, that considers a single cluster is formed over batch windows. By not using clustering, we investigated the usefulness of trace clustering in the drift detection and localization methods.

For the batch clustering strategy, we utilized the offline clustering algorithm *k-means* with the landmark window model following the steps presented in figure 11 on the (a) clustering strategy. The *k-means* initialization was based on the strategy proposed in Arthur and Vassilvitskii (2007) with the *k-means++* approach. For every clustered window, the algorithm runs for 10 iterations with different initial centroids. The best result in terms of minimum sum of squared distances of every point to its closest centroid is selected as the result of the clustering step. When using the *CluStream* approach, the algorithm is fed with every trace individually, in a streaming manner, and both the micro and macro clusters are obtained after each trace has been processed.

The “no clustering” strategy computes the time-dependent and time-independent evolution features in batches using the landmark window model without performing any clustering, just considering the set of traces in the window. For example, when calculating the “*avg_dist_between_centroids*” we consider the average distance between all traces in a window as they were centroids; or when calculating the feature “*avg_diff_centroids*” we consider the mean of all traces as a unique centroid in each window and then the distance between the centroids in two subsequent windows is calculated. The goal of this analysis is to understand how the usage of clustering is able to improve performance of our drift detection and localization methods.

Our experiments explored the impact of specific parameters in results and the search for the best combination of parameters. The first set of parameters is related to the clustering step of our approach, while the second set of parameters are related to the drift detection and localization algorithm, as presented in section 3.2.4. The clustering parameters depend on the specific strategy being applied:

- For the batch strategy, we set the windowing model as landmark with fixed size, established by the parameter *window_size* and the parameter for number of clusters to be obtained in the *K-Means++* algorithm is defined in *k*.
- For the *CluStream*, there is the following parameters: *m* sets the number of micro-clusters to be maintained; *h* determines the size of the horizon window that controls the expiration of micro-clusters; *t* controls how large the radius of a micro-cluster is to append new data points to it; and finally *k_macro* sets the number of macro-clusters to be obtained by using the *KMeans* algorithm fed with the micro-clusters.
- For the “no clustering” strategy, the only required parameter is the *window_size* considered when calculating the features.

This chapter is organized as follows: Section 4.1 contains the results and discussion of experiments based on synthetic event logs, while section 4.2 describes results and discussion for real-world event logs.

4.1 Synthetic event logs

In this section, we present results from experiments performed over two sets of event logs: one based on a elementary process model (Loan Applications) with several change patterns, presented in Maaradji *et al.* (2015a); the second one based on a more complex process model, to simulate a high trace variability and noisy scenario, presented in Ostovar, Leemans and Rosa (2020).

For both event logs there is ground truth information available for the detection task, indicating when the concept drift occurs. That allow us to evaluate our approach with the F-Score measure to account for the balance between false positives and true positive classification into drift or not drift. For the drift localization task, however, there is only information on what activities are exactly involved into the change patterns available for the first set of event logs. With that information, we evaluate the quality of our drift localization results using the Area Under the ROC Curve (AUC) for the first event log. For the second set of event logs there is no ground truth available for the localization task. In that case, we examined our localization results assisted by process model discovery before and after the drifts. Also, there is no baseline results for the drift detection available to compare our approach.

The discussion on the first set of event log is present in subsection 4.1.1, followed by the discussion on the second set of event logs in subsection 4.1.2.

4.1.1 Loan Applications

This collection contains 72 event logs with different types of process drift based on Loan Applications baseline process model. This model has fifteen different activities, one start event, three end events, loops, parallelism and alternative branches, as presented in figure 15.

Figure 15 – Loan Applications: Baseline process model

Source: Maaradji *et al.* (2015a)

This set of event logs is made available by *4TU.Centre for Research Data*, at the *4TU.ResearchData*¹ repository. Specifically, the event logs studied in this section are from the collection named *Business Process Drift*² presented in Maaradji *et al.* (2015a).

The event logs were generated to simulate concept drifts in different types of control-flow changes. The baseline model was altered according to patterns of changes commonly identified in process models, such as adding, removing or swapping fragments. The changes are grouped into three categories: insertion (I), resequentialization (R) and optionalization (O). Table 13 shows a list of the control-flow changes present on the dataset with their corresponding categories. To simulate more intricate change patterns, there is also composite event logs generated by applying one change for each category in a nested manner (IOR, IRO, OIR, ORI, RIO and ROI). For example, the change pattern IOR was

¹ This repository has a collection of real and synthetic event logs for process mining research accessible at <https://data.4tu.nl/>.

² Available at https://data.4tu.nl/articles/dataset/Business_Process_Drift/12712436

obtained by first adding a new activity (category I), making this activity parallel (R) and putting it in a loop structure (O).

Table 13 – Loan Applications: Control-flow change patterns present in the dataset

Change pattern	Description	Category
cm	Move fragment into/out of conditional branch	I
cp	Duplicate fragment	I
pm	Move fragment into/out of parallel branch	I
re	Add/remove fragment	I
rp	Substitute fragment	I
sw	Swap two fragments	I
cd	Synchronize two fragments	R
cf	Make two fragments conditional/sequential	R
pl	Make two fragments parallel/sequential	R
cb	Make fragment skippable/non-skippable	O
fr	Change branching frequency	O
lp	Make fragment loopable/non-loopable	O

Source: Maaradji *et al.* (2015b)

Event logs were generated from the baseline model and for each altered model. Then, they were combined by alternating a fragment of event log from the baseline model and one from the altered model, in order to obtain four logs of sizes 2,500, 5,000, 7,500 and 10,000 cases with nine drifts at each 10% of the log size (MAARADJI *et al.*, 2015b). Figure 16 shows this operation to generate a log of 5,000 cases.

Figure 16 – An example of synthetic event log generation with concept drift

In this process, the position of each generated drift is used as the corrected answer to the detection task, providing an effective way to measure the efficiency of this method. To evaluate the accuracy of our method, we analyzed the F-score measure.

As previously mentioned in 3.2.1, the change patterns in this event log are subject to the limitations of the trace vector representations. Table 14 presents the complete relationship between trace representations and change patterns. The trace representation based on the transitions of activities can support all the different control-flow changes in the event log of this experiment. The checkmark (✓) indicates that a given trace representation supports the change pattern in the synthetic log in this experiment. The *fr* change pattern is a special case denote by the asterisk (*) because it is a case of a control-flow change that does not alter the structure of the process model, instead just alters it most frequent behavior. In this case, there is no explicit change in the traces that would suggest a limitation of any trace representation. To ensure fairness when evaluating our detection method, these limitations were considered for each trace representation.

Table 14 – Loan Applications: Support of trace representation for each control-flow change patterns

Change pattern	Category	Activity			Transitions binary
		binary	frequency	order	
cm	I	✓	✓	✓	✓
cp	I		✓	✓	✓
pm	I	✓	✓	✓	✓
re	I	✓	✓	✓	✓
rp	I	✓	✓	✓	✓
sw	I			✓	✓
cd	R			✓	✓
cf	R	✓	✓	✓	✓
pl	R			✓	✓
cb	O	✓	✓	✓	✓
fr	O	*	*	*	*
lp	O		✓	✓	✓
RIO	-		✓	✓	✓
ROI	-	✓	✓	✓	✓
ORI	-		✓	✓	✓
OIR	-	✓	✓	✓	✓
IRO	-	✓	✓	✓	✓
IOR	-	✓	✓	✓	✓

Source: Rafael Gaspar de Sousa, 2021

Parameters

For the experiments with the Loan Applications set of event logs, the parameters related to the batch strategy are present in table 15a; table 15b presents the ones related to the *CluStream* strategy; and table 15c shows the parameters considered in the “no clustering” strategy.

The values domains of each parameter were chosen empirically through exploratory experiments. For the *window_size* parameter we have considered values that are able to hold a significant amount of traces but also not being too large. For the k parameter we applied the elbow method to select reasonable values for k (YUAN; YANG, 2019). Other parameters were chosen by visually inspecting the results obtained in these event logs, interpreting the feature curves and their relationship to the drifts in the logs. The *mb* parameter for the *CluStream* is set at higher values when compared to the *batch* and “no clustering” strategy due to the streaming algorithm behavior of adapting and more variation at the beginning of the stream.

Batch strategy

In the following subsection a few analyses will be presented to illustrate the advantages and disadvantages of our approach when using the batch clustering strategy. We will show how different features and parameters reveal distinct results according to the nature of the drifts and specific change patterns of each event log.

Figure 17 shows the evolution of the feature *average distance between centroids* for all event logs of 5,000 cases, considering only the change patterns that can be perceived through the activity-binary representation. In these examples, we considered $k = 2$ and a *window_size* = 100 for better understanding. In these event logs, there is a drift in every 500 traces, alternating the baseline process behavior with different behavior for each change pattern. The vertical dashed lines identify the instant when the concept drift occurs, indicating that the following 500 traces come from a change pattern process.

As a result of the trace clustering steps, we can observe a recurring pattern of stability followed by an abrupt change in the feature behavior over time. Pattern *cf* displays a substantial increase in the average distance between the centroids right after

Table 15 – Loan Applications: Parameters in the experiment for the Loan Applications set of event logs.

(a) Clustering and detection parameters for the batch strategy

Parameter	Values
<i>window_size</i>	{50, 75, 100, 125, 150, 200, 250}
<i>k</i>	{2, 3, 6}
<i>rw</i>	{3, 4}
θ	{1.0, 1.5, 2.0}
<i>mb</i>	{0.3%, 0.5%, 0.7%, 1.0%}

(b) Clustering and detection parameters for *CluStream*

Parameter	Values
<i>m</i>	{15, 30, 50}
<i>h</i>	{50, 150, 200}
<i>t</i>	{2, 25, 50}
<i>k_macro</i>	{2, 3, 6}
<i>rw</i>	{150, 200}
θ	{1.5, 2.5}
<i>mb</i>	{0.3%, 3.0%}

(c) Windowing and detection parameters for the “no clustering” strategy

Parameter	Values
<i>window_size</i>	{100, 125, 150, 200}
<i>rw</i>	{3, 4}
θ	{1.25, 1.5, 1.75, 2.0}
<i>mb</i>	{0.3%, 0.5%, 0.7%, 1.0%}

Source: Rafael Gaspar de Sousa, 2021

the concept drift. With a similar effect, patterns *cm*, *pm* and *cb* have a substantial decrease in the feature value after the concept drift. The pattern *fr* also shared that behavior, but exhibit more subtle variations. Patterns *rp* and *re* have not reproduced the same effect in this feature because they involve disappearance and reappearance of one specific activity. This causes changes in all centroids positioning but keeps the relative positioning between them intact, meaning the average distance between cluster suffer no significant change.

Using the same parameters and configurations above, figure 18 shows the evolution of the *average distance between centroids* for all the other patterns that conceptually are not supported by the limitation of the activity-binary representation in the activity profile. We can observe there are no significant changes in the feature and no recurring

Figure 17 – Loan Applications: Evolution of the *average distance between centroids* for the 5,000 cases event logs, with activity-binary representation, window of size 100 and *k-means* with $k = 2$

Source: Rafael Gaspar de Sousa, 2021

pattern since the representation is limited and is not able to distinguish the behavior of the baseline process to the drift.

Figure 18 – Loan Applications: Evolution of the *average distance between centroids* for the 5,000 cases event logs, using activity-binary representation, window of size 100 and *k-means* with $k = 2$ in not supported change patterns

Source: Rafael Gaspar de Sousa, 2021

When running the same set of event logs but with transitions-binary representation instead, the results show the expected recurring pattern, as presented in figure 19. Though *sw*, *cd* and *sw* still have subtle variations, for the change patterns *lp* and *cp* there is a clear and significant change after every drift.

Each feature extracted from the cluster tracking process is focused on the description of a specific characteristic in the clustering, hence it is susceptible to specific types of control-flow changes. For example, figure 17 suggests that the *average distance between centroids* may not be the best feature to describe the drift for the change pattern *re*, since

Figure 19 – Loan Applications: Evolution of the *average distance between centroids* for the 5,000 cases event logs, using transitions-binary representation, window of size 100 and *k-means* with $k = 2$ in the activity not supported change patterns

Source: Rafael Gaspar de Sousa, 2021

it remains unchanged during the baseline and the drift log fragments. Figure 20 shows other features for the *re* change pattern. The *total MSE* presents the recurring behavior expected with a subtle increase right after each drift. Since it is a time-dependent feature, in the next window its value goes back to the previous level because it is comparing windows inside the drift fragment, remaining stable until the next drift.

Figure 20 – Loan Applications: Evolution of different features for the *re* change pattern with the 5,000 cases event log

Source: Rafael Gaspar de Sousa, 2021

The *window_size* parameter impacts directly on the detection delay, since the larger the window, more traces it needs to detect a drift. Figure 21 shows the results when varying the *window_size*. Larger windows tend to be more stable since it makes it possible to gather a more significant amount of traces to cluster, however also have a greater delay in reacting after a drift.

