



Utrecht University

Department of Information and Computing Sciences

# Hybrid Trace Clustering

by

Mehrad Abdollahi

*A thesis submitted in fulfilment of the requirements for the  
degree of Master of Science*

*in*

Business Informatics

*Supervisors:*

First Supervisor: Dr. ir. Xixi Lu

Second Supervisor: Dr. ir. Jan Martijn E. M. van der Werf

March 2021

## Abstract

Process mining is a relatively young analytical discipline that is used as a bridge between data mining and business process management. Experts have been using process mining to extract insights about how processes work in real life, how many deviations they have compared to their anticipations, and how these processes can be improved. However, when it comes to large and complex data, discovered process models by process mining algorithms might be quite complicated. In such cases, called the discovery of spaghetti-like models, one cannot simply understand the required knowledge from a model. One possible approach to avoid this type of models is to cluster traces that share homogeneous behaviors. While existing approaches offer promising results, each of them suffers from a drawback, including having high computational complexity, not producing high quality models, and not explaining the existence of irrelevant traces in some clusters to name a few. This thesis aims to hybridize two trace clustering types by introducing an algorithm that makes a balance between the quality of cluster models and run time of the algorithm. Evaluation of our technique (Hybrid) on six real-life event logs show meaningful improvements against applying similarity-based or model-driven techniques individually in terms of quality of process models. The obtained results of performance and scalability evaluation also reveal that Hybrid technique delivers clusters on average with lower running times compared to a state-of-the-art model-driven technique.

**Keywords:** Trace clustering, Process mining, Process Discovery, Data Mining

## **Acknowledgements**

This thesis is a result of nine months of demanding but also satisfying work. I started working on this thesis at the beginning of the COVID-19 global pandemic, a catastrophe that touched not only my life but the whole world. Doing my master thesis in lockdown and amid the Corona crisis was a big disappointment and frustration for me. In the end, however, I am delighted and satisfied with the knowledge I gained in the context of process mining and output results of the research I did.

I would like to express my deep gratitude to my first supervisor, Xixi Lu, for her thorough and constructive feedbacks, her patience with my problems and questions, and her supportive attitude when times were difficult for me.

Also, I want to give my special thanks to my parents and my brother. I could not have finished my thesis without their endless love, support, and encouragement.

To all my friends who were in contact with me while writing this thesis, thank you. I will never forget your help and support when I needed it the most.

# Contents

<b>List of Figures</b>	<b>4</b>
<b>List of Tables</b>	<b>5</b>
<b>1 Introduction</b>	<b>7</b>
1.1 Problem Statement . . . . .	8
1.2 Research Questions . . . . .	9
1.3 Expected Contributions . . . . .	10
1.4 Research Method . . . . .	11
<b>2 Preliminaries</b>	<b>13</b>
2.1 Basic Notations . . . . .	13
2.2 Event Log . . . . .	14
2.3 Process Mining . . . . .	15
2.4 Data Clustering . . . . .	16
2.5 Trace Clustering . . . . .	17
<b>3 Related Literature</b>	<b>19</b>
3.1 Feature vector-based similarity . . . . .	19
3.2 Syntax-based similarity . . . . .	21
3.3 Model-driven . . . . .	21
3.4 Mixed-paradigm . . . . .	22
3.5 Discussion . . . . .	23
<b>4 Hybrid Trace Clustering</b>	<b>25</b>
4.1 Elements . . . . .	25
4.2 Algorithm . . . . .	27
4.2.1 Step 1: Initial Clustering . . . . .	29
4.2.2 Step 2: Trace Re-distribution . . . . .	29
4.2.3 Step 3: New Cluster Initialization . . . . .	30
4.2.4 Step 4: Iterative Clustering . . . . .	30

<i>CONTENTS</i>	3
4.2.5 Step 5: Final Trace Assignment . . . . .	31
<b>5 Implementation</b>	<b>33</b>
<b>6 Evaluation</b>	<b>36</b>
6.1 Experimental Setup . . . . .	37
6.2 SQ1: Distance and clustering technique . . . . .	38
6.2.1 Discussion . . . . .	46
6.3 SQ2: Input Parameter Configuration . . . . .	47
6.3.1 Discussion . . . . .	48
6.4 SQ 3: Comparative Analysis . . . . .	49
6.4.1 Discussion . . . . .	54
6.4.2 Performance Analysis . . . . .	54
6.5 Overall Discussion . . . . .	57
<b>7 Conclusion</b>	<b>60</b>
7.1 Conclusion . . . . .	60
7.2 Limitations . . . . .	61
7.3 Future Work . . . . .	62
<b>Bibliography</b>	<b>63</b>
<b>APPENDICES</b>	<b>67</b>
<b>A Output Process Models</b>	<b>68</b>

# List of Figures

1.1	High Level framework of the Hybrid technique . . . . .	9
1.2	Engineering Cycle . . . . .	12
2.1	Trace Clustering . . . . .	18
4.1	Hybrid technique framework . . . . .	26
5.1	Hybrid technique code architecture . . . . .	34
6.1	The effects of Distance Metric and Clustering Techniques on Weighted Average F1-score. . . . .	39
6.2	Weighted Average F1-score against Clustering technique. . . . .	40
6.3	The effects of flag on Weighted Average F1-score with different Clustering Techniques. . . . .	41
6.4	Effect of Distance metric and Clustering Technique on Run- ning time. . . . .	45
6.5	Effect of 3 input parameters on four real-life logs . . . . .	48
6.6	Weighted Average F1-score against Clustering techniques . . . . .	50
6.7	Scalability of Hybrid technique in terms of the number of trace variants . . . . .	57
6.8	Scalability of Hybrid technique in terms of the number of unique activities . . . . .	58

# List of Tables

2.1	Example of an artificial event log of a hospital . . . . .	15
3.1	Existing Trace Clustering techniques in the literature . . . . .	24
6.1	Selected Real-life event logs . . . . .	38
6.2	Sampled BPI 2012 event logs . . . . .	38
6.3	Initial clusters found by Hybrid’s first phase for KIM event log	42
6.5	Initial clusters found by Hybrid’s first phase for Road Traffic fine event log . . . . .	42
6.4	Process models discovered from KIM event log for 4 clusters using Hybrid technique, grouped by clustering algorithm used	43
6.6	Process models discovered from Road Traffic Fine event log for 4 clusters using Hybrid technique, grouped by clustering algorithm used . . . . .	44
6.7	Process models discovered from BPI 2013 event log for 4 clus- ters by Hybrid technique and ActiTraC . . . . .	52
6.8	Process models discovered from Road Traffic Fine event log for 3 clusters by Hybrid technique and ActiTraC . . . . .	53
6.9	ActiTraC Real-life Event Logs . . . . .	55
6.10	Average runtimes of different techniques in seconds . . . . .	55
6.11	Runtime (in seconds) on artificial logs with 15 unique activities	56
6.12	Runtime (in seconds) on artificial logs with 1000 trace variants	57
A.1	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2013 event log for 4 clusters(first and second clusters) . . . . .	69
A.2	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2013 event log for 4 clusters(third and fourth clusters) . . . . .	70

A.3	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from KIM event log for 4 clusters(first and second clusters) . . . . .	71
A.4	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from KIM event log for 4 clusters(third and fourth clusters) . . . . .	72
A.5	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from Road Traffic Fine event log for 4 clusters(first and second clusters) . . . . .	73
A.6	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from Road Traffic Fine event log for 4 clusters (third and fourth clusters) . . . . .	74
A.7	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from TSL event log for 4 clusters(first and second clusters) . . . . .	75
A.8	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from TSL event log for 4 clusters(third and fourth clusters) . . . . .	76
A.9	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2012 event log for 4 clusters(first and second clusters) . . . . .	77
A.10	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2012 event log for 4 clusters (third and fourth clusters) . . . . .	78
A.11	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2018-Reference event log for 4 clusters (first and second clusters) . . . . .	79
A.12	Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2018-Reference event log for 4 clusters (third and fourth clusters) . . . . .	80



# Chapter 1

## Introduction

With the proliferation of stored data in the twenty-first century, various new disciplines emerged to complement established fields like statistics in translating raw data into valuable insights. Machine learning and data mining as examples of these new disciplines are applied in a wide spectrum of areas, ranging from image and speech recognition to malware filtering and fraud detection. Detection of spam emails is a tangible and daily example of a data mining problem that falls within data clustering. As one of the main tasks of data mining, data clustering groups similar objects together, whereas objects in different clusters are dissimilar. Therefore, a clustering algorithm is able to find spam emails and put them in a separate folder.

*Process-aware information systems (PAIS)* is another domain profiting from data science solutions. *Business Process Management (BPM)*, *Enterprise Resource Planning (ERP)*, *Customer Relationship Management (CRM)* are only a few of the systems that support processes in an organisation. However, unlike other areas that use spreadsheet data, these systems' data relates to processes. In other words, data about a process, so-called *event data*, involves a collection of continuous execution of activities in the flow of time. Since tabular data science tools are process-agnostic, they are not useful for PAIS. This obstacle was overcome by the introduction of process mining.

Process mining is a group of techniques developed to monitor and optimise processes through event data produced by information systems in an organisation. The advent of process mining techniques has opened new opportunities to address gaps between traditional and process-oriented data analysis tools and methods [38].

This thesis aims to address data clustering, one of the classical data sci-

ence problems, in the process mining context. The solution provided is a novel approach that combines two currently available methods in the literature to produce a competitive technique both in terms of scalability and quality. Therefore, in the rest of the current chapter, the research problem is explained, and the respective research questions are discussed. Then the contributions of this thesis and research methods used are reviewed. In chapter 2 and 3, preliminary notations and notions, and related works in the literature are reviewed and discussed, respectively. Chapter 4 explains the Hybrid technique and its algorithm's input in detail. Chapter 5 reviews the implementation of the Hybrid technique, followed by the evaluation discussions in chapter 6. Finally, chapter 7 summarise the findings of this thesis and discusses limitations and future work.

## 1.1 Problem Statement

When it comes to process model discovery from event logs that occurred at different abstraction levels, current discovery algorithms cannot function accurately. To this end, a wide range of clustering algorithms have been proposed in the context of process mining. According to a comparative analysis of trace clustering techniques [37], ActiTraC technique [16] is one of the best performing approaches that offers promising results in terms of quality of resulted process models. In ActiTraC, the focus is on the way traces are distributed among clusters by iteratively measuring the quality of discovered models.

However, this approach faces two main challenges. First, the running time of this approach could grow exponentially in the number of distinct process variants [16]. Second, there could be perfect clusters as well as extremely poor clusters that decreases the overall average F1-score. On the other hand, other approaches that concentrate on finding the similarity between traces sometimes cluster traces that look similar but belong to different process behaviours. The reason for this is that these approaches take process agnostic, data mining techniques that have not been primarily designed for event data.

This thesis proposes a hybrid approach of existing trace clustering methods to address the drawbacks of each of the mentioned techniques. The target methods considered are ActiTraC [16], a model-driven clustering technique and K-means using feature-vector, a similarity-based technique . In the hybrid approach, first a similarity-based clustering technique is applied to find the possible clusters with high levels of fitness and precision. This is done

to reduce the amount of computation in the next steps when model-driven clustering starts. Moreover, unlike ActiTraC [16] that examine trace variants one by one, hybrid techniques take into account the similarity between traces when building new clusters.

Overall, the Hybrid technique is composed of two main components: initial clustering and re-clustering. An initial clustering is first applied on the original event log through data clustering algorithms. Next, traces are re-clustered with respect to input parameters and results of initial clustering. The output of the technique is several event log clusters in which produce less complex process models. A high-level overview of the hybrid technique is shown in Figure 1.1.

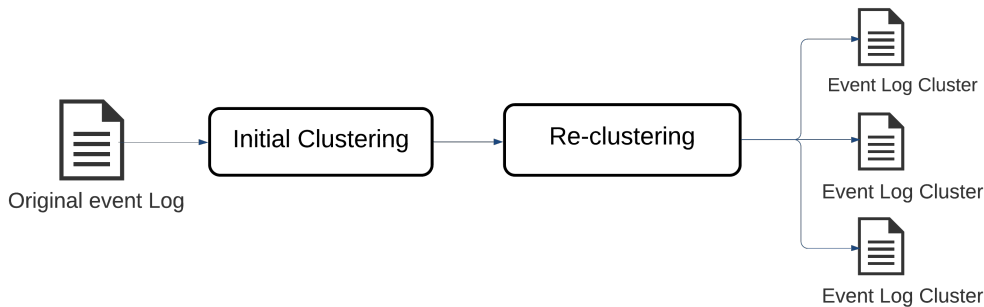


Figure 1.1: High Level framework of the Hybrid technique

## 1.2 Research Questions

While model-driven techniques result in precise models, in each iteration they need to re-discover the model to evaluate the participation of a new trace. Calculating fitness on each iteration for each new trace is a cumbersome task in terms of computational time. In trace similarity techniques, the quality of the clusters is revealed only after the clustering task is finalized. Therefore, two similar traces might be put in the same cluster, while they do not improve the process model. This thesis aims to improve current clustering techniques by offering an algorithm that makes a balance between run time complexity and quality of cluster models.

This thesis is about design and investigation of a hybrid clustering algorithm in the context of process mining. According to [44], research problems

in design science associates with *design problems (DP)* and *knowledge problems (KP)*. Although the main research question (RQ) is a design problem, its sub-questions (SQ) include two design problems and one knowledge problem. The main research question is formulated as follows:

**How to combine similarity-based and model-driven clustering techniques to propose a novel technique that makes a balance between quality of cluster models and run time of the algorithm? (DP)**

This question can be decomposed into the following sub-questions:

- **SQ1:** *Which distance metric and clustering technique result in fastest clustering while not compromising quality of process models? (DP)*

Finding similar traces of a trace can be done in different ways, in which each of them affect the performance of the algorithm. For this thesis, data clustering algorithm and distance metric used could substantially influence both quality of cluster models and runtime of the technique. Therefore, the effect of these two measures is examined individually and collectively for this sub-research question.

- **SQ2:** *What are the effects of different input parameters on the results of clustering? (KP)*

Hybrid algorithm has more input parameters that influence the results of clustering. In this thesis, the effect of different parameter values will be reviewed.

- **SQ3:** *How does the Hybrid technique perform compared to existing approaches? (DP)*

The performance of Hybrid technique is against existing clustering techniques with different clustering types, both in terms of quality of process models and algorithm's running time.

### 1.3 Expected Contributions

In this thesis, a new way of trace clustering is proposed. The effect of distance metrics between traces and the evaluation metrics of the models are investigated in this thesis. Therefore, researchers can use findings of this thesis not only in their works in trace clustering but also in other scopes of

process mining, such as process discovery.

Moreover, one can look at the Hybrid technique as a framework. Input parameters can be configured by the user, more clustering algorithms could be embedded in the technique, and other process model quality criteria can be applied to adapt the results according to the problem requirements.

Finally, the Hybrid technique is implemented in Python, the most popular programming language, according to GitHub and Google trends [33]. The implementation is based on a process-mining compliant Python library, PM4PY, and it is publicly available as a Python package at GitHub.