The k parameter specifies the number of clusters to be found by the *k-means* algorithm. Variations on this parameter impacts directly the features extracted from the

Figure 21 – Loan Applications: Impact of the *window_size* parameter reflected on the evolution of *average distance between centroids* for windows of 50, 100, 150 and 250 traces

Source: Rafael Gaspar de Sousa, 2021

clustering evolution. Figure 22 shows the execution of the experiment when varying the k parameter. Higher values are more sensitive to small changes since each cluster is related to fewer traces.

Figure 22 – Loan Applications: Impact of variation of the k parameter reflected on the evolution of *average distance between centroids* for values of 2, 3 and 6

Source: Rafael Gaspar de Sousa, 2021

CluStream

Regarding the *CluStream* strategy, figure 23 shows the execution of our approach for the change pattern supported by the activity-binary trace representation, as in figure 17. Though noisier patterns are observed in this results, the recurring behavior was presented, similar to the results obtained with the batch strategy. However, there is a particular behavior at the beginning of each series, when the micro clusters are still being initialized by the first traces fed to the algorithm. The strategy achieve stability as more traces arrive in the incoming stream.

Figure 23 – Loan Applications: Evolution of the *average distance between macro centroids* for the 5,000 cases event logs, with activity-binary representation, $h = 150$, $k_macro = 3$, $m = 30$, $t = 2$

Source: Rafael Gaspar de Sousa, 2021

The *CluStream* strategy produced feature series with stronger changes after the drifts for the change pattern *cb*, *cm* and *re*. For the change patterns *cf*, *rp* and *pm*, the series showed the same pattern but in an unsteady manner. Similar to what happened to the batch strategy, the change pattern *fr* is the one with most subtle changes, which indicates the clustering result is intrinsically connected to the type of change occurring in the process behavior. Some change patterns are made easier to analyze, while others require a more sensitive point of view.

The horizon parameter h impacts on the forgetting behavior of the clustering algorithm. The micro clusters expire faster for lower values of h and, otherwise, their weights last for longer periods for higher values. Figure 24 presents the execution of the *CluStream* strategy with several values of h to illustrate such a dynamic. The number of

micro clusters set by the parameter m present a similar dynamic. Figure 25 shows the behavior of clustering for different values of m . Balancing sensitivity on these parameters is important to detect deviations without pointing out false positives.

Figure 24 – Loan Applications: Evolution of the *average distance between macro centroids* for several values of h in the 5,000 cases event log of change pattern cb , with activity-binary representation, $k_macro = 3$, $m = 30$ and $t = 2$

Source: Rafael Gaspar de Sousa, 2021

Figure 25 – Loan Applications: Evolution of the *average distance between macro centroids* for several values of m in the 5,000 cases event log of change pattern cb , with activity-binary representation, $k_macro = 3$, $h = 150$ and $t = 2$

Source: Rafael Gaspar de Sousa, 2021

The t parameter establishes a multiplier for the radius of micro clusters to include new data. As for the effect of t in this set of event logs, there was not a significant difference perceived, although the intensity of noise and variation changes with it, as shown in figure 26. As for the k_macro parameter, there is a similar behavior to the batch strategy, where a larger number of macro clusters tend to show variations not necessarily related to the drifts that might lead to false positives. Figure 27 shows the execution of this strategy over different values of the parameter.

Figure 26 – Loan Applications: Evolution of the *average distance between macro centroids* for several values of t in the 5,000 cases event log of change pattern cb , with activity-binary representation, $k_macro = 3$, $m = 30$ and $h = 150$

Source: Rafael Gaspar de Sousa, 2021

Figure 27 – Loan Applications: Evolution of the *average distance between macro centroids* for several values of k_macro in the 5,000 cases event log of change pattern cb , with activity-binary representation, $h = 150$, $m = 30$ and $t = 2$

Source: Rafael Gaspar de Sousa, 2021

No clustering

When analyzing the event log evolution as the time-independent and time-dependent features over time without performing clustering, a messier behavior is perceived. The recurring drift pattern we have obtained with clustering is not observed in the “no clustering” strategy. Figure 28 shows the “no clustering” approach for the change patterns supported by the activity-binary representation.

In figure 28, although an intense variation is present in the feature value after each drift, we observe there is also variation in indexes not related to the drift and the stability in-between drifts is not present as it was presented in the results obtained with clustering strategies. This type of visualization shows that there is not enough abstraction of the process behavior to smooth the process behavior in periods known to not have had

Figure 28 – Loan Applications: Evolution of the *squared sum* for the 5,000 cases event logs and “no clustering” strategy, with activity-binary representation and *window_size* = 100

Source: Rafael Gaspar de Sousa, 2021

any drifts. When using clustering, these periods present less noise which helps the drifts' analysis.

Clustering parameters

Each of the parameters selected when clustering affected the results in a different way. In the following list we summary our findings for the impact of the parameters in our approach.:

- *window_size*: This parameter is applicable in the batch and “no clustering” strategy. It states the amount of traces being considered in each window. Higher values tend to produce more stable results while lower values might not be able to fully capture the process behavior. However, larger window sizes will require waiting for more traces to be completed between each batch then possibly delaying the drift detection.
- *k*, *m* and *k_macro*: These parameters are used to set the number of clusters in the batch strategy; micro and macro clusters in the *CluStream* strategy respectively. The selection of number of clusters is a well known problem in the data mining field and there are several ways to tackle it (YUAN; YANG, 2019). Although higher values tend to stabilize the behavior in feature curves in the context of our approach, that might affect the sensibility of our approach to drifts leading to higher false negatives.
- *h* and *t*: These parameters are exclusive to the *CluStream* strategy. Both have a similar effect to the *window_size* parameter, where there is much less stability in

the feature curves when lower values are selected. The h parameter can lead to false negatives for higher values, while the impact of the t parameter is more related to the data distribution and in our experiments showed less intense effects.

Trace Clustering

To explore what parts of the process behavior are being clustered together by the trace clustering algorithms, we can take a magnifying glass and look into the specific traces that were put into each cluster. As this event log is based on a fairly simple process, we decided to use the Inductive Miner process discovery technique presented in [Leemans, Fahland and Aalst \(2013\)](#) to obtain the process model for each cluster separately and inspect the differences before and after the drift visually.

Figure 29 shows the process model for each cluster before and after the drift in change pattern *cb*. The green background in the process model after the drift represents that there were no changes found in the process, while yellow backgrounds with red circle highlights the changes found in the process. In this case with $k = 2$, the first cluster (Cluster 0) includes only traces that end with “Loan Application Rejected” while the second one (Cluster 1) includes the rest of traces that can either finish with “Loan Application Canceled” or “Loan Application Approved”. There are differences in the processes models in Cluster 1, while the models in Cluster 0 exhibits no change. The activities “Prepare Acceptance Pack” and “Check if home insurance quote is requested” now become conditional after a XOR gateway before each of them is included in the processes.

Figure 29 – Loan Applications: Process model discovery applied to each cluster before and after the drift for change pattern *cb*. Batch strategy with $k = 2$, *window_size* = 100 at the first drift.

Source: Rafael Gaspar de Sousa, 2021

The fact that the first cluster (Cluster 0) remains unaltered is an example of when the ability of trace clustering to isolate similar behaviors is useful. Figure 30 shows a comparison of the feature series for the approaches using and clustering and when not using clustering. The drifts can be more easily perceived as a recurring pattern when using the clustering strategies (lines green and yellow) as opposed to the “no clustering” strategy (red line) in which the feature series is not as stable and the recurring pattern is not as well defined.

Figure 30 – Loan Applications: Evolution of the *average distance between centroids* when using the three different strategies for the change pattern *cb*.

Source: Rafael Gaspar de Sousa, 2021

For the change pattern *fr* presented in figure 31, there were no changes in the process models discovered. This was expected, according to the change made in the process behavior that only impacts the frequency of traces that have parallel activities occur in a specific order, involving activities “Check credit history”, “Asses loan risk” and “Appraise property”.

Figure 31 – Loan Applications: Process model discovery applied to each cluster before and after the drift for change pattern *fr*. Batch strategy with $k = 2$, *window_size* = 100 at the first drift at index 5,000.

Source: Rafael Gaspar de Sousa, 2021

Drift detection

The drift detection algorithm processes the feature values obtained after trace clustering as a stream. Figure 32 shows the execution of the algorithm for the *cb* change pattern with 10,000 cases using $k = 2$, $window_size = 100$ and the “average distance between centroids” as feature. The parameter rw defines the size of the rolling window to smooth the values in the stream and is plotted in the dotted green curve. The blue areas around the series represent the tolerance boundaries estimated by the rolling mean and standard deviation, at each value. Drifts are detected when the value lies outside the tolerance boundaries. In this example, all the drifts (vertical lines) were correctly found.

Figure 32 – Loan Applications: Execution of the drift detection algorithm for the *cb* with the 10,000 cases event log using $k = 2$, $window_size = 100$. The detection parameters are $\theta = 2$, $mb = 1\%$ and $rw = 3$

Source: Rafael Gaspar de Sousa, 2021

Impact of detection parameters

The mb parameter sets a minimum tolerance based on a percentage of the rolling mean in order to deal with noise and smaller variations. Figure 33 shows the execution of the experiment for different values of mb . When mb is set to 0, the boundaries can be very small, causing some false positive detections, as indicated by the purple arrow. Alternatively, when it is set to a large value, the drift detection strategy can ignore correct drifts by considering a wider range of values inside of the tolerance boundaries.

Figure 33 – Loan Applications: Execution of the drift detection algorithm for the *cb* with the 10,000 cases event log using $k = 2$, $window_size = 100$. The detection parameters are $\theta = 2$ and $rw = 3$

Source: Rafael Gaspar de Sousa, 2021

The θ parameter impacts the size of the boundaries as a multiplier of the rolling standard deviation. Figure 34 presents the execution of the experiment for different values of θ . This parameter usually affects segments with higher variation, for example from indexes 1,000 to 2,000, since more stable segments are dealt by the mb parameter. The purpose of this parameter is to adapt the tolerance boundaries according to the rolling standard deviation, to avoid the detection of false positives drifts in more unstable segments of the event log. For smaller values of θ , shorter tolerance boundaries are considered and more false positives can occur. When the value is too large, abrupt and significant changes may lie within the boundaries, leading to false negatives.

The rw parameter controls how many values of the analyzed feature to consider inside the rolling window. Figure 35 shows the execution of the experiment for different values of rw . It controls how much smoother the series will be and is useful for softening more unstable segments and reduce false positives. Similar to the other parameters, lower values may induce more false positives and higher ones can lead to more false negatives. This parameter also affects the detection delay, since the algorithm considers a larger window of values, it may need more windows to detect a drift. For example with $rw = 7$,

Figure 34 – Loan Applications: Execution of the drift detection algorithm for the *cb* with the 10,000 cases event log using $k = 2$, $window_size = 100$. The detection parameters are $mb = 1\%$ and $rw = 3$

Source: Rafael Gaspar de Sousa, 2021

five out of seven detected drifts took more than 1 window, and for $rw = 3$, all nine drifts were detected within 1 window in the batch strategy.

Detection results

To evaluate the results of the experiments we used the average F1 in the event logs of 5,000, 7,500, and 10,000 cases for each change pattern, to consider our method independently of the size of the log, as performed in [Maaradji et al. \(2015b\)](#). Drifts detected after two times the window size are considered as false positives, to discard detections that might have occurred by chance. Table 16 presents the results for each change pattern when considering the best parameters setting by each trace representation within the change patterns they support. Tables 17a, 17b and 17c describe the parameters when using each clustering strategy.

The results have shown that the *fr* change pattern is the hardest problem on our approach throughout all strategies. The change pattern in this case has only a slight impact on the process behavior since it changes the frequency of two specific activities that have only a small influence in the overall clustering situation. For the batch strategy, the

Figure 35 – Loan Applications: Execution of the drift detection algorithm for the *cb* with the 10,000 cases event log using $k = 2$, $window_size = 100$. The detection parameters are $mb = 1\%$ and $\theta = 2$

Source: Rafael Gaspar de Sousa, 2021

Table 16 – Loan Applications: F1 score results for the drift detection task for all change patterns. The best results for each case are highlighted in bold.