## 1.4 Research Method

This thesis follows the design science methodology offered by [44]. The primary research question is a design problem, and thus, the engineering cycle is selected as the research framework. Besides, this thesis took an iterative approach towards the engineering cycle. For example, the Hybrid technique algorithm' design and inputs were revised after initial results of first prototypes. Figure 1.2 shows the tasks executed in each four steps of the engineering cycle.

The problem was identified and explored further following a thorough literature study in the problem investigation step. The literature study was conducted using forward\backward snowballing on terms such as *trace clustering*, *process instance clustering*, *clustering in process mining*, *data mining in process mining*, and so forth.

Based on the literature study findings, a novel clustering technique is suggested in the treatment design step. Decisions on Hybrid algorithm design were affected both by deficiencies in similar techniques in literature and results of first versions of algorithm. Input parameters were also modified since the thesis began to have definitive configurations.

For the treatment validation step, several real-life event logs were employed to evaluate the Hybrid technique. These event logs have been used in the literature extensively, and thus, they can be considered benchmark data. The Hybrid technique was also evaluated against other techniques in literature with different input parameters to generalize results. More information concerning evaluation can be found in chapter 6.

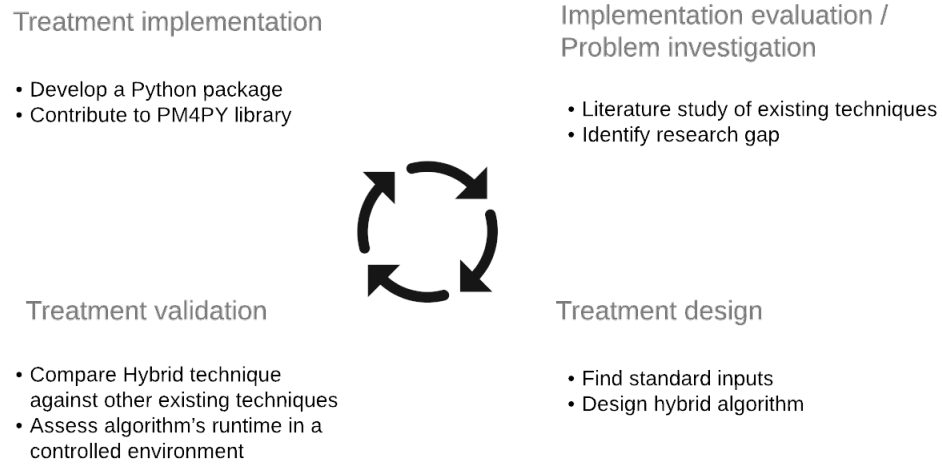


Figure 1.2: Engineering Cycle

Treatment implementation step was conducted in the Python programming language. The Hybrid technique will be added to available open-source process mining Python libraries and can be accessed openly.

# Chapter 2

## Preliminaries

In this chapter, all required concepts and notations used in this thesis are defined formally. Definitions begin with basic mathematical notations which is then followed by event log and process mining related concepts respectively. In the last part, data and trace clustering techniques are introduced.

### 2.1 Basic Notations

**Definition 2.1.1** (Set). A *set* is a distinct collection of elements. The number of elements in a set denotes the size a set. If a set does not have any member, it is called an empty set and is denoted by  $\emptyset$ . Let  $A$  and  $B$  be two sets, then following concepts are defined:

- **Element.**  $x \in A$  denotes  $x$  is an element(member) of the set  $A$
- **Union.**  $A \cup B = \{x : x \in A \vee x \in B\}$  is denoted as the union of sets  $A$  and  $B$
- **Intersection.**  $A \cap B = \{x : x \in A \wedge x \in B\}$  is denoted as the intersection of sets  $A$  and  $B$
- **Subset.**  $A \subset B$  denotes set  $A$  is a subset of  $B$  if  $A \cap B = A$

**Definition 2.1.2** (function).  $f$  is a function from  $A$  to  $B$  that maps an element in  $A$  to an element in  $B$  and is denoted as  $f : A \rightarrow B$

**Definition 2.1.3** (Multiset). A multiset  $m$  over set  $A$  is a collection of elements that unlike a set, each element instance can appear multiple times. It is denoted as a funtion  $m : A \rightarrow \mathbb{N}$  which maps every element to its number of occurrences in the multiset. For example, given multi set  $m = [a_1^3, a_2^2, a_3^4]$  over  $A = \{a_1, a_2, a_3\}$ , then  $m(a_1) = 3$ ,  $m(a_2) = 2$ , and  $m(a_3) = 4$ .

**Definition 2.1.4** (Sequence). A sequence  $\sigma = \langle a_1, a_2, \dots, a_n \rangle$  over the set  $A$  is an enumerated and ordered collection of elements of size  $n$ , where  $a_i \in A$  and  $1 \leq i \leq n$ .

## 2.2 Event Log

Process mining techniques can be leveraged only if pertinent input data is provided. These types of data are recorded by information systems like Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) in the form of so-called *Event logs*. As the name suggests, event logs contain a set of *events* recorded during the execution of a process. Each event denotes an *activity* that has been executed at a certain time by a particular performer and within the context of a particular *case*. A sequence of events that represents an orderly list of the same process instance executions from the start to the end is referred to as *trace*. Each process instance is manifested as a trace and an event log is represented by a multiset of traces.

Table 2.1 shows a fragment of an event log of a healthcare process. Every row in this table outlines an event, and each column represents an *attribute* of an event. The *Case* column indicates to which case or process instance an event belongs to. The *timestamp* column shows the completion execution time of an activity that corresponds to the event. The values of the column *Activity* are associated with the event's activity name and columns *Resource* and *Location* show the event's corresponding performer and their location in the hospital, respectively. It is worth noting that essential requirements for an event log are only the first three columns, namely, *case id*, *activity*, and *timestamp*. If an event log contains these features, it can be used as input for process mining techniques. The remainder of the columns provide additional information that could be missing in other event logs.

**Definition 2.1 (Event, Attribute).** Let  $E$  be the set of all possible event identifiers,  $N$  be the set of all possible attribute names, and  $V$  be the set of all possible attribute values. For any  $e \in E$ ,  $n \in N$ , and  $v \in V$ , mapping function  $\pi_n : E \rightarrow V$  maps event  $e$  to the corresponding value  $v$  for the attribute name  $n$ . As indicated before, the minimum number of attributes for an event in an event log is three. Therefore, if  $e$  does not have attribute  $n$ , then we define it as  $\pi_n(e) = \perp$ .

**Definition 2.2 (Trace).** Let  $\sigma = \langle e_1, e_2, \dots, e_n \rangle \in E^*$  be a finite sequence of events recorded for a process instance (case), where each event appears



Case	Activity	Timestamp	Location	Resource
1	Registration	2020-11-09 10:30	Emergency	Kim
1	Triage request	2020-11-09 10:35	Emergency	Kim
1	Triage	2020-11-09 10:41	Emergency	Bao
3	Surgery	2020-11-09 10:34	Operation room	Ali
2	Admit to hospital	2020-11-09 11:00	Reception	Raj
1	Discharge to home	2020-11-09 10:52	Emergency	Kim
2	Blood test	2020-11-09 11:30	Laboratory	Pete
2	MRI	2020-11-09 12:00	Operation room	Bao
3	Doctor appointment	2020-11-09 13:00	Ward	Ali
3	Registration	2020-11-09 13:30	ICU	Raj
4	kidney function test	2020-11-09 12:45	Laboratory	Pete
...	...	...	...	...

Table 2.1: Example of an artificial event log of a hospital

only once.  $\sigma$  is called a trace if for any  $i$  and  $j$  where  $1 \leq i < j \leq n$ ,  $\pi_{timestamp}(e_i) \leq \pi_{timestamp}(e_j)$ . In other words, the ordering in a trace should respect the timestamp attribute of events.

**Definition 2.3 (Event Log).** An event log  $L = \{\sigma_1, \sigma_2, \dots, \sigma_n\} \subseteq E^*$  is a finite set of traces where each event appears only once in the entire log, i.e., for any  $\sigma, \sigma' \in L, \sigma \neq \sigma'$  and for any events  $e \in \sigma$  and  $e' \in \sigma', e \neq e'$ .

**Definition 2.4 (Trace Variant).** Let  $A$  be the set of all possible activity names. A trace variant is a unique sequence of activities over  $A^*$ . Let  $\sigma^v$  be the set of all traces  $\sigma_1, \sigma_2, \dots, \sigma_n \in L$  such that share a same sequence of activities. Assume  $TV = \{\sigma_1^v, \sigma_2^v, \dots, \sigma_n^v\}$  is the set of all trace variants in the log, then for any  $i, j$  ( $i \neq j$ ) and  $\sigma_i^v, \sigma_j^v \in TV, \sigma_i^v \neq \sigma_j^v$ .

## 2.3 Process Mining

Finding the bottlenecks and predicting delays in a process, and supporting process redesign decisions are only a few benefits of business processes analysis [19]. The results of these analysis are then projected onto process models which bring opportunities for relevant experts to optimize processes. While traditional business intelligence tools and dashboards are useful in data-driven contexts, they are not able to produce valuable insights for op-

erational processes and process-centric contexts [38]. Van der Aalst argues that data mining techniques do not take into account the end-to-end nature of process data [38]. Consequently, he offered process mining, a novel approach that incorporates both process model-driven and data mining approaches.

Process mining has three main domains: Process discovery, conformance checking, and process enhancement. In process discovery, an abstract representation of an event log is built in well-known notations, e.g., BPMN [43] or Petri net [32]. In conformance checking, an existing process model is compared to the event log of the same process and then, deviations of the model from the log are identified. Process enhancement is about improving and extending process models in order to derive models that describe the behavior in the log more precisely.

## 2.4 Data Clustering

Clustering is a very well-known technique in data mining that has been applied in many fields such as biology, image pattern recognition, and business intelligence. In this unsupervised classification technique, data points are grouped together based on their similarity in such a way that objects in the same cluster are most similar to each other, while they have minimum similarity with objects in other clusters [25]. Similarity in the most of the cases is defined by a distance function such as Euclidean distance, Manhattan distance, and Cosine similarity. Han et al. [23] group basic clustering techniques into the following categories: *partitioning*, *hierarchical*, *density-based*, and *Grid-based* methods.

In partitioning methods, data is divided into  $k$  groups, in which  $k$  is provided by the user. Then, in an iterative manner, clusters are improved by substituting objects between clusters until a local optimum is reached for each cluster. Greedy approaches like  $k$ -means and  $k$ -medoids are among the best performing partitioning algorithms.

In hierarchical clustering, as the name suggests, a hierarchy of clusters is made. Then groups of objects that are close to each other in the hierarchy are merged in a bottom-up approach. We can also start with one cluster and divide it into smaller clusters in a top-down approach.

In density-based approaches, the absolute distance between objects does not

define similarity, but density between objects defines it. Clustering of the objects continue until the minimum number of objects in a certain radius, density, is satisfied and does not exceeds some threshold.

Grid-based approaches build a space with finite number of cells that forms a grid. Object space is then transformed into this grid structure and the whole process of clustering is done in this space, regardless of number of data objects.

## 2.5 Trace Clustering

Trace clustering is the intersection of data mining and process mining. Discovering a single model from a complex event log leads to a complex model with twisted and tangled arcs and transitions that is extremely difficult for a domain expert to read. Therefore, data mining approaches are applied in the context of the process mining to divide the complex log into smaller logs. Consequently, models discovered from smaller logs are less complicated and more comfortable to read. Many studies have been made into this field that includes both adaptations of the data mining ideas and novel techniques designed particularly for event logs. Trace clustering could be model-driven or trace similarity driven. While the former looks for traces that improve the model, the latter cluster traces that are close in terms of their distance. Figure 2.1 shows an overview of how application of trace clustering can make a difference.

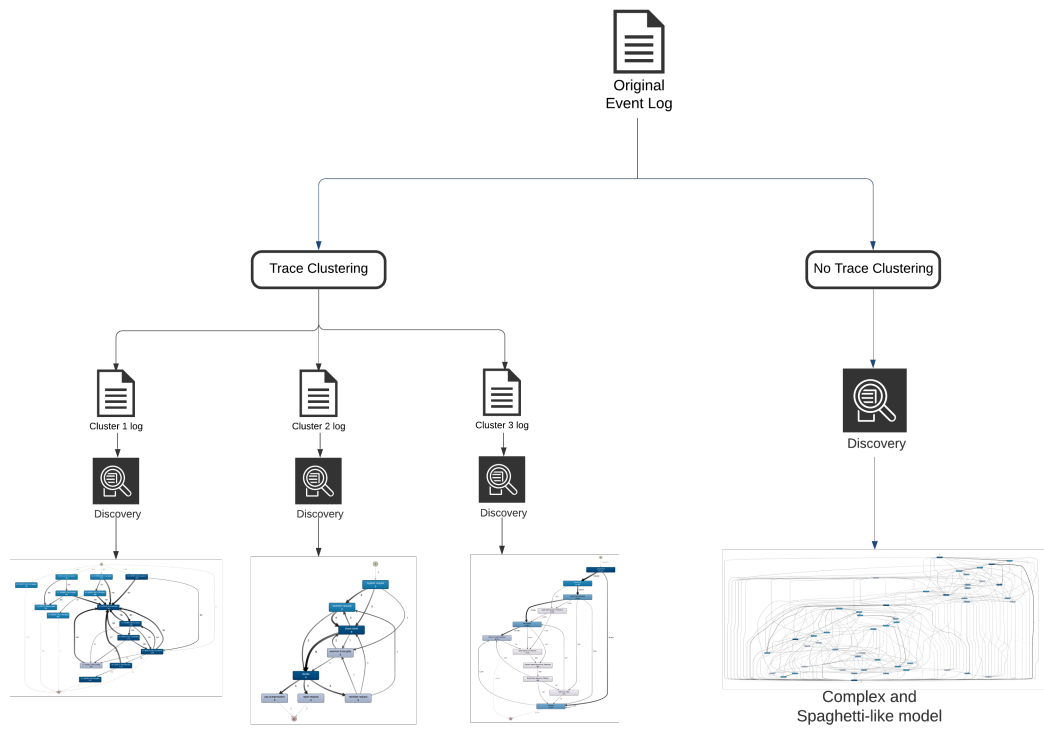


Figure 2.1: Trace Clustering

# Chapter 3

## Related Literature

In this section, existing trace clustering studies are investigated. Generally, these studies can be put into three main types in which the first two approaches focus on the similarity between traces, while the third methods are model-driven. In the first type, traces are mapped into vectors so that data mining distance metrics can be applied. In the second type, string similarity metrics are used to find similar traces. In the third type, the focus is directly on the quality of discovered models and distribution of traces among clusters.

In addition to established techniques in literature, a novel technique based on model-driven and similarity-based paradigms was introduced in 2019. This research will be discussed, and its differences with this thesis will be highlighted separately after the main clustering types have been presented.

### 3.1 Feature vector-based similarity

One way to use data mining clustering algorithms in the context of process mining is to transform each trace to a vector that represents one or more quantitative characteristic of that trace. Different perspectives of a trace have been quantified and used to vectorize a trace in the past works. Greco et al. [22] as one of the early works in trace clustering considered frequent sub-sequences of traces as feature vectors and used two thresholds to define the frequent sub-sequences. Then, each trace is mapped into a point in vectorial space so a data clustering techniques like k-means can be applied to group similar traces with each other.