Pattern	No clustering				Batch				<i>CluStream</i>			
	Activity			Transitions	Activity			Transitions	Activity			Transitions
	binary	frequency	order	binary	binary	frequency	order	binary	binary	frequency	order	binary
cb	0.71	0.68	0.71	0.70	0.98	0.98	0.76	0.79	0.62	0.61	0.38	0.56
cf	0.64	0.77	0.67	0.68	1.00	1.00	0.71	0.98	0.36	0.61	0.30	0.45
cm	0.74	0.75	0.74	0.71	0.95	0.95	0.75	0.81	0.64	0.52	0.42	0.53
rp	0.75	0.84	0.75	0.74	0.97	0.97	0.74	0.95	0.74	0.52	0.65	0.57
pm	0.79	0.77	0.78	0.78	0.98	0.98	0.82	0.96	0.70	0.54	0.42	0.56
fr	0.77	0.79	0.74	0.77	0.18	0.18	0.75	0.30	0.36	0.59	0.41	0.46
re	0.68	0.78	0.63	0.74	1.00	0.41	0.72	0.77	0.52	0.57	0.30	0.51
lp	-	0.74	0.68	0.72	-	1.00	0.60	0.72	-	0.51	0.62	0.47
cp	-	0.74	0.73	0.77	-	0.93	0.66	0.93	-	0.50	0.39	0.52
pl	-	-	0.73	0.74	-	-	0.63	0.98	-	-	0.47	0.48
cd	-	-	0.62	0.71	-	-	0.66	0.81	-	-	0.57	0.53
sw	-	-	0.68	0.78	-	-	0.63	0.96	-	-	0.37	0.60
RIO	-	0.77	0.75	0.80	-	0.97	0.71	0.95	-	0.60	0.33	0.54
ROI	0.81	0.79	0.81	0.71	0.97	0.96	0.78	0.96	0.88	0.56	0.62	0.54
ORI	-	0.75	0.72	0.74	-	1.00	0.78	0.95	-	0.58	0.44	0.47
OIR	0.72	0.69	0.70	0.71	0.97	0.98	0.75	0.98	0.71	0.58	0.47	0.58
IRO	0.79	0.71	0.79	0.72	0.98	0.88	0.70	0.98	0.46	0.56	0.39	0.51
IOR	0.70	0.85	0.69	0.73	0.95	0.95	0.71	0.96	0.59	0.51	0.57	0.49
Average	0.74	0.76	0.72	0.74	0.81	0.90	0.71	0.88	0.61	0.56	0.46	0.52

Source: Rafael Gaspar de Sousa, 2021

Table 17 – Loan Applications: Parameters settings for the best average results for each trace representations.

(a) Best set of parameters for batch strategy

Trace representation	Feature	Clustering parameters		Detection parameters		
		<i>window_size</i>	<i>k</i>	<i>rw</i>	θ	<i>mb</i>
activity-binary	count_non_zero_MSE	125	2	3	1	0.3%
activity-frequency	count_non_zero_MSE	125	2	3	1	0.3%
transitions-binary	count_non_zero_MSE	125	2	3	2	0.5%
activity-order	total_MSE	200	2	3	2	1.0%

(b) Best set of parameters for *CluStream*

Trace representation	Feature	Clustering parameters				Detection parameters		
		<i>h</i>	<i>m</i>	<i>t</i>	<i>k_macro</i>	<i>rw</i>	θ	<i>mb</i>
activity-binary	avg_macro_interclusters_distances	200	50	2	3	200	1.5	1.0%
activity-frequency	count_non_zero_MSE_macro	200	50	2	3	200	3.5	3.0%
transitions-binary	std_micro_interclusters_distances	200	50	2	3	200	3.5	3.0%
activity-order	avg_macro_centroids_std	200	50	2	3	200	2.5	1.0%

(c) Best set of parameters for “no clustering”

Trace representation	Feature	Window size	Detection parameters		
			<i>rw</i>	θ	<i>mb</i>
activity-binary	avg_interpoints_distance	200	3	1.25	0.3%
activity-frequency	std_interpoints_distance	200	3	1.25	0.7%
transitions-binary	std_interpoints_distance	200	3	1.25	0.7%
activity-order	avg_interpoints_distance	200	3	1.25	1.0%

Source: Rafael Gaspar de Sousa, 2021

activity-binary and activity-frequency representations performed well on all other problems except *fr*, with F-scores averaging 0.96. With the transitions-binary representation, we obtained the best results overall, considering that it is able to represent all change patterns. However, the activity-binary and activity-frequency representations led to better results in the change patterns they support. For example, change pattern *lp* consists of the inclusion of a loop structure that increases the frequency of a specific activity inside a trace. In that case, we obtained better results when using the activity-frequency representation that adequately deals with this behavior.

The drift detection method showed the same level of results for the composite change patterns (*RIO*, *ROI*, *ORI*, *OIR*, *IRO* and *IOR*) as for the single change patterns. These results provided no indication that more complex changes can be harder to be detected as drifts. Actually, more subtle change patterns such as *fr* proved to be harder to detected since the drifts have less intensity.

Regarding the trace representations, both *transitions-binary* and *activity-order* have the advantage of supporting all change patterns, however they do not achieve better results than the activity-binary and activity-frequency representations. That might be due to the increase in complexity both in terms of number of dimensions and variance of the vectors inherent to these representation schemes.

The results from all variations of parameters in this experiment were ordered according to find the combination with best result in average, as presented in tables 16, 17a, 17b, and 17c. Consequently, good results for specific change patterns have been left out of this summary, such as for the *fr* for example since it was obtained with a particular combination of parameters that might not perform well in other problems. In a different parameters setting for the batch strategy (exhibit in table 18), we have obtained a F1 of 0.91 for the *fr* pattern which is superior than our baseline.

Table 18 – Loan Applications: Parameters for the best result for change pattern *fr*

Trace representation	Feature	Clustering parameters		Detection parameters			
		<i>window_size</i>	algorithm	<i>k</i>	<i>rw</i>	θ	<i>mb</i>
activity-frequency	avg_dist_between_centroids	150	<i>k-means</i>	2	3	2	0.7%

Source: Rafael Gaspar de Sousa, 2021

Over the different clustering strategies, the best results were found with the batch strategy, that was able to outperform both the “no clustering” and *CluStream* over all change patterns independently of the representations themselves, except for *fr* (in which “no clustering” achieved a better F-Score of 0.77 in table 16). As discussed previously, the “no clustering” approach lacks the benefits of segregating the process behavior to detect the drift obtained when performing trace clustering.

Although the *CluStream* strategy could not exceed the results of when not performing clustering in this analyzes, that is due to the more delicate parameter tuning required depending on each change pattern when using the streaming algorithm. That dynamic can be verified with the results of table 19 that consider the best combination of parameters for each change pattern. Also, we have compared our results with the approach presented in Maaradji *et al.* (2015b) on the same set of event logs.

When considering the best parameters combination for each change pattern, we are able to see that there is little improvement of the average result when using the “no clustering” strategy (from 0.72 on average to 0.81), where the *CluStream* results increases

Table 19 – Loan Applications: Comparison between the baseline (MAARADJI *et al.*, 2015b) and our concept drift detection method (F-Score results) when considering the best parameters setting per change pattern.

Pattern	Baseline	No clustering	Batch	<i>CluStream</i>
	(MAARADJI <i>et al.</i> , 2015b)	activity-order	activity-order	activity-order
cb	0.92	0.87	1.00	0.96
cf	0.98	0.83	1.00	0.96
cm	1.00	0.80	1.00	0.98
rp	0.96	0.79	1.00	0.96
pm	1.00	0.89	0.96	0.87
fr	0.75	0.79	0.91	0.68
re	1.00	0.79	1.00	0.90
lp	1.00	0.75	0.97	0.91
cp	1.00	0.86	1.00	0.78
pl	1.00	0.80	0.80	0.67
cd	0.88	0.73	0.83	0.76
sw	1.00	0.84	0.96	0.82
RIO	0.98	0.81	0.96	0.64
ROI	1.00	0.82	1.00	0.96
ORI	1.00	0.78	1.00	1.00
OIR	0.97	0.85	1.00	0.92
IRO	1.00	0.79	1.00	0.96
IOR	1.00	0.77	1.00	0.86
Average	0.97	0.81	0.96	0.87

Source: Rafael Gaspar de Sousa, 2021

to 0.87 on average. For all three strategies, the activity-order trace representation was the one in which the drift detection method achieved the best results in this analysis, even though it was not the best one overall (table 16). This indicate that this representation can be valuable under specific parameters settings.

Also, regarding the baseline comparison, we can notice that a slightly superior performance in the majority of cases is obtained by the baseline and our best configuration overall presents an equivalent or better result in 6 out of 12 change patterns. However, when considering the best configuration for each change pattern, our method presents close results. This indicates that better results can be achieved by our approach with adjustment of the parameters according to the change pattern being investigated, specially with the *CluStream* approach.

Drift localization results

This part of the experiment can be understood as a sequel to the detection experiment, as both are done in the same settings. The same clusters obtained during the detection steps were employed as resources to this localization experiment, in the same conditions.

To evaluate our approach, we have annotated, for all change patterns, the activities that were affected in the process drift. Then, the resulting MSE list per dimension is compared to the ground truth indicating whether the given activity is involved in the drift. To illustrate, figure 36 shows the activities involved in change pattern *cp*, in comparison to the baseline process model. In this case, *Check credit history* and *Assess loan risk* are the two activities annotated to be involved with the process drift.

Figure 36 – Loan Applications: Process model variation corresponding to change pattern *cp*

Source: Rafael Gaspar de Sousa, 2021

In the evaluation process, all the nine ground truth drifts are considered when analyzing the localization for each log. To isolate the analysis of the localization results, we simulated a perfect detection algorithm since the goal is to understand the performance

of the localization method by itself and analyze its capacity in providing meaningful information on all real drifts. The metric applied in this evaluation is the Area Under the ROC Curve that is based on the true positive and false positive rates and is able to provide a sense of how the MSE is able to distinguish the activities involved in the drift. As in the detection method, we have considered the average values throughout the event logs of sizes 5,000, 7,500, and 10,000 cases for each change pattern, in order to consider our method independently of log size. When using the *CluStream* strategy, we have considered the parameter h as the distance in terms of window reference to compare the centroids before and after the drift.

To better illustrate the output of our localization method, figure 37 shows the MSE for every dimension on the first drift of the event log *cp*. As seen in figure 36, the activities expected to be involved in this drift are *Check credit history* and *Assess loan risk*, both holding the highest values for MSE when comparing the clustering situations before and after the drift.

Figure 37 – Loan Applications: Drift localization results for the first drift in the 5,000 cases *cp* event log with activity-frequency representation, *k-means* with $k = 2$ and *window_size* = 100.

Source: Rafael Gaspar de Sousa, 2021

Other activities, such as *Check application form completeness* present small MSE due to fluctuations on the centroids positioning over time. The remainder activities have presented $MSE = 0$, showing no variation in the behavior before and after the drift. Figure 38 shows the MSE over the windows in the batch strategy for three activities. The *Check credit history* spikes recurring immediately after each drift, as the other activities have no significant variations over time.

Figure 38 – Loan Applications: MSE over time for three activities in the 5,000 cases *cp* event log with activity-frequency representation and batch with $k = 2$ and *window_size* = 100.

Source: Rafael Gaspar de Sousa, 2021

When the detection method finds a false drift, the localization method identifies activities that are not the correct ones involved in the drift, which indicates that the drift was probably detected by some natural fluctuation on the process behavior that caused deviation on clusters. As an example, in the *cp* event log with 10,000 cases, a false drift is found at index 6,750 while the closest real drift begins in index 7,000. In the results presented in figure 39, we observe clear differences in the false positive and true positive situations. Figure 39a presents the localization results for the false positive drift detected and figure 39b the results for a ground truth drift.

In the false drift situation, the identified changes are not related to a drift but due to natural fluctuations in the process behavior. For example, activities involved in loops or XOR gateways might vary their occurrence between a window and another, leading to a window having more loops or other specific behavior more frequently. Depending on the context, those fluctuations can also be considered as drift, just like change pattern *fr* in which the change in the process behavior is very subtle. Also, when applying our localization method to this situations, much smaller MSEs are found, as compared than the ones obtained in the detection of a correct drift. These subtle changes might have been

Figure 39 – Loan Applications: Drift localization results for the 10,000 cases *cp* event log with activity-frequency representation, *k-means* with $k = 2$ and *window_size* = 125

Source: Rafael Gaspar de Sousa, 2021

considered as a drift for the detection method, hence leading to a false positive detection and to an misleading localization.

The localization results can also provide a visual asset to analyze how the clusters evolve over time. Figure 40 shows the results of localization as a heatmap when performed for each window, independently of drifts, for each clustering strategy. Darker cells indicate higher values of MSE, while white ones express lower values for each specific activity (y-axis) at each case index. The horizontal black dashed lines point out the true drifts.

In the change pattern *cb*, the activities involved in the drift are *Check if home insurance quote is requested* and *Prepare acceptance pack*, which is correctly made clear by the illustrations for all clustering strategies. However, the best visual result is obtained by the batch strategy, as the “no clustering” and *CluStream* strategies present other activities at other indexes besides the drifts with occasionally higher values for the MSE. This behavior indicates that changes on trace distribution window to window are being more emphasized than needed, since they are not related to the drift itself, which might lead to lower performance on localization.

The present experiment has been based on the exploration of the same clustering parameters that have been examined in the detection experiment. The best results per change pattern for each trace representation are shown in table 20. Table 21a describes the clustering parameters for each case.