Work of [15] continued and improved the study of [22] by introducing a discovery algorithm independent approach that avoid over-generalization in the resulted process models. After transforming traces into a vectoral space, k-means technique with Euclidean distance are applied in order to cluster the similar cases. However, for overfit clusters, structural patterns in which lead the model to show behaviors that are not in the log are captured. Therefore, this type of clusters will split into smaller clusters until optimal models are discovered from these clusters.

Bose et al. [6] presented a robust context-aware approach that leverage the so-called conserved patterns, sub-sequences of activities that are conserved across multiple cases. These conserved patterns are then considered as features in the space vector model, and traces that share many conserved patterns between them are clustered together. Authors also used Euclidean metric to calculate the distance between features and applied the Agglomerative hierarchical clustering technique as the clustering algorithm.

Reviewed literature so far only considered control-flow perspective as the baseline for trace comparison without incorporating context. Following studies have incorporated context by considering other perspectives.

Song et al. [36] introduced a novel approach to divide the log into homogeneous subset of traces by using activities' information, like their transitions, originators and their performance. In this work, other perspectives are considered in the process of finding similarities between traces by defining them as trace profiles. Profiling can go beyond well-known perspectives as custom profiles can be built if additional information is available in the event log. Afterwards, each trace with candidate profiles is converted to a vector to calculate its distance with other traces with similar profiles. Authors have done several experiments with three different distance measures and four clustering techniques which makes this study a flexible approach for trace clustering. The pitfall of this approach could be the curse of dimensionality. Too many dimensions make the distance between many traces look equal and the results of clustering inaccurate.

In [8], Ceravolo et al. adopted the trace profiling [36] approach and introduced position profile, a triple that considers the occurrence of an activity at a certain position with its respective frequency. In order to be able to perform different distance metrics, they transformed the whole event log to an integer matrix based on the notion of position profile. For measuring the similarity, unlike Cosine that work with normal distribution, they chose a

probability distribution independent approach that is based on inferential statistics.

## 3.2 Syntax-based similarity

In this approach, traces are compared with each other in terms of their syntactic difference. In other words, the distance between two traces is defined as the number of operations (deletion, insertion, substitution) needed for the first trace to transform to the second one. One of the most common techniques in this area is Levenshtein distance [28].

In [5], Bose and van der Aalst measured the similarity between traces with General Edit Distance, however they applied a cost function in order to automatically penalize the edit operations of uncorrelated and contrasting activities. Quadratic time complexity could be the main drawback of this approach.

Di Francescomarino et al. [18] also used edit distance in order to find similarity matrix between traces. Then, they used DBSCAN technique to cluster traces based on the density of similar process instances.

## 3.3 Model-driven

Unlike the previous two methods, in the model-driven approach, a trace joins a cluster if only it improves the quality of the cluster model. Consequently, the distance between traces is ignored when clustering traces.

The work of [7] as one the first works of sequence clustering introduced a model-based clustering. In this work, Cadez et al. used the Expected-maximization algorithm [17] to cluster users of a web site based on their behaviors by learning a mixture of first-order Markov models. Ferreira et al. [21] did the same approach as [7] but in the context of process mining to cluster sequences of activities.

In [16], authors tried to find the optimal distribution of traces between clusters that leads to maximum quality of process models of clusters. In other words, they do not aim to find the similarity between traces, but rather they

cluster traces that fit in a certain process model. They argue that data mining techniques does not consider the quality of underlying process models generated for each cluster during the clustering. Authors refer to this problem as the divergence between the evaluation bias and clustering bias and propose a new approach based on active learning that first take unique cases and, based on their distance or frequency, they are clustered together as primal clusters. Clusters accept members only if the fitness is optimized, otherwise traces are allocated to a thrash cluster or are distributed equally between other clusters. Active trace clustering (ActiTraC) which will be referred in this thesis extensively, is quadratic in terms of distinct process instances.

In [42], Veiga and Ferreira combined trace clustering with First order Markov models using a hierarchical approach. Initially, random clusters are built, and traces are distributed among them. Consequently, the cluster models (state transition probabilities of the Markov chain of each cluster) are evaluated. Then iteratively, traces are re-assigned to the clusters and evaluation is done again until algorithm converges, and cluster models do not change. Additionally, a probabilistic approach suits logs with diverse behaviors. Nevertheless, authors add a pre-processing step to deal with the noise by removing uncommon and rare sequences that affect probabilistic models. This step involves dropping sequences with low support.

The work of Hompes et al. [24] combines the model-based and feature vector-based approaches. In this work, case data have also been considered, as the event log is first split up into consecutive time windows (time perspective). Then cases are transformed into vectors and a similarity matrix is calculated by applying Cosine similarity measure. A vector of case age factors for instance can be used to decrease the case similarity to incorporate the effect of time. Eventually, similarity matrix is the input of the MCL algorithm (Markov Cluster)[41].

### 3.4 Mixed-paradigm

De Koninck and De Weerdts [12] introduced a novel trace clustering that adopts the ideas of both distance-based and model-driven clustering. The evaluation results show that the mixed-paradigm approach performed better in three domains, namely internal consistency, process model quality, and computational complexity, compared to existing trace clustering techniques. Although the mixed-paradigm and hybrid approach proposed in this thesis look similar, there are differences between them. The main differ-



ence is the way distance-driven and model-driven paradigms are applied. In the mixed-paradigm, traces are first divided into super-instances and sub-instances based on their distance, and then clusters are built based on super-instances. In the final step, the same approach in ActiTraC [16] is applied to distribute traces among clusters. In the hybrid approach, the distribution step is not model-driven only, as it takes into account the close traces of a candidate trace as well. Also, an initialization step pick accurate clusters before trace distribution starts, which could lead to improvements in performance in some event logs.

### 3.5 Discussion

Feature vector-based allows researchers to apply data mining algorithms and concepts on event data. Also, all perspectives of an event log can be mapped into vectors and used in the clustering, which leads to higher quality results. However, using a vectorial space with too many features brings a large computation load that results in an extremely long run time of a technique.

Syntax-based approaches also are expensive in terms of computation. Calculating the distance between two traces with a simple syntax-based approach like general edit distance takes quadratic time. Also, ignoring the order of activities when computing syntactical difference between two traces is another drawback of this method.

Besides, according to a study that incorporated expert knowledge in clustering [13], both of these approaches suffer from another drawback in clustering, and that is irrelevant clustering of traces. This is because two traces might be similar according to the distance measures, but do represent different behaviours.

In model-driven trace clustering traces are grouped if together they improve the quality of the models quantitatively. However, the main drawback of model-driven approaches is the high computational complexity. Qualification of traces in cluster membership is achieved through model quality evaluation which is an expensive task in terms of calculation. The evaluation also depends on many factors, including the underlying discovery algorithm, model quality metrics used, and the kind of evaluation (token-based or alignment-based). For example, ActiTraC could require 50 times more resources than similarity-based approaches, as demonstrated in the ActiTraC paper.

The idea of this thesis is inspired by the drawbacks of techniques in the literature. Delivering an accurate technique with reasonable computation time by combining the advantages of existing methods is the main task of this thesis. Table 3.1 shows an overview of existing trace clustering techniques in the literature, grouped by clustering type, method used, and clustering technique.

Technique	Clustering Type	Method	Clustering Technique
Greco et al. [22]	Feature Vector	Frequent features	K-means
De Medeiros et al. [15]	Feature Vector	Frequent Patterns	K-means
Bose et al. [6]	Feature Vector	Conserved patterns	Hierarchical
Song et al. [36]	Feature Vector	Trace Profiling	Multiple techniques
Ceravolo et al. [8]	Feature Vector	Trace Profiling	Hierarchical
Bose et. al [5]	Syntax	General edit distance	Hierarchical
Ferreira et. al [21]	Model-driven	Markov model + EM	K-cluster
De Weerd et al. [16]	Model-driven	Active learning	K-cluster
Veiga et al. [42]	Model-driven	Markov Chains	Hierarchical
Hompes [24]	Model-driven	Markov cluster algorithm	K-cluster
De Koninck et al. [12]	Feature Vector + Model-driven	Active learning	K-means

*Technique* column is the citation of the respective paper; *Clustering Type* column shows the type of trace clustering; *Method* column indicates what method the paper used; *Clustering Technique* column refers to the underlying clustering technique used, which is mostly a data clustering algorithm. *K-cluster* in this column includes techniques in which the number of clusters,  $k$ , is set beforehand, however it does not indicate a specific algorithm.

Table 3.1: Existing Trace Clustering techniques in the literature

# Chapter 4

## Hybrid Trace Clustering

This chapter discusses the approach proposed by this thesis and addresses the arguments surrounding decisions. It is worth noting that Hybrid technique is not limited to the configuration used in this thesis. A clustering problem could be approached by a more generalized Hybrid technique using different input parameter values.

Figure 4.1 illustrates the main steps of Hybrid technique framework. After providing the event log and input parameters, a data clustering algorithm is applied on the event log to identify and exclude high quality clusters from the rest of the clustering task. Unqualified traces are then marked in the second step and initialize a new cluster in the third step. In the fourth step, traces are assigned based on their performance on cluster models, and in the fifth step, remaining traces are put together as a trash cluster. Full details of the algorithm is described in section 4.2.

### 4.1 Elements

In the Hybrid technique, process models are iteratively evaluated quantitatively. Many of the techniques in the literature have used quantitative process model quality metrics to evaluate clusters [16, 5, 6, 36]. *Fitness*, *precision*, *generalizability*, and *simplicity*, are the four quality criteria proposed by Van der Aalst [38] and are considered as the baseline in the quantitative evaluation of process models.

Quality metrics can be used in different ways. For instance, in [35], authors used fitness and precision but gave them different weights for their quantitative model evaluation. The same authors also used Cardoso and the

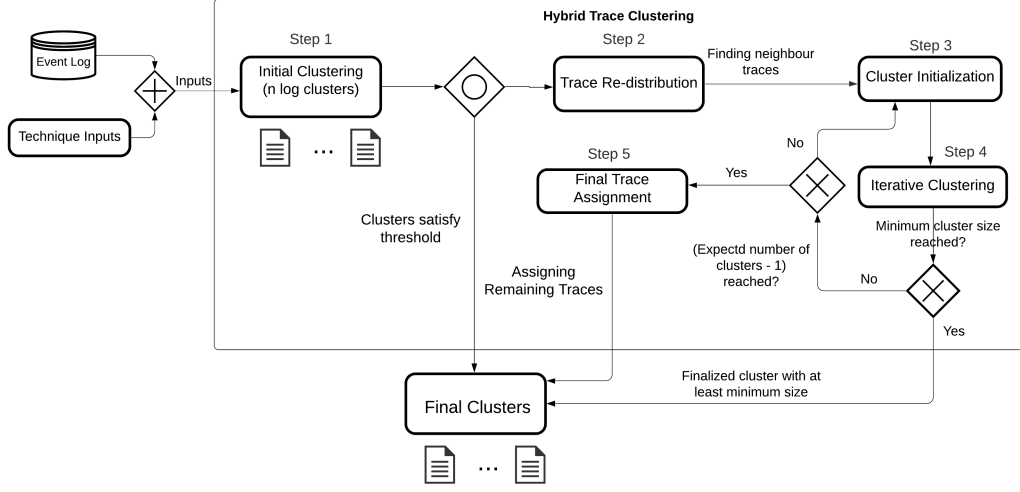


Figure 4.1: Hybrid technique framework

size of the model (total number of transitions, arcs, and places of a Petri net), to qualitatively evaluate models. Authors of ActiTraC [16] used only a specific type of fitness (ICS-Fitness) [14] as their sole evaluation metric in the iteration phase.

In this thesis, fitness, precision, and  $F1$ -score are chosen as model evaluation metrics. Regarding the rationale behind this decision, it can be argued that in a clustering problem, we are looking for readable models such that each represent a certain behaviour in the event log. Therefore, fitness is chosen as we are looking for models that are able to represent the behaviour seen in the event log, while precision is chosen because each cluster should not cover non-homogeneous behaviour or traces that belong to other clusters. To balance fitness and precision,  $F1$ -score of each cluster is calculated, giving both metrics equal importance.  $F1$ -score is calculated as in Equation 4.1:

$$F1 = 2 * \frac{fitness * precision}{fitness + precision} \quad (4.1)$$

Also, since clusters have different sizes, cluster weight (size) is incorporated in the  $F1$ -score to have a better indication of average quality of all clusters. Let  $l$  be the size of a cluster and  $1 \leq i \leq n$ . Then, the *weighted average  $F1$ -score* ( $F1_{WA}$ ) is calculated as in Equation 4.2:

$$F1_{WA} = \frac{\sum_{i=1}^n l_i * F1_i}{\sum_{i=1}^n l_i} \quad (4.2)$$

Weighted average F1-score is an appropriate metric for evaluation of Hybrid technique which will be used later frequently in chapter 6.

Moreover, the Hybrid technique algorithm also depends on the task of process discovery. Cluster models are continuously discovered and assessed, and thus, the selection of process discovery algorithm is a critical task. In this thesis, *Inductive miner (IM)* [27] without any filtering, and default setting, has been chosen as the primary discovery algorithm. Inductive miner is one of the most compelling discovery algorithms when the target is to have models with the highest F1-scores [2]. According to the same study, [2], Inductive miner also discovers less complicated process models compared to many other available techniques .

## 4.2 Algorithm

The Pseudo-code in Algorithm 1 describes the algorithm of the Hybrid technique and it will be referenced again throughout this chapter. Furthermore, the algorithm's inputs, parameters, and outputs are explained first, and then each step of the algorithm is presented in detail. It is worthy to note that while different values of algorithm's input are investigated in this thesis, algorithm parameters use a default setting.

### Input Parameters

- **Event log.** An event log that process discovery algorithms are unable to produce a single comprehensible process model from it. The Hybrid clustering technique divides into several clusters.
- **Number of Clusters (NC).** The number of clusters that are going to be obtained should be set beforehand. This parameter also decides when the algorithm should stop looking for more clusters.
- **Initial F1-score Threshold (FT).** This is the F1-score threshold that distinguishes appropriate clusters in the first step. Setting it too low lead to qualification of unnecessary clusters and setting it too high only qualify highly accurate clusters in this step.

- **Neighborhood Size (NS).** This parameter decides how many similar traces of a single frequent trace should be considered when building a new cluster. A large size results in too many traces in a starting cluster, and small size puts the bias of the technique towards the model-driven side.
- **Distance Metric.** This parameter lets the user choose the distance metric. The distance metric can be either a syntax-based or feature vector-based.
- **Clustering Technique.** This parameter defines which data clustering technique is used in the first step of the Hybrid technique.

### Algorithm parameters

- **Minimum Cluster Size (MCS).** Experiments done in this thesis have shown that sometimes clustering technique in the first step results in clusters with very few traces, but with a high F1-score. Without any condition, these clusters satisfy the F1-score threshold and therefore are added to the final clusters. An example of this phenomena is cluster number 4 found by K-means in table A.2 in appendix that includes only 5 traces, but has a perfect F1-score. In order to avoid these sort of clusters, a minimum cluster size condition is practiced. In this thesis, minimum size is set to 300.
- **Discovery Algorithm.** This parameter indicates the process discovery algorithm used in each step of the hybrid technique. All models in this thesis were discovered using Inductive miner.
- **Intermediate Model quality measure.** Users can choose the model quality criteria used in the mode-driven part of Hybrid technique. The intermediate model quality evaluates models whenever a new trace variant is tested on a cluster. Examples for this parameter are fitness, precision, and complexity to name a few. The weight of each metric could also be tuned with respect to the needs of problem. In this thesis, harmonic mean of fitness and precision is adopted as default setting.

### Output

- A collection of sub-event logs that each belong to a cluster.

### 4.2.1 Step 1: Initial Clustering

The goal of the first step is to see if we can find proper clusters in terms of F1-score, using only a trace similarity-based approach with the least computational complexity. According to [5], mapping traces into vectors with a *bag-of-activities (BOA)* approach and calculating the distance between them using Euclidean measure can be done in linear time with accordance to the length of the trace, which is the number of activities in the trace.

Moreover, traditional clustering algorithms with the least amount of computational complexity are applied that can utilise the BOA model. Based on a survey on clustering algorithms [45], *K-means* [29] and *DBSCAN* [20] are among the fastest algorithms and therefore, are chosen for this step.

Line 3 in the algorithm 1 executes this step. The resulting clusters are used later in the next step.

### 4.2.2 Step 2: Trace Re-distribution

In this step, the F1-score of resulted cluster models is measured and compared to the initial F1-score threshold (FT) that is set by the user as an input. Clusters that satisfy FT are qualified as final clusters, and their traces will not participate in the next steps of the algorithm. Clusters that do not meet the FT are dissolved with the condition that traces that were put in the same cluster are marked with the same flag. These flags are the used in the next step to build a new cluster.

Moreover, initial clusters are evaluated using *token-based replay* [39] fitness and precision. Although token-based conformance checking provides less accurate diagnostics than *alignment-based* [34], it is less time-consuming [38]. Since in the first step we are looking for highly accurate initial clusters while not compromising time, token-based replay conformance checking is selected for evaluation in this step.

Lines 5-12 in algorithm 1 are responsible for execution of this step. First a cluster model is discovered using Inductive Miner in lines 5 and 6. Then, the F1-score is checked in line 7. If a cluster is qualified, it will be joined to the final clusters in line 8, otherwise, the flag function is executed in line 12.

### 4.2.3 Step 3: New Cluster Initialization

The purpose of this step is to create a new cluster with the condition that we have not reached the expected number of clusters (NC) yet. If we reach this step and there exist (NC - 1) clusters, step 5 is executed. Otherwise, the following procedure is performed.

From the remaining traces, the most frequent trace variant is picked in order to start a new cluster. Next, the neighbourhood of this variant is checked for similar variants that have not been flagged in the previous step, which then are added to the cluster. The size of the neighbourhood (NS) is fixed by the user as an input. The reason for checking neighbourhood is that we want to group as many traces that are close and have not been clustered together yet as possible. In other words, we want to try a different trace distribution between clusters. After the most frequent variant and the variants in its neighbourhood are identified, all traces belonging to these variants comprise a new cluster.

In line 13 of the algorithm 1, loop conditions indicate this step's requirement. Also, line 17 shows the participation's flag and distance conditions for traces.

### 4.2.4 Step 4: Iterative Clustering

In this step, remaining traces are examined on the new cluster. The changes in the cluster model's F1-score decide the participation of the candidate traces in the cluster.

Initially, the most frequent variant is identified and added to the cluster. The changes in the F1-score of the cluster model is monitored.

Two following scenarios could happen:

- If F1-score of the model decreases, the minimum cluster size(MCS) is checked. If the cluster satisfy *MCS*, the selected variant is ignored, the cluster is finalized, no other trace is added to the cluster anymore, and step 3 is repeated again. If the size condition is not satisfied, step 4 is repeated again.
- If F1-score increases, all traces belonging to the nominated trace variant are added to the cluster and step 4 is repeated again.

Lines 20-31 in the algorithm 1 show how this step is executed. Lines 20-27 are repeated until there is a variant that does not improve the F1-score of



the model. After the repetition is interrupted by an unfit variant, the cluster is finalized in line 30 and conditions of the big loop in line 13 are rechecked.

#### 4.2.5 Step 5: Final Trace Assignment

Remaining variants (the ones that have not been assigned to any cluster yet) are put together in a so called *trash cluster* as the final cluster (line 32 in algorithm 1. Trash clusters' traces are only put together whenever pre-set number of clusters is reached. Absence of any similarity-based or model-driven clustering task on these traces mostly results in poor cluster models in terms of fitness and precision.

**Algorithm 1:** Hybrid Trace Clustering Algorithm

---

**Input:** Original event log  $L$ , Number of clusters  $k$ , Distance metric  $DM$ , Clustering technique  $CT$ , neighbourhood size  $ns$ , F1-score threshold  $ft$ , and minimum cluster size  $mcs$

**Output:** collection of event log clusters  $FC$

```

1  $RT \leftarrow \emptyset$  // Remaining traces that are not clustered yet
2  $IC \leftarrow \emptyset$  // Denotes the initial set of clusters
3  $IC \leftarrow CT(L, k)$  // Initial clustering using K-means or DBSCAN
4  $counter \leftarrow 0$  // Cluster counter that increments whenever a cluster is
   finalized
5 for every cluster in  $IC$  do
6    $PM_{cluster} \leftarrow IM(cluster)$  // Process model  $PM_{cluster}$  of each cluster
   is discovered using Inductive miner algorithm  $IM$ 
7 if  $F1\text{-score}(cluster, PM) > ft \wedge \text{length}(cluster) \geq mcs$  then
8    $FC \leftarrow cluster$ 
9    $RT \leftarrow L \setminus cluster$ 
10 else
11   for  $trace_{variant}$  in cluster do
12      $flag(variant)$  // Variants in an unqualified cluster get the
     same flag
13 while  $RT \neq \emptyset \wedge counter \neq k-1$  do
14    $NC \leftarrow \emptyset$  // A new cluster is initiated
15    $trace\_variants \leftarrow frequent\_variant\_finder(RT, ns)$ 
16   for  $variant$  in  $trace\_variants$  do
17     if  $dist_{DM}(variant_{frequent}, variant) < 2 \wedge variant_{frequent}^{flag} \neq$ 
      $variant^{flag}$  then
18        $NC \leftarrow NC \cup \{trace_{variant}^i\}$  // All traces with the same
     trace variant as  $variant$  are added to the new cluster
19        $RT \leftarrow RT \setminus \{trace_{variant}^i\}$ 
20   repeat
21      $PM \leftarrow IM(NC)$ 
22      $variant_{frequent} \leftarrow frequent\_variant\_finder(RT, 1)$ 
23      $NC_{new} \leftarrow NC \cup \{trace_{variant_{frequent}}\}$ 
24      $PM_{new} \leftarrow IM(NC_{new})$ 
25     if  $F1\text{-score}(NC_{new}, PM_{new}) \geq F1\text{-score}(NC, PM)$  then
26        $NC \leftarrow NC_{new}$ 
27        $RT \leftarrow RT \setminus \{trace_{variant_{frequent}}\}$ 
28   until
29      $F1\text{-score}(NC_{new}, PM_{new})$ 
      $\leq F1\text{-score}(NC, PM) \wedge \text{length}(NC) \geq mcs$ 
30    $FC \leftarrow FC \cup NC_{new}$ 
31    $count \leftarrow count + 1$ 
32  $FC \leftarrow FC \cup RT$  // Remaining traces are grouped as trash cluster
33
```

---

# Chapter 5

## Implementation

This section discusses the implementation of the Hybrid technique and explains all other required information pertaining to implementation. The algorithm used in the Hybrid technique is implemented using Python programming language and is publicly available at Github. Thus, first, all libraries used in the implementation are reviewed. Then the main functions of the Hybrid technique algorithm are described.

- PM4PY [4]: An open source library that supports process mining algorithms in Python. The Hybrid technique is developed based on this library, meaning discovery and conformance checking algorithms are borrowed from PM4PY.
- NumPy: An open source library that supports high level mathematical functions as well as multi-dimensional matrices. For this thesis, it has been used to calculate the trace similarities according to their distance. In other words, implementation of both bag of activities and Levenshtein has used Numpy.
- Scikit-learn: An open-source library that supports machine learning algorithms. K-means and DBSCAN algorithms used in this thesis are utilised from scikit-learn library.

For the calculation of process models' fitness in the algorithm, token-based variant of the replay fitness is used. The implementation of this variant in PM4PY is based on a new and faster version of token-based replay fitness [3]. Regarding the calculation of precision, ETConformance (using token-based replay) implementation of PM4PY is used.

For the calculation of fitness and precision in the evaluation of the Hybrid technique, alignment-based replay fitness and align-ETConformance implementations of fitness and precision in the PM4PY are used respectively.

The code architecture of implementation is visualized in figure 5.1. There are three main python modules in the implementation: *algorithm*, *dist\_calc*, and *clustering\_algo*. The main module, *algorithm*, includes three main functions, in which *apply* is the the function that provokes the algorithm. This function calls other two main functions, *new\_cluster* and *trace\_distribution*. K-means and DBSCAN are implemented in *clustering\_algo* module and will be used in the *apply* function. Functions in the *dist\_calc* are used in the *new\_cluster*.

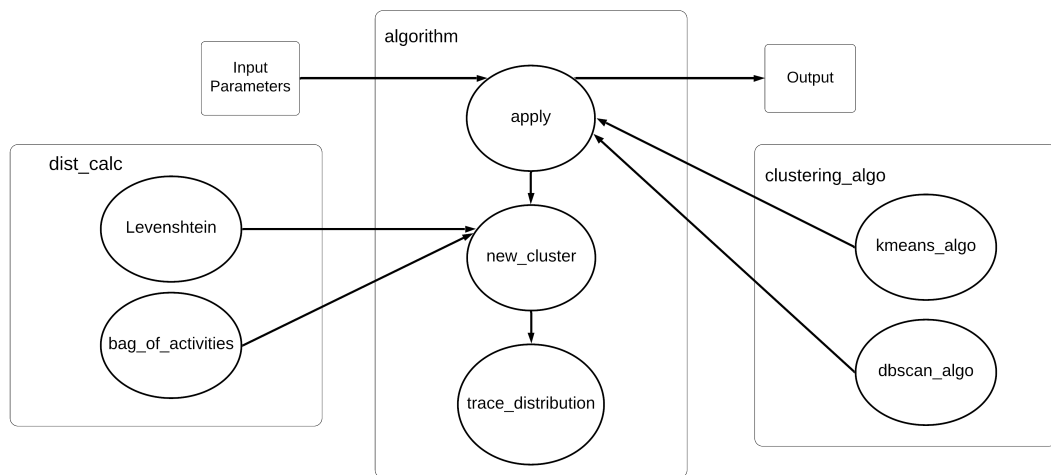


Figure 5.1: Hybrid technique code architecture

```

1 import pm4py
2 import hybrid_trace_clustering
3
4 DEFAULT_DISCOVERY_TECHNIQUE = 'inductive miner'
5 DEFAULT_CLUSTERING_TECHNIQUE = 'K-means'
6 DEFAULT_NUMBER_OF_CLUSTERS = 4
7 DEFAULT_DISTANCE_TECHNIQUE = 'levenshtein'
8 DEFAULT_F1_SCORE_THRESHOLD = 0.95
9 DEFAULT_NEIGHBORHOOD_SIZE = 10
10
11 event_log = pm4py.xes_importer.apply("example_log.xes")
12 clusters = hybrid_trace_clustering.algorithm.apply(
13     event_log,
14     initial_f1_score=DEFAULT_F1_SCORE_THRESHOLD,
15     number_of_clusters=DEFAULT_NUMBER_OF_CLUSTERS,
16     neighbourhood_size=DEFAULT_NEIGHBORHOOD_SIZE,
  
```

```
17 distance_technique=DEFAULT_DISTANCE_TECHNIQUE ,  
18 clustering_technique=DEFAULT_CLUSTERING_TECHNIQUE ,  
19 discovery_technique=DEFAULT_DISCOVERY_TECHNIQUE)
```

Listing 5.1: Hybrid trace clustering code usage example in Python

# Chapter 6

## Evaluation

In a data clustering problem, evaluation of resulted clusters centers on how well clusters fit the data. If ground truth is available, cluster quality could be assessed by comparing the homogeneity or completeness of clusters to labelled data. When the ground truth is unavailable, scientists switch to methods that quantify how compact or separated clusters are as an alternative approach.

Within the context of process mining, there is another opportunity for cluster assessment, regardless of the availability of ground truth. Since clustering on an event log aims to derive simpler process models, the quality of cluster models is of the highest importance. Therefore, the cluster models' quality can be considered as an evaluation metric alongside data clustering metrics for overall performance assessment.

Process models have been assessed both qualitatively and quantitatively in the literature. While in the former domain experts' knowledge is utilised, four main quality metrics are used for the latter. In [30], authors took advantage of the knowledge of healthcare experts and asked them to divide the list of patients of a hospital into homogeneous groups. Then, they used these groups as clusters to train their own technique. While this approach leads to significant accurate results, it is out of scope of this thesis. Nevertheless, readability of resulted cluster models of different techniques will be discussed in this thesis.

In this chapter, the experimental setup is explained in the first section. The next three sections are aligned with the research questions and their respective results. The final section discusses the overall evaluation of Hybrid technique.

## 6.1 Experimental Setup

In order to evaluate the hybrid technique, over 300 hundred experiment runs have been done. All the experiments for this thesis are conducted on a MacBook Pro 2020 with a 2 GHz Quad-Core Intel Core i5 processor and 16GB of RAM. Python 3.8 on PyCharm IDE was chosen as the experiment environment.

Since the hybrid technique is evaluated in terms of both run time and quality of cluster models, two different experiment settings are used for evaluation. Experiments concerning models' quality have been conducted on six real-life event logs such that each belong to various domains. *KIM* event log belong to Help desk process of the ICT service of *KU Leuven* [1]. *TSL* event log belongs to a second-line CRM process [1]. Both *KIM* and *TSL* are borrowed from ActiTraC research paper [16]. *Road Traffic Fine* event log is taken from an information system that manages road traffic fines [31]. *BPI 2012* is the event log of a Dutch financial institute [9]. *BPI 2013* event log belongs to IT department of *Volvo* in Belgium [10]. Data of *BPI 2018-Reference* is provided by *data experts*, a medium-sized IT company [11]. Table 6.1 shows event logs used for experiments.

Experiments have been repeated for three, four, and five clusters for every input configuration and for each log with inductive miner as the discovery algorithm and alignment-based F1-score as the the main evaluation metric. In addition, Hybrid technique is also compared with other clustering techniques. For this purpose, Hybrid technique has been evaluated on standard input parameters in order to keep the consistency.

It is also important to note that among experiments' real-life event logs, *BPI 2012* has been sampled due to high computational demands of alignment-based conformance-checking on this log. Therefore, to avoid long hours of experiments only for this one log, a random sampling has been done on *BPI 2012*. *BPI 2012* log originally has 36 unique activities, which this amount is reduced to 8, 9, and 11 for sampled logs. For all input parameters, the average of 10 experimental runs for each sampled log is recorded and the mean of all results, run times and F1-scores, is recorded for *BPI 2012* value. Table 6.2 shows the sampling results of *BPI 2012*.