Figure 40 – Loan Applications: Drift localization for all index and the three different strategies. Using 5,000 cases *cb* event log with activity-binary representation and *window_size* = 125, *k* = 2, *h* = 200, *k_macro* = 3, *m* = 50 and *t* = 2

Source: Rafael Gaspar de Sousa, 2021

Table 20 – Loan Applications: Area under the ROC curve results for the drift localization task per change patterns

Pattern	No clustering				Batch				CluStream			
	Activity			Transitions	Activity			Transitions	Activity			Transitions
	binary	frequency	order	binary	binary	frequency	order	binary	binary	frequency	order	binary
cb	1.00	1.00	1.00	0.96	1.00	1.00	1.00	0.95	0.98	0.83	0.98	0.93
cf	0.99	1.00	1.00	0.98	1.00	1.00	1.00	0.98	0.94	0.80	0.98	0.93
cm	0.99	0.99	0.98	1.00	0.99	0.99	0.98	0.94	0.97	0.86	0.98	1.00
rp	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.90	0.99	0.99
pm	1.00	1.00	0.99	0.91	1.00	1.00	1.00	0.92	0.95	0.84	0.97	0.94
fr	0.96	0.98	0.99	0.97	1.00	0.99	1.00	0.99	0.90	0.80	0.91	0.66
re	1.00	1.00	1.00	0.99	1.00	0.99	1.00	0.99	1.00	0.93	1.00	0.70
lp	-	1.00	0.69	0.79	-	0.97	0.73	0.77	-	0.92	0.73	0.81
cp	-	1.00	0.36	0.97	-	1.00	0.41	0.97	-	0.91	0.39	0.96
pl	-	-	0.54	0.91	-	-	0.72	0.88	-	-	0.61	0.59
cd	-	-	0.42	0.88	-	-	0.63	0.88	-	-	0.65	0.69
sw	-	-	0.82	0.93	-	-	0.91	0.92	-	-	0.82	0.93
RIO	0.78	0.94	0.77	1.00	0.77	0.98	0.87	1.00	0.67	0.82	0.67	0.99
ROI	1.00	1.00	1.00	0.92	1.00	1.00	1.00	0.90	0.98	0.99	0.98	0.93
ORI	0.91	0.92	0.88	1.00	0.85	0.96	0.83	1.00	0.73	0.90	0.82	0.99
OIR	0.34	0.93	0.87	0.98	0.33	0.93	0.92	0.97	0.53	0.97	0.83	0.96
IRO	0.76	0.84	0.76	0.86	0.78	0.93	0.89	0.87	0.69	0.91	0.81	0.89
IOR	0.99	1.00	1.00	0.96	1.00	1.00	1.00	0.96	0.95	0.89	0.97	0.74
Average	0.89	0.97	0.83	0.94	0.90	0.98	0.88	0.94	0.86	0.91	0.83	0.86

Source: Rafael Gaspar de Sousa. 2020

Table 21 – Loan Applications: Parameters for the best average results for each trace representations for the drift localization task

(a) Best set of parameters for Batch

Trace representation	Clustering parameters	
	<i>window_size</i>	<i>k</i>
activity-binary	125	2
activity-frequency	175	2
transitions-binary	125	2
activity-order	125	2

(b) Best set of parameters for *CluStream*

Trace representation	Clustering parameters			
	<i>h</i>	<i>m</i>	<i>t</i>	<i>k_macro</i>
activity-binary	150	50	2	3
activity-frequency	150	50	2	3
transitions-binary	200	30	2	3
activity-order	200	30	2	3

(c) Best set of parameters for “no clustering”

Trace representation	Window size
activity-binary	150
activity-frequency	125
transitions-binary	125
activity-order	125

Source: Rafael Gaspar de Sousa, 2021

Similar to the results found in the detection task in terms of clustering strategy, the batch strategy stood out with best results overall, closely followed by the “no clustering” and *CluStream*. In terms of trace representations, the activity-frequency and transitions-binary showed the best results throughout all clustering strategies and change patterns.

When considering the batch strategy, the activity-binary and activity-frequency representations have performed nearly perfect for all their supported change patterns, achieving a score higher than 0.99 in every case, including the *fr* change pattern, that has been the most challenging one for the detection approach. In the *fr* change pattern, the process drift in the control-flow is not directly related to changes in a specific activity, but to a branching XOR condition in the process that slightly alters the frequency of some activities. As a result, the highest MSE achieved in the *fr* problem is 0.08 while, for example, the *cp* change pattern it reaches about 0.5. That is what makes dealing with this problem harder because change patterns as *fr* can easily be misunderstood as tolerable fluctuations in the process behavior.

The transitions-binary representation presented adequate performance overall, being able to cover all the change patterns with regular effectiveness. A setback on this representation is its high dimensionality since there can be many possible transitions in a process model to be mapped into a dimension. In this situation, centroids are subject to more natural fluctuations in the dimensions, which affect the performance negatively. A possible extension to deal with this problem would be to handle smaller variations or discount these natural fluctuations when calculating the MSE based on the variations on previous windows.

When analyzing the clustering parameters involved in these configurations, we notice that $k = 2$ presents the best performance, as in the detection experiment, alongside with *window_size* = 125 in almost all cases, except the activity-frequency representation. A possible explanation for the fact that this representation is the only one not bounded to a 0 and 1 interval, as loops imply that activities occur more than once, hence requiring a larger value for *window_size* as higher values impact the euclidean distance used when clustering.

The best clustering parameters found for the localization are different from the ones obtained for the detection task. Table 22 presents the results when considering the best clustering parameters obtained in the drift detection, previously displayed in table 17a. The same results are obtained for the activity-binary and transitions-binary

representations, where there is no difference in the *window_size* parameter. Activity-frequency and activity-order showed small alterations in the results due to the larger difference in the *window_size* parameter (from 175 and 125 in the localization respectively, to 125 and 200 in the detection), but on average remained the same.

Table 22 – Loan Applications: Area under the ROC curve results for the drift localization task per change patterns for the clustering parameters from the drift detection task using the batch strategy. Highlighted items indicate differences from the results according table 20

Change pattern	Activity			Transitions
	binary	frequency	order	binary
cb	1.00	1.00	1.00	0.94
cf	1.00	1.00	1.00	0.97
cm	0.99	0.99	0.97	0.93
rp	1.00	1.00	1.00	0.99
pm	1.00	1.00	0.99	0.91
fr	1.00	0.99	0.99	0.99
re	1.00	1.00	1.00	0.99
lp	-	0.94	0.70	0.77
cp	-	1.00	0.39	0.97
pl	-	-	0.74	0.88
cd	-	-	0.62	0.87
sw	-	-	0.91	0.92
RIO	0.76	0.98	0.86	1.00
ROI	1.00	1.00	1.00	0.89
ORI	0.84	0.95	0.83	1.00
OIR	0.32	0.94	0.91	0.97
IRO	0.77	0.90	0.88	0.86
IOR	1.00	1.00	1.00	0.96
Average	0.90	0.98	0.88	0.93

Source: Rafael Gaspar de Sousa, 2021

4.1.2 High trace variability process

The event logs presented in [Ostovar, Leemans and Rosa \(2020\)](#) are based on a more complex process model when compared to the Loan Applications presented in the the previous section. The goal with this event log is to simulate a noisy environment with high trace variability. The process model consists of 42 activities, five XOR-gateways, six AND-gateways, and three loop structures modeled in a intertwined manner, as shown in

the baseline process model in figure 41. The trace variability in these logs range from 80% to 99%.

Figure 41 – High trace variability process: Baseline model

Source: Adapted from [Ostovar, Leemans and Rosa \(2020\)](#)

There are initially five event logs for each change pattern (Table 23), with two drifts at trace indexes 900 and 1,900 that involve a different number of activities (from one to five activities) and 3,000 cases in total. Also, there are versions of the logs with 2.5% and 5.0% of random noise, totalizing fifteen event logs per change patterns. Additionally, there are two sets of more complex change patterns that combines multiple changes in the process (IOR, IRO, OIR, ORI, RIO and ROI). One set for composite changes, analogous to what has been presented in the previous set of event logs that combine process variations from multiple change patterns, they will be referred with the suffix “c” (e.g. IORc, IROc, etc.). Also, event logs composed of combining change patterns in a nested form (i.e. adding one activity and then making that new activity parallel) will be named with the suffix “n”.

Table 23 – High trace variability process: Change patterns present in event log and support by trace representation profile of our approach

Change pattern	Description	Category	Activity-binary	Activity-frequency	Transitions-binary	Activity-order
ConditionalMove	Move a fragment into/out of conditional branch	I	x	x	x	x
ConditionalRemoval	Insert/delete a fragment in/from conditional branch	I	x	x	x	x
ConditionalToSequence	Make fragments mutually exclusive/sequential	R	x	x	x	x
Frequency	Change branching frequency	O	x	x	x	x
Loop	Make a fragment loopable/non-loopable	O		x	x	x
ParallelMove	Move a fragment into/out of parallel branch	I			x	x
ParallelRemoval	Insert/delete a fragment in/from parallel branch	I	x	x	x	x
ParallelToSequence	Make fragments parallel/sequential	R			x	x
SerialMove	Move a fragment to between two fragments	I			x	x
SerialRemoval	Insert/delete a fragment between two fragments	I	x	x	x	x
Skip	Make a fragment skippable/non-skippable	O	x	x	x	x
Substitute	Substitute a fragment	I	x	x	x	x
Swap	Swap two fragments	I			x	x

Source: Rafael Gaspar de Sousa, 2021

Parameters

For the experiments with this event log, the parameters explored are present in table 24a related to the batch strategy; table 24b presents the ones related to the *CluStream* strategy; and table 24c shows the domain of the parameters considered in the “no clustering” strategy. The values domain of the parameters values were chosen empirically through exploratory experiments.

Batch

Despite the high trace variability present in these event logs, our approach is able to present an stable recurring pattern expected for the majority of cases, as shown in figure 42 with an example using the batch clustering strategy and using the “centroids sum of squares” feature. In this setting, only the change patterns *Substitute*, *ConditionalRemoval* and *Frequency* show no sensitivity to the drifts at indexes 900 and 1,900, while others show intense changes by the drifts and an stability along the log.

Table 24 – High trace variability process: Parameters in the experiment for the Loan Applications set of event logs.

(a) Clustering and detection parameters for batch strategy

Parameter	Values
<i>window_size</i>	{100, 125, 150, 175, 200, 250}
<i>k</i>	{2, 3, 6, 8}
<i>rw</i>	{3, 4, 5, 7}
θ	{1.25, 1.5, 1.75, 2.0}
<i>mb</i>	{0.3%, 0.5%, 0.7%, 1.0%}

(b) Clustering and detection parameters for *CluStream*

Parameter	Values
<i>m</i>	{30, 50}
<i>h</i>	{150, 200}
<i>t</i>	{2, 50}
<i>k_macro</i>	{2, 3, 6}
<i>rw</i>	{150, 200}
θ	{1.5, 2.5, 3.5}
<i>mb</i>	{1.0%, 7.0%, 10.0%}

(c) Windowing and detection parameters for the “no clustering” strategy

Parameter	Values
<i>window_size</i>	{100, 125, 150, 175, 200}
<i>rw</i>	{3, 4}
θ	{1.25, 1.5, 1.75, 2.0}
<i>mb</i>	{0.3%, 0.5%, 0.7%, 1.0%}

Source: Rafael Gaspar de Sousa, 2021

Noise levels

This event log allows us to investigate our approach performance in different levels of noise, which is presented in figure 43. As expected, benefiting from trace clustering generalization ability, our method is quite robust in those scenarios, exhibiting a steady pattern between drifts in all noisy scenarios. The *CluStream* approach is more responsive and susceptible to fluctuations overall, as it process each trace as they appear. That makes it possible to detect drifts with a shorter delay in terms of number of traces, as it is not bound to batches and does not require waiting for a specific number of new traces. Interestingly though, for the second drift in the log with 5.0% of noise (around index

Figure 42 – High trace variability process: Evolution of the *centroids sum of squares* with activity-binary trace representation for the supported change patterns, using the batch strategy with *window_size* = 150 and *k* = 3.

Source: Rafael Gaspar de Sousa, 2021

2,000), the delay is longer than in with the batch strategy, due to the tolerance boundaries being a little more flexible as result of the higher variation of the feature value over time.

Under the same conditions and parameters as mentioned above for the batch strategy, figure 44 shows the execution when clustering is not performed. In these cases, although all true drifts are correctly detected and the expected recurring pattern is present, there are occasional false positives in this level of sensitivity, as the “no clustering” strategy produces less steady lines, specially in the highest noise level.

Drift detection results

In terms of overall detection results, table 25 shows the F1 score resulted by the execution of our approach throughout all event logs in this set using the different strategies and trace representations. For each change pattern, we have averaged the results for all different number of activities involved in the drift (from one to five activities) and all different levels of noise (0%, 2.5% and 5%). Identically to what has been performed for the previous set of event logs, we have selected the best configuration of parameters and clustering feature for each specific representation. The best parameters are available in tables 26a, 26b and 26c.

As seen in the previous set of event logs, the batch strategy achieved the best results overall, being able to reach an F-Score of 1.0 for most change patterns. The transitions-binary representation achieves the best result overall, with an average F-Score

Figure 43 – High trace variability process: Execution of the drift detection algorithm for the log ConditionalMove-1 with different levels of noise. a) Using $window_size = 100$, $k = 3$ for the batch strategy with $rw = 3$, $std_tol = 2.5$ and $min_tol = 0.01$ for drift detection; b) Using $h = 200$, $k_macro = 3$, $m = 50$ and $t = 2$ for the *CluStream* strategy with $rw = 200$, $std_tol = 1.5$ and $min_tol = 0.02$ for drift detection.

Source: Rafael Gaspar de Sousa, 2021

of 0.97 for all change patterns. The same level of performance obtained in the simpler change patterns is attained in the more complex change patterns, both the nested and composite examples.

However, both the *CluStream* and the “no clustering” strategies did not show the same level of performance, averaging an F-Score of around 0.53 throughout the different representations and change patterns. As we have seen an example of execution with good results in figure 43, the *CluStream* approach is adequate for this task but requires more sensitive parameter settings, as it also produces features with more variability that can lead to false positives. In that manner, table 27 shows the results with the best combination of parameters found for each change pattern, indicating that better outcomes can be achieved with adjusted tuning of parameters associated with each problem particularity.

Even when considering specific parameters for each change pattern, the results for the “no clustering” strategy does not improve as in the Loan Applications process.

Figure 44 – High trace variability process: Execution of the drift detection algorithm for the log ConditionalMove-1 with different levels of noise with “no clustering” $window_size = 100$ and $rw = 3$, $std_tol = 2.5$ and $min_tol = 0.01$ for drift detection.

Source: Rafael Gaspar de Sousa, 2021

Different from the results seen in the previous set of event logs, the *CluStream* is in a higher standard when compared to the “no clustering” strategy. As this is a more complex process, the models provided by clustering produced better results by summarizing the key aspects of process behavior.

Drift localization experiments

Although details on which activities are involved in drifting for each change pattern are lacking, our localization method can still be applied and analyzed in an unsupervised manner. Additionally, we utilized the inductive miner process discovery technique in the segments before and after the drift to visually inspect and validate the localization results.

Table 25 – High trace variability process: Best results achieved for drift detection, considering the best parameter combinations for each change pattern individually

Change pattern	No clustering				Batch				CluStream			
	Activity		Transitions		Activity		Transitions		Activity		Transitions	
	binary	frequency	order	binary	binary	frequency	order	binary	binary	frequency	order	binary
ConditionalMove	0.75	0.53	0.61	0.45	1.00	0.93	1.00	1.00	0.63	0.52	0.65	0.48
ConditionalRemoval	0.84	0.46	0.73	0.60	1.00	0.93	1.00	1.00	0.50	0.45	0.63	0.63
ConditionalToSequence	0.75	0.48	0.54	0.66	1.00	0.91	1.00	1.00	0.70	0.58	0.74	0.56
Frequency	0.35	0.48	0.43	0.40	0.86	0.97	0.94	0.90	0.22	0.22	0.25	0.36
Loop	-	0.53	0.35	0.55	-	1.00	0.95	0.94	-	0.55	0.16	0.38
ParallelMove	-	-	0.28	0.64	-	-	0.86	1.00	-	-	0.17	0.57
ParallelRemoval	0.66	0.55	0.74	0.45	1.00	0.80	1.00	1.00	0.25	0.36	0.37	0.62
ParallelToSequence	-	-	0.39	0.46	-	-	0.74	1.00	-	-	0.18	0.67
SerialMove	-	-	0.28	0.55	-	-	0.83	1.00	-	-	0.15	0.38
SerialRemoval	0.65	0.51	0.70	0.44	1.00	0.77	1.00	1.00	0.19	0.38	0.33	0.51
Skip	0.64	0.44	0.47	0.52	1.00	0.82	1.00	0.97	0.37	0.30	0.34	0.42
Substitute	0.54	0.47	0.61	0.39	1.00	0.97	1.00	1.00	0.63	0.50	0.62	0.54
Swap	-	-	0.40	0.53	-	-	0.98	1.00	-	-	0.15	0.41
RIOc	0.63	0.53	0.63	0.63	1.00	1.00	1.00	1.00	0.66	0.46	0.57	0.69
ROIc	0.78	0.54	0.81	0.62	1.00	1.00	1.00	1.00	0.74	0.64	0.71	0.52
RIOn	0.64	0.51	0.60	0.66	1.00	1.00	1.00	1.00	0.71	0.58	0.65	0.44
ROIn	0.74	0.43	0.59	0.69	1.00	1.00	1.00	1.00	0.74	0.67	0.76	0.48
OIRc	0.67	0.48	0.62	0.56	1.00	1.00	1.00	1.00	0.69	0.61	0.68	0.72
ORIC	0.36	0.33	0.44	0.33	1.00	0.93	1.00	1.00	0.52	0.54	0.43	0.53
OIRn	0.82	0.46	0.75	0.53	1.00	0.89	1.00	1.00	0.48	0.46	0.54	0.59
ORIn	0.58	0.46	0.58	0.56	1.00	1.00	1.00	1.00	0.60	0.60	0.64	0.68
IORc	0.57	0.36	0.46	0.61	1.00	1.00	1.00	0.97	0.67	0.67	0.75	0.66
IROc	0.69	0.46	0.70	0.61	1.00	1.00	1.00	1.00	0.78	0.60	0.76	0.71
IORn	0.55	0.41	0.67	0.51	1.00	0.97	1.00	1.00	0.69	0.67	0.71	0.45
IROn	0.46	0.53	0.45	0.48	1.00	0.87	1.00	1.00	0.61	0.57	0.59	0.54
Average	0.57	0.47	0.55	0.54	0.95	0.81	0.92	0.97	0.57	0.52	0.50	0.54

As a visual asset, figure 45 shows the execution of our drift localization approach over all traces of the *ConditionalMove* change pattern with the three different strategies. The activities n and p stand out as the ones that present more intense MSE when comparing the process before and after the drift index.

As expected according to the drift detection results, both clustering strategies provide a cleaner visualization of the variations on the log over time, minimizing natural fluctuations on most variables. Different from the batch strategy, the *CluStream* strategy indicates an intense variation in activity r right at the start of the log. This might be related to the initial adjustment of the algorithm as the traces are still being accommodated in the clustering model gradually. On the other hand, the “no clustering” strategy signalizes variations with medium intensity over a great number of activities during the log, which are due to the natural difference in trace distribution from one batch window to another that might mislead the localization results.

As a way to verify the correctness of this results, we also analyzed the process model obtained when running process discovery before and after the drift. Figure 46 shows the process model for the traces in each cluster before and after the drift.

Table 26 – High trace variability process: Parameters for the best average results for each trace representations

(a) Best set of parameters for Batch

Trace representation	Feature	Clustering parameters		Detection parameters		
		<i>window_size</i>	<i>k</i>	<i>rw</i>	θ	<i>mb</i>
activity-binary	avg_diff_centroids	250	3	3	1.5	1.0%
activity-frequency	avg_diff_centroids	250	6	3	1.5	1.0%
transitions-binary	avg_diff_centroids	250	8	3	1.5	2.5%
activity-order	avg_diff_centroids	250	8	3	1.5	1.0%

(b) Best set of parameters for *CluStream*

Trace representation	Feature	Clustering parameters				Detection parameters		
		<i>h</i>	<i>m</i>	<i>t</i>	<i>k_macro</i>	<i>rw</i>	θ	<i>mb</i>
activity-binary	avg_micro_interclusters_distances	200	50	2	3	200	2.5	7.0%
activity-frequency	count_non_zero_MSE_micro	200	50	2	3	200	2.5	7.0%
transitions-binary	avg_micro_interclusters_MSE	200	50	2	3	200	2.5	7.0%
activity-order	avg_micro_interclusters_MSE	200	50	2	3	200	2.5	7.0%

(c) Best set of parameters for “no clustering”

Trace representation	Feature	Window size	Detection parameters		
			<i>rw</i>	θ	<i>mb</i>
activity-binary	avg_interpoints_max_distance	200	3	1.25	1.0%
activity-frequency	abs_error	175	3	1.25	0.3%
transitions-binary	avg_points_std	200	3	1.25	1.0%
activity-order	squared_sum	200	3	1.25	1.0%

Source: Rafael Gaspar de Sousa, 2021

In this case, the process model changed in all clusters in the same step, pinpointed by the red circle in the figure. The activities n and p that before the drift happened sequentially, now are put into conditional gates. There were no changes found when comparing the other segments of the process model, therefore confirming the accuracy of our localization output.

4.2 Real-world event logs

Next, we evaluated our approach with two real-world event logs made available in BPICs (Business Process Intelligence Challenges). In the next subsections, we describe the business context of such event logs and uncover all processing steps and results from the application of our approach in each of them.

Table 27 – High trace variability process: Best results achieved for drift detection, considering the best parameter combinations for each change pattern individually

Change pattern	No clustering	Batch	<i>CluStream</i>	
	activity-order	transitions-binary	activity-binary	transitions-binary
ConditionalMove	0.64	1.00	0.89	0.66
ConditionalRemoval	0.74	1.00	0.82	0.84
ConditionalToSequence	0.64	1.00	0.89	0.72
Frequency	0.53	0.90	0.40	0.48
Loop	0.50	0.94	-	0.53
ParallelMove	0.67	1.00	-	0.83
ParallelRemoval	0.70	1.00	0.72	0.83
ParallelToSequence	0.54	1.00	-	0.91
SerialMove	0.51	1.00	-	0.69
SerialRemoval	0.64	1.00	0.58	0.72
Skip	0.78	0.97	0.62	0.60
Substitute	0.50	1.00	0.86	0.69
Swap	0.60	1.00	-	0.87
RIOc	0.58	1.00	0.89	0.89
ROIc	0.74	1.00	1.00	0.79
RIO _n	0.50	1.00	0.93	0.67
ROI _n	0.68	1.00	0.98	0.71
OIRc	0.68	1.00	0.93	0.95
ORIc	0.80	1.00	0.71	0.79
OIR _n	0.59	1.00	0.83	0.85
ORI _n	0.50	1.00	0.88	0.91
IORc	0.72	0.97	0.83	0.91
IROc	0.67	1.00	1.00	0.95
IOR _n	0.62	1.00	0.91	0.64
IRO _n	0.54	1.00	0.81	0.79
Average	0.62	0.99	0.82	0.77

Source: Rafael Gaspar de Sousa, 2021

4.2.1 BPIC 2015

The set of event logs from the BPIC 2015 relate to the process of issuing permits (to formally allow constructions, demolition, etc.) in five different Dutch municipalities. The events range from late 2010s until early 2015, and there were around 1,000 cases and 50,000 events for each of the five municipalities (we will refer to each of them as M1 to M5).

The underlying process for all municipalities are the same, though they present minor differences. Along the four years period covered by the event logs, changes are made to procedures, rules or regulations, producing potential concept drifts that we analyzed with our approach. Over the five event logs there are 356 different activities in the lowest

Figure 45 – High trace variability process: localization for all index and the three different strategies in the log ConditionalMove-1. Using $window_size = 150$, $k = 3$, $h = 200$, $m = 50$, $t = 2$ and $k_macro = 3$

Source: Rafael Gaspar de Sousa, 2021

level of abstraction. In the highest level, there are 35 activities that denotes landmark stages of the process, starting with the name “phase”. To better understand the process, Teinemaa et al. (TEINEMAA; LEONTJEVA; MASING, 2015) obtained a process model by analyzing the most frequent behavior in a high level abstraction for M1, showed in figure 47. The process starts by receiving the application and for asking additional information when needed. Once the application is received, there is a step of decision making that involves getting advice from stakeholders involved in each case. Then, the decision is sent after it has been taken and the process ends.

To analyze possible changes in the process behavior over time, we have executed our approach for the five event logs. The experiments were performed with the three different

Figure 46 – High trace variability process: Process model discovery before and after the drift for each cluster.

Source: Rafael Gaspar de Sousa, 2021

Figure 47 – BPIC 2015: Most frequent behavior for high level abstraction in M1

Source: Adapted from [Teinemaa, Leontjeva and Masing \(2015\)](#)

clustering strategies previously mentioned, and the parameters were selected empirically. Exploratory experiments were performed considering the four trace representations and several distinct parameters settings. The curves generated from the trace clustering features were inspected visually against each other to determine the elected parameters. No major differences were found among the representations, then the activity-binary representation was selected for further analysis due to its simplicity.