Event Log	#Cases	#Variants	#Unique Activities
BPI challenge 2012 [9]	13087	4366	36
BPI challenge 2013 [10]	7554	2278	13
BPI challenge 2018-Reference [11]	43802	515	6
KIM [1]	24770	1174	18
TSL [1]	17812	1908	42
Road Traffic Fine [31]	150370	231	11

Table 6.1: Selected Real-life event logs

Sampled Log	#Cases	#Variants	#Activities
BPI 2012-8	13087	4366	8
BPI 2012-9	13087	2278	9
BPI 2012-11	13087	515	11

Table 6.2: Sampled BPI 2012 event logs

## 6.2 SQ1: Distance and clustering technique

This section addresses the first sub-research question by showing the results of the relevant experiments. The first sub-research question is formulated as follows:

**SQ1:** *Which distance metric and clustering technique result in the fastest clustering while not compromising the quality of process models?*

In order to find the most accurate and fastest distance and clustering techniques, experiments conducted while all other input parameters were fixed. In the rest of this section, first cluster models are evaluated quantitatively and qualitatively, and then the running time of experiments is reviewed.

### Quality in terms of F1-score

The individual effects of distance metric and clustering technique on the weighted average F1-score are depicted in Figure 6.1 for all logs. Starting with the distance metric, it can be observed that levenshtein has higher mean F1-score among all logs. Only in BPI 2018-ref two metrics has the same value. However, no clustering technique offers a consistent improvement. DBSCAN performed better in KIM, Road, and TSL event logs, while K-means resulted in higher F1-scores in BPI 2013 and BPI 2012. Regarding BPI 2018, DB-



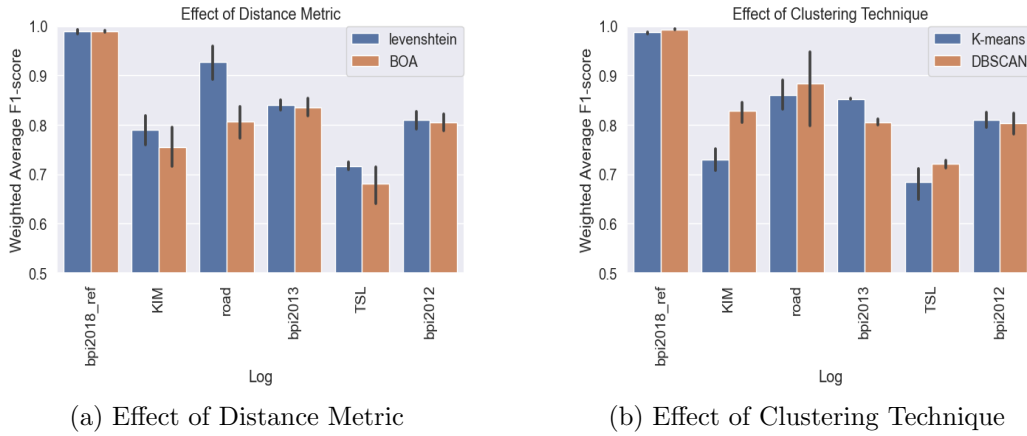


Figure 6.1: The effects of Distance Metric and Clustering Techniques on Weighted Average F1-score.

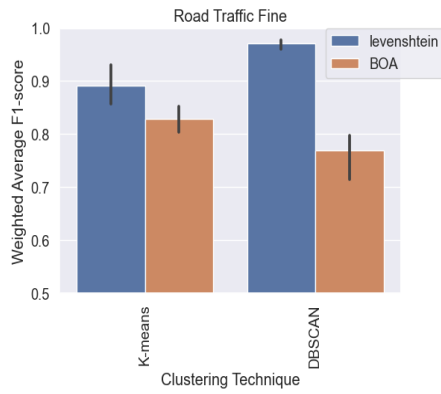
SCAN performed slightly better.

Figure 6.2 shows the combined effect of distance metric and clustering technique for each event log. Based on the plots, it can be deduced that application of Levenshtein with DBSCAN results in significantly higher scores in Road 6.2a and KIM 6.2b logs. This combination has slightly better results in 6.2c, 6.2d, and 6.2f as well. Only in 6.2e this combination is not the best performing one.

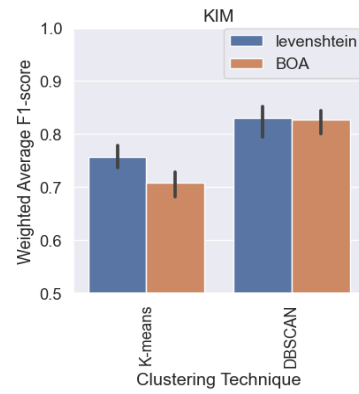
### The Effect of flag in Model's quality

The inclusion of flags in the algorithm affects how the initial clustering results are used in the next steps of the algorithm. Therefore, before starting the qualitative analysis of cluster models, this sub-section investigates the effect of flag on the clustering technique used in the first step of Hybrid's algorithm.

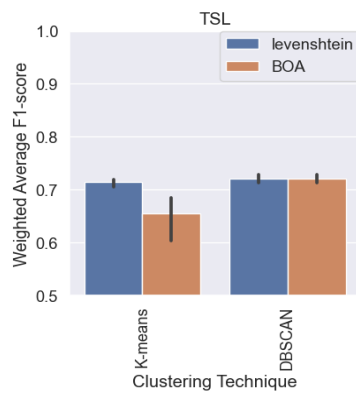
As can be seen in Figure 6.3, except for one log (TSL), employment of flag results in the higher F1-scores when DBSCAN is used as the clustering technique. On the other hand, except for one log (BPI 2013), having flag results in lower F1-scores. This is because K-means could barely miss high quality clusters because of the threshold. The traces in these clusters are not allowed to be together because of the flag condition. However, DBSCAN could find few but huge clusters, or many but small clusters. The flag condition then penalizes the clusters not satisfying the threshold correctly. The next



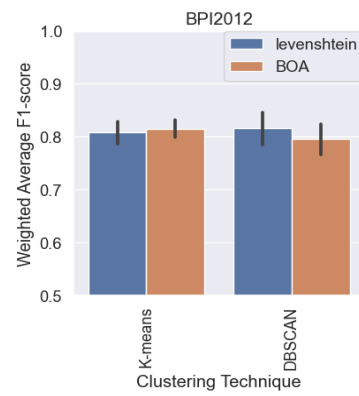
(a) Road Traffic Fine.



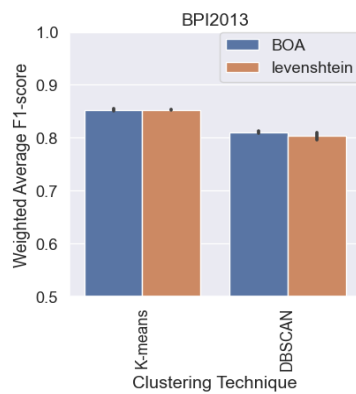
(b) KIM.



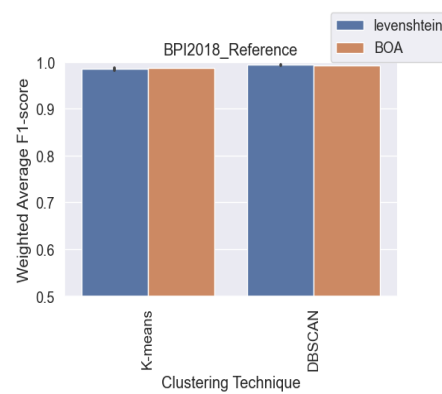
(c) TSL.



(d) BPI 2012.



(e) BPI 2013.



(f) BPI 2018-Reference.

Figure 6.2: Weighted Average F1-score against Clustering technique.

sub-section discusses the relation between the quality of models and flag’s contribution.

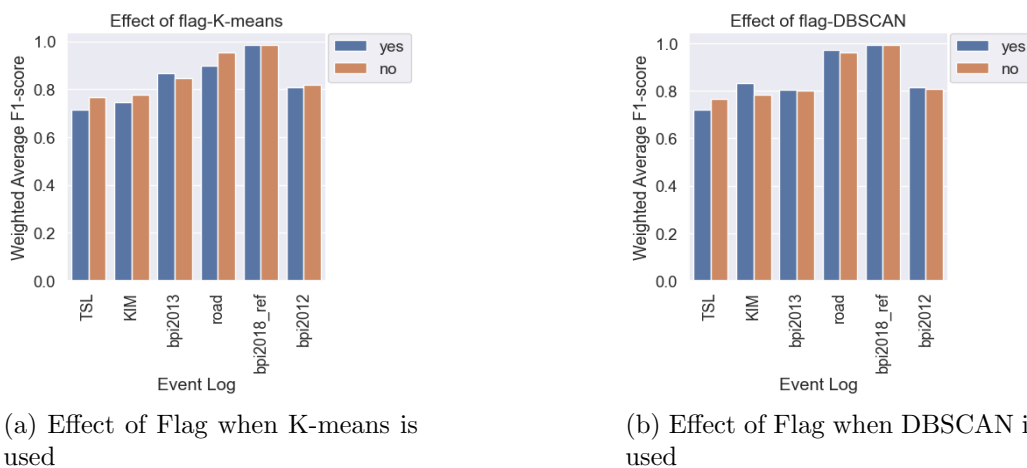


Figure 6.3: The effects of flag on Weighted Average F1-score with different Clustering Techniques.

### Cluster Models’ Quality

In order to observe the effect of the clustering algorithm in practice, discovered process models are investigated qualitatively in this part. According to figure 6.1b, DBSCAN on logs KIM and Road traffic fine results in higher weighted average F1-score compared to when K-means is used. Therefore, we seek more insight in these two event logs with Levenshtein as the underlying distance metric for the comparison, and 0.95 as the initial F1-score threshold. In order to fit the models in one table for the sake of simplicity, trash clusters of both types are also ignored in tables.

Table 6.3 shows the results of first phase of Hybrid with different clustering algorithms. Looking at the F1-scores, it is apparent that no cluster is qualified as a final cluster in the first phase. However, what is important for us to recognise in the table below is that, DBSCAN only found one cluster. Since this huge cluster is not qualified, all traces are marked with the same flag. When a new cluster is built in the second step of the algorithm, only the one most frequent variant initialises the new cluster. All other variants have the same flag and are not allowed to be grouped together. The outcome of

this event can be seen in the clusters of table 6.4, when the first and second clusters of DBSCAN have perfect F1-score. Overall, clusters of DBSCAN only comprise of a few number of variants because of the flag condition.

#Traces/ F1-score \ Cluster number	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Hybrid Variant				
Hybrid <sub>DBSCAN</sub>	24770/0.662	-	-	-
Hybrid <sub>K-means</sub>	5092/0.523	6622/0.739	6242/0.489	6814/0.690

Table 6.3: Initial clusters found by Hybrid’s first phase for KIM event log

On the other hand, K-means resulted in four unqualified clusters, which results in four flags. This allows the next step of the algorithm more flexibility to chose variants. The outcome of this event is visualised in the K-means clusters in the table 6.4. Although the clusters of K-means have lower F1-scores compared to DBSCAN, its models cover more behaviours.

Road traffic fine is another event log which clustering algorithm used makes a meaningful difference in the final weighted average F1-scores. As table 6.5 shows, only one cluster from K-means is qualified as the final clusters. Table 6.6 shows the resulted clusters of Hybrid technique when DBSCAN and K-means are employed.

#Traces/ F1-score \ Cluster number	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Hybrid Variant				
Hybrid <sub>DBSCAN</sub>	70510/0.828	79814/0.76	46/0.736	-
Hybrid <sub>K-means</sub>	20540/0.798	56887/0.59	22973/0.711	<b>49970/0.999</b>

Table 6.5: Initial clusters found by Hybrid’s first phase for Road Traffic fine event log

Like KIM event log in the previous part, DBSCAN found two perfect clusters which are basically comprise of single variants. This is due to the fact that DBSCAN marks the most of the traces with only two flags, according to table 6.5. On the contrary, K-means clusters are comprised of multiple variants, which makes them cover several behaviours.

Hybrid Variant	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid <sub>DBSCAN</sub>	4	1	3660	1.0
Hybrid <sub>K-means</sub>	4	1	8956	0.92
Hybrid <sub>DBSCAN</sub>	4	2	6762	1.0
Hybrid <sub>K-means</sub>	4	2	10051	0.72
Hybrid <sub>DBSCAN</sub>	4	3	3442	0.747
Hybrid <sub>K-means</sub>	4	3	721	0.73

Table 6.4: Process models discovered from KIM event log for 4 clusters using Hybrid technique, grouped by clustering algorithm used

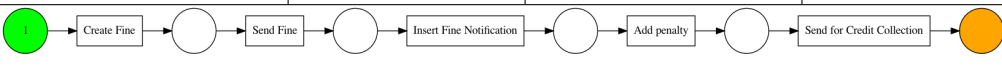
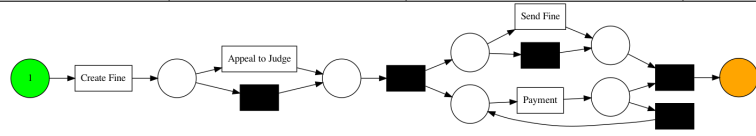
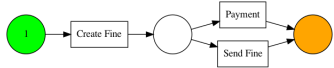
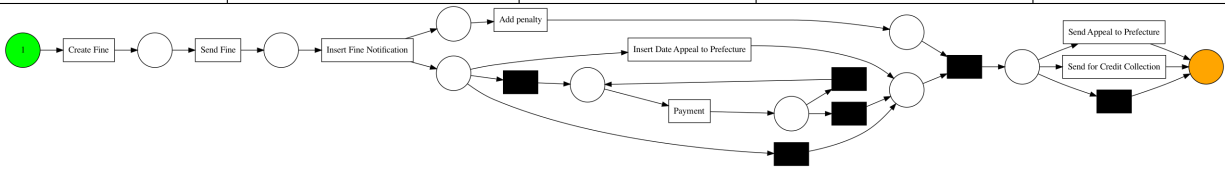
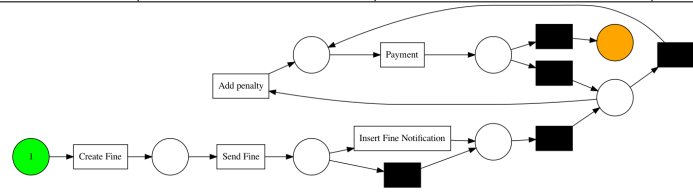
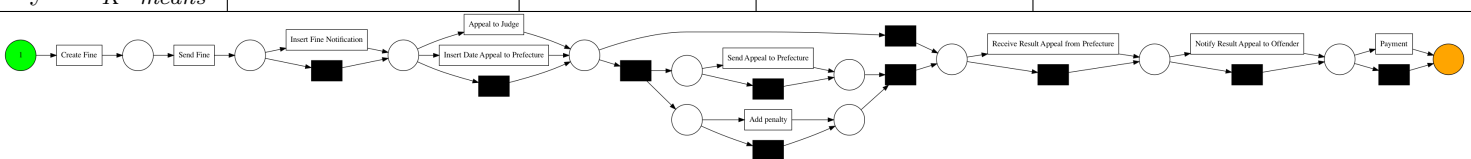
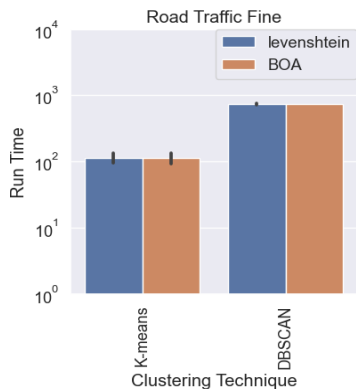
Hybrid Variant	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid <sub>DBSCAN</sub>	4	1	56482	1.0
				