Since each log has a different number of traces, we have set specific window sizes for each log as 7% of the size of the full log (resulting in sizes of 83 for M1, 58 for M2, 98 for M3, 73 for M4 and 80 for M5). For the batch clustering step, we used $k = 2$ and for the *CluStream* the parameters were $h = 150$, $m = 20$, $t = 2$ and $k_macro = 3$. The drift detection parameters were $rw = 3$, $std_tol = 3.5$ and $min_tol = 1.0\%$ for both the batch and “no clustering” strategy while for the *CluStream* the parameters were $rw = 100$, $std_tol = 2$ and $min_tol = 10\%$. Figure 48 shows the result of the detection step for all municipalities considering the *average intracusters distances* feature over time.

Figure 48 – BPIC 2015: Execution of our approach for all five municipalities with the different clustering strategies.

Source: Rafael Gaspar de Sousa, 2021

A similar tendency is present through all event logs, showing a growth trend starting at early 2012 with the feature value increasing and stabilizing after 2012. With the batch and *CluStream* strategy, there is one drift detected around that period in all municipalities. Additionally, for municipalities M2, M3 and M4 a drift is found near the end of the event log with the batch strategy, while with *CluStream* there were drifts identified at earlier points in the logs M2, M3 and M4. The “no clustering” approach identified a drift around early 2012 in every municipality except M2. The changes in the process do not occur at

the same time in the five municipalities, probably due to the fact that each have different starting dates for changes in legislation, regulations that requires different procedures changes or some freedom when implementing those changes (HAM, 2015). The fact that different clustering strategies and parameters detect a drift in the same instant can be considered as a factor that indicates a certain level of confidence that there is a true drift at that specific.

Although there is not a definitive answer concerning the existence of concept drift in this set of event logs, previous work have also encountered a possible process drift at the same period near early 2012 (TEINEMAA; LEONTJEVA; MASING, 2015; HAM, 2015). To further investigate the drifts found, figure 49 shows the result of executing our drift localization step for all municipalities when considering the batch clustering strategy. The activities *Phase Decision Ready*, *Phase Decision Sent* and *Phase Decision Taken* were the most involved throughout all municipalities, especially in M3, M4 and M5.

Figure 49 – BPIC 2015: Results of drift localization for the first drift found in all event logs with the batch strategy. Showing the activities that had an MSE above 0 in at least one event log.

Source: Rafael Gaspar de Sousa, 2021

To assist our understanding of the process behavior and the drifts detected, we have also analyzed the occurrence of the activities over time. Figure 50 shows a heatmap with the average occurrence of the top 10 most frequent activities in a 50 cases rolling mean. From this visualization, we can see that some high frequent activities start appearing in all the logs only after 2012. Even though the sudden appearance of new activities is a direct and simple new behavior to be easily identified by process analysts, such task is

not as straight forward in a automatic drift analysis. Additionally, van der Ham (HAM, 2015) compared process models obtained before and after that period and found significant differences as well.

Figure 50 – BPIC 2015: Occurrence of the top 10 most frequent activities over time for each municipality. Vertical lines indicate found drifts. Activities that presented the highest MSE in each municipality in localization are highlighted.

Source: Rafael Gaspar de Sousa, 2021

When visually analyzing the curves obtained with our approach, one can raise the question of sensibility when detecting a drift, specially in a real-world event log without supervised labels. There are several variations over time that might be identified by our approach as a drift or not, depending on the sensibility set in the parameters. Potential drifts can be validate by specialists on the process behavior to determine what is considered

or not as a drift. In conclusion, while these experiments supports the hypothesis of a process drift in the log, the results should be validated with domain experts to definitive answers.

4.2.2 BPIC 2020

This set of event logs consists of data about travel expense claims from the Eindhoven University of Technology (TU/e) [Dongen \(2020\)](#) from 2017 to 2018 collected for several university departments. The five logs that compose this set follow a similar process flow. In terms of size, the *Domestic declarations* log has around 10,000 cases and 55,000 events, while *International declarations*, *Permit Log* and *Requests for payment* (expenses not related to travels, but to hardware purchases) have around 6,000 cases and 62,000 events each. Finally, *Pre-paid travel costs* is the smaller log with 2,007 cases and 17,512 events.

The process is described in [Dongen \(2020\)](#) as the procedures required when arranging the reimbursement of costs to university staff. Based on the challenge description, [Bano Maximilian Völker and Weske \(2020\)](#) produced a model to visualize the process flow presented in figure 51. Initially, the request is submitted to be approved in first instance for the travel administration. Next, it follows to request the approval of the budget owner and the supervisor later on if applicable. In some cases, there is also the need of the director's permission. The process ends when the request is approved in all instances and the payment is handled, or when it is rejected if any of the required approvals are not granted.

Figure 51 – BPIC 2020: Described business process model BPI 2020

Source: Adapted from [Bano Maximilian Völker and Weske \(2020\)](#).

We have executed our approach over all logs to analyze for concept drifts using the different clustering strategies, as done with the previous set of event logs in subsection 4.2.1.

In terms of parameters, the *window_size* for the batch and “no clustering” strategies was set in 225 for all logs except for *Pre-paid travel costs* which was set at 150 since it is an smaller log. The *CluStream* parameters selected were $h = 200$, $m = 30$, $t = 2$ and $k_macro = 3$. The drift detection parameters were $rw = 3$, $std_tol = 3.5$ and $min_tol = 7.0\%$ for both the batch and “no clustering” strategy, while for the *CluStream* the parameters were $rw = 200$, $std_tol = 3.5$ and $min_tol = 10\%$. Figure 52 shows the results of the detection step for all municipalities considering the *average intraclusters distances* feature over time. Vertical lines indicates drifts detected by each strategy.

Figure 52 – BPIC 2020: Execution of our approach for all five municipalities with different clustering strategies

Source: Rafael Gaspar de Sousa, 2021

The results for each event log are discussed individually in the following topics.

- *Domestic declarations*: There is a growth trend from 2017 to 2018. Also, there is more stability in the curves after 2018 for the three strategies, while in 2017 there is more variance captured by the *CluStream* strategy. Visually, the major change occurs in the beginning of 2018. The batch and “no clustering” strategies both found a drift around that period, while the *CluStream* strategy found three drifts in 2017.
- *International declarations*: In this log a more steady trend is found in all strategies. The batch strategy found a drift at early 2018 when the major change in the feature curve occurs. While the *CluStream* strategy did not detect any drift, the “no clustering” strategy three drifts.

- *Permit log*: Similar to the *International declarations* event log, this log resulted in a more a stable trend overall, although a strong difference in early 2018 is found when using the batch strategy. Two other drifts were found with the “no clustering” strategy in middle 2018.
- *Pre paid travel cost*: In this event log, there is a growth trend presented by the feature curves, where considerable changes can be identified in the beginning of 2018, when the process behavior starts to stabilize according to the *CluStream* strategy. A drift is found in early 2018 by the batch strategy and an additional drift is found next in April 2018 by the “no clustering” strategy.
- *Request for payment*: Similar to the trend presented in *Domestic declarations*, the most intense changes occurred around the change of the year 2017 to 2018. The three clustering strategies found a drift around that same period of year change. The *CluStream* strategy found additional drifts in the middle of 2017 in a less stable period of the process behavior.

On the different clustering strategies, drifts have been found during 2017 by the *CluStream* strategy in both *Domestic declarations* and *Request for payment*, while have not shown significant differences in the other three logs. On the other hand, using the batch approach we have identified one drift in the first month of 2018 in every log except in *International Declarations*, in which it was detected early in the fourth month. The “no clustering” strategy also found drifts in the first third of 2018 in all logs, but additional drifts in October of 2017 and July of 2018 in the *International Declarations* and *Permit log* event logs.

Identical to the situation encountered in the previous real set of event logs in subsection 4.2.1, there is not a conclusive answer provided by domain specialists concerning the presence of drifts. However, the study in [Bano Maximilian Völker and Weske \(2020\)](#) also found supportive evidence of changes in the process from 2017 to 2018. As speculative reasoning, they mention a possible digitalization of the process that required all cases that were filled by hand to be filled digitally and capture by the event logs, as there is a great increase in the number of cases in that period as well. Additionally, the dataset description in [Dongen \(2020\)](#) mentions that during 2017 the data was collected from only two departments and the rest of departments were only included in 2018.

To further comprehend the drifts found by our approach, we have performed the localization of the first drift found with the batch strategy at early 2018. Figure 53 shows the results obtained in the localization step for each log.

Figure 53 – BPIC 2020: Results of drift localization for the first drift found in all event logs with the batch strategy

Source: Rafael Gaspar de Sousa, 2021

The role of the Administration is present in the most involved activities in the logs *Permit Log*, *Request for Payment* and *Pre-paid travel cost*, while in the other logs there is higher influence of the Supervisor, Employee and Pre Approver. Additionally, though the process in each log is similar, the activities with higher involvement in drift is not that common between the processes, due to particularities of each subject.

To correlate these findings with the log itself, figure 54 shows a heatmap with the occurrence of the top 15 activities in a rolling window of 50 traces. The idea is to provide a alternative to visually how frequent an activity is in the traces over time. Darker values in the illustration indicate higher frequency, while lighter values indicate lower presence of the activity. Vertical lines highlight drifts found.

Over all event logs, it is clear that there are a set of activities that started occurring only after 2018, such as *Declaration_APPROVED_by_BUDGET_OWNER* in the *Domestic declarations* and *Permit_APPROVED_by_BUDGET_OWNER* in *International declarations*. Other activities, however, began to stop occurring in 2018, such as *Request_For_Pay-*

Figure 54 – BPIC 2020: occurrence of the top twelve most frequent activities over time for each municipality. Vertical lines indicate drifts found and highlighted activities are the ones with highest MSE in the localization task.

Source: Rafael Gaspar de Sousa, 2021

ment_APPROVED_by_PRE_APPROVER and *Permit_APPROVED_by_PRE_APPROVER* in *Pre-paid travel cost*.

The logs *Pre-paid travel cost* and *International declarations* show stronger behavior both in activities that disappear and activities that appear in 2018, leading their drift localization results (figure 53) to have a greater number of activities with high values of MSE. In the other logs, the stronger behavior is related only to the appearing of activities, leading to fewer activities with high values of MSE in the localization.

We can also understand other drifts, such as the second drift detected by the “no clustering” strategy in the *International declarations* event log, in late March 2018. Right before the drift, during the middle of the month, there is a uncommon and abrupt

absence of at least 4 activities (*Permit_APPROVED_by_BUDGET_OWNER* and *Permit_APPROVED_by_ADMINISTRATION* for example).

Although we can related our drift detection with changes in activities and traces frequency, there is yet a lack of domain expertise to confirm those changes. Our findings corroborate to the analysis in [Bano Maximilian Völker and Weske \(2020\)](#) of a possible concept drift between 2017 and 2018, and the event log description in [Dongen \(2020\)](#) which mentions an increase in the breadth of data since they started to log the remaining departments in 2018.

4.3 Summary

Experiments have been carried out to test the proposed approach in synthetic event logs with different change patterns and clustering strategies. We discussed the effect of parameters and analyzed the impact of different parameters and trace representations on the detection results. When comparing to a baseline, our approach failed to consistently present better or equivalent results when using a fixed configuration of parameters, but it has shown potential when adjusting the parameters to change pattern being investigated, even in event logs with high trace variability and complex processes.

Our approach using batch strategy and activity-frequency trace representation when using fixed parameters, outperforms the baseline in three and matches the performance in two out of the total seventeen change patterns for the Loan Applications set of event logs. When considering the optimized parameters for each change pattern, our approach exceeds or matches the performance in twelve out of the seventeen change patterns with the batch strategy and activity-order trace representation.

In terms of drift localization, we have shown the ability of providing information on the activities involved in the drift with our drift localization approach using the same clusters obtained in the drift detection step. In our experiments, our approach presented an high performance when indicating only the correct activities involved with higher intensity.

Although the results in our experiments with real-world event logs can only be confirmed in a supervised manner with the assistance of domain experts, we showed that our approach is suitable for such cases, being able to provide some insights and possible concept drifts.

5 Final considerations

In this document, we presented a research project in the process mining and clustering fields. Specifically, we proposed a study to investigate the usage of trace clustering in concept drift detection and localization in process mining. To close this document, this chapter discuss the following matters: limits in section 5.1; contributions in section 5.2; and possible avenues for further research in section 5.3.

5.1 Limits

The drift detection and localization methods proposed in this project rely on the comparison of clustering situations. That is possible by assuming both situations are bound to the same dimensional space. In the experiments, we have assumed to know *a priori* all the dimensions involved in the log (i.e. knowing all the possible activities and transitions present in the event log beforehand). Although that is a limitation of this work, knowing at least all possible activities is feasible in many occasions in a real-world business process.

Our approach is subject to the trace representation and clustering algorithms. As previously mentioned in subsection 2.5.2, each trace representation is only capable of characterizing a subset of change patterns. As for the clustering algorithms, each of them have they own limitations (e.g. only identifying spherical clusters, in case of *k-means* based on euclidean distances) that influence the results of our approach.