Hybrid <sub>K-means</sub>	4	1	49970	0.999
				
Hybrid <sub>DBSCAN</sub>	4	2	66756	1.0
				
Hybrid <sub>K-means</sub>	4	2	78016	0.942
				
Hybrid <sub>DBSCAN</sub>	4	3	19688	0.910
				
Hybrid <sub>K-means</sub>	4	3	20899	0.912
				

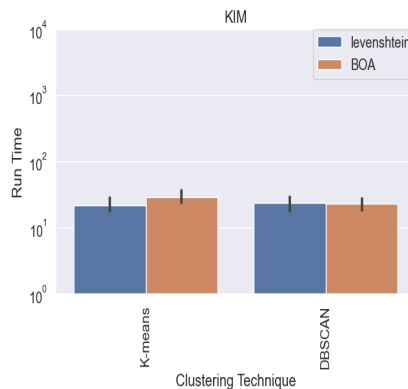
Table 6.6: Process models discovered from Road Traffic Fine event log for 4 clusters using Hybrid technique, grouped by clustering algorithm used

## Run Time

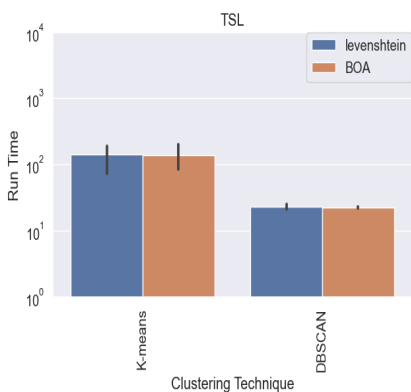
This sub-section investigates the effect of distance metric and clustering techniques on the run time of the algorithm. Figure 6.4 shows the results of running time of the Hybrid algorithm on the six event logs. What stands out from the figure below is that no single configuration is always associated



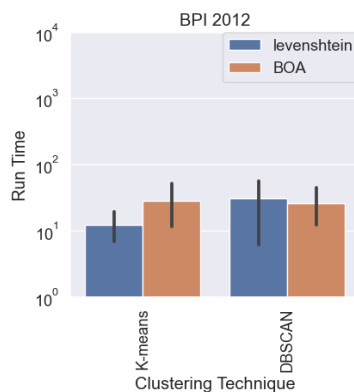
(a) Road Traffic Fine.



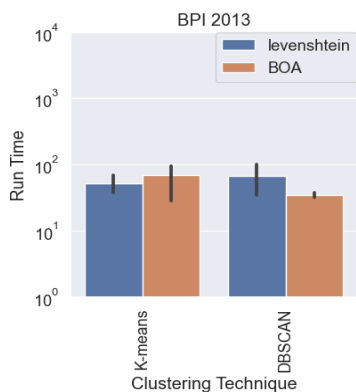
(b) KIM.



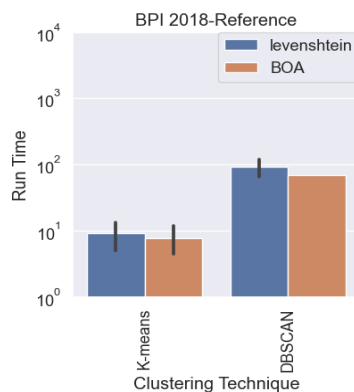
(c) TSL.



(d) BPI 2012.



(e) BPI 2013.



(f) BPI 2018-Reference.

Figure 6.4: Effect of Distance metric and Clustering Technique on Running time.

with the lowest run time. In addition, 6.4 shows that the distance metric has no effect on the run time on the Road and TSL logs. Another observation is that while BOA distance metric with DBSCAN associates with lower run times on BPI 2012 and BPI 2013 logs, Levenshtein distance metric performs more efficient when combined with K-means. For KIM and BPI 2018, the difference that distance metric makes is negligible.

Regarding the effect of only clustering technique, the number of cases which K-means performs better, 6.4a, 6.4d, and 6.4f is greater than the number of cases which DBSCAN does. Except for TSL event log (6.4c) which DBSCAN is considerably faster than K-means, K-means is faster or slightly slower than DBSCAN. In addition, K-means results in faster clustering when combined with Levenshtein distance metric, compared to BOA.

### 6.2.1 Discussion

There is no consistent pattern for a combination of distance metric or clustering technique that results in the best performance in terms of quality. While Levenshtein clearly surpasses BOA in terms of quality of cluster models, there is not enough proof that Levenshtein always performs more efficiently than BOA. Unlike having no pattern between distance and clustering techniques for the quality of models, using K-means and Levenshtein associates with lower run times, while this DBSCAN and BOA together achieves a faster clustering.

Concerning the effect of the clustering algorithm on quality of cluster models, several observations could be noticed. DBSCAN offers the same or higher scores compared to K-means. Only in the BPI 2013 event log, K-means produced models with higher quality.

The outcome of incorporating flag is that the highest quality K-means clusters cover more behaviours, while DBSCAN clusters lean towards atomic behaviours. This could occur while both might produce models with approximately close F1-scores. We can argue that there is a higher chance for unqualified traces in K-means to get different flags compared to DBSCAN. This is because K-means always results in  $k$  clusters, while there is no guarantee for DBSCAN to find a specific number of clusters. Therefore, as observed in the qualitative part, all traces could get the same flag. This will substantially affect the distribution of traces among clusters for the rest of the algorithm



## 6.3 SQ2: Input Parameter Configuration

The Hybrid technique has five input parameters. In the section 6.2, the effect of distance and clustering technique was investigated individually and collectively. In this section, the rest of the parameters are examined. The plots in Figure 6.5 illustrate which parameter values lead to the highest possible F1-score. It is worthy to note that K-means and Levenshtein were chosen as the clustering technique and distance metric respectively, and the rest of the experiments' parameter space was considered as follows:

- **Number of Clusters:** {3, 4, 5}
- **Neighbourhood Size:** {10, 30}
- **F1-Score Threshold:** {0.85, 0.95}

Figure 6.5 shows results of different input values on F1-score of cluster models. The order of values in x-axis of plots is as follows: *neighbourhood size/ F1-score threshold*

What is interesting about the data in Figure 6.5 is that the neighbourhood size does not have a significant effect on the F1-score, except for slight variations in KIM and Road logs. For KIM event log, reducing the size from 30 to 10 improves the quality, regardless of the number of clusters. This case also applies for Road on 4 and 5 clusters, except that for 3 clusters, neighbourhood of size 30 improves the quality. Despite differences, the effect of the neighbourhood size is insignificant individually.

Moreover, F1-Score threshold used in the first step may change the final results. For example, in the TSL log, for 4 and 5 clusters and size of 10 for neighbourhood, 0.85 as threshold results in higher F1-scores compared to 0.95. Nonetheless, for the same number of clusters but with a neighbourhood size of 30, the effect of threshold is neutralised. In Road, there is an immediate improvement in F1-score when the threshold is lowered as well.

In the BPI 2013 log, it is apparent that a lower threshold achieves higher F1-score. However, this effect is only applicable to 4 and 5 clusters and not to 3 clusters. This finding conforms to the results of TSL log and Road that when looking for 3 clusters, input parameter variations does not make a significant difference.

### 6.3.1 Discussion

It can be argued that collective effects of input parameters is much greater than each parameters' individual effect. The initial F1-score appears to have a significant effect on F1-score only when there more than 3 clusters. The neighbourhood's size shows its meaningful impact when the initial F1-score threshold is set to 0.85. While the effect could lose its significance when the threshold is set to 0.95. In addition, since every event log represents a different behaviour, it is infeasible to find an input configuration for Hybrid technique that is optimal for all event logs.

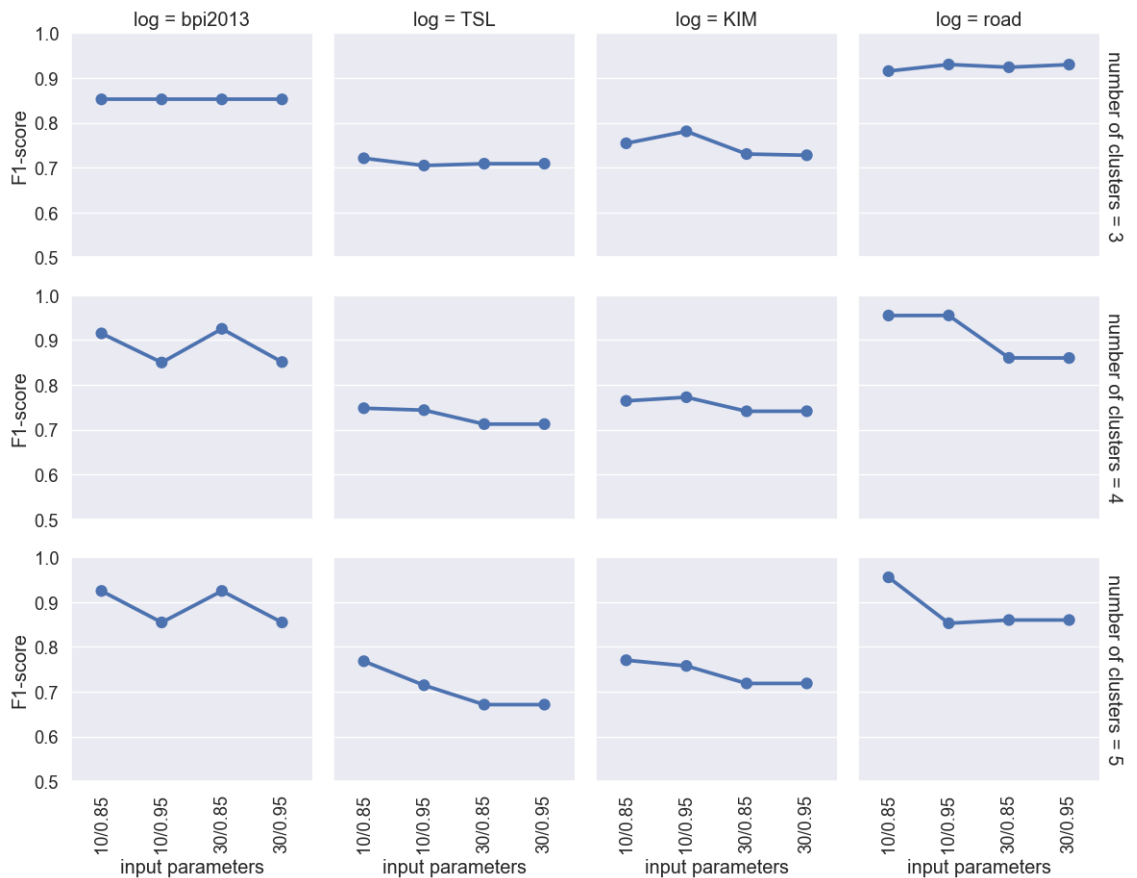


Figure 6.5: Effect of 3 input parameters on four real-life logs

Other input parameters were initially considered. However, since they did not change the final results significantly, even with collaboration with other parameters, they were dropped eventually. For instance, the minimum dis-

tance that qualifies two traces as close to be put in the same cluster was initially an input parameter with 2, 5, and 8 as the parameter space. However, no significant change was made into the final models' F1-score.

## 6.4 SQ 3: Comparative Analysis

This section addresses the third sub-research question by showing the results of the relevant experiments. The third research question is formulated as follows:

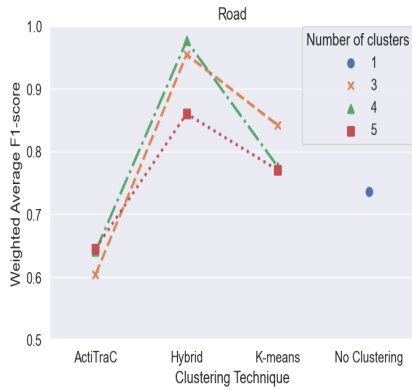
**SQ3:** *How does the Hybrid technique perform compared to existing approaches?*

In this section, Hybrid technique is evaluated in terms of quality against two other techniques, each with a different underlying clustering paradigm: *ActiTraC*, and *K-means*. Output models of mentioned techniques for all logs are available in the appendix A. Input parameters used for Hybrid technique are K-means as clustering technique, Levenshtein as distance metric, 0.95 and 10 as F1-score threshold and neighbourhood size respectively. In this thesis, these values are called *default input parameters*.

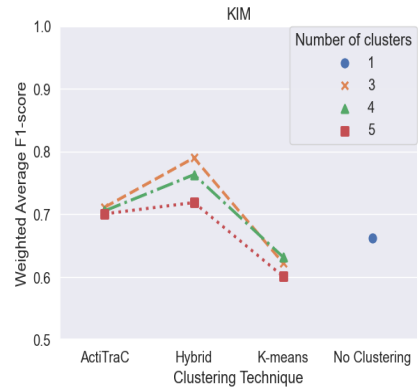
In order to show how much trace clustering improves the quality of process models, the results of applying *No Clustering* is also included in the comparative analysis. Six plots in figure 6.6 shows the weighted average F1-scores against clustering techniques for each log. Each point in a plot reveals the mean of weighted average F1-scores of cluster models when the goal was to find a certain number of clusters.

The most striking aspect of plots in figure 6.6 is that except for BPI 2013 log in 6.6e, Hybrid technique is able to find clusters with the highest mean F1-scores compared to other techniques. In 6.6e, Hybrid technique only fails to perform better than traditional K-means in 4 and 5 clusters. Although *ActiTraC* is a clustering technique with model-driven approach, it is yet to deliver models with higher F1-score compared to K-means. This case can be noticed in Road (6.6a), BPI 2012 (6.6d), BPI 2013 (6.6e), and BPI 2018 (6.6f) logs, where K-means has higher weighted average F1-scores. Bearing in mind that Hybrid technique uses K-means, it can be observed that combining similarity-based and model-driven techniques result in better overall performance.

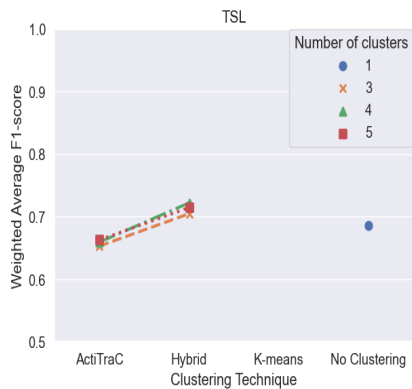
Although Hybrid technique results outperform other techniques in quantita-



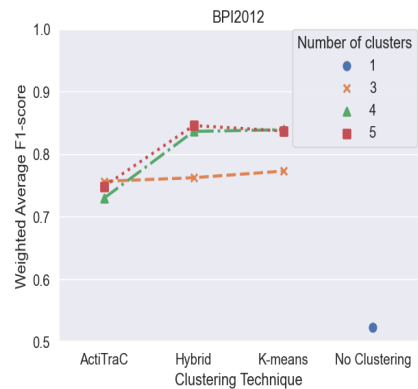
(a) Road Traffic Fine.