5.2 Contributions

The main expected contribution in this project is a method able to perform drift detection and localization online based on the same data structure obtained by applying trace clustering online. The representation obtained by trace clustering is expected to reflect the changes in a process over time, hence providing sufficient input to perform both drift detection and localization, each with its different mechanism, by processing traces only once and with no need for additional data structures or high computational cost.

The partial results obtained in this project were published at an international conference. The full reference for the correlated paper is:

- SOUSA, R. G. de; PERES, S. M.; FANTINATO, M.; REIJERS, H. A. Concept drift detection and localization in process mining: An integrated and efficient approach enabled by trace clustering. In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2021. (SAC '21), p. 364–373. ISBN 9781450381048. DOI:<https://doi.org/10.1145/3412841.3441918>.

5.3 Future work

An optional pre-processing step might be applied to group frequent segments able to simplify event logs, as presented in [Bose \(2012\)](#), hence resulting in lower dimensions when transforming traces into vectors. In terms of trace representations, we suggest to explore different techniques in order to capture more aspects of the process behavior, such as embeddings ([KONINCK; BROUCKE; WEERDT, 2018](#)). Based on the trace clustering approach in [Appice and Malerba \(2016\)](#) that showed improvement in trace clustering when using a multi-view clustering approach, we also recommend to extend this research by combining information from different representations in order to improve clustering and thus our detection and localization approach.

In the clustering step, density-based streaming clustering algorithms could be explored, as done in [Tavares et al. \(2019\)](#), as well as novel clustering algorithms. For the concept drift localization task, a noise-reducing strategy can be developed in order to discard small fluctuations in the process behavior that can interfere in the results. Also, the localization results can be used to improve the drift detection results by considering the intensity of the MSEs obtained when a drift is found and avoid false positives.

Another important avenue for future work is to pursue methods for automatic parameters selection, as our approach performance is influenced to a number of parameters, such as the size of windows, number of clusters or standard deviation tolerance when dealing with drifts.

Bibliography

- AALST, W. M. P. van der; PESIC, M.; SCHONENBERG, H. Declarative workflows: Balancing between flexibility and support. *Computer Science - Research and Development*, v. 23, n. 2, p. 99–113, May 2009. Referenced at page 54.
- AALST, W. van der. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. 1st. ed. [S.l.: s.n.], 2011. 352 p. Referenced 2 times at pages 20 and 27.
- AALST, W. van der. Process mining in the large: A tutorial. In: *3rd Eur. Summer Sch. on Bus. Intell.* [S.l.: s.n.], 2014. p. 33–76. Referenced at page 19.
- AALST, W. van der. *Process Mining: Data Science in Action*. 2nd. ed. [S.l.: s.n.], 2016. Referenced 4 times at pages 21, 27, 28, and 43.
- ACCORSI, R.; STOCKER, T. Discovering workflow changes with time-based trace clustering. In: *2nd Int. Symp. on Data-driven Proc. Discov. and Anal.* [S.l.: s.n.], 2012. p. 154–168. Referenced 5 times at pages 24, 32, 52, 57, and 58.
- AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for clustering evolving data streams. In: *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*. [S.l.]: VLDB Endowment, 2003. (VLDB '03), p. 81–92. ISBN 0127224424. Referenced at page 41.
- APPICE, A.; MALERBA, D. A co-training strategy for multiple view clustering in process mining. *IEEE Trans. Serv. Comput.*, v. 9, n. 6, p. 832–845, 2016. Referenced 4 times at pages 31, 45, 47, and 126.
- ARMAS-CERVANTES, A.; BALDAN, P.; DUMAS, M.; GARCIA-BAÑUELOS, L. Diagnosing behavioral differences between business process models. *Inf. Syst.*, v. 56, n. C, p. 304–325, 2016. ISSN 0306-4379. Referenced 4 times at pages 20, 33, 55, and 59.
- ARTHUR, D.; VASSILVITSKII, S. K-means++: The advantages of careful seeding. In: *18th Annual ACM-SIAM Symp. on Discrete Algorithms*. [S.l.: s.n.], 2007. (SODA '07), p. 1027–1035. Referenced 2 times at pages 39 and 74.
- BANO MAXIMILIAN VÖLKER, S. R. H. L. D.; WESKE, M. Multi-perspective analysis of approval processes based on multiple event logs. In: *Tenth International Business Process Intelligence Challenge (BPIC'20)*. [S.l.: s.n.], 2020. Referenced 3 times at pages 119, 121, and 124.
- BEEST, N. R. T. P. van; DUMAS, M.; GARCÍA-BAÑUELOS, L.; ROSA, M. L. Log delta analysis: Interpretable differencing of business process event logs. In: *Bus. Process Manage.* [S.l.: s.n.], 2015. p. 386–405. Referenced 3 times at pages 33, 55, and 59.
- BOLT, A.; LEONI, M. de; AALST, W. van der. Process variant comparison: Using event logs to detect differences in behavior and business rules. *Inf. Syst.*, v. 74, p. 53 – 66, 2018. Referenced 3 times at pages 33, 54, and 59.
- BOSE, R. J. C. *Process mining in the large: preprocessing, discovery, and diagnostics*. Thesis (Doctorate) — Department of Mathematics and Computer Science, 2012. Proefschrift. Referenced 3 times at pages 21, 44, and 126.

- BOSE, R. J. C.; AALST, W. van der. Context aware trace clustering: Towards improving process mining results. In: *SIAM Int. Conf. on Data Mining*. [S.l.: s.n.], 2009. p. 401–412. Referenced 5 times at pages [21](#), [24](#), [32](#), [43](#), and [44](#).
- BOSE, R. P. J. C.; AALST, W. van der; ŽLIOBAITÉ, I.; PECHENIZKIY, M. Handling concept drift in process mining. In: *23rd Int. Conf. on Adv. Inf. Syst. Eng.* [S.l.: s.n.], 2011. p. 391–405. Referenced 6 times at pages [19](#), [20](#), [51](#), [54](#), [57](#), and [58](#).
- BOSE, R. P. J. C.; AALST, W. van der; ŽLIOBAITÉ, I.; PECHENIZKIY, M. Dealing with concept drifts in process mining. *IEEE Trans. on Neural Networks and Learning Syst.*, v. 25, n. 1, p. 154–171, 2014. Referenced 8 times at pages [31](#), [33](#), [34](#), [35](#), [50](#), [51](#), [57](#), and [58](#).
- BROCKHOFF, T.; UYSAL, M. S.; AALST, W. M. van der. Time-aware concept drift detection using the earth mover's distance. In: *2020 2nd International Conference on Process Mining (ICPM)*. [S.l.: s.n.], 2020. p. 33–40. Referenced 3 times at pages [54](#), [57](#), and [58](#).
- CALI, C.; LONGOBARDI, M. Some mathematical properties of the roc curve and their applications. *Ricerche di Matematica*, v. 64, 2015. Referenced at page [34](#).
- CARMONA, J.; GAVALDÁ, R. Online techniques for dealing with concept drift in process mining. In: *11th Int. Symp. Adv. in Intel. Data Anal.* [S.l.: s.n.], 2012. p. 90–102. Referenced 6 times at pages [20](#), [32](#), [53](#), [54](#), [57](#), and [58](#).
- CARNEIN, M.; TRAUTMANN, H. Optimizing data stream representation: An extensive survey on stream clustering algorithms. *Bus. & Inf. Syst. Eng.*, v. 61, p. 277–297, 2019. Referenced 4 times at pages [39](#), [40](#), [41](#), and [63](#).
- DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. *IEEE Trans. on Pattern Analysis and Machine Intell.*, PAMI-1, n. 2, p. 224–227, 1979. Referenced 2 times at pages [37](#) and [38](#).
- DONGEN, B. van. *BPI Challenge 2020*. [S.l.]: 4TU.ResearchData, 2020. Referenced 3 times at pages [119](#), [121](#), and [124](#).
- DUMAS, M.; ROSA, M. L.; MENDLING, J.; REIJERS, H. A. *Fundamentals of Business Process Management*. [S.l.: s.n.], 2013. Referenced 2 times at pages [27](#) and [28](#).
- FARIA, E. R.; GAMA, J. a.; CARVALHO, A. C. P. L. F. Novelty detection algorithm for data streams multi-class problems. In: *28th Annual ACM Symp. on Appl. Comput.* [S.l.: s.n.], 2013. (SAC '13), p. 795–800. Referenced at page [43](#).
- GAMA, J. *Knowledge discovery from data streams*. [S.l.: s.n.], 2010. Referenced at page [39](#).
- GAMA, J.; ŽLIOBAITÉ, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. *ACM Comput. Surv.*, v. 46, n. 4, p. 44:1–44:37, 2014. Referenced 3 times at pages [22](#), [30](#), and [39](#).
- GARCIA, C. dos S.; MEINCHEIM, A.; JUNIOR, E. R. F.; DALLAGASSA, M. R.; SATO, D. M. V.; CARVALHO, D. R.; SANTOS, E. A. P.; SCALABRIN, E. E. Process mining techniques and applications – a systematic mapping study. *Expert Syst. Appl.*, v. 133, p. 260 – 295, 2019. ISSN 0957-4174. Referenced at page [23](#).

- GARCIA, K. D.; POEL, M.; KOK, J. N.; CARVALHO, A. C. P. L. F. de. Online clustering for novelty detection and concept drift in data streams. In: *Progr. in Artif. Intell.* [S.l.: s.n.], 2019. p. 448–459. Referenced 2 times at pages [24](#) and [43](#).
- GONG, S.; ZHANG, Y.; YU, G. Clustering stream data by exploring the evolution of density mountain. *Proc. VLDB Endow.*, v. 11, n. 4, p. 393–405, 2017. ISSN 2150-8097. Referenced at page [64](#).
- GRECO, G.; GUZZO, A.; PONTIERI, L.; SACCÀ, D. Discovering expressive process models by clustering log traces. *IEEE Trans. on Knowl. and Data Eng.*, v. 18, p. 1010–1027, 2006. Referenced at page [24](#).
- HAM, U. van der. Benchmarking of five dutch municipalities with process mining techniques reveals opportunities for improvement. In: *Fifth International Business Process Intelligence Challenge (BPIC'15)*. [S.l.: s.n.], 2015. Referenced 2 times at pages [117](#) and [118](#).
- HAN, J.; KAMBER, M.; PEI, J. *Data Mining Concepts and Techniques*. 3rd. ed. [S.l.: s.n.], 2012. Referenced 4 times at pages [35](#), [36](#), [37](#), and [66](#).
- HOMPES, B. F. A.; BUIJS, J.; van der Aalst, W.; DIXIT, P.; BUURMAN, J. Detecting change in processes using comparative trace clustering. In: *5th Int. Symp. on Data-driven Proc. Discov. and Anal.* [S.l.: s.n.], 2015. p. 95–108. Referenced 5 times at pages [24](#), [32](#), [45](#), [52](#), and [57](#).
- IMPEDOVO, A.; MIGNONE, P.; LOGLISCI, C.; CECI, M. Simultaneous process drift detection and characterization with pattern-based change detectors. In: *Discovery Science*. Cham: Springer International Publishing, 2020. p. 451–467. Referenced 5 times at pages [53](#), [55](#), [57](#), [58](#), and [59](#).
- IWASHITA, A. S.; PAPA, J. P. An overview on concept drift learning. *IEEE Access*, v. 7, p. 1532–1547, 2019. Referenced 2 times at pages [24](#) and [30](#).
- JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: A review. *ACM Comput. Surv.*, v. 31, n. 3, p. 264–323, 1999. Referenced 2 times at pages [36](#) and [37](#).
- JUNIOR., P. M. G.; SANTOS, S. G. de C.; BARROS, R. S.; VIEIRA, D. C. A comparative study on concept drift detectors. *Expert Syst. Appl.*, v. 41, n. 18, p. 8144 – 8156, 2014. Referenced 2 times at pages [19](#) and [30](#).
- JUNIOR, S. B.; TAVARES, G. M.; COSTA, V. G. T. d.; CERAVOLO, P.; DAMIANI, E. A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. In: *Companion The Web Conf. 2018*. [S.l.: s.n.], 2018. p. 319–326. Referenced at page [24](#).
- KONINCK, P. D.; BROUCKE, S. vanden; WEERDT, J. D. act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes. In: *Bus. Process Manage.* [S.l.: s.n.], 2018. p. 305–321. Referenced 3 times at pages [45](#), [47](#), and [126](#).
- KONINCK, P. D.; WEERDT, J.; BROUCKE, S. vanden. Explaining clusterings of process instances. *Data Mining and Knowl. Discovery*, v. 31, p. 1–35, 2016. Referenced at page [45](#).

- KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, v. 2, n. 1-2, p. 83–97, 1955. Referenced 2 times at pages 65 and 72.
- KUMAR, M.; THOMAS, L.; BASAVA, A. Capturing the sudden concept drift in process mining. In: *Int. Works. on Algor. Theories for the Anal. of Event Data*. [S.l.: s.n.], 2015. p. 132–143. Referenced 5 times at pages 32, 51, 52, 57, and 58.
- KURNIATI, A.; MCINERNEY, C.; ZUCKER, K.; HALL, G.; HOGG, D.; JOHNSON, O. *A Multi-level Approach for Identifying Process Change in Cancer Pathways*. 2020. Referenced at page 34.
- LEEMANS, S. J. J.; FAHLAND, D.; AALST, W. M. P. van der. Discovering block-structured process models from event logs - a constructive approach. In: COLOM, J.-M.; DESEL, J. (Ed.). *Application and Theory of Petri Nets and Concurrency*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 311–329. Referenced at page 88.
- LI, T.; HE, T.; WANG, Z.; ZHANG, Y.; CHU, D. Unraveling process evolution by handling concept drifts in process mining. In: *IEEE Int. Conf. on Serv. Comput.* [S.l.: s.n.], 2017. p. 442–449. Referenced at page 34.
- LIU, N.; HUANG, J.; CUI, L. A framework for online process concept drift detection from event streams. In: *IEEE Int. Conf. on Serv. Comput.* [S.l.: s.n.], 2018. p. 105–112. Referenced 5 times at pages 53, 55, 57, 58, and 59.
- LU, X. *Using behavioral context in process mining: exploration, preprocessing and analysis of event data*. Thesis (Doctorate) — Department of Mathematics and Computer Science, 2018. Referenced 2 times at pages 44 and 45.
- LU, Y.; CHEN, Q.; POON, S. K. A robust and accurate approach to detect process drifts from event streams. *CoRR*, abs/2103.10749, 2021. Referenced 3 times at pages 52, 57, and 58.
- LUENGO, D.; SEPÚLVEDA, M. Applying clustering in process mining to find different versions of a business process that changes over time. In: *Bus. Process Manage. Workshops*. [S.l.: s.n.], 2012. p. 153–158. Referenced at page 24.
- MAARADJI, A.; DUMAS, M.; ROSA, M. L.; OSTOVAR, A. *Business Process Drift*. 2015. Dataset, Queensland Univ. of Technology, Australia. Referenced 2 times at pages 75 and 76.
- MAARADJI, A.; DUMAS, M.; ROSA, M. L.; OSTOVAR, A. Fast and accurate business process drift detection. In: *Bus. Process Manage.* [S.l.: s.n.], 2015. p. 406–422. Referenced 13 times at pages 14, 20, 31, 32, 33, 51, 52, 57, 58, 77, 92, 95, and 96.
- MAARADJI, A.; DUMAS, M.; ROSA, M. L.; OSTOVAR, A. Detecting sudden and gradual drifts in business processes from execution traces. *IEEE Trans. Knowl. Data Eng.*, v. 29, n. 10, p. 2140–2154, 2017. Referenced 3 times at pages 52, 57, and 58.
- MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: *In 5th Berkeley Symposium on Mathematical Statistics and Probability*. [S.l.: s.n.], 1967. p. 281–297. Referenced at page 38.

- MAITA, A. R. C.; MARTINS, L.; PAZ, C. R. L.; RAFFERTY, L.; HUNG, P. C. K.; PERES, S. M.; FANTINATO, M. A systematic mapping study of process mining. *Enterprise Inf. Syst.*, v. 12, p. 1–45, 2017. Referenced 3 times at pages 21, 23, and 43.
- MARTJUSHEV, J.; BOSE, R. P. J. C.; AALST, W. van der. Change point detection and dealing with gradual and multi-order dynamics in process mining. In: *Perspectives in Bus. Informatics Research*. [S.l.: s.n.], 2015. p. 161–178. Referenced 3 times at pages 51, 57, and 58.
- NGUYEN, H.; DUMAS, M.; ROSA, M. L.; HOFSTEDE, A. H. M. ter. Multi-perspective comparison of business process variants based on event logs. In: *Conceptual Modeling*. [S.l.: s.n.], 2018. p. 449–459. Referenced 2 times at pages 55 and 59.
- NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. *Knowl. and Inf. Syst.*, v. 45, n. 3, p. 535–569, 2015. Referenced at page 40.
- OLIVEIRA, M.; GAMA, J. Mec - monitoring clusters' transitions. In: *5th Starting AI Researchers' Symp.* [S.l.: s.n.], 2010. p. 212–224. Referenced 2 times at pages 64 and 65.
- OSTOVAR, A. *Business process drift: Detection and characterization*. Thesis (Doctorate) — Queensland University of Technology, 2019. Referenced 4 times at pages 20, 21, 32, and 50.
- OSTOVAR, A.; LEEMANS, S. J. J.; ROSA, M. L. Robust drift characterization from event streams of business processes. *ACM Trans. Knowl. Discov. Data*, v. 14, n. 3, 2020. ISSN 1556-4681. Referenced 6 times at pages 33, 56, 59, 75, 104, and 105.
- OSTOVAR, A.; MAARADJI, A.; ROSA, M. L.; HOFSTEDE, A. H. ter; DONGEN, B. van. Detecting drift from event streams of unpredictable business processes. In: *35th Int. Conf. on Conceptual Model.* [S.l.: s.n.], 2016. Referenced 3 times at pages 52, 57, and 58.
- OSTOVAR, A.; MAARADJI, A.; ROSA, M. L.; HOFSTEDE, A. ter. Characterizing drift from event streams of business processes. In: *29th Int. Conf. on Adv. Inf. Syst. Eng.* [S.l.: s.n.], 2017. p. 210–228. Referenced 3 times at pages 33, 56, and 59.
- PAUWELS, S.; CALDERS, T. Detecting and explaining drifts in yearly grant applications. *CoRR*, abs/1809.05650, 2018. Referenced 2 times at pages 55 and 59.
- PEREIRA, C. M. M.; MELLO, R. F. de. A comparison of clustering algorithms for data streams. In: HRUSCHKA, E. R.; WATADA, J.; NICOLETTI, M. do C. (Ed.). *Integrated Computing Technology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 59–74. Referenced 2 times at pages 41 and 42.
- RODRIGUEZ, M. Z.; COMIN, C. H.; CASANOVA, D.; BRUNO, O. M.; AMANCIO, D. R.; COSTA, L. d. F.; RODRIGUES, F. A. Clustering algorithms: A comparative approach. *PLOS ONE*, v. 14, p. 1–34, 2019. Referenced at page 37.
- ROUSSEEUW, P. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *comput. appl. math.* 20, 53–65. *Journal of Computational and Applied Mathematics*, v. 20, p. 53–65, 1987. Referenced 2 times at pages 37 and 38.
- SEELIGER, A.; NOLLE, T.; MÜHLHÄUSER, M. Detecting concept drift in processes using graph metrics on process graphs. In: *9th Conf. on Subject-Oriented Bus. Process Manage.* [S.l.: s.n.], 2017. Referenced 2 times at pages 19 and 33.

- SEELIGER, A.; NOLLE, T.; MÜHLHÄUSER, M. Finding structure in the unstructured: Hybrid feature set clustering for process discovery. In: *Bus. Process Manage. - 16th Int. Conf., BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings*. [S.l.: s.n.], 2018. (Lecture Notes in Computer Science, v. 11080), p. 288–304. Referenced at page [24](#).
- SETHI, T. S.; KANTARDZIC, M. On the reliable detection of concept drift from streaming unlabeled data. *Expert Syst. Appl.*, v. 82, p. 77 – 99, 2017. Referenced 2 times at pages [24](#) and [42](#).
- SETHI, T. S.; KANTARDZIC, M.; HU, H. A grid density based framework for classifying streaming data in the presence of concept drift. *Journal of Intelligent Inf. Syst.*, v. 46, n. 1, p. 179–211, 2016. Referenced at page [43](#).
- SILVA, J.; FARIA, E.; BARROS, R.; HRUSCHKA, E.; CARVALHO, A. de; GAMA, J. Data stream clustering: A survey. *ACM Comput. Surv.*, v. 46, 2014. Referenced at page [22](#).
- SONG, M.; GÜNTHER, C. W.; AALST, W. van der. Trace clustering in process mining. In: *4th Workshop on Bus. Process Intell.* [S.l.: s.n.], 2008. p. 51–62. Referenced 2 times at pages [24](#) and [45](#).
- SPILIOPOULOU, M.; NTOUTSI, I.; THEODORIDIS, Y.; SCHULT, R. Monic: Modeling and monitoring cluster transitions. In: *12th ACM SIGKDD Int. Conf. on Knowl. Discovery and Data Mining*. [S.l.: s.n.], 2006. p. 706–711. Referenced 2 times at pages [64](#) and [65](#).
- TANCA, L.; AZZALINI, D.; AZZALINI, F.; MAZURAN, M. Tracking the evolution of financial time series clusters. In: *5th Workshop on Data Science for Macro-Modeling with Financial and Economic Datasets*. [S.l.: s.n.], 2019. (DSMM'19). Referenced at page [65](#).
- TAVARES, G. M.; CERAVOLO, P.; COSTA, V. G. T. D.; DAMIANI, E.; JUNIOR, S. B. Overlapping analytic stages in online process mining. In: *2019 IEEE International Conference on Services Computing (SCC)*. [S.l.: s.n.], 2019. p. 167–175. Referenced 5 times at pages [24](#), [53](#), [57](#), [58](#), and [126](#).
- TEINEMAA, I.; LEONTJEVA, A.; MASING, K.-O. BPIC 2015: Diagnostics of building permit application process in dutch municipalities. In: *Fifth International Business Process Intelligence Challenge (BPIC'15)*. [S.l.: s.n.], 2015. Referenced 3 times at pages [114](#), [115](#), and [117](#).
- THARWAT, A. Classification assessment methods. *Appl. Comput. and Inf.*, 2018. Referenced at page [33](#).
- TSYMBAL, A. The problem of concept drift: Definitions and related work. *Computer Science Department, Trinity College Dublin*, v. 106, p. 58, 2004. Referenced at page [30](#).
- VALLIM, R. M.; FILHO, J. A. A.; MELLO, R. F. de; CARVALHO, A. C. P. L. F. de; GAMA, J. Unsupervised density-based behavior change detection in data streams. *Intelligent Data Analysis*, v. 18, p. 181–201, 2014. Referenced at page [42](#).
- WANG, S.; SCHLOBACH, S.; KLEIN, M. What is concept drift and how to measure it? In: *Knowl. Eng. and Manage. by the Masses*. [S.l.: s.n.], 2010. p. 241–256. Referenced at page [29](#).

- WAZLAWICK, R. S. *Metodologia de Pesquisa para Ciência da Computação*. [S.l.: s.n.], 2017. Referenced at page [25](#).
- WEBB, G. I.; HYDE, R.; CAO, H.; NGUYEN, H. L.; PETITJEAN, F. Characterizing concept drift. *Data Min. Knowl. Discov.*, v. 30, n. 4, p. 964–994, 2016. Referenced 2 times at pages [30](#) and [50](#).
- WEBER, B.; REICHERT, M.; RINDERLE-MA, S. Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowl. Eng.*, v. 66, n. 3, p. 438–466, 2008. Referenced at page [31](#).
- WEERDT, J.; BROUCKE, S. vanden; VANTHIENEN, J.; BAESSENS, B. Active trace clustering for improved process discovery. *IEEE Trans. Knowl. Data Eng.*, v. 25, p. 2708–2720, 2013. Referenced at page [24](#).
- WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, v. 23, n. 1, p. 69–101, 1996. Referenced at page [29](#).
- XU, D.; TIAN, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science*, v. 2, p. 165–193, 2015. Referenced at page [37](#).
- YESHCENKO, A.; CICCIO, C. D.; MENDLING, J.; POLYVYANYYY, A. Visual drift detection for sequence data analysis of business processes. *IEEE Transactions on Visualization Computer Graphics*, IEEE Computer Society, Los Alamitos, CA, USA, n. 01, p. 1–1, jan 2021. Referenced 6 times at pages [24](#), [54](#), [55](#), [57](#), [58](#), and [59](#).
- YUAN, C.; YANG, H. Research on k-value selection method of k-means clustering algorithm. *J*, v. 2, n. 2, p. 226–235, 2019. ISSN 2571-8800. Referenced 2 times at pages [79](#) and [87](#).
- ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, Association for Computing Machinery, New York, NY, USA, v. 25, n. 2, p. 103–114, jun. 1996. ISSN 0163-5808. Referenced at page [41](#).
- ZHENG, C.; WEN, L.; WANG, J. Detecting process concept drifts from event logs. In: *Int. Conf. on Coop. Inf. Sys.* [S.l.: s.n.], 2017. p. 524–542. Referenced 5 times at pages [20](#), [24](#), [53](#), [57](#), and [58](#).
- ŽLIOBAITÉ, I.; PECHENIZKIY, M.; GAMA, J. An overview of concept drift applications. In: *Big Data Analysis*. [S.l.: s.n.], 2016. v. 16, p. 91–114. Referenced 2 times at pages [19](#) and [29](#).