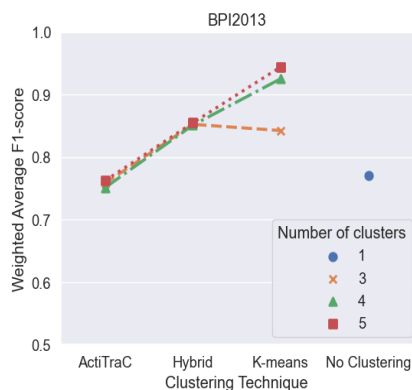
(b) KIM.



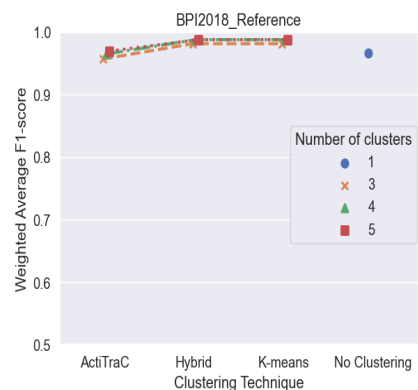
(c) TSL.



(d) BPI 2012.



(e) BPI 2013.



(f) BPI 2018-Reference.

Figure 6.6: Weighted Average F1-score against Clustering techniques

tive analysis, this thesis also analyzes the quality of process models of each cluster for each technique. This is done to investigate the differences between resulting process models, in terms of size of clusters, homogeneity, and most importantly, the models' simplicity. Therefore, first Hybrid and ActiTraC output models are compared and discussed on three different logs.

Table 6.7 illustrates cluster models of both techniques when applied on BPI 2013 log with four as the number of desired clusters. In the first glance, it can be observed from table 6.7 that while the first two output clusters of the Hybrid technique have a higher F1-score, they look more complicated than the first two clusters of ActiTraC. However, the first two clusters of Hybrid technique hold more traces (3038 and 1171) than ActiTraC (1749 and 524). Another observation is that activity *Queued* as the third most frequent activity (17.62%), is absent in the first two big clusters of ActiTraC. At the same time, *Queued* is represented in the second and third cluster of Hybrid technique.

Table 6.8 shows another example. Note that for this log, all models initiate with *Create Fine* activity. For the sake of better understanding of comparison, we ignore this activity when counting the number of activities in a model or discussing model's complexity.

While the first two clusters' size for both techniques is close to each other, ActiTraC produced models with higher complexity and lower F1-score. The Hybrid technique's first cluster only includes 3 activities, while ActiTraC covers 10 activities. For example, in the first cluster of Hybrid in figure 6.8, after a fine is created, a process can be finished when either one of *Appeal to Judge*, *Send Fine*, and *Payment*, the most frequent activities of this log, is executed. However, this behaviour is present in all clusters of ActiTraC alongside other behaviours. Even in the second cluster of Hybrid technique, you could not finish a process with just executing one of the *Appeal to Judge*, *Send Fine*, and *Payment* activities. Also, all sorts of behaviour can be seen in the third cluster of Hybrid technique only because it is the trash cluster. The trash clusters is only made after fit and precise clusters are finalized first.

Type	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	1	3038	0.983
ActiTraC	4	1	1749	0.749
Hybrid	4	2	1171	0.862
ActiTraC	4	2	524	0.8
Hybrid	4	3	534	0.775
ActiTraC	4	3	352	0.667
Hybrid	4	4	2811	0.717
ActiTraC	4	4	4929	0.752

Table 6.7: Process models discovered from BPI 2013 event log for 4 clusters by Hybrid technique and ActiTraC

Type	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	3	1	56482	1.0
ActiTraC	3	1	58813	0.687
Hybrid	3	2	66756	1.0
ActiTraC	3	2	59585	0.478
Hybrid	3	3	27132	0.747
ActiTraC	3	3	31972	0.68

Table 6.8: Process models discovered from Road Traffic Fine event log for 3 clusters by Hybrid technique and ActiTraC

### 6.4.1 Discussion

Clusters resulting from ActiTraC tend to be more general than the Hybrid technique, meaning that they include more trace variants which could belong to other clusters. ActiTraC attempts to put as many traces as possible in a cluster only if they fit the model and consider no punishment for traces existing in other clusters. On the other hand, output models of Hybrid are mainly limited to a set of specific trace variants. Accordingly, Hybrid technique is more successful than ActiTraC in satisfying the fundamental provision of clustering, building clusters with a high degree of intracluster homogeneity while these clusters have a high degree of intercluster heterogeneity. It can be argued that since Hybrid and ActiTraC underlying algorithms are essentially different, no concrete evidence can be given about the reasons of differences in resulting cluster models. Nevertheless, the fact that Hybrid technique considers both fitness and precision and the absence of any precision metric in ActiTraC as the intermediate model quality criteria could be one of reasons of the difference between two techniques.

Another argument regarding the difference between ActiTraC and Hybrid could be that the focus of ActiTraC is on the quality of initial clusters. This can be observed in the number of traces in each cluster. Many traces are pushed back to trash cluster, making these clusters worthless since the derived models are extremely complex. For example, trash clusters of ActiTraC in both Road Traffic fine and BPI 2013 event logs are larger than Hybrid, as can be seen in tables 6.8 and 6.7 respectively. This could also be the reason why ActiTraC performed poorly in calculation of weighted average F1-scores of cluster models compared to Hybrid in figure 6.6. Large trash clusters with low F1-scores penalize the overall performance significantly.

### 6.4.2 Performance Analysis

This section discusses runtime performance and scalability of the Hybrid technique. First, the runtime of Hybrid on four real-life event logs is compared to ActiTraC's runtime. Then, the scalability of Hybrid technique is assessed individually based on artificial logs.



### Runtime Comparison

Table 6.9 shows the four event logs used in the assessment against ActiTraC. These event logs have been specifically selected to have the same experimental setting. Due to technical difficulties, which will be later explained in the limitation part, it was opted to narrow the evaluation to event logs in table 6.9. Since these four logs are used for performance evaluation in the ActiTraC origin paper [16], they set a sound ground for a valid comparison with Hybrid technique.

Event Log	#Cases	#Variants	#Unique Activities
ICP	12391	1411	70
MCRM	956	212	22
KIM	24770	1174	18
TSL	17812	1908	42

Table 6.9: ActiTraC Real-life Event Logs

Table 6.10 presents the performance numbers of average runtimes for both ActiTraC variants (*freq* and *MRA*) and Hybrid techniques, best results embolden for each log.

It is apparent from the table 6.10 that Hybrid technique outperform both ActiTraC variants in *ICP*, *KIM*, and *TSL* logs. *MCRM* is the only log that *MRA* variant of ActiTraC performs faster than Hybrid, though *freq* variant performs only slightly better than Hybrid technique.

Event Log \ Technique	Hybrid	ActiTraC <sup>freq</sup>	ActiTraC <sup>MRA</sup>
ICP	<b>52.873</b>	348.682	618.691
MCRM	82.165	61.739	<b>13.328</b>
KIM	<b>15.277</b>	279.912	178.496
TSL	<b>228.222</b>	302.383	371.68

Table 6.10: Average runtimes of different techniques in seconds

### Scalability

Hybrid algorithm's computational complexity depends considerably on sev-

eral factors. Underlying process discovery technique, how fitness and precision are calculated in each iteration, and distance metrics used, all affect the performance of the technique. However, in order to assess how performance scales with respect to the input log characteristics, two main criteria are considered in this section, namely the number of trace variants and the number of unique activities. Two sets of artificial logs were created to investigate the effect of these two criteria. Event logs are generated using the PTandLogGenerator [26] package in ProM [40], an open source process mining tool. For scalability assessments in this section, each log has been tested with 5 experiment runs, and the run time results are provided with a 95 % confidence interval.

As can be seen in the Table 6.11, the first set is comprised of five event logs. To determine the effect of increasing the number of trace variants, all of the logs have the same number of unique activities. This number is set to 15. According to the table below, the runtime increases linearly in terms of number of trace variants. Looking at Figure 6.7, it is also apparent that for 3, 4, and 5 clusters, runtime is linear in terms of number of trace variants.

#Clusters \ #Variants	100	300	597	996	1985
3	15.2	21.2	54.6	98.7	216.9
CI	[8.45, 20.94]	[21.19, 21.35]	[30.38, 82.94]	[80.97, 113.90]	[139.52, 283.00]
4	18.0	37.8	76.2	142.1	1429.3
CI	[11.49, 25.69]	[34.73, 40.45]	[73.96, 78.10]	[131.61, 154.48]	[796.11, 1970.02]
5	21.8	55.8	133.5	206.4	1503.7
CI	[17.51, 26.77]	[34.29, 80.97]	[128.52, 139.37]	[151.06, 253.67]	[1168.21, 1790.17]

Table 6.11: Runtime (in seconds) on artificial logs with 15 unique activities

The second set of artificial event logs were made to examine how Hybrid technique scales with respect to the number of unique activities in a log. It is worthy to note that although the second set of artificial event logs have different number of unique activities, all of them have 1000 trace variants. Table 6.12 shows the average run times of event logs for 3, 4, and 5 clusters. Figure 6.8 also plots the increase of run time in accordance with the increase of unique activities on a logarithmic scale.

According to table 6.12, while the number of unique activities increases 5 units on each step, the run time does not follow a same pattern. For example for 4 clusters, from 10 to 20 activities, the run time increases from 55.45 seconds to 254.12 seconds. This means while number of activities doubled,

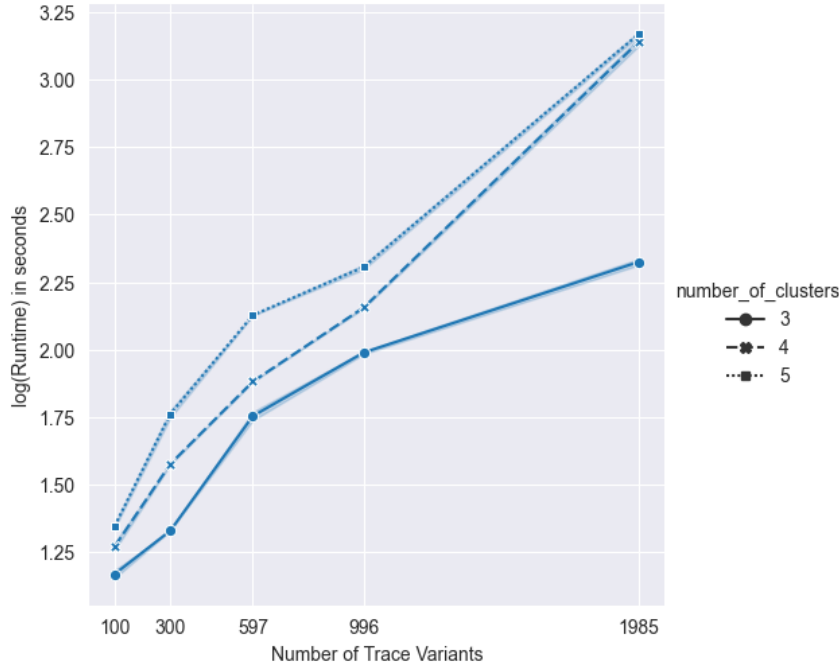


Figure 6.7: Scalability of Hybrid technique in terms of the number of trace variants

run time was 4 times more than 55.45. It can be observed that, growth rate from 15 to 20 is not equal to growth from 20 to 30. However, the growth rate is not exponential in overall.

#Clusters \ #Activities	10	15	20	25	30
3	39.46	53.57	206.42	204.98	153.77
CI	[14.36, 64.55]	[43.17, 63.98]	[76.75, 336.09]	[121.01, 288.96]	[91.19, 216.36]
4	55.45	71.61	254.12	314.68	230.91
CI	[35.66, 75.23]	[60.41, 82.81]	[157.39, 350.85]	[198.05, 431.31]	[174.61, 287.21]
5	104.09	90.89	301.01	354.11	273.41
CI	[28.74, 179.43]	[76.26, 105.53]	[224.89, 377.13]	[289.93, 418.28]	[205.01, 341.81]

Table 6.12: Runtime (in seconds) on artificial logs with 1000 trace variants

## 6.5 Overall Discussion

In this section, an overall discussion of the algorithm's results is delivered. The findings of this thesis are based on pre-defined algorithm parameters,

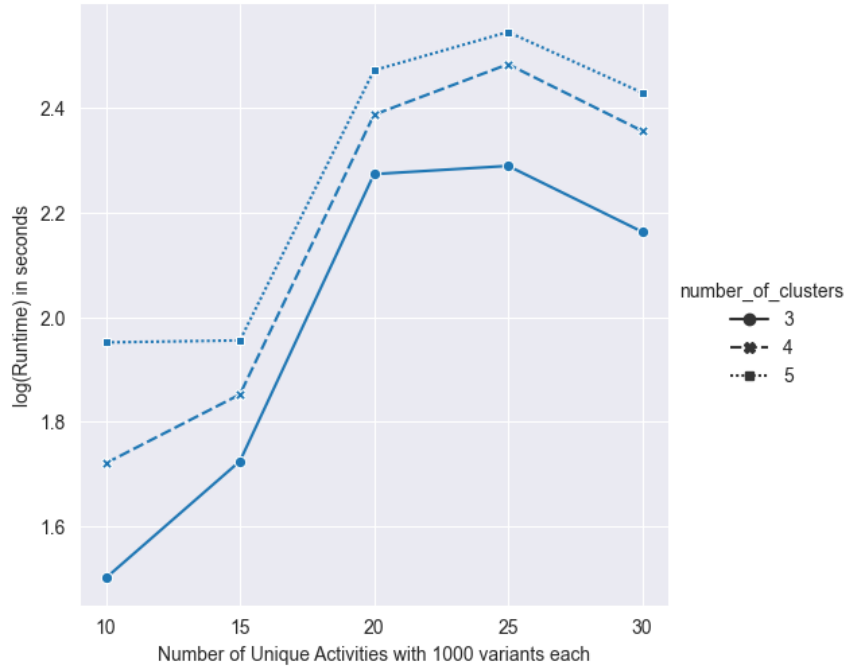


Figure 6.8: Scalability of Hybrid technique in terms of the number of unique activities

and algorithm only takes into consideration the control-flow perspective of traces when clustering.

Results of Hybrid algorithm in figure 6.6 (comparison with other techniques), are based on a default input parameters. However, this default setting did not necessarily resulted in the best performance. For example, in most real-life logs, DBSCAN resulted in higher quality clusters with run time being compromised compared to K-means, as can be observed in figures 6.2 and 6.4. Therefore, if taking a long time is not a concern, DBSCAN with Levenshtein might be the better first option for clustering. On the other hand, there are other options for having a faster clustering. Setting a lower F1-score threshold would make algorithm accept more clusters in the first step, which helps the model-driven part of the algorithm to process fewer trace variants. Choosing a larger size for the neighbourhood too builds a greater initial cluster that consequently, reduces the computational complexity of the model-driven part. However, as discussed in section 6.3, a larger neighbourhood resulted in lower average F1-scores. While the combination of K-means with Levenshtein, lower F1-score thresholds, and larger neighbourhood sizes may be a more rational option for a faster clustering, it could compromise

the quality of process models.

Moreover, since determining the optimal number of clusters is a classic problem in data clustering, evaluation of Hybrid technique was conducted only on 3, 4, and 5 clusters. However, these three numbers are not optimal for all event logs, as figure 6.6c (TSL) suggests that only a slight improvement was witnessed through clustering. Therefore, the task of discovering the number of clusters could be assigned to domain experts in order to have cluster models with the highest quality.

Lastly, current evaluation results are limited to only six real-life logs with specific algorithm parameters, including Inductive miner and F1-score as discovery algorithm and intermediate model quality criteria, respectively. Hybrid technique is designed in a way that different distance metrics, clustering algorithms, discovery algorithm, and model quality criteria can be integrated in it. Therefore, a more comprehensive benchmarking setting could be devised to find the most optimal results, in terms of both run time and model quality, for each event log.

# Chapter 7

## Conclusion

### 7.1 Conclusion

Although research on improvement of process discovery algorithms continues, the motivation for dividing the event logs to smaller logs remains the same. Trace clustering techniques improve both quality of process models, as well as reduces the amount of time needed to discover a single model. The algorithm in this thesis is designed to achieve the same goals of a trace clustering technique, with higher quality and performance. Hybrid technique is devised after a thorough literature review and identification of current trace clustering deficiencies.

Data clustering algorithms group data points based on their distance in a feature vector space. However, they are unable to perform strongly under process-oriented event logs. Researchers attempted to overcome this deficiency by adopting data clustering ideas in the process mining context. However, some of the resulting clusters may contain irrelevant traces or may still be rather complicated. On the other hand, a second type of techniques was developed that only focused on the quality of process models. Nevertheless, these type of techniques suffer from considerable computational complexity. Hybrid technique proposed in this thesis takes advantage of traditional data clustering algorithms in the first step to find high quality clusters in a reasonable run time. In the next step of the algorithm, based on the distances between traces, clusters are initialized. These new clusters then follow a models-driven approach to find traces that inherently improve process model's fitness and precision at the same time.

The Hybrid technique is evaluated on six real-life event logs both in terms of

run time and quality. The results show that it produces clusters with higher overall average F1-score compared to ActiTraC, a prominent clustering technique in the literature and K-means, a data clustering technique. Process models produced by Hybrid technique also shown to have more homogeneity compared to ActiTraC and K-means models. In other words, Hybrid's models more distinguishable than other techniques since mostly they do not share the same trace variants, or even close variants with each other.

In addition, the run time comparison results demonstrated that Hybrid technique runs faster than ActiTraC in 3 out of 4 event logs used in the ActiTraC original paper. Results of scalability assessments also show that Hybrid technique's run time grows linearly in terms of the number of trace variants and also, it does not grow exponentially in terms of number of distinct activities.

Finally, in order to test the Hybrid technique, it has been implemented in Python and is publicly available at GitHub. Researchers are capable of configuring input parameters and investigate different variations of the algorithm. To this reason, the Hybrid technique can contribute to researchers' work both in the process mining and data mining fields.

## 7.2 Limitations

An important limitation lies in the fact that the performance evaluation of Hybrid technique and ActiTraC could not take place in the same environment. This is because ActiTraC was implemented in ProM using Java, while Hybrid technique is implemented in Python. Therefore, runtime comparisons was only limited to four logs used in the ActiTraC paper. Due to technical differences between Java and Python, the runtime numbers could not be entirely accurate.

Secondly, the study was also limited by the high computational complexity of the alignment-based evaluation on two occasions. The First one is regarding BPI 2012 log, as F1-score of produced clusters could not be computed in a practical state in terms of time. Although the log was sampled with three different sizes and 5 times for each, caution must be applied as the findings might not be completely transferable to the original log. Due to the same reason, the F1-scores of clusters produced by K-means for TSL log were not be computed. However, by looking at the extremely complex cluster models in the A.7 and A.8, it can be concluded that they have a low F1-score. Consequently, they would not invalidate the fact that Hybrid

technique performed better.

### 7.3 Future Work

There are several avenues for future work. Firstly, further studies regarding the role of data clustering algorithms in the first step would be worthwhile. These data clustering algorithms can be the subject of optimisation, and could also be incorporated in the further steps of the Hybrid algorithm again.

Finding neighbour traces in the Hybrid algorithm is now only based on frequency. Although techniques like K-nearest neighbour (KNN) compromise the algorithm's performance with training time requirement, they would make the algorithm more accurate if optimised. Therefore, KNN and similar approaches could be investigated in the future studies.

Accurately incorporating more input parameters within the algorithm could be another avenue for future work. An example of this could be the minimum distance that qualifies two close traces. Also, as of now, only one syntax-based and one vector-based distance metrics are used. In future investigations, it might be possible to use different distance metrics.

Lastly, further research could also be conducted to determine the effectiveness of different intermediate process model quality criteria. Different weights could be given to fitness and precision, or metrics concerning the model's complexity could be incorporated.



# Bibliography

- [1] ActiTraC: Active Trace Clustering. process mining research from researchers in belgium. <http://processmining.be/actitrac/>.
- [2] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, Andrea Marrella, Massimo Mecella, and Allar Soo. *Automated discovery of process models from event logs: Review and benchmark*, volume 31. IEEE, 2018.
- [3] Alessandro Berti and Wil MP van der Aalst. Reviving token-based replay: Increasing speed while improving diagnostics. In *ATAED@ Petri Nets/ACSD*, pages 87–103, 2019.
- [4] Alessandro Berti, Sebastiaan J van Zelst, and Wil van der Aalst. Process mining for python (pm4py): bridging the gap between process-and data science. *arXiv preprint arXiv:1905.06169*, 2019.
- [5] RP Jagadeesh Chandra Bose and Wil MP Van der Aalst. Context aware trace clustering: Towards improving process mining results. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 401–412. SIAM, 2009.
- [6] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. Trace clustering based on conserved patterns: Towards achieving better process models. In *International Conference on Business Process Management*, pages 170–181. Springer, 2009.
- [7] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Model-based clustering and visualization of navigation patterns on a web site. *Data mining and knowledge discovery*, 7(4):399–424, 2003.
- [8] Paolo Ceravolo, Ernesto Damiani, Mohammadsadegh Torabi, and Sylvio Barbon. Toward a new generation of log pre-processing methods for

- process mining. In *International Conference on Business Process Management*, pages 55–70. Springer, 2017.
- [9] BPI Challenge. 4tu. centre for research data, 2012.
- [10] BPI Challenge. 4tu. centre for research data, 2013.
- [11] BPI Challenge. 4tu. centre for research data, 2018.
- [12] Pieter De Koninck and Jochen De Weerd. Scalable mixed-paradigm trace clustering using super-instances. In *2019 International Conference on Process Mining (ICPM)*, pages 17–24. IEEE, 2019.
- [13] Pieter De Koninck, Klaas Nelissen, Bart Baesens, Seppe vanden Broucke, Monique Snoeck, and Jochen De Weerd. An approach for incorporating expert knowledge in trace clustering. In *International Conference on Advanced Information Systems Engineering*, pages 561–576. Springer, 2017.
- [14] Ana Karla A de Medeiros, Anton JMM Weijters, and Wil MP van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.
- [15] Ana Karla Alves De Medeiros, Antonella Guzzo, Gianluigi Greco, Wil MP Van Der Aalst, AJMM Weijters, Boudewijn F Van Dongen, and Domenico Saccà. Process mining based on clustering: A quest for precision. In *International Conference on Business Process Management*, pages 17–29. Springer, 2007.
- [16] Jochen De Weerd, Seppe Vanden Broucke, Jan Vanthienen, and Bart Baesens. Active trace clustering for improved process discovery. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2708–2720, 2013.
- [17] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
- [18] Chiara Di Francescomarino, Marlon Dumas, Fabrizio Maria Maggi, and Irene Teinemaa. Clustering-based predictive process monitoring. *IEEE transactions on services computing*, 2016.
- [19] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A Reijers. *Business process management*. Springer, 2013.

- [20] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.
- [21] Diogo Ferreira, Marielba Zacarias, Miguel Malheiros, and Pedro Ferreira. Approaching process mining with sequence clustering: Experiments and findings. In *International conference on business process management*, pages 360–374. Springer, 2007.
- [22] Gianluigi Greco, Antonella Guzzo, Luigi Pontieri, and Domenico Sacca. Discovering expressive process models by clustering log traces. *IEEE Transactions on knowledge and data engineering*, 18(8):1010–1027, 2006.
- [23] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
- [24] BFA Hompes, Joos CAM Buijs, Wil MP van der Aalst, Prabhakar M Dixit, and Johannes Buurman. Detecting changes in process behavior using comparative case clustering. In *International Symposium on Data-Driven Process Discovery and Analysis*, pages 54–75. Springer, 2015.
- [25] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.
- [26] Toon Jouck and Benoît Depaire. Ptdandloggenerator: A generator for artificial event data. *BPM (Demos)*, 1789:23–27, 2016.
- [27] Sander JJ Leemans, Dirk Fahland, and Wil MP van der Aalst. Discovering block-structured process models from event logs—a constructive approach. In *International conference on applications and theory of Petri nets and concurrency*, pages 311–329. Springer, 2013.
- [28] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.
- [29] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.
- [30] Xixi Lu, Seyed Amin Tabatabaei, Mark Hoogendoorn, and Hajo A Reijers. Trace clustering on very large event data in healthcare using frequent sequence patterns. In *International Conference on Business Process Management*, pages 198–215. Springer, 2019.

- [31] Felix Mannhardt, Massimiliano De Leoni, Hajo A Reijers, and Wil MP Van Der Aalst. Balanced multi-perspective checking of process conformance. *Computing*, 98(4):407–437, 2016.
- [32] Carl Adam Petri and Wolfgang Reisig. Petri net. *Scholarpedia*, 3(4):6477, 2008.
- [33] Willem Roper. *Python Remains Most Popular Programming Language*. <https://www.statista.com/chart/21017/most-popular-programming-languages/>, March 2020.
- [34] Anne Rozinat and Wil MP Van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.
- [35] Mohammadreza Fani Sani, Mathilde Boltenhagen, and Wil van der Aalst. Prototype selection based on clustering and conformance metrics for model discovery. *arXiv preprint arXiv:1912.00736*, 2019.
- [36] Minseok Song, Christian W Günther, and Wil MP Van der Aalst. Trace clustering in process mining. In *International conference on business process management*, pages 109–120. Springer, 2008.
- [37] Tom Thaler, Simon Felix Ternis, Peter Fettke, and Peter Loos. A comparative analysis of process instance cluster techniques. *Wirtschaftsinformatik*, 2015:423–437, 2015.
- [38] Wil Van Der Aalst. Data science in action. In *Process mining*, pages 3–23. Springer, 2016.
- [39] Wil Van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
- [40] Boudewijn F Van Dongen, Ana Karla A de Medeiros, HMW Verbeek, AJMM Weijters, and Wil MP van Der Aalst. The prom framework: A new era in process mining tool support. In *International conference on application and theory of petri nets*, pages 444–454. Springer, 2005.
- [41] Stijn vanDongen. A cluster algorithm for graphs. *Information Systems [INS]*, (R 0010), 2000.

- [42] Gabriel M Veiga and Diogo R Ferreira. Understanding spaghetti models with sequence clustering for prom. In *International conference on business process management*, pages 92–103. Springer, 2009.
- [43] Stephen A White. Introduction to bpmn. *Ibm Cooperation*, 2(0):0, 2004.
- [44] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.
- [45] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

# Appendix A

## Output Process Models

Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	1	3038	0.984
ActiTraC	4	1	1749	0.75
K-means	4	1	4509	0.885
Hybrid	4	2	1171	0.862
ActiTraC	4	2	524	0.8
K-means	4	2	3038	0.984

Table A.1: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2013 event log for 4 clusters(first and second clusters)

Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	3	534	0.775
ActiTraC	4	3	352	0.667
K-means	4	3	2	0.565
Hybrid	4	4	2811	0.717
ActiTraC	4	4	4929	0.752
K-means	4	4	5	1.0

Table A.2: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2013 event log for 4 clusters(third and fourth clusters)



Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	1	3660	1.0
ActiTraC	4	1	18422	0.758
K-means	4	1	5092	0.523
Hybrid	4	2	6762	1.0
ActiTraC	4	2	2761	0.573
K-means	4	2	6622	0.758

Table A.3: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from KIM event log for 4 clusters(first and second clusters)

Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	3	3442	1.0
ActiTraC	4	3	953	0.543
K-means	4	3	6242	0.49
Hybrid	4	4	10906	0.632
ActiTraC	4	4	2634	0.527
K-means	4	4	6814	0.718

Table A.4: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from KIM event log for 4 clusters(third and fourth clusters)

Type	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	1	56482	1.0
ActiTraC	4	1	58813	0.687
K-means	4	1	20540	0.798
Hybrid	4	2	66756	1.0
ActiTraC	4	2	59585	0.478
K-means	4	2	56887	0.590

Table A.5: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from Road Traffic Fine event log for 4 clusters (first and second clusters)

Type	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	3	19688	0.910
ActiTraC	4	3	20767	0.908
K-means	4	3	22973	0.725
Hybrid	4	4	7444	0.758
ActiTraC	4	4	11205	0.758
K-means	4	4	49970	0.999

Table A.6: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from Road Traffic Fine event log for 4 clusters (third and fourth clusters)

Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	1	2877	0.845
ActiTraC	4	1	5587	0.769
K-means	4	1	9182	-
Hybrid	4	2	2783	0.881
ActiTraC	4	2	4591	0.682
K-means	4	2	3115	-

Table A.7: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from TSL event log for 4 clusters(first and second clusters)

Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	3	1116	0.837
ActiTraC	4	3	246	1.0
K-means	4	3	2510	-
Hybrid	4	4	11036	0.638
ActiTraC	4	4	7388	0.55
K-means	4	4	3005	-

Table A.8: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from TSL event log for 4 clusters(third and fourth clusters)

Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	1	4696	1.0
ActiTraC	4	1	3480	1.0
K-means	4	1	2246	0.73
Hybrid	4	2	1876	1.0
ActiTraC	4	2	1876	1.0
K-means	4	2	4696	1.0

Table A.9: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2012 event log for 4 clusters(first and second clusters)

Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	3	1806	0.657
ActiTraC	4	3	1220	0.676
K-means	4	3	3376	0.813
Hybrid	4	4	4709	0.676
ActiTraC	4	4	6511	0.516
K-means	4	4	2769	0.685

Table A.10: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2012 event log for 4 clusters (third and fourth clusters)



Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	1	30738	1.0
ActiTraC	4	1	30734	1.0
K-means	4	1	30738	1.0
Hybrid	4	2	10332	0.957
ActiTraC	4	2	5448	0.914
K-means	4	2	10332	0.957

Table A.11: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2018-Reference event log for 4 clusters (first and second clusters)

Clustering technique	Number of clusters	cluster number	Number of Traces	F1-score
Hybrid	4	3	2633	0.973
ActiTraC	4	3	2252	1.0
K-means	4	3	2633	0.973
Hybrid	4	4	99	0.75
ActiTraC	4	4	5368	0.8
K-means	4	4	99	0.75

Table A.12: Process models discovered by Hybrid, ActiTraC, and K-means algorithms from BPI 2018-Reference event log for 4 clusters (third and fourth clusters